Data Structures for Restricted Triangular Range Searching

Nadia M. Benbernou* Mashhood Ishaque† Diane L. Souvaine‡

Abstract

We present data structures for triangular emptiness and reporting queries for a planar point set, where the query triangle contains the origin. The data structures use near-linear space and achieve polylogarithmic query times.

1 Introduction

Simplex range searching (emptiness, reporting, counting) [1] is a fundamental problem in computational geometry. Given a set S of n points, a simplex emptiness query asks whether a given query simplex contains a non-empty subset of S. A simplex reporting query asks for a report of all points of S inside the query simplex, and a simplex counting query asks for the total number of such points.

In this paper we consider the restricted version of simplex (triangular) emptiness and reporting queries for points in a plane, where each query triangle contains the origin. Since we can triangulate any triangle containing origin into three triangles such that each triangle has one vertex incident on origin, we can assume wlog that each query triangle has one vertex at origin. The same idea works for any convex polygon containing the origin, but the number of query triangles is equal to the number of sides in the polygon.

1.1 Our Results

We present near-linear-space data structures for the following queries. All reporting queries incur an additional cost of $O(k)$ where k is the number of objects to be reported.

- Restricted triangular emptiness and reporting queries in $O(\lg^2 n)$ time. [Section 2]
- Restricted triangular emptiness and reporting queries in $O(2^{1/\epsilon} \lg n)$ time. [Section 3]
- Restricted triangular emptiness queries in $O(\lg n)$ time. [Section 4]
- Triangular emptiness and reporting queries in $O(\text{polylog } n)$ time with high probability, where the vertices of the query triangle are randomly chosen from the given point set. [Section 5]
- Ray intersection detection and reporting queries among an arrangement of n lines in $O(\text{polylog } n)$ time. [Section 6]
- Non-orthogonal square emptiness and reporting queries in $O(\text{polylog } n)$ time. [Section 7]

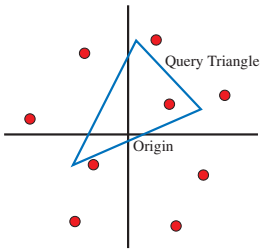

Figure 1: Restricted Triangular Range Queries

1.2 Related Results

Chazelle [5] showed that in the arithmetic model $\Omega(n^{1/2})$ time is needed to answer a triangular counting query using linear space. Similarly Brönnimann *et al.* [4] gave a lower bound of $\Omega(n^{1/3})$ for halfplane range counting in semigroup arithmetic model. For triangular reporting queries, Chazelle and Rosenberg [9] showed that, on a pointer machine, a query time of $O(n^{\delta}+k)$ requires $\Omega(n^{2(1-\delta)-\epsilon})$ space. Consequently polylogarithmic query time requires close to quadratic space. Although halfplane range counting is as hard as triangular range counting, halfplane reporting is significantly easier than triangular reporting. Chazelle *et al.* [6] gave a linear-space data structure for halfplane range reporting that achieves $O(\lg n+k)$ query time. The data structure maintains nested (peeling) convex layers for the given point set. Similarly for halfplane emptiness queries, a linear-space data structure that maintains convex hull of the given point set allows the queries to be answered in $O(\lg n)$ time.

The best known data structure for triangular emptiness (reporting) that uses $O(n \lg n)$ space, achieves a query time of $O(n^{1/2+\epsilon})$ (additional $O(k)$ for reporting). The data structure is based on Matoušek's technique of simplicial partitioning with low crossing number [16].

Using near quadratic space it is possible to support triangular range searching in polylogarithmic time.

*Massachusetts Institute of Technology, nbenbern@mit.edu. Partially supported by AFOSR grant FA9550-07-1-0538.

†Tufts University, mishaq01@cs.tuft.edu. Partially supported by NSF grant CCF-0431027.

‡Tufts University, dls@cs.tuft.edu. Partially supported by NSF grant CCF-0431027.

Chazelle *et al.* [10] gave a $O(n^{2+\epsilon})$ space data structure that supports reporting queries in $O(\lg n + k)$ time. Goswami *et al.* [15] presented a $O(n^2)$ space data structure that can support triangular reporting queries in $O(\lg^2 n + k)$ and triangular counting (and hence emptiness) queries in $O(\lg n)$ time.

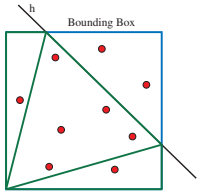

Figure 2: Halfplane Queries to Triangular Queries

The best lower bounds known for the triangular emptiness queries comes from the lower bounds for halfplane emptiness queries. There is a simple reduction from halfplane range queries to triangular range queries (see Figure 2). Erickson [14] showed that for any data structure supporting halfplane emptiness queries will have a lower bound of $\Omega(\lg n)$ for query, $\Omega(n)$ for space, and $\Omega(n \lg n)$ for preprocessing. While the bounds for halfplane emptiness are tight, there is no matching upper bound for triangular emptiness queries.

