Adaptive Point Location in Planar Convex Subdivisions

Siu-Wing Cheng(✉) and Man-Kit Lau

Department of Computer Science and Engineering, HKUST,
Hong Kong, China
scheng@cse.ust.hk

Abstract. We present a planar point location structure for a convex subdivision S. Given a query sequence of length m, the total running time is $O(\mathrm{OPT} + m \log\log n + n)$, where n is the number of vertices in S and OPT is the minimum running time to process the same query sequence by any linear decision tree for answering planar point location queries in S. The running time includes the preprocessing time. Therefore, for $m \geq n$, our running time is only worse than the best possible bound by $O(\log\log n)$ per query, which is much smaller than the $O(\log n)$ query time offered by an worst-case optimal planar point location structure.

Keywords: Point location · Convex subdivision · Adaptive data structure

1 Introduction

There has been extensive research on planar point location—a fundamental problem in computational geometry—to obtain worst-case optimal query time, preprocessing time, and space complexity [2,16,20–25]. Some of them are now standard results in textbooks in computational geometry [8,11]. Planar point location can be seen as a generalization of the one-dimensional dictionary problem to two dimensions. In any dimension, the information theoretic lower bound in processing a sequence of m queries follows from Shannon's work [26] and the *entropy-based* lower bound is $\sum_z f(z) \cdot \log \frac{m}{f(z)}$, where $f(z)$ denotes the access frequency of an item z in the sequence of length m. The splay tree [27] has been designed such that, given an initially empty structure and a sequence of m insertions, deletions, and queries, the total running time for manipulating the data structure to process these operations is $O\left(\sum_z f(z) \cdot \log \frac{m}{f(z)}\right)$, where every insertion and deletion of z also contributes one to the access frequency of z. Notice that the access frequencies of items are unknown beforehand. As a result, $o(\log n)$ amortized query time is possible in one dimension if the access frequencies of the items are substantially unequal.

Supported by FSGRF14EG26, HKUST.

K. Elbassioni and K. Makino (Eds.): ISAAC 2015, LNCS 9472, pp. 14–22, 2015.
DOI: 10.1007/978-3-662-48971-0_2

For point location in a planar subdivision S, there are also previous works on making the performance adaptive to the access frequencies. When the regions in S have constant complexities, and the query distribution is fixed and available as part of the input, there are several works by Arya et al. [4–7] and Iacono [17] to construct a data structure such that the expected query time is $O\left(\sum_z p_z \log \frac{1}{p_z}\right)$, where p_z is the probability of a query point falling into the region z. The algorithm of Iacono [17] uses $O(n)$ space and $O(n)$ preprocessing time. The algorithm of Arya et al. [7] uses $O(n)$ space and $O(n \log n)$ preprocessing time, and its expected running time per query is optimal up to the leading constant factor modulo some additive lower-order terms. Subsequently, analogous results have been obtained for connected subdivisions [13] and disconnected subdivisions [1,9,10] in which the regions may have arbitrary complexities. In the aforementioned results, the query distribution is fixed and available as part of the input. A natural question is whether we can obtain a self-adjusting planar point location structure that can adapt to a query sequence without knowing the access frequencies of the regions beforehand. There has been only one such result in the case that S is a triangulation by Iacono and Mulzer [19]. They present a method that achieves a total running time of $O\left(n + \sum_z f(z) \cdot \log \frac{m}{f(z)}\right)$, including the preprocessing time to construct the initial structure before processing the query sequence.

