I/O-efficient triangular range search and its application

Gautam K. Das[*] Bradford G. Nickerson[†]

Abstract

In this paper, we consider a special type of triangular range search for a planar point set S in the I/O-model. Here, two sides of the query triangle are parallel to the coordinate axes (x- and y-axis). We call such a triangle an *axis parallel triangle*. For the axis parallel triangular range search problem, we propose a data structure of size $O(\frac{N}{B}(\frac{\log_B^{1+\epsilon} N}{\log_B \log_B N})^2)$ disk blocks and a query algorithm of complexity $O(\log_B N(\frac{\log_B N}{\log_B \log_B N})^2 + K/B)$ I/Os, where $0 < \epsilon < 1$, N is the number of points in S, B is the number of points transferred in one I/O, and K is the number of points in the answer to the query. We also consider the application of this data structure and query algorithm to oriented rectangular range search.

1 Introduction

Transfer of data between internal and external memory is a major performance bottleneck for storage and retrieval of the extremely large datasets that are now commonplace. In some applications, geometric data such as points, lines and polygons are the primary elements stored in these massive datasets. Such applications often perform queries like searching for objects satisfying certain spatial constraints, including reporting the objects intersecting a search region. In this paper, given a set S of N points in the plane, we design the data structures and query algorithms for (i) open triangular range search (Section 2), (ii) axis parallel triangular range search (Section 3), and (iii) arbitrarily oriented rectangular range search (Section 4). To the best of our knowledge, the best previous result for arbitrarily oriented triangular and rectangular range search requires near quadratic space in the pointer machine model and there are no I/O-model results for these problems.

To achieve a query time $O(N^\delta + K)$ requires $\Omega(N^{2(1-\delta)-\epsilon})$ space on a pointer machine for the triangular range reporting problem where $\delta, \epsilon > 0$ and K is the number of points in the query region [8]. In [7], Chazelle et al. proved that in the arithmetic model $\Omega(N^{1/2})$ time is required to answer a triangular range counting query using linear space. The first work addressing the triangular range counting query appeared in [10], where two near quadratic data structures are presented for points in the plane with complexity (i) query time $O(\log N)$ using space $O(N^{2+\epsilon})$ and (ii) query time $O(\log N \log\log N)$ using space $O(\frac{N^2}{\log N})$. In the same paper (for 3D points), the authors presented a simplex range counting algorithm using $O(\log N)$ time and $O(N^{7+\epsilon})$ space. For points in the plane, a near-optimal upper bound for the triangular range counting problem is available in [9]. The space complexity is $O(N^{2+\epsilon})$ whereas the query complexity for counting and reporting are $O(\log N)$ and $O(\log N + K)$, respectively. In [13], Matousek presented a data structure of size $O(N^2)$ which answers counting and reporting queries in $O(\log^3 N)$ and $O(\log^3 N + K)$ time, respectively for points in the plane. In [11], Goswami et al. presented a $O(N^2)$ space data structure that can support triangular reporting queries in $O(\log^2 N + K)$ time and triangular counting queries in $O(\log N)$ time. If the query triangle contains the origin, then near linear space data structures are available for emptiness and reporting queries in $O(\log N)$ and $O(\log^2 N)$ time, respectively [12]. The authors in [12] also proved that $O(2^{1/\epsilon} \log N)$ query time is possible for the same problems and for the axis parallel triangular range search $O(\log^3 N + K)$ time query using $O(N \log^2 N)$ space. Ishaque et al. [12] proposed a near linear space data structure for (i) triangular emptiness and reporting queries in $O(polylogN)$ time with high probability (where the vertices of the query triangle are randomly chosen from the given point set) and (ii) non-orthogonal square emptiness and reporting queries in $O(polylogN)$ time. For detailed surveys on geometric range search see [6, 14, 15].

2 Open triangular range search

Given a set S of N points in the plane, the open triangular range search problem is to design a data structure supporting range queries between two infinite rays (ρ_1 and ρ_2) emanating from a source point p such that one of the rays (say, ρ_2) is horizontal, pointing to the right and the ray ρ_1 is above ray ρ_2. Figure 1(a) demonstrates the problem where the points in the shaded region fall within the query region. Without loss of generality, we assume that all the points in S are in the first quadrant of the coordinate framework.

The basic idea behind our data structure for open triangular range search is as follows: given two rays, one of

[*] University of New Brunswick, Fredericton, Canada

[†] University of New Brunswick, Fredericton, Canada

which is parallel to the x-axis, we first find the appropriate points in the query region based on y-coordinates, and then we apply an algorithm for 2-dimensional halfspace range search on these 2-dimensional points.

