Data Structures for Range-Aggregate Extent Queries

Prosenjit Gupta[*] Ravi Janardan[†] Yokesh Kumar[†] Michiel Smid[‡]

July 22, 2008

Abstract

A fundamental and well-studied problem in computational geometry is *range searching*, where the goal is to preprocess a set, S, of geometric objects (e.g., points in the plane) so that the subset $S' \subseteq S$ that is contained in a query range (e.g., an axes-parallel rectangle) can be reported efficiently. However, in many situations, what is of interest is to generate a more informative "summary" of the output, obtained by applying a suitable *aggregation function* on S'. Examples of such aggregation functions include *count*, *sum*, *min*, *max*, *mean*, *median*, *mode*, and *top-k* that are usually computed on a set of weights defined suitably on the objects. Such *range-aggregate* query problems have been the subject of much recent research in both the database and the computational geometry communities.

In this paper, we further generalize this line of work by considering aggregation functions on point-sets that measure the extent or "spread" of the objects in the retrieved set S'. The functions considered here include *closest pair*, *diameter*, and *width*. The challenge here is that these aggregation functions (unlike, say, *count*) are not efficiently decomposable in the sense that the answer to S' cannot be inferred easily from answers to subsets that induce a partition of S'. Nevertheless, we have been able to obtain space- and query-time-efficient solutions to several such problems including: closest pair queries with axes-parallel rectangles on point sets in the plane and on random point-sets in $\mathbb{R}^d$ $(d \geq 2)$, closest pair queries with disks on random point-sets in the plane, diameter queries on point-sets in the plane, and guaranteed-quality approximations for diameter and width queries in the plane. Our results are based on a combination of geometric techniques, including multilevel range trees, Voronoi Diagrams, Euclidean Minimum Spanning Trees, sparse representations of candidate outputs, and proofs of (expected) upper bounds on the sizes of such representations.

[*]Mentor Graphics, Hyderabad 5000082, India and International Institute of Information Technology, Gachibowli, Hyderabad 500032, India. Research supported in part by grants SR/S3/EECE/22/2004 and DST/INT/US/NSF-RPO-0155/04 from the Department of Science and Technology, Government of India. Email: `prosenjit_gupta@acm.org`

[†]Department of Computer Science & Engineering, University of Minnesota, Minneapolis, MN 55455, U.S.A. Email: {`janardan,kumaryo`}`@cs.umn.edu`. Research supported, in part, by the National Science Foundation under grant INT–0422775 and CCF–0514950.

[‡]School of Computer Science, Carleton University, Ottawa, Ontario, K1S 5B6, Canada. Email: `michiel@scs.carleton.ca`. Research supported by NSERC.

1 Introduction

Range searching is an important and well-studied class of problems in computational geometry. In a typical instance of this problem, called *range reporting*, we are given a set, S, of geometric objects (say, points in the plane) that we wish to preprocess into a data structure, so that given any query object Q (say, an axes-parallel rectangle), the subset $S' \subseteq S$ that is contained in Q can be reported efficiently. (Thus, $S' = S \cap Q$.) The paper by Agarwal and Erickson [1] provides a comprehensive survey of geometric range searching.

There are situations where it is not sufficient to merely report the objects of S'; instead, what is desired is a more informative "summary" of the output, such as an order-statistic on S'. For instance, a realtor would be interested in knowing the average or the median price of homes (the "objects") in different neighborhoods (the "queries") of a large city. This can be accomplished by applying a suitable function, called an *aggregation function*, on S'. Examples of aggregation functions include *count*, *sum*, *min*, *max*, *mean*, *median*, *mode*, and *top-k* that are usually computed on a set of weights defined suitably on the objects (in the above example, the weights are house prices). Such *range-aggregate* query problems have been the subject of much recent research in both the database and the computational geometry communities; see, for instance, [3, 8, 10, 14, 20, 22, 23, 24, 25].

