Optimal Planar Point Location

John Iacono[*]

Abstract

Given a fixed distribution of point location queries among the regions of a triangulation of the plane, a data structure is presented that achieves, within constant multiplicative factors, the entropy bound on the expected point location query time.

1 Introduction

Point location is one of the fundamental problems in computational geometry. In a point location problem, a subdivision of space is provided, and a data structure is constructed such that, for each query point, a pointer to the region in the subdivision which the query point lies in is returned. This problem has been well studied, and different restrictions on the dimension and nature of the subdivision have yielded numerous data structures. In two dimensions, data structures have been created (e.g. [4] using [2] for preprocessing) which have attained the best possible asymptotic query time of $O(\log n)$, and construction time of $O(n)$, for a planar polygonal subdivision, where n is the size of the subdivision.

Given a triangulation of the plane, T, into n regions $r_1 \dots r_n$, if the probability of accessing region r_i on every point location query is $p(r_i)$ then the lower bound for the expected cost per operation is given by the entropy: $\sum_{i=1}^{n} p(r_i) \log \frac{1}{p(r_i)}$. Thus, for non-uniform access distributions, expected $o(\log n)$ point location queries are possible.

The main result of this paper is a data structure that achieves the entropy bound, within a constant multiplicative factor, when the access probabilities are known. Our data structure, which is a variant of Kirkpatrick's [4], can be constructed in $O(n)$ time and space, and has a worst-case cost of $O(\log n)$ per point location query.

This data structure can also be used to efficiently perform offline point location. Given a sequence of m point location queries in a triangulation of the plane into n regions, where $f(r_i)$ of the queries are in region r_i, our data structure can process these queries in expected time $O(n + \sum_{i=1}^{n} f(r_i) \log \frac{m}{f(r_i)})$, and linear space. Moreover, this data structure achieves this runtime without knowing the $f(r_i)$ values a priori.

[*]Department of Computer Science, Rutgers University, New Brunswick, NJ 08903. `iacono@john.rutgers.edu`. Research Supported by NSF grant CCR-9732689

2 Previous Results

Arya et. al. [1] recently provided a data structure for planar point location that achieved expected performance within a small multiplicative constant of the entropy bound. However, their structure has a number of disadvantages. First, they require the horizontal and vertical coordinates of the point location queries to be generated from two independent probability distributions. Secondly, their structure requires $O(n \log n)$ preprocessing time. Their structure allows a general polygonal subdivision of the plane, as opposed to our algorithm which only works on triangulations. However, they require that the perimeter of each region be bounded in a way which restricts increasingly complex polygons into regions of increasingly small probability. This restriction circumvents the fact that it is impossible to achieve, within constant factors, the entropy bound for point location among the regions of a general polygonal subdivision of the plane.

Goodrich, Orletsky and Ramaiyer [3] provided a data structure for the point location problem that achieves the same $O(n + \sum_{i=1}^{n} f(r_i) \log \frac{m}{f(r_i)})$ runtime as our structure over a point location query sequence, with two advantages. Their structure is online, and is deterministic. As their structure is based on splay trees, the runtimes of individual point location queries are amortized and may take linear time. The disadvantage of the structure is that the runtime is relative not to the user provided subdivision of the plane, but rather to one constructed by passing a vertical line through every point in the original subdivision. This restriction can cause their structure to perform asymptotically worse than ours over certain classes of of subdivisions and point location query sequences. Their structure also requires $O(n \log n)$ time for construction, as opposed to $O(n)$ for our structure.

Thus, the significance of our result is to achieve the entropy bound for point location in a triangulation, within constant multiplicative factors, without any unnatural assumptions or restrictions.

3 Data Structure

For a triangulation T of n regions, $r_1, r_2 \dots r_n$, and a set of n positive values $x_1, x_2 \dots x_n$, it is possible to build a data structure in time $O(n)$ that can answer a point location query on a point in r_i in deterministic time $O\left(\log \frac{\sum_{j=1}^{n} x_j}{x_i}\right)$.

By choosing x_i to be $p(r_i)$, the probability of accessing region r_i, the entropy bound can be achieved.

The data structure is a variant of Kirkpatrick's planar point location structure [4]. We assume that the triangulation has been preprocessed so that it contains only three exterior points, and the the x_i values are normalized so that $\sum_{i=1}^{n} x_i = 1$. It is also assumed that the triangulation T contains at least 25 vertices.