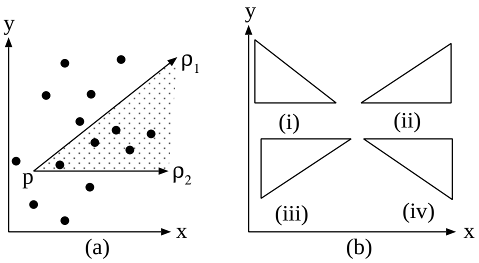

Figure 1: Demonstration of (a) the open triangular range search problem and (b) different types of axis parallel triangles.

Data Structure: The basic building block of the data structure is the weight-balanced B-tree [3]. Let the B-tree be $\mathcal{T}$ with branching parameter $a = O(\log_B^{\epsilon} N)$ and leaf parameter $k = B$ based on the y-coordinate of the points in S, for an appropriately chosen $\epsilon \in (0, 1)$. All leaf nodes of $\mathcal{T}$ are on the same level, and thus the height of the tree $\mathcal{T}$ is $O(\frac{\log_B N}{\log_B \log_B N})$ and it uses $O(N/B)$ space. $\mathcal{T}$ can be constructed in $O(\frac{N}{B} \log_{\frac{M}{B}} \frac{N}{B})$ I/Os by first sorting the points based on their y-coordinates and then constructing the tree level-by-level in a bottom-up fashion, where M is the maximum number of points that can be stored in main memory. In each internal node v of $\mathcal{T}$ we attach a secondary structure.

The secondary structure is the data structure for answering two-dimensional halfspace range queries [4]. Consider a node v_i and its siblings, ordered from left to right they form the sequence $v_1, v_2, \ldots, v_{i-1}, v_i, v_{i+1}, \ldots, v_a$. The elements in the secondary structure at node $v = v_i$ are the points in the subtrees with roots $v_i, v_{i+1}, \ldots, v_a$. The secondary structure at the root node contains all the points in S.

Query Algorithm: Let ρ_1 and ρ_2 be the two rays emanating from (α, β) of the query Q (open triangle) where ρ_2 is the horizontal ray.

In the root node v of $\mathcal{T}$, if $\beta \leq p.y$ for all the points in S where $p.y$ is the y-component of the point p, then report all the points in the secondary structure associated with v that satisfy the halfspace query. Otherwise, visit the path from the root in a downward direction to find the topmost node u_i ($\neq v$) in the base tree $\mathcal{T}$ such that (i) the β lies in the y-range of u_i and (ii) $i < a$. Note that a is the branching parameter of $\mathcal{T}$. The search Q can be decomposed in two parts: (1) in the node u_i and (2) all the nodes $u_{i+1}, u_{i+2}, \ldots, u_a$; i.e. node u_i and the secondary structure stored at node u_{i+1}. At the node u_{i+1}, we perform a 2D halfspace range search in the secondary structure stored at node u_{i+1} using $O(\log_B N + K/B)$ I/Os. To search for points satisfying the query in node u_i, we follow the path in base tree $\mathcal{T}$ from u_i to leaf node. At each level of the path we apply a 2D halfspace range search appropriately.

The complexity of the data structure follows from the following theorem.

Theorem 1 *A set S of N points in the plane can be stored in a data structure of size $O(\frac{N}{B} \frac{\log_B^{1+\epsilon} N}{\log_B \log_B N})$ disk blocks that supports open triangular range search in $O(\frac{\log_B^2 N}{\log_B \log_B N} + K/B)$ I/Os.*

Due to space constraints, all proofs are omitted from this paper.

3 Axis parallel triangular search

In axis parallel triangle search, one side of the triangle is parallel to the x-axis, one side is parallel to the y-axis and the other side is in arbitrary orientation. Four types of axis parallel triangles are possible based on their orientation as follows:

Type-1: The corner with internal angle $\pi/2$ is the left-bottom corner (see Figure 1(b)(i)).

Type-2: The corner with internal angle $\pi/2$ is the right-bottom corner (see Figure 1(b)(ii)).

Type-3: The corner with internal angle $\pi/2$ is the left-top corner (see Figure 1(b)(iii)).

Type-4: The corner with internal angle $\pi/2$ is the right-top corner (see Figure 1(b)(iv)).

Here, we describe the data structure and query algorithm for the query triangle of Type-1. Similarly, we can design the data structure and query algorithm for the three other triangle types.