In this paper, we make further contributions to range-aggregate query processing by considering aggregation functions that measure the extent or "spread" of the objects in S'. These functions include *closest pair*, *diameter* (or *farthest pair*), and *width*. Extent measures find applications in collision detection, shape-fitting, clustering etc. [2]. Often, instead of computing the measure on the entire set—which can be both expensive and unnecessary—it is more useful to "zoom in" on a region of interest that is specified by a query range and compute quickly the desired measure only for this region. (For example, given the instantaneous positions of all aircraft over a busy airport, it is important that an air-traffic controller be able to determine rapidly the closest pair within any prescribed region of airspace, in order to identify potential collisions.)

A major challenge in working with these aggregation functions is that they are not decomposable efficiently, in the sense that the answer for S', under one of these funtions, cannot be inferred quickly from answers for subsets that form a partition of S'. (For instance, given a partition of S' into sets S'_1 and S'_2, the closest pair in S' cannot be inferred in sublinear time from the closest pair information for S'_1 and S'_2.) Despite this, however, we have been able to obtain space- and query-time-efficient solutions (either exact or approximate solutions with guaranteed error bounds) to several range-aggregate extent queries, as summarized in Table 1. Our results are based on a combination of techniques, including the use of multilevel range trees, Voronoi Diagrams, Euclidean Minimum Spanning Trees, generating sparse representations of candidate output sets, and establishing proofs of (expected) upper bounds on the sizes of such representations.

Shan *et al.* [17] were among the first to consider the range-aggregate closest pair problem. For axes-parallel query rectangles, they gave a solution based on R-trees and showed that this performed well in practice; however, they did not provide a theoretical analysis of their method. Gupta [9] obtained a solution to this problem in one-dimension (resp., two-dimensions), again for axes-parallel query ranges, where the query time was $O(1)$ using $O(n)$ space (resp., $O(\log^3 n)$ query time using $O(n^2 \log^3 n)$ space). The two-dimensional result was improved recently by Sharathkumar and Gupta [19] to $O(\log^3 n)$ query time using $O(n \log^3 n)$ space. In [18, 19], they also considered a variant (motivated by applications in VLSI design rule checking [21]), where the goal is to determine if the closest pair in an axes-parallel query rectangle is within a user-specified tolerance; their

Objects	Query	Aggregation function	Query time	Space	Sec.
Points in $\mathbb{R}^2$	Rect.	Closest pair	$\log^2 n$	$n \log^5 n$	2
Random points in $\mathbb{R}^d$ ($d \geq 2$)	Hyper-rect.	Closest pair	$\log^{2d} n$	$n \log^{3d-2} n$ (expected)	3
Random points in $\mathbb{R}^2$	Disk	Closest pair	$n^{2/3+\epsilon}$ (expected)	$n^{1+\epsilon}$ (expected)	4
Points in $\mathbb{R}^d$ ($d \geq 2$)	Hyper-rect.	Closest pair (“partly in” query)	$\log^d n$	$n \log^d n$	5
	Half-space		$n^{1-1/d+\epsilon}$	$n^{1+\epsilon}$	
	Ball		$n^{1-1/(d+1)+\epsilon}$	$n^{1+\epsilon}$	
Points in $\mathbb{R}^2$	Rect.	Farthest pair	$k \log^5 n$ $1 \leq k \leq n$	$(n + (n/k)^2) \log^2 n$	6
Points in $\mathbb{R}^2$	Rect.	$(1-\delta)$-farthest pair	$\frac{1}{\sqrt{\delta}} \log^2 n$	$\frac{1}{\sqrt{\delta}} n \log n$	7
		$(1-\delta)$-farthest pair; variable δ	$\frac{1}{\sqrt{\delta}} \log n + \log^3 n$	$n \log^2 n$	
Points in $\mathbb{R}^2$	Rect.	$(1+\delta)$-width	$\frac{1}{\sqrt{\delta}} \log^3 n$	$\frac{1}{\sqrt{\delta}} n \log^2 n$	8

Table 1: *Summary of results. All results are big-O and, unless noted otherwise, worst-case. Query rectangles are axes-parallel. By “random points” we mean that the points are chosen independently and uniformly at random in the unit-square. For the result in Section 5, by “partly in” we mean that one point in the closest pair is in the query and the other is outside. Here k is a tunable parameter, $1 \leq k \leq n$, δ is an error tolerance parameter $0 < \delta < 1$, $\epsilon > 0$ is a constant, and $d \geq 2$ is a constant. By “variable δ”, we mean that it is part of the query; otherwise, it is fixed.*

---

approach answered queries in $O(\log^2 n)$ time using $O(n \log^{2+\epsilon} n)$ space, for any constant $\epsilon > 0$. To the best of our knowledge, there has been no previous work on the range-aggregate versions of the diameter or width problems.