In this paper, we study the adaptive point location problem for a convex subdivision S. That is, every region in S is a convex polygon (except the outer unbounded region). We do not require the regions in S to have constant complexities. One cannot just triangulate S, apply the result for triangulation by Iacono and Mulzer [19], and hope to achieve the entropy-based lower bound. Suppose that we encode the names of the regions using bit vectors of possibly different lengths. Then, the entropy-based lower bound is the minimum number of bits needed to encode the sequence of output region names corresponding to the m queries under the prescribed access frequencies. Each output bit requires at least one unit of processing time, and therefore, the entropy-based lower bound is also a lower bound for the total running time. Consequently, geometry is not taken into consideration at all. Arya et al. [7] show that one can design a convex polygon of n sides and a query distribution so that a query point lies in the polygon with probability $1/2$ and the expected number of point-line comparisons needed to decide whether a query point lies in the polygon is $\Omega(\log n)$. However, the entropy-based lower bound for a single query is only a constant in this case. This shows that the entropy-based lower bound is too weak for a convex subdivision. As in [13], we compare our result with the best linear decision tree for answering point location queries in S. This is reasonable because the linear decision tree models the process for answering a query by point-line comparisons, and many existing point location structures are based on point-line comparisons.[1]

[1] Methods that employ indexing (e.g. [15]) and bit tricks (e.g. [12]) do not fall under the linear decision tree model.

Given a sequence of m queries, our method runs in $O(\mathrm{OPT} + m \log\log n + n)$ total time, where OPT is the minimum time to process the same query sequence by any linear decision tree for answering point location queries in S. Our time bound includes the preprocessing time before processing the query sequence. Therefore, for $m \geq n$, our running time is only worse than the best possible bound by $O(\log\log n)$ time per query, which is much smaller than the $O(\log n)$ query time offered by an worst-case optimal planar point location structure.

One can build another auxiliary planar point location structure so that a query can be executed on our adaptive structure and this auxiliary point location structures simultaneously until one of the two structures returns an answer. The advantage is that this auxiliary point location structure can offer additional properties. For example, if one uses the distance-sensitive planar point location structure [3], it means that queries far away from any region boundary can be answered fast too. Alternatively, if one uses the proximate planar point location structure [18] as the auxiliary structure, then a query can be answered faster if the query point is close to the previous one.

2 Triangulation of a Convex Polygon

Let P be a convex region in S with n_P vertices in counterclockwise order $(v_0, v_1, ..., v_{n_P-1})$. We triangulate P as follows. Select every other vertex of P. (When n_P is odd, the last vertex selected is adjacent to the first vertex selected.) Let P_1 be the convex hull of these selected vertices. Clearly, $P_1 \subset P$, $P \setminus P_1$ is a collection of triangles, and the number of vertices of P_1 is at most $\lceil n_P/2 \rceil$. Then, we recurse on P_1 to construct P_2 and so on until we produce a convex hull P_j that is a single triangle or a single line segment. The triangulation of P is the collection of triangles in $P \setminus P_1$, $P_1 \setminus P_2$, etc. We denote this triangulation of P by T_P. Figure 1 shows an example. This hierarchical triangulation was first introduced by Dobkin and Kirkpatrick [14] in the context of detecting intersection between two convex polygons and polyhedra. Note that $O(\log n)$ P_i's are constructed because the size of the P_i's decreases repeatedly by a constant factor. The time to produce each P_i is $\lceil n_P/2^i \rceil$. Therefore, the total time to compute T_P is $O\left(\sum_{i=0}^{\infty} n_P/2^i\right) = O(n_P)$. A line segment ℓ in P intersects the boundary of each P_i's in at most two points. It follows that ℓ intersects at most two triangles in $P_{i-1} \setminus P_i$, and therefore, ℓ intersects $O(\log n)$ triangles in T_P. Interestingly, this simple hierarchical triangulation T_P leads to a query performance that is adaptive and only slightly worse than the best possible bound. In the following, we prove an upper bound on the entropy of T_P that is closely related to the performance of any linear decision tree.