2 Simple Data Structures with $O(\lg^2 n)$ Query

Given a set of n points in the plane, sort the points rotationally in counter-clockwise order around the origin. Assign as ID to each point its order in the sorted list of points. Now consider any single wedge formed by two rays emanating from the origin. Let i be the first and j be the last point inside the wedge that would be hit if we were to sweep rotationally around origin using a ray that goes from the right boundary of the wedge to the left boundary. Observe that all the points inside the wedge are consecutive (with wrap-around) in the sorted order, see Figure 3. For any given wedge, we can find the points i and j in $O(\lg n)$ time. For any triangular emptiness/reporting query Δ_{abc} with a vertex, b, coincident with the origin, we can extend the two sides as rays $\overrightarrow{ba}$ and $\overrightarrow{bc}$ away from the origin to form a wedge. Now if we had a halfplane emptiness/reporting data structure over the points in this wedge, we could answer triangular emptiness and reporting queries by querying the halfplane bounded by $\overleftrightarrow{ac}$.

A naïve data structure for restricted triangular emptiness queries would be to store a convex hull for each pair of indicies (i, j), supporting queries in $O(\lg n)$ time, and using $O(n^3)$ space. The preprocessing time would be $O(n^3)$ because we can compute the convex hull in linear time for points in sorted order. For triangular reporting we would store nested convex layers instead of convex hulls. The query time would be $O(\lg n + k)$, space would be $O(n^3)$, and the preprocessing would be $O(n^3 \lg n)$.

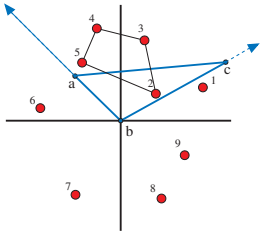

Figure 3: Points Sorted Around the Origin

Since the halfplane emptiness and reporting queries are decomposable, we could build a simpler data structure. Given the set of points in sorted order, build the dynamic convex hull structure of Overmars and van Leeuwen [18]. The data structure needs $O(n)$ space, $O(n \lg n)$ preprocessing and the query takes $O(\lg^2 n)$. To answer an emptiness query, the leaves in the dynamic convex hull tree corresponding to two extreme points inside the wedge are located. Then by climbing up the tree from the two leaves until the least common ancestor is reached, up to $O(\lg n)$ convex hull structures are collected. Together these convex hull cover all the points between and including the two extreme points. The query is answered by querying each of these $O(\lg n)$ structures for halfplane emptiness. For restricted triangular reporting, a convex peeling layers structure is stored at each node. The data structure needs $O(n \lg n)$ space , $O(n \lg^2 n)$ preprocessing and support queries in $O(\lg^2 n + k)$ time. Daescu *et al.* [11] used a similar idea to build a data structure for halfplane farthest-point queries.

3 Data Structures with $O(2^{1/\epsilon} \lg n)$ Query

In this section we present an near-linear-space data structure for restricted triangular emptiness/reporting queries that achieves logarithmic query time. We start with the naïve data structures from previous section that use $O(n^3)$ space and achieves $O(\lg n)$ query time. Then we recursively apply the space-reducing transformation from Aronov *et al.* [2] to provide a $O(n^{1+\epsilon})$ space data structure. The space-reducing transformations preserve the $O(\lg n)$ query time, but the constant is exponential in $1/\epsilon$.

Here is how the transformation works. Given the set of n points sorted around the origin, select every mth point to be a breakpoint. For each breakpoint m_i, compute the convex hull (convex layers for reporting) for each sequence of points starting at m_i whose length is a power of two. This constitutes linear space for each of the breakpoints. In addition, we compute this data structure recursively for each half-open interval $[m_i, m_{i+1})$ formed by the breakpoints m_i and m_{i+1}. Let $M(n)$ be the size of the data structure on n points.

The recurrence relation (same as in [2]) for the space of the data structure after one space-reducing transformation:

$$M(n) = (n/m + 1)M(m) + O(n^2/m)$$

Applying $(1/2 + 1/\epsilon)$ transformations for any given $\epsilon > 0$, as in [2], yields the desired space and preprocessing of $O(n^{1+\epsilon})$. Although the published proof for the recurrence relation is incorrect, there is an easy fix (omitted) suggested by Erik Demaine [12]–an author of that paper.