2 Computing the closest pair in a query rectangle

Let S be a set of n points in the plane. We will show how to preprocess the points of S into a data structure such that queries of the following form can be answered: Given an axes-parallel rectangle Q, report the closest pair in $S \cap Q$. We will solve this problem using a multi-level tree structure, where higher levels are used to solve simpler variants of this query problem. We will describe this structure in a bottom-up fashion. Thus, we start with the simplest version of the query problem, and then successively solve more general versions of the problem.

Computing the closest pair in a quadrant or vertical strip: We start by showing how to answer queries of the form “given a north-east query quadrant Q, report the closest pair in $S \cap Q$”. Let G be the graph with vertex set S in which any two points p and q are connected by an edge if and only if there exists a north-east quadrant Q such that (p, q) is the closest pair in $S \cap Q$. It can be shown that G is a plane graph and, therefore, consists of $O(n)$ edges. In other words, even though the total number of "distinct" north-east query quadrants is $\Theta(n^2)$, the number of distinct answers to these queries is only $O(n)$.

For each edge $e = (p, q)$ of the graph G, we define the following point r_e in the plane: $r_e = (\min(p_x, q_x), \min(p_y, q_y))$. Let $R = \{r_e :\ e \text{ is an edge in } G\}$. We give each point r_e of R a weight which is defined to be the distance between the endpoints of e. Then, answering closest pair queries in a north-east quadrant is equivalent to answering queries of the following form: Given a north-east query quadrant Q, report the point of minimum weight in $R \cap Q$. Using a two-dimensional range tree and fractional cascading, we obtain the following result:

Lemma 1 *The points of S can be stored in a data structure of size $O(n \log n)$ such that for any north-east query quadrant Q, the closest pair in $S \cap Q$ can be reported in $O(\log n)$ time.*

We also need a data structure for answering closest pair queries in a vertical query strip:

Lemma 2 ([19]) *The points of S can be stored in a data structure of size $O(n \log^2 n)$ such that for any vertical query strip Q, the closest pair in $S \cap Q$ can be reported in $O(\log n)$ time.*

The opposite-quadrant lemma: Let A be the set consisting of all points of S whose x- and y-coordinates are less than zero, and let B be the set consisting of all points of S whose x- and y-coordinates are larger than zero. Let A_5 be the subset of A consisting of the $\min(5, |A|)$ points that are L_∞-closest to the origin, and let B_5 be the subset of B consisting of the $\min(5, |B|)$ points that are L_∞-closest to the origin.

Lemma 3 *Let (p, q) be the closest pair in S and let $p \in A$ and $q \in B$. Then, $p \in A_5$ and $q \in B_5$.*

We give a proof of this lemma, so that the reader may see the reason for the magic number "5" used above.

Proof. Assume the claim is not true. Then we may assume w.l.o.g that $p \notin A_5$. Observe that this implies that A_5 consists of five elements. Let δ be the largest L_∞-distance between the origin and any point in A_5. Since $p \notin A_5$, and p and q are in opposite quadrants, we have $d(p, q) \geq \delta$. The box $[-\delta, 0]^2$ contains all points of A_5. If we partition this box into four subboxes with sides of lenght $\delta/2$, then one of the subboxes contains two points a and a' of A_5. Since $d(a, a') \leq \sqrt{2} \cdot \delta/2 < \delta \leq d(p, q)$, it follows that (p, q) is not a closest-pair in S. This is a contradiction. ∎

Computing L_∞-neighbors in a quadrant: Assume that the x- and y-coordinates of all points of S are positive. We want to preprocess S such that queries of the following form can be answered: Given a query point q with positive coordinates, let Q_q be the south-west quadrant of q. Report the $\min(5, |S \cap Q_q|)$ points in $S \cap Q_q$ that are L_∞-closest to the origin.