Data Structure: Like the data structure for open triangular range search proposed in Section 2, here also the data structure is based on the weight-balanced B-tree. Let the B-tree be $\mathcal{T}$ with branching parameter $a = O(\log_B^{\epsilon} N)$ and leaf parameter $k = B$ based on the x-coordinates of the points in S, for an appropriately chosen $\epsilon \in (0, 1)$. Here, all the leaf nodes of $\mathcal{T}$ are on the same level and thus the height of the tree $\mathcal{T}$ is $O(\frac{\log_B N}{\log_B \log_B N})$ and it uses $O(N/B)$ disk blocks. In each internal node v of $\mathcal{T}$ we attach two structures namely *structure-1* and *structure-2*.

Structure-1 is the optimal data structure for answering two-dimensional 3-sided range queries [2]. We maintain a set U of points in the plane such that given a 3-sided query $Q = (\alpha, \beta, \gamma)$, we report all the points $(x, y) \in U$ with $\alpha \leq x$ and $\beta \leq y \leq \gamma$. The elements of the node v are the set of points that are in the subtree rooted at v. Structure-2 is the data structure for answering open triangular range queries (see Section 2).

The elements included in structure-2 corresponding to node v are as follows:

Consider a node v_i and its siblings, ordered from left to right they form the sequence $v_1, v_2, \ldots, v_{i-1}, v_i, v_{i+1}, \ldots, v_a$. The elements of structure-2 at node $v = v_i$ are the points in the subtrees with roots $v_i, v_{i+1}, \ldots, v_a$. Structure-2 at the root node contains all the points in S.

Query Algorithm: Let ΔABC be the axis parallel query triangle where $A = (\alpha, \beta)$, $B = (\gamma, \beta)$ and $C = (\alpha, \delta)$.

In the root node v of $\mathcal{T}$, if $\alpha \leq p.x$ for all the points in S where $p.x$ is the x-component of the point p, then report all the points in the structure-2 associated with v that satisfy the open triangular range search. Otherwise visit the path from the root in a downward direction to find the topmost node u_i ($\neq v$) in the base tree $\mathcal{T}$ such that (i) the α lies in the x-range of u_i and (ii) $i < a$, where a is the branching parameter of $\mathcal{T}$. Let the right boundary of the node u_i be at $x = \mu$. Also assume that the line $x = \mu$ intersects the non-axis parallel side of ΔABC at $y = \nu$.

The search ΔABC can be decomposed into three parts; namely (i) a 3-sided query $Q = (\alpha, \beta, \nu)$ in the node u_i (dotted region in Figure 2(a)), (ii) an axis parallel triangular range search ΔDEC ($D = (\alpha, \nu)$ and $E = (\mu, \nu)$) at node u_i (dark shaded region in Figure 2(a)), and (iii) all the nodes from u_{i+1} to u_a i.e., the structure-2 stored at node u_{i+1} (light shaded region in Figure 2(a)). At node u_{i+1}, we can perform a 2D range query between two infinite rays in the structure-2 stored at node u_{i+1}. The search at this node takes $O(\log_B^2 N + K/B)$ I/O operations (see Section 2). For the 3-sided query $Q = (\alpha, \beta, \nu)$ in node u_i, we can use structure-1. For the axis parallel triangular range search ΔDEC in node u_i, we follow the path in the base tree $\mathcal{T}$ from node u_i to the appropriate leaf node. At each level of the path, we apply a 3-sided range query and an open triangular range query.

The complexity of the data structure follows from the following theorem.

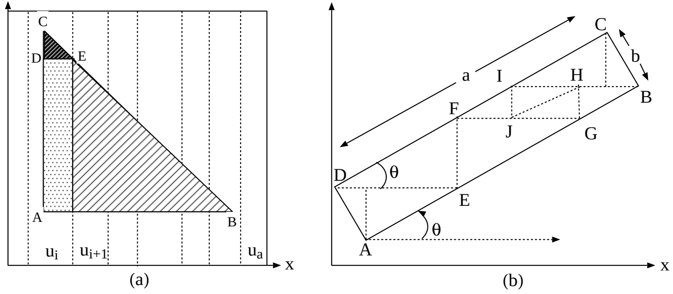

Figure 2: (a) Demonstration of axis parallel triangular range search and (b) partition of an arbitrarily orientated rectangle.