To answer a triangular emptiness (reporting) query, identify in $O(\lg n)$ time the extreme points i and j inside the wedge; let m_i and m_j be the breakpoints inside the wedge, closest to points i and j respectively; open intervals (i, m_i) and (m_j, j) do not contain any breakpoint, thus we must have a recursive data structure for them; two convex hull (convex layers) structures will cover all the points in the interval $[m_i, m_j]$. For a reporting query we may report some points twice. The recurrence relation for query time is given as follows, with a solution of $O(2^{1/\epsilon} \lg n)$ (as in [2]):

$$Q(n) = 2Q(m) + O(\lg n)$$

4 Emptiness Queries in $O(\lg n)$ Time

We apply the fractional cascading technique [7] to the $O(\lg^2 n)$ time emptiness query data structure given in Section 2. The modified data structure supports emptiness queries in $O(\lg n)$ time, but the space becomes $O(n \lg n)$.

The basic idea is to reduce a halfplane emptiness query to the problem of finding the extreme point in a given direction. A query halfplane is empty if and only if the extreme (farthest) point in the direction perpendicular to the query line (defining the halfplane) is not contained in the halfplane. With each extreme point we can associate a half-open interval of slopes. We can store a sorted array of these intervals and for a given slope find the extreme point in $O(\lg n)$ time using binary search. We store one such sorted array (corresponding to extreme points in a node's subtree) at each node in the data structure. Notice to answer a restricted triangular emptiness query we still need to perform $O(\lg n)$ extreme point queries, but we can apply fractional cascading here. So only the first extreme point query takes $O(\lg n)$ time, and the queries after that can be answered in constant time.

Achieving $O(\lg n + k)$ time for reporting queries remains a key open problem, as Chazelle and Liu [8] showed that the fractional cascading technique does not generalize to planar maps.

5 A Probabilistic Data Structure

For a given set of n points, there are $\binom{n}{3}$ triangles with vertices in the point set. For a query triangle chosen randomly out of these $\binom{n}{3}$ triangles, there exist near-linear-space data structures that support triangular emptiness and reporting queries in $O(\lg n)$ time with high probability. Even in case of failure the data structures do not answer queries incorrectly; instead they identify in $O(\lg n)$ time whether the given query can be answered. The key idea is to use the result by Aronov *et al.* [3] (also see [13, 17]) which says that for a subset of these $\binom{n}{3}$ triangles of size at least m $(\geq n^2)$, there exists a point that is inside of at least $\Omega(\frac{m^3}{n^6 \lg^5 n})$ triangles. Although it might seem that the same result could be achieved using ϵ-nets, the obvious strategies for doing so fail.

Starting from the set of all $\binom{n}{3}$ triangles, we find the deepest point (there is a $O(n^6)$ naïve algorithm). Using this point as origin we build restricted triangular query data structures over the given point set. We then remove the triangles containing this deepest point, from the set of triangles. We repeat the method on the remaining subset until we have created $O(\lg n)$ data structures.

For a triangular query we can find in $O(\lg n)$ time if there is some data structure whose origin is contained in the query triangle. If there exists such a data structure, we use it to answer the triangular emptiness/reporting query; otherwise we indicate failure.

6 Ray Intersection Detection and Reporting

In the data structures for restricted triangular range queries if the points are sorted by their x-coordinate instead of radially sorted, the data structures can support axis-parallel three-sided trapezoidal queries. Using two such trapezoidal queries, we can answer double-wedge emptiness and reporting queries where one of the lines forming the wedge is vertical (see Figure 4).

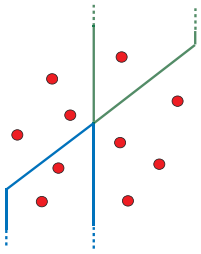

Figure 4: Trapezoidal Range Queries

Since such a double-wedge corresponds to a ray in the dual world, the data structures can be used to answer ray intersection detection and reporting queries among an arrangement of n lines. The data structure for detecting intersection uses $O(n \lg n)$ space and support queries in $O(\lg n)$ time. For intersection reporting the query time is $O(\lg^2 n + k)$ using $O(n \lg n)$ space, or $O(\lg n + k)$ using $O(n^{1+\epsilon})$ space.

7 Non-Orthogonal Square Range Searching

By building a range tree of three-sided trapezoidal query data structures, we could support right-triangular

queries, where the base of the triangle is axis-parallel. The space for the data structure increases and the query time slows down by a factor of $O(\lg n)$.

Since a non-orthogonal square can be partitioned into eight axis-parallel right triangles, the data structures for axis-parallel right triangles also support non-orthogonal square (or rectangles with constant aspect ratio–fat rectangles) emptiness and reporting queries .

A brief sketch of the proof: from the highest vertex of the given square draw a vertical line segment down to one of the non-adjacent sides. Similarly from the lowest vertex draw a vertical line segment upwards. Let x be the side-length then each vertical segment has a length in the interval $[x, \sqrt{2}x]$. The two diagonals of the square intersect at a distance $\frac{1}{\sqrt{2}}x$ from each vertex. Therefore, the downward vertical segment goes below and the upward vertical segments goes above this point of intersection. Thus we can draw horizontal segment from the lower (upper) endpoint of the downward (upward) vertical segment to the other vertical segment. Notice the argument does not hold for long skinny non-orthogonal rectangles.