Let A be the set of all points of S that are on or below the diagonal $y = x$, and let $B := S \setminus A$. Then, for each point p in A (resp. B), the L_∞-distance between p and the origin is equal to the x-coordinate (resp. y-coordinate) of p. This observation leads to the following data structure, which can be used to answer queries for the case when q is on or above the diagonal $y = x$: We maintain (i) an array storing the points of A sorted by their x-coordinates, and (ii) an array storing the points of B sorted by their x-coordinates; with each entry p in this array, we store the following information: Let $B_p := \{b \in B : b_x \leq p_x\}$. We store with p the $\min(5, |B_p|)$ lowest points in B_p.

Lemma 4 *Assume all points of S are in the positive quadrant. These points can be stored in a data structure of size $O(n)$ such that for any query point q with positive coordinates, the* $\min(5, |S \cap Q_q|)$ *points in $S \cap Q_q$ that are L_∞-closest to the origin, can be reported in $O(\log n)$ time.*

Computing the closest pair in an anchored 3-sided rectangle: Let ℓ be a fixed vertical line. An *anchored 3-sided rectangle* Q is an axes-parallel rectangle that is unbounded in the positive y-direction and that is intersected by ℓ. Thus, such a rectangle can be written as $Q = [a, b] \times [c, \infty)$, where the x-coordinate of ℓ is between a and b. We consider queries of the following form: Given an anchored 3-sided rectangle Q, report the closest pair in $S \cap Q$.

Let T be a balanced binary search tree storing the points of S at its leaves, sorted by their y-coordinates. Consider Q as above. Let u be the highest node on the right spine of T such that the horizontal line ℓ'_u that separates the left and right subtrees of u intersects Q. Let X_u be the intersection point between ℓ and ℓ'_u; this point partitions the plane into four quadrants. Let u_1 and u_2 be the left and right children of u, respectively. Let $S^l_{u_1}$ be the set of points of S_{u_1} that are to the left of the line ℓ, and let $S^r_{u_1}$ be the set of points of S_{u_1} that are to the right of ℓ. (Throughout, we use the notation S_v, where v is a node in a binary search tree, to denote the subset of S stored at the leaves of v's subtree.) Define $S^l_{u_2}$ and $S^r_{u_2}$ similarly with respect to S_{u_2}.

Consider the closest pair (p, q) in $S \cap Q$. There are six possible cases. (1) If p and q are both to the left of ℓ, then (p, q) is the closest pair of the subset of $S^l_{u_1} \cup S^l_{u_2}$ which is in the north-east quadrant of the point (a, c). Thus, we find (p, q) by storing at u the data structure of Lemma 1 storing the set $S^l_{u_1} \cup S^l_{u_2}$. (2) If p and q are both to the right of ℓ, then (p, q) is the closest pair of the subset of $S^r_{u_1} \cup S^r_{u_2}$ which is in the north-west quadrant of the point (b, c). This is symmetric to (1). (3) If p and q are both above the line ℓ'_u, then (p, q) is the closest pair of the subset of $S^l_{u_2} \cup S^r_{u_2}$ which is in the strip bounded by the vertical lines through (a, c) and (b, c). Thus, we find (p, q) by storing at u the structure of Lemma 2 storing the set $S^l_{u_2} \cup S^r_{u_2}$. (4) Assume p is in the south-west quadrant of X_u and q is in the north-east quadrant of X_u. By storing at u appropriate variants of the structure of Lemma 4, and using Lemma 3, we can compute 25 point-pairs, such that (p, q) is among them. (5) The case when p is in the north-west quadrant of X_u and q is in the south-east quadrant of X_u is symmetric to (4). (6) If p and q are both below the line ℓ'_u, then both points are in the subtree of u_1. Thus, we can find (p, q) by recursively querying this subtree.

Thus, by storing at each node u of T the data structures of Lemmas 1, 2, and 4, for the appropriate subsets of S_u, we can answer closest pair queries for anchored 3-sided rectangles.