Theorem 2 *A set S of N points in the plane can be stored in a data structure of size $O(\frac{N}{B}(\frac{\log_B^{1+\epsilon} N}{\log_B \log_B N})^2)$ disk blocks supporting axis parallel triangular range query using $O(\log_B N(\frac{\log_B N}{\log_B \log_B N})^2 + K/B)$ I/Os.*

The lower bound for axis parallel triangular range search comes from the optimal data structure for axis aligned rectangular range search. In [1], Arge showed that the size of the optimal data structure for axis aligned rectangular range search is $O(\frac{N}{B}\frac{\log_B N}{\log_B \log_B N})$ and requires $O(\log_B N + K/B)$ query I/Os.

4 Application of axis parallel triangular range search

The data structure and query algorithm for axis parallel triangular range search can be used for arbitrarily oriented rectangular range search. The space for the data structure remains the same as the data structure for axis parallel triangular range search, but the query complexity slows down by a factor that depends on the "skinniness" of the rectangle. The basic idea behind the arbitrarily oriented rectangular range search is to partition the rectangle into $O(m)$ (say) axis parallel triangles and for each triangle to use the data structure and query algorithm for axis parallel triangular range search. In [12], the authors show that an arbitrarily oriented square can be partitioned into eight axis parallel triangles. For an arbitrarily oriented square range search, the size of the data structure and the query I/Os remains the same as for axis parallel triangular range search. To partition an arbitrarily oriented rectangle into axis parallel triangles, we consider a rectangle $ABCD$ where lengths of the sides are a and b with $a \geq b$, and the side AB of the rectangle forms an angle θ with the positive direction of the x-axis. We call this rectangle $\mathcal{R}_\theta(a, b)$. Now, we provide a sketch of the partitioning procedure for $\mathcal{R}_\theta(a, b)$:

Step 1: Draw a horizontal line from the right-most vertex (B in Figure 2(b)) of the rectangle $\mathcal{R}_\theta(a, b)$ to the non-adjacent side (edge BI in Figure 2(b)).
Step 2: Draw a horizontal line from the left-most vertex (D in Figure 2(b)) of the rectangle $\mathcal{R}_\theta(a, b)$ of the non-adjacent side (edge DE in Figure 2(b)).
Step 3: Draw a vertical line from E to the side opposite to the side containing E (EF in Figure 2(b)).
Step 4: Repeat Steps 2 and 3 by considering the point F as the top-left vertex of the rectangle $\mathcal{R}_\theta(a, b)$ until the vertical line in Step 3 intersects the edge BI. Let the intersection point be H.
Step 5: Draw vertical lines from vertex A and C to the edges DE and BI, respectively.
Step 6: Draw a vertical line from I to a horizontal line generated in Step 2. Let the intersection point be J (edge IJ in Figure 2(b)).
Step 7: Join the points H and J.

Lemma 3 *If $n(\mathcal{R}_\theta(a,b))$ is the number of axis parallel triangles in the above partition algorithm of $\mathcal{R}_\theta(a,b)$, then $n(\mathcal{R}_{\frac{\pi}{4}}(a,b)) \geq n(\mathcal{R}_\theta(a,b))$ for $0 \leq \theta \leq \pi/4$.*

Similarly, we can say that if the value of θ varies from $\pi/4$ to $\pi/2$ then also $n(\mathcal{R}_{\frac{\pi}{4}}(a,b)) \geq n(\mathcal{R}_\theta(a,b))$ by a slight modification of the partitioning algorithm. Thus, we have the following lemma:

Lemma 4 *For fixed values of a and b, $n(\mathcal{R}_{\frac{\pi}{4}}(a,b)) \geq n(\mathcal{R}_\theta(a,b))$ for $0 \leq \theta \leq \pi/2$.*

From now on, we consider the value of θ equal to $\pi/4$ for worst case complexity analysis of arbitrarily oriented rectangular range search. We also use the notation $n(\mathcal{R}(a,b))$ for $n(\mathcal{R}_{\frac{\pi}{4}}(a,b))$. The estimate of $n(\mathcal{R}(a,b))$ in terms of a and b follows from the following lemma:

Lemma 5 $n(\mathcal{R}(a,b)) = \lfloor \frac{a}{b} \rfloor + 7$.