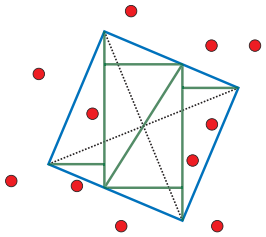

Figure 5: Non-Orthogonal Square Range Queries

8 Concluding Remarks

It is impossible to achieve $O(\text{polylog } n)$ time for triangular counting queries using $O(n \text{ polylog } n)$ space, even when the query triangle contains the origin; since a halfplane counting query can be reduced in constant time to a restricted triangular counting query. Thus the lower bound for halfplane range counting queries [4] applies to restricted triangular range counting as well.

The fact that the data structures for restricted triangular queries support both emptiness and reporting queries in $O(\text{polylog } n)$ time indicates that the techniques for restricted case will not extend to general triangular emptiness. The lower bound on triangular reporting queries [9] on a pointer machine, indicates that any near-linear-space data structure achieving $O(\text{polylog } n)$ time for triangular emptiness must handle emptiness queries differently from reporting queries.

Acknowledgment

This paper has benefited from the suggestions of anonymous referees.

References

[1] P. Agarwal and J. Erickson. Geometric range searching and its relatives. In B. Chazelle, J.E. Goodman, and R. Pollack, editors, *Adv. in Disc. and Comp. Geometry*, volume 23, pages 1–56. AMS Press, Providence, RI, USA, 1999.

[2] Boris Aronov, Prosenjit Bose, Erik D. Demaine, Joachim Gudmundsson, John Iacono, Stefan Langerman, and Michiel H. M. Smid. Data structures for halfplane proximity queries and incremental voronoi diagrams. In *LATIN*, pages 80–92, 2006.

[3] Boris Aronov, Bernard Chazelle, Herbert Edelsbrunner, Leonidas J. Guibas, Micha Sharir, and Rephael Wenger. Points and triangles in the plane and halving planes in space. *Discrete Comput. Geom.*, 6(5):435–442, 1991.

[4] Hervé Brönnimann, Bernard Chazelle, and János Pach. How hard is half-space range searching? *Discrete & Computational Geometry*, 10:143–155, 1993.

[5] Bernard Chazelle. Lower bounds on the complexity of polytope range searching. *JAMS*, 2:637–666, 1989.

[6] Bernard Chazelle, Leo J. Guibas, and D. T. Lee. The power of geometric duality. *BIT*, 25(1):76–90, 1985.

[7] Bernard Chazelle and Leonidas J. Guibas. Fractional cascading: I. a data structuring technique. *Algorithmica*, 1(2):133–162, 1986.

[8] Bernard Chazelle and Ding Liu. Lower bounds for intersection searching and fractional cascading in higher dimension. *J. Comput. Syst. Sci.*, 68(2):269–284, 2004.

[9] Bernard Chazelle and Burton Rosenberg. Simplex range reporting on a pointer machine. *Comput. Geom. Theory Appl.*, 5(5):237–247, 1996.

[10] Bernard Chazelle, Micha Sharir, and Emo Welzl. Quasi-optimal upper bounds for simplex range searching and new zone theorems. *Algorithmica*, 8:407–429, 1992.

[11] Ovidiu Daescu, Ningfang Mi, Chan-Su Shin, and Alexander Wolff. Farthest-point queries with geometric and combinatorial constraints. *Comput. Geom. Theory Appl.*, 33(3):174–185, 2006.

[12] Erik D. Demaine. Personal communication, 2007.

[13] David Eppstein. Improved bounds for intersecting triangles and halving planes. *J. Comb. Theory Ser. A*, 62(1):176–182, 1993.

[14] Jeff Erickson. Space-time tradeoffs for emptiness queries. *SIAM J. Comput.*, 29(6):1968–1996, 2000.

[15] Partha P. Goswami, Sandip Das, and Subhas C. Nandy. Simplex range searching and k nearest neighbors of a line segment in 2d. In *SWAT*, pages 69–79, London, UK, 2002. Springer-Verlag.

[16] Jirí Matousek. Geometric range searching. *ACM Comput. Surv.*, 26(4):421–461, 1994.

[17] Gabriel Nivasch and Micha Sharir. Eppstein's bound on intersecting triangles revisited. *CoRR*, abs/0804.4415, 2008. informal publication.

[18] M. H. Overmars and J. van Leeuwen. Maintenance of configurations in the plane. *J. Comput. Syst. Sci.*, 23:166–204, 1981.