Lemma 5 *The points of S can be stored in a data structure of size $O(n \log^3 n)$ such that for any anchored 3-sided query rectangle Q, the closest pair in $S \cap Q$ can be reported in $O(\log^2 n)$ time.*

Computing the closest pair in an anchored rectangle: Let ℓ be a fixed horizontal line. An *anchored rectangle* Q is an axes-parallel rectangle that is intersected by ℓ. Such a rectangle can be written as $Q = [a, b] \times [c, d]$, where the y-coordinate of ℓ is between c and d. We consider queries of the following form: Given an anchored rectangle Q, report the closest pair in $S \cap Q$.

Let T be a balanced binary search tree storing the points of S at its leaves, sorted by their x-coordinates. Using T, we can reduce a closest pair query for an anchored rectangle Q to four queries for anchored 3-sided rectangles and two closest pair queries for opposite quadrants. Depending on Q, these queries can be performed for appropriate subsets of S_u for any node u of T.

Lemma 6 *The points of S can be stored in a data structure of size $O(n \log^4 n)$ such that for any anchored query rectangle Q, the closest pair in $S \cap Q$ can be reported in $O(\log^2 n)$ time.*

General closest pair rectangle queries: We now show how to answer general queries of the form "given an axes-parallel rectangle Q, report the closest pair in $S \cap Q$".

Let T be a balanced binary search tree storing the points of S at its leaves, sorted by their y-coordinates. For each internal node u of T, we define the horizontal line ℓ'_u as before. Each internal node u of T stores the data structure of Lemma 6. This structure stores the set S_u and supports queries of the form "report the closest pair in an anchored query rectangle", for rectangles that are anchored with respect to the line ℓ'_u.

To answer a query with a given axes-parallel rectangle Q, we start at the root of T and follow the path until the first node u is reached such that the horizontal line ℓ'_u stored at u intersects Q. Then we use the structure storing S_u to find the closest pair in the query rectangle Q (which is anchored with respect to ℓ'_u). The correctness follows from the fact that $S \cap Q = S_u \cap Q$.

Theorem 7 *A set S of n points in the plane can be preprocessed into a data structure of size $O(n \log^5 n)$ such that for any axes-parallel query rectangle Q, the closest pair in $S \cap Q$ can be reported in $O(\log^2 n)$ time.*

3 Closest pair rectangle queries on randomly distributed points

Let S be a set of n points in the plane, chosen independently and uniformly at random in the unit-square. We obtain a data structure of expected size $O(n \log^4 n)$ and query time $O(\log^4 n)$ time. Though not as efficient, asymptotically, as the one in [19], our solution is simple and practical, and, moreover, extends naturally to any fixed dimension $d > 2$. (Also, for $d = 2$ our data structure can be constructed more efficiently than the one in [19].)

Our approach is to precompute each point-pair (p, q), with $p, q \in S$, that is the closest pair for at least one axes-parallel query rectangle. We then store each such pair as a weighted point in a four-dimensional range tree. The four dimensions are, successively, the x- and y-coordinates of p and the x- and y-coordinates of q. The weight is the Euclidean distance $d(p, q)$. Given an axes-parallel query rectangle Q, we can find the closest pair in $S \cap Q$ by doing a range-minimum query [7] on the tree with the four-dimensional hyper-rectangle $Q \times Q$.

Formally, let $\mathcal{Q}$ be the (infinite) set of all axes-parallel query rectangles. Let Λ be the number of pairs (p, q), with $p, q \in S$ and p to the left of q, such that there is a rectangle $Q \in \mathcal{Q}$ for which (p, q) is a closest pair in $S \cap Q$. Then our structure uses $O(\Lambda \log^3 n)$ space and has a query time of $O(\log^4 n)$. Moreover, it can be built in time equal to that needed to compute the Λ pairs plus $O(\Lambda \log^3 n)$ time. Thus, if Λ is "small", then this will be an efficient and practical solution.

Unfortunately, Λ can be $\Theta(n^2)$ in the worst case: Consider two sets of $n/2$ points on the boundary of the unit-circle, in opposite quadrants. Every pair of points, one from each set, contributes one to Λ since it is the closest pair for the rectangle it defines. (See Figure 1.)