The data structure consists of a series of triangulations $t_0, t_1 \dots t_m$. Each region in a triangulation t_i contains links to those regions in t_{i+1} and t_{i-1} that are not disjoint with it. A region is said to be terminal if it is found in the original triangulation T. Every region has an associated weight. The sum of weights in each triangulation is 1.

The final triangulation t_m is T, all other triangulations t_i are derived from t_{i+1} by removing an independent set of vertices, and arbitrarily retriangulating the resultant nontriangular regions. The weight of the region r_i in t_m is x_i. The weights of newly formed nonterminal regions is an arbitrary redistribution of the weights of the regions incident to the vertices whose removal caused the creation of these regions. The weight of a region in t_i which is also in t_{i+1} remains the same. The key to making this data structure work is carefully picking the right independent set of vertices for removal.

Suppose t_i has n_i regions and v_i vertices. At most $\frac{v_i}{4}$ vertices have degree higher then 24. At most $\frac{n_i}{12}$ of the regions have weights greater than $\frac{12}{n_i}$. As each region is triangular, at most $\frac{n_i}{4} < \frac{v_i}{4}$ vertices have an region incident whose weight is at least $\frac{12}{n_i}$. Therefore, at least $\frac{v_i}{2}$ vertices have degree at most 24 and do not have a region incident whose weight is more than $\frac{12}{n_i}$. We then may, by using a brute force algorithm, pick an independent set of size $\lfloor \frac{n_i}{25} \rfloor$ of vertices for removal with both of these properties. The removal of this set from t_i, and the retriangulation of the resultant nontriangular regions, is the method by which t_i is derived from t_{i+1}. Since each of the removed vertices has a degree of at most 24, the triangulation of the region formed by the removal of a vertex can be done in constant time, and each newly formed region in t_i will have at most 24 regions in t_{i+1} that it intersects in. The time to create t_i from t_{i+1} is $O(n_i)$. The process of removing vertices to create new triangulations stops when the number of vertices is 24. Thus $n_0 = 25$.

Point location is on a query point p is performed by locating the region in each triangulation $t_0, t_1 \dots$ that p lies in, until p is found to be in a terminal region of some triangulation t_j. For each step in this process, if the region that p lies in in t_i is known to be ρ, the region that p lies in in t_{i+1} may be determined by looking at the at most 24 regions in t_{i+1} that intersect with, and thus are linked to, ρ. Thus point location on a region that is terminal in triangulation t_j takes time $O(j)$, which is $O(\log n)$ in the worst case.

Analysis: Since $0.98 n_{i+1} \leq n_i$, and $n_0 \leq 23$, it can be seen that $n_i \leq 23(0.98)^{-i}$. Given a region r_j with weight x_j, r_j is guaranteed to remain a terminal region in all triangulations $t_j, \dots t_m$ where $x_j > \frac{12}{n_i}$. Thus the time to perform a point location query on a point in r_j is $O(\log \frac{1}{x_j})$.

4 Offline Point Location

Given a triangulation of the plane into n regions $r_1 \dots r_n$ and a sequence of m point location queries, where $f(r_i)$ of the point location queries in the sequence are to region r_i, the point location queries may be executed offline in expected time $O(n + \sum_{i=1}^{n} f(r_i) \log \frac{m}{f(r_i)})$.

Sample, independently and at random $\frac{m}{\log n}$ query points from the access sequence. Construct Kirkpatrick's data structure for planar point location for the triangulation, T. Perform point location using this data structure on the sampled points. Let g_i be the number of points in the randomly selected set that were found to be in r_i. Let x_i be 1 if $g_i = 0$ and $g_i \log n$ otherwise. Construct our data structure if the previous section using the triangulation T and x_i values that have just been computed. Answer all of the m point location queries using this data structure. The proof that this method achieves the stated runtime has been omitted.

References

[1] S. Arya, S. Cheng, D. Mount, and H. Ramesh. Efficient expected-case algorithms for planar point location. In *Proc. 7th Scandanavian Workshop on Algorithm Theory. LNCS 1851*, pages 353–366, 2000.

[2] B. Chazelle. Triangulating a simple polygon in linear time. *Discrete Computational Geometry*, 6:485–524, 1991.

[3] 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.

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