Lemma 1. *Let P be a convex polygon in $\mathbb{R}^2$. Let $H(T_P)$ denote the entropy of T_P. Let $\mathcal{D}$ be an arbitrary linear decision tree for determining whether a query point in $\mathbb{R}^2$ lies in P. Let $L_{\mathcal{D}}$ be the set of leaves of $\mathcal{D}$ and for every leaf $\nu \in L_{\mathcal{D}}$, let r_ν denote the convex region represented by ν. Consider an arbitrary query sequence of length m. For any region $r \subseteq \mathbb{R}^2$, let $f(r)$ denote the number of queries that fall inside r. Then, the following inequality is satisfied.*

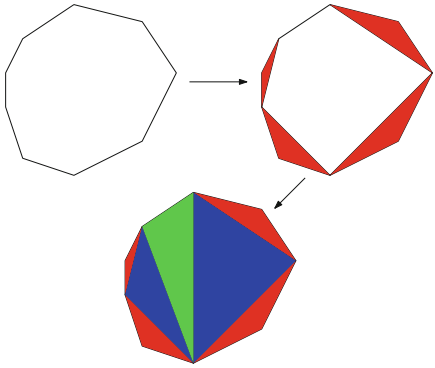

Fig. 1. Triangulation by convex hull. The red, blue and green triangles are obtained from the first, second and third convex hulls respectively (Color figure online)

$$H(T_P) = \sum_{t \in T_P} f(t) \cdot \log \frac{m}{f(t)}$$
$$\leq \sum_{\nu \in L_{\mathcal{D}}} f(r_\nu) \cdot (\text{depth}(\nu) + O(\log(\text{depth}(\nu))) + O(\log \log n))$$

Proof. For any line segment ℓ inside P, ℓ intersects at most two triangles in $P_i \setminus P_{i+1}$ in each level of the hierarchical triangulation T_P. Therefore, ℓ intersects at most $O(\log n)$ triangles in T_P. Let q be a query point that falls in the convex polygon r_ν for some leaf $\nu \in L_{\mathcal{D}}$. Let k be the number of sides of r_ν. We have $\text{depth}(\nu) \geq k$ because each internal node on the path from the root of $\mathcal{D}$ to ν corresponds to a cut along a line.

We can expand the linear decision tree $\mathcal{D}$ to another linear decision tree $\mathcal{D}''$ that allows us to identify the triangle $t \in T_P$ containing q. The construction of $\mathcal{D}''$ works in two steps as follows. For each leaf $\nu \in L_{\mathcal{D}}$, we recursively add a chord to split r_ν into two convex polygons, each having at most $(\lceil \frac{k}{2} \rceil + 1)$ sides. At the same time, we attach two child nodes of ν to represent these smaller convex polygons. The recursion stops when r_ν is triangulated. Figure 2 gives an example of the recursive triangulation of r_ν. The recursive triangulation of the leaves in $L_{\mathcal{D}}$ produces a subtree rooted at ν of height $O(\log k) = O(\log(\text{depth}(\nu)))$. Let $\mathcal{D}'$ denote this intermediate linear decision tree obtained. Each leaf of $\mathcal{D}'$ represents a triangle t' that lies in r_ν for some $\nu \in L_{\mathcal{D}}$. The boundary of t' intersects $O(\log n)$ triangles in T_P. Therefore, for any query point that lies in t', we can determine which triangle $t \in T_P$ contains that query point in $O(\log \log n)$ time by applying binary search on the $O(\log n)$ triangles that intersect t'. This motivates us to expand $\mathcal{D}'$ further as follows. For every leaf ν' of $\mathcal{D}'$, replace ν' by a linear decision tree that corresponds to a binary search on the triangles in T_P that intersects the triangle corresponding to ν'. The resulting linear decision tree is $\mathcal{D}''$. The height of $\mathcal{D}''$ is $O(\log \log n)$ more than the height of $\mathcal{D}'$.