However, the situation is much better if the points of S are chosen independently and uniformly at random in the unit-square, as it then turns out that the expected value of Λ is $O(n \log n)$.

Lemma 8 *Let (p, q) be an ordered point-pair, with $p, q \in S$ and p to the left of q. This pair contributes a count of one to Λ if and only if the axes-parallel rectangle, $R(p, q)$, that has $\overline{pq}$ as a diagonal is empty, i.e., it contains no point of $S \setminus \{p, q\}$.*

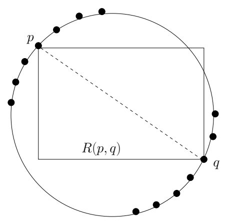

Figure 1: *An example point-set for which $\Lambda = \Theta(n^2)$.*

By Lemma 8, Λ equals the number of empty rectangles $R(p,q)$, taken over all ordered point-pairs (p,q), with $p, q \in S$ and p to the left of q. If the points of S are chosen independently and uniformly at random in the unit-square, then they are "well-distributed" and there will not be too many empty rectangles $R(p,q)$, as formalized by the following lemma. (This result is also stated, without proof, by Felsner [6].)

Lemma 9 *For a set S of n points that are chosen independently and uniformly at random in the unit-square, the expected value, $E(\Lambda)$, of Λ is $O(n \log n)$.*

From the preceding discussion it follows that the closest pair in $S \cap Q$ can be computed in $O(\log^4 n)$ time using a structure of expected size $O(n \log^4 n)$.

The above approach can be generalized to answer closest pair queries for axes-parallel query hyper-rectangles in $\mathbb{R}^d$ also, for any fixed $d \geq 3$. This is based on a result from [13] (see also [4]) that there are $O(n \log^{d-1} n/(d-1)!)$ so-called direct domination pairs (p,q) among n points drawn independently at random from the unit-hypercube in $\mathbb{R}^d$. (Pair (p,q) is a *direct domination pair* if p dominates q and there is no other point r such that p dominates r and r dominates q.) Each such pair (p,q) defines the diagonal of an empty hyper-rectangle. Thus the expected number Λ of point-pairs in $\mathbb{R}^d$ such that each is a closest pair for at least one query hyper-rectangle is $O(n \log^{d-1} n)$. Our problem reduces now to storing each such pair (p,q) as a weighted point in a $2d$-dimensional range tree, where the $2d$ dimensions are, successively, the x_1-, x_2-, …, x_d-coordinates of p and the x_1-, x_2-, …, x_d-coordinates of q, and the weight is the distance $d(p,q)$. The expected space used is $O(E(\Lambda) \log^{2d-1} n) = O(n \log^{3d-2} n)$ and the query time is $O(\log^{2d} n)$.

Theorem 10 *Let S be a set of n points chosen independently and uniformly at random in the unit-hypercube in $\mathbb{R}^d$, $d \geq 2$. S can be preprocessed into a structure of expected size $O(n \log^{3d-2} n)$ so that for any axes-parallel query rectangle Q, the closest pair in $S \cap Q$ can be reported in $O(\log^{2d} n)$ time.*

4 Closest pair disk queries on randomly distributed points

The approach in Section 3 extends naturally to closest pair queries in random point-sets in the plane when the query is a disk. Let $\mathcal{D}$ be the set of all disks. Let S_d be the set of all ordered point-pairs (p,q), with $p, q \in S$ and p to the left of q, such that there is a disk $D \in \mathcal{D}$ for which (p,q) is the closest pair in $S \cap D$. We give each pair (p,q) in S_d a weight, which is equal to $d(p,q)$.

Given a query disk D, finding the closest pair in $S \cap D$ is then equivalent to finding the point-pair (p,q) in S_d of minimum weight for which both p and q are in D. Let $\Lambda = |S_d|$. By using halfspace composition techniques (see [1]), we can solve this problem with a query time of $O(\Lambda^{2/3+\epsilon})$ using $O(\Lambda^{1+\epsilon})$ space for any constant $\epsilon > 0$.