Theorem 6 *A set S of N points in the plane can be stored in a data structure of size $O(\frac{N}{B}(\frac{\log_B^{1+\epsilon} N}{\log_B \log_B N})^2)$ disk blocks such that arbitrarily oriented rectangular range search can be performed in $O(\frac{a}{b} \log_B N(\frac{\log_B N}{\log_B \log_B N})^2 + K/B)$ I/Os in the worst case, where a and b $(\leq a)$ are the side lengths of the query rectangle.*

Corollary 7 *A set S of N points in the plane can be stored in a data structure of size $O(\frac{N}{B}(\frac{\log_B^{1+\epsilon} N}{\log_B \log_B N})^2)$ disk blocks such that sufficiently fat rectangular range search can be performed in $O(\log_B N(\frac{\log_B N}{\log_B \log_B N})^2 + K/B)$ I/Os.*

Proof. For sufficiently fat rectangles, $\frac{a}{b}$ can be considered a constant. Thus, the Corollary follows from Theorem 6. □

5 Conclusion

In this paper, we have proposed a data structure and query algorithm for the open triangular range search and the axis parallel triangular range search problems on 2-dimensional point sets. The size of the data structures are $O(\frac{N}{B}\frac{\log_B^{1+\epsilon} N}{\log_B \log_B N})$ and $O(\frac{N}{B}(\frac{\log_B^{1+\epsilon} N}{\log_B \log_B N})^2)$ disk blocks, respectively and the query complexities are $O(\frac{\log_B^2 N}{\log_B \log_B N} + K/B)$ and $O(\log_B N(\frac{\log_B N}{\log_B \log_B N})^2 + K/B)$ I/Os, respectively. We have also proposed a data structure and query algorithm for arbitrarily oriented rectangular range search using the result of axis parallel triangular range search, and showed that for sufficiently fat query rectangles, the size of the data structure and query I/Os remains the same as that of axis parallel triangular range search. The axis parallel triangular range search can also be adapted to other geometric queries (e.g. points inside regular polygons, points between parallel lines). It can be extended to 3-dimensional points using the result of 3-dimensional halfspace range query proposed in [5].

The lower bound of the axis parallel triangular range search is $\Omega(\frac{N}{B}\frac{\log_B N}{\log_B \log_B N})$ disk blocks for space and $\Omega(\log_B N + K/B)$ I/Os for query. An open problem is to reduce the gap between the lower bound and our proposed complexity for the same problem.

References

[1] L. Arge External memory data structures. *Handbook of Massive Data Sets*, J. Abello, P. M. Pardalos, M. G. C. Resende (eds.), pp. 313-358, Kluwer Academic Publishers, Dordrecht, 2002.

[2] L. Arge, V. Samoladas and J. S. Vitter On two-dimensional indexability and optimal range search indexing. *Symp. on Principles of Database Systems*, pp. 346-357, 1999.

[3] L. Arge and J.S. Vitter Optimal external memory interval management. *SIAM J. on Computing*, 32:1488-1508,2003.

[4] P. K. Agarwal, L. Arge, J. Erickson, P. Franciosa and J. S. Vitter, Effient searching with linear constraints, *J. of Computer and System Sc.*, 61:194-216, 2000.

[5] P. Afshani and T. M. Chan Optimal halfspace range reporting in three dimensions. *Symp. on Discrete Algorithms*, 180-186, 2009.

[6] P. K. Agarwal and J. Erickson Geometric range searching and its relatives. In: B. Chazelle, J. E. Goodman and R. Pollack, Editors, Advances in Discrete and Computational Geometry, *Contemporary Mathematics*, 223:158, 1999.

[7] B. Chazelle Lower bounds on the complexity of polytope range searching. *J. of the American Mathematical Society*, 2:637-666, 1989.

[8] B. Chazelle and B. Rosenberg Simplex range reporting on a pointer machine. *Computational Geometry: Theory and Applications*, 5:237-247, 1996.

[9] B. Chazelle, M. Sharir and E. Welzl Quasi-optimal upper bounds for simplex range searching and new zone theorems. *Algorithmica*, 8:407-429, 1992.

[10] R. Cole and C. K. Yap Geometric retrieval problems. *Information and Control*, 63:39-57, 1984.

[11] P. P. Goswami, S. Das and S. C. Nandy Triangular range counting query in 2D and its application in finding k nearest neighbors of a line segment. *Computational Geometry: Theory and Applications*, 29:163-175, 2004.

[12] M. Ishaque, D. L. Souvaine and N. Benbernou Data structures for restricted triangular range searching. *Canadian Conference on Comp. Geom.*, 2008.

[13] J. Matousek Range searching with efficient hierarchical cutting. *Discrete Com. Geom.*, 10:157-182, 1993.

[14] J. Matousek Geometric range searching. *ACM Computing Serveys*, 26:421-461, 1994.

[15] Y. Nunez Non-orthogonal range searching. *Technical Report*, 2007.