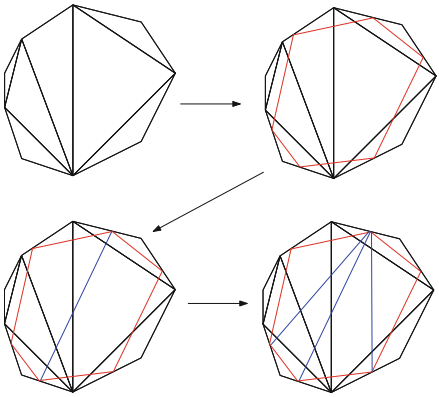

Fig. 2. The red lines represent the boundary of the convex k-gon of a leaf node of $\mathcal{D}$ and the blue lines represent the split of the convex k-gon (Color figure online).

If q is a query point inside r_ν for some $\nu \in L_\mathcal{D}$, then we can follow the path from the root of $\mathcal{D}$ to ν and then from ν to a leaf ν'' of $\mathcal{D}''$. The length of the path traversed is $\text{depth}(\nu'') \leq \text{depth}(\nu) + O(\log(\text{depth}(\nu))) + O(\log\log n)$. The entropy of T_P is an information-theoretic lower bound to answering point location queries in T_P. In particular, this lower bound applies to the linear decision tree $\mathcal{D}''$. Therefore,

$$\begin{aligned} H(T_P) &= \sum_{t \in T_P} f(t) \cdot \log \frac{m}{f(t)} \\ &\leq \sum_{\text{leaf } \nu'' \text{ of } \mathcal{D}''} f(r_{\nu''}) \cdot \text{depth}(\nu'') \\ &\leq \sum_{\nu \in L_\mathcal{D}} f(r_\nu) \cdot (\text{depth}(\nu) + O(\log(\text{depth}(\nu))) + O(\log\log n)) \end{aligned}$$

□

3 Point Location in a Convex Subdivision

Let S be an input convex subdivision. For each convex region P in S, we triangulate P hierarchically as described in Sect. 2. The collection of all triangles in all convex regions in S form a triangulation T of S. Clearly, $\sum_{P \in S} n_P = O(n)$, and therefore, T has $O(n)$ triangles and T can be constructed in $O(n)$ time. Next, we invoke the previous work of Iacono and Mulzer [19] for building an adaptive point location structure for planar triangulations. This gives us a point location data structure for T. We will prove that this point location data structure guarantees that any query sequence of length m can be answered in $O(\text{OPT} + m \log\log n + n)$ time, where OPT is the minimum time needed by any linear decision tree to process that query sequence.

The method of Iacono and Mulzer [19] is based on rebuilding from time to time. Initially, an optimal worst-case data structure W_0 is built on all triangles in T, and we start answering queries using W_0 until $\Theta(n^\alpha)$ queries have been answered for some $\alpha \in (0,1)$. Then we identify the n^β most frequently queried triangles for some $\beta \in (0,1)$ such that $\alpha \in (\beta, 1-\beta)$, triangulate their exterior, and then build a point location structure W_1 that is distribution-sensitive with respect to frequency counts in these n^β triangles [17]. These frequency counts are fixed when the rebuilding starts. The counts and this distribution-sensitive structure will not be updated as more queries are processed. Until the next rebuilding after another $\Theta(n^\alpha)$ queries, we first submit every query to W_1, and if W_1 does not report a triangle in the input triangulation, we resort to W_0 to answer the query. The challenge in [19] lies in proving that the total time to answer any query sequence of length m matches the entropy bound.

We prove below that by constructing Iacono and Mulzer's data structure on the triangulation T of S, we can obtain a query performance that is adaptive to the query sequence.

Theorem 1. *Let S be a convex subdivision of n vertices in $\mathbb{R}^2$. Our algorithm is a point-line comparison based algorithm that answers any point location query sequence of length m in $O(\mathrm{OPT} + m \log\log n + n)$ time, where OPT is the minimum time to process the same query sequence by any linear decision tree for answering point location queries in S. The preprocessing time is included in our running time bound.*