We will show below that $E(\Lambda)$ is $O(n \log n)$. It follows that there is a structure whose expected query time is proportional to $E(\Lambda^{2/3+\epsilon})$ and whose expected size is $E(\Lambda^{1+\epsilon})$. Since $\Lambda \leq n^2$, we have $\Lambda^{1+\epsilon} \leq \Lambda n^{2\epsilon}$ and, thus, $E(\Lambda^{1+\epsilon}) \leq E(\Lambda) n^{2\epsilon} = O(n^{1+3\epsilon})$.

Recall Jensen's inequality [15] which states that for any convex function f, $E(f(\Lambda)) \geq f(E(\Lambda))$. The function $f(x) = -x^{2/3+\epsilon}$ (for $x > 0$ and fixed ϵ with $0 < \epsilon < 1/3$) is convex. It follows that $E(\Lambda^{2/3+\epsilon}) \leq (E(\Lambda))^{2/3+\epsilon} = O(n^{2/3+2\epsilon})$. Thus, by replacing ϵ by $\epsilon/3$, we obtain a structure whose expected query time is $O(n^{2/3+\epsilon})$ and whose expected size is $O(n^{1+\epsilon})$.

We can show that $E(\Lambda) = O(n \log n)$ as follows. Let $\mathcal{T}$ be the set of all axes-parallel right-triangles (i.e., the non-hypotenuse sides are parallel to the coordinate axes). Let S_t be the set of all ordered pairs (p, q), with $p, q \in S$ and p to the left of q, such that there is an axes-parallel right triangle $T \in \mathcal{T}$ for which (p, q) is the closest pair in $S \cap T$.

Lemma 11 *$S_d \subseteq S_t$.*

By Lemma 11, it suffices to show that $E(|S_t|)$ is $O(n \log n)$. Let $R(p, q)$ be the axes-parallel rectangle with $\overline{pq}$ as a diagonal. Let $T^+ = T^+(p, q)$ (resp. $T^- = T^-(p, q)$) be the subset of $R(p, q)$ lying on or above (resp. on or below) $\overline{pq}$. Note that T^+ and T^- are both in $\mathcal{T}$.

Lemma 12 *(p, q), with $p, q \in S$ and p to the left of q, is in S_t if and only if T^+ or T^- is empty.*

An upper bound on $E(|S_t|)$ now follows from the following lemma.

Lemma 13 *For a set S of n points chosen independently and uniformly at random in the unit-square, the expected number of point-pairs (p, q), where $p, q \in S$ and p is to the left of q, and such that $T^+(p, q)$ or $T^-(p, q)$ is empty, is $O(n \log n)$.*

Combined with the earlier discussion we have:

Theorem 14 *Let S be a set of n points chosen independently and uniformly at random in the unit-square. For any constant $\epsilon > 0$, there is a structure of expected size $O(n^{1+\epsilon})$ such that for any query disk D, the closest pair in $S \cap D$ can be reported in $O(n^{2/3+\epsilon})$ expected time.*

5 Computing the closest pair "partially inside" a query rectangle

Let S be a set of n points in the plane. We consider queries of the following form: Given a query region $Q \subseteq \mathbb{R}^2$, report the pair (p, q), where $p \in S \cap Q$ and $q \in S \setminus Q$, whose distance is minimum.

We will use the following property of the Euclidean Minimum Spanning Tree (EMST) of S [16].

Lemma 15 *Let T be an EMST of S, let $(\hat{S}, S \setminus \hat{S})$ be any partition of S, and let L be the set of shortest line-segments among all line-segments that have one endpoint in S and the other in $S \setminus \hat{S}$. Then, at least one line-segment of L is an edge of T.*

By Lemma 15, it suffices to consider just the $n - 1$ point-pairs that define edges of the EMST of S (rather than all $\binom{n}{2}$ pairs in S). Specifically, if the boundary ∂Q of Q is a simple closed curve (so that the "inside" and "outside" of Q are well-defined by the Jordan Curve Theorem), then $\overline{pq}$

intersects ∂Q, where (p, q) is the desired closest pair. Thus, our problem becomes: Among all edges of the EMST of S intersected by ∂Q and having exactly one endpoint in Q, find the shortest edge.

Suppose that Q is an axes-parallel rectangle $[a, b] \times [c, d]$. Let e be an edge of the EMST of S and let (e_x, e_y) and (e'_x, e'_y) be its endpoints, where (e_x, e_y) is to the left of (e'_x, e'_y).