Proof. Let T be the triangulation of S obtained by triangulating every convex region in S as described in Sect. 2. We apply Theorem 2 in [19] to construct a point location structure on T. This total time spent by this structure on any query sequence of length m is

$$O\left(n + \sum_{t \in T} f(t) \cdot \log \frac{m}{f(t)}\right).$$

By manipulating the terms, we obtain

$$O\left(n + \sum_{t \in T} f(t) \cdot \log \frac{m}{f(t)}\right) = O\left(n + \sum_{P \in S} \sum_{t \in T_P} f(t) \cdot \log \frac{m}{f(t)}\right).$$

Then Lemma 1 implies that

$$\begin{aligned} &O\left(n + \sum_{t \in T} f(t) \cdot \log \frac{m}{f(t)}\right) \\ &= O\left(n + \sum_{P \in S} \sum_{\nu \in L_{\mathcal{D}}|_P} f(r_\nu) \cdot (\mathrm{depth}(\nu) + O(\log(\mathrm{depth}(\nu))) + O(\log\log n))\right), \end{aligned}$$

where $\mathcal{D}$ is an arbitrary linear decision tree for answering point location queries in S and we use $L_{\mathcal{D}}|_P$ to denote the subset of leaves of $\mathcal{D}$ that correspond to subset of points in P. Some explanation is in order why Lemma 1 is applicable. Clearly, a linear decision tree for answering point location queries in S is also a linear decision tree for answering point location queries in P, so Lemma 1 is applicable.

Since a leaf of $\mathcal{D}$ must correspond to a subset of points in at most one convex region P in S, the total running time for answering any query sequence of length m is

$$O\left(n + \sum_{\nu \in L_{\mathcal{D}}} f(r_\nu) \cdot (\text{depth}(\nu) + O(\log(\text{depth}(\nu))) + O(\log\log n))\right)$$
$$= O\left(\sum_{\nu \in L_{\mathcal{D}}} f(r_\nu) \cdot \text{depth}(\nu)\right) + O(m \log\log n + n).$$

The first term is $O(\text{OPT})$ because we can choose $\mathcal{D}$ to be the optimal linear decision tree. □

4 Conclusion

One can build another auxiliary planar point location structure so that a query can be executed on our adaptive structure and this auxiliary point location structures simultaneously until one of the two structures returns an answer. The advantage is that this auxiliary point location structure can offer additional properties. For example, if one uses the distance-sensitive planar point location structure [3], it means that queries far away from any region boundary can be answered fast too. Alternatively, if one uses the proximate planar point location structure [18] as the auxiliary structure, then a query can be answered faster if the query point is close to the previous one. Notice that the performance of these auxiliary structures are independent from the access frequencies. Therefore, such an auxiliary structure is constructed only once at the beginning, and it does not need to be rebuilt periodically as our point location structure.

Acknowledgment. We thank the anonymous referees for their helpful comments.

References

1. Afshani, P., Barbay, J., Chan, T.: Instance optimal geometric algorithms. In: Proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science, pp. 129–138 (2009)
2. Adamy, U., Seidel, R.: On the exact worst case query complexity of planar point location. In: Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 609–618 (1998)

3. Aronov, B., de Berg, M., Roeloffzen, M., Speckmann, B.: Distance-sensitive planar point location. In: Dehne, F., Solis-Oba, R., Sack, J.-R. (eds.) WADS 2013. LNCS, vol. 8037, pp. 49–60. Springer, Heidelberg (2013)
4. Arya, S., Cheng, S.W., Mount, D.M., Ramesh, H.: Efficient expected-case algorithms for planar point location. In: Proceedings of the 7th Scandinavian Workshop on Algorithm Theory, pp. 353–366 (2000)
5. Arya, S., Malamatos, T., Mount, D.M.: Nearly optimal expected-case planar point location. In: Proceedings of the 41st Annual Symposium on Foundations of Computer Science, pp. 208–218 (2000)
6. Arya, S., Malamatos, T., Mount, D.M.: A simple entropy-based algorithm for planar point location. ACM Trans. Algorithms **3**(2), article 17 (2007)
7. Arya, S., Malamatos, T., Mount, D., Wong, K.: Optimal expected-case planar point location. SIAM J. Comput. **37**(2), 584–610 (2007)
8. Boissonnat, J.D., Yvinec, M.: Algorithmic Geometry. Cambridge University Press, Cambridge (1998)
9. Bose, P., Devroye, L., Douïeb, K., Dujmovic, V., King, J., Morin, P.: Point location in disconnected planar subdivisions. arXiv:1001.2763v1 [cs.CG], 15 January 2010
10. Bose, P., Devroye, L., Douïeb, K., Dujmovic, V., King, J., Morin, P.: Odds-On Trees, arXiv:1002.1092v1 [cs.CG], 5 February 2010
11. de Berg, M., Cheong, O., van Kreveld, M., Overmars, M.: Computational Geometry: Algorithms and Applications. Springer, New York (2008)
12. Chan, T.M., Pătraşcu, M.: Transdichotomous results in computational geometry, I: point location in sublogarithmic time. SIAM J. Comput. **39**(2), 703–729 (2009)
13. Collette, S., Dujmović, V., Iacono, J., Langerman, S., Morin, P.: Entropy, triangulation, and point location in planar subdivisions. ACM Trans. Algorithms **8**(3), article 29 (2012)
14. Dobkin, D.P., Kirkpatrick, D.G.: Determining the separation of preprocessed polyhedra–a unified approach. In: Proceedings of the 17th International Colloquium on Automata, Languages and Programming, pp. 400–413 (1990)
15. Edahiro, M., Kokubo, I., Asano, T.: A new point-location algorithm and its practical efficiency–comparison with existing algorithms. ACM Trans. Graph. **3**(2), 86–109 (1984)
16. Edelsbrunner, H., Guibas, L.J., Stolfi, J.: Optimal point location in a monotone subdivision. SIAM J. Comput. **15**(2), 317–340 (1986)
17. Iacono, J.: Expected asymptotically optimal planar point location. Comput. Geom. Theory Appl. **29**(1), 19–22 (2004)
18. Iacono, J., Langerman, S.: Proximate planar point location. In: Proceedings of the 19th Annual Symposium on Computational Geometry, pp. 220–226 (2003)
19. Iacono, J., Mulzer, W.: A static optimality transformation with applications to planar point location. Int. J. Comput. Geom. Appl. **22**(4), 327–340 (2012)
20. Kirkpatrick, D.G.: Optimal search in planar subdivisions. SIAM J. Comput. **12**(1), 28–35 (1983)
21. Lee, D.T., Preparata, F.P.: Location of a point in a planar subdivision and its applications. SIAM J. Comput. **6**(3), 594–606 (1977)
22. Mulmuley, K.: A fast planar partition algorithm, I. J. Symbolic Comput. **10**(3–4), 253–280 (1990)
23. Preparata, F.P.: A new approach to planar point location. SIAM J. Comput. **10**(3), 473–483 (1981)
24. Sarnak, N., Tarjan, R.E.: Planar point location using persistent search trees. Commun. ACM **29**(7), 669–679 (1986)

25. Seidel, R.: A simple and fast incremental randomized algorithm for computing trapezoidal decompositions and for triangulating polygons. Comput. Geom. Theory Appl. **1**(1), 51–64 (1991)
26. Shannon, C.E.: A mathematical theory of communication. ACM SIGMOBILE Mob. Comput. Commun. Rev. **5**(1), 3–55 (2001)
27. Sleator, D.D., Tarjan, R.E.: Self-adjusting binary search trees. J. ACM **32**(3), 652–686 (1985)

http://www.springer.com/978-3-662-48970-3

Algorithms and Computation
26th International Symposium, ISAAC 2015, Nagoya, Japan, December 9-11, 2015, Proceedings
Elbassioni, K.; Makino, K. (Eds.)
2015, XXII, 793 p. 131 illus., Softcover
ISBN: 978-3-662-48970-3