Lemma 16 *If ∂Q intersects e and Q contains exactly one endpoint of e, then at least one of the following holds: (1) $(e_x, e_y) \in Q$ and $e'_y > d$, (2) $(e_x, e_y) \in Q$ and $e'_y < c$, (3) $(e_x, e_y) \in Q$ and $e'_x > b$, (4) $(e'_x, e'_y) \in Q$ and $e_y > d$, (5) $(e'_x, e'_y) \in Q$ and $e_y < c$, (6) $(e'_x, e'_y) \in Q$ and $e_x < a$.*

Our structure consists of four parts, D_1, D_2, D_3, and D_4: For each edge e of the EMST of S, we create the point (e_x, e_y, e'_y) in $\mathbb{R}^3$, with weight equal to the length of e. Let S_1 be the resulting point-set. D_1 is a 3-level range tree on S_1, where the level-1 tree is built on the e_x's, the level-2 trees are built on the e_y's, and the level-3 trees are built on the e'_y's. Each node of each level-3 tree stores the minimum of the weights of the points in its subtree. Similarly, D_2, D_3, and D_4 are defined on the sets of weighted points (e_x, e_y, e'_x), (e'_x, e'_y, e_y), and (e'_x, e'_y, e_x), respectively.

The query algorithm consists of the following six queries: D_1 is queried first with the 3-dimensional query rectangle $Q \times (d, \infty)$ and then with $Q \times (-\infty, c)$; this covers cases 1 and 2 of Lemma 16. D_2 is queried with $Q \times (b, \infty)$; case 3. D_3 is queried first with $Q \times (d, \infty)$ and then with $Q \times (-\infty, c)$; cases 4 and 5. D_4 is queried with $Q \times (-\infty, a)$; case 6. Each of these six query returns a real number which is the length of the shortest edge of the EMST of S satisfying one of the six conditions in Lemma 16. The smallest of these is the desired answer.

In more detail, the query on (say) D_1 with $Q \times (d, \infty)$ involves querying levels 1–3 of D_1 successively with the intervals $[a, b]$, $[c, d]$, and (d, ∞), which yields a grand total of $O(\log^3 n)$ canonical nodes in the level-3 subtrees. The union of the points contained in the subtrees of these level-3 canonical nodes is exactly the set of points of S_1 that are in $Q \times (d, \infty)$. The minimum of the weights stored at the $O(\log^3 n)$ level-3 canonical nodes is then returned.

Clearly the space is $O(n \log^2 n)$ and the query time is $O(\log^3 n)$. The latter can be improved to $O(\log^2 n)$, via *layering* (see de Berg *et al.* [5]), since the third interval in the query is semi-infinite.

This can be generalized to axes-parallel query hyper-rectangles in $\mathbb{R}^d$, $d > 2$. The analog of Lemma 16 has $4d - 2$ conditions, each of which can be expressed as a range restriction in $\mathbb{R}^{d+1}$ and, hence, can be handled using a $(d + 1)$-dimensional range tree, with layering at the innermost level.

Theorem 17 *Let S be a set of n points in $\mathbb{R}^d$, where $d \geq 2$ is a fixed constant. There is a structure of size $O(n \log^d n)$, such that for any d-dimensional axes-parallel query hyper-rectangle Q, the closest pair (p, q), where p is in $S \cap Q$ and q is in $S \setminus Q$, can be reported in $O(\log^d n)$ time.*

Remark 1 This approach works for any query object Q for which a tree structure exists such that $S \cap Q$ and $S \setminus Q$ can each be grouped into a "small" number of canonical nodes. For each edge $e = (p_e, q_e)$ of the EMST, we store (say) p_e, in such a tree structure, D, and at each node u of D we store a pointer to a secondary structure of the same type as D; this structure stores all endpoints q_e for which the corresponding points p_e are stored in the subtree of u. We query the resulting structure first with Q and then the subtrees of the resulting canonical nodes with Q's complement.

For instance, if Q is a halfplane, then a partition tree can be used for D; if Q is a disk, then again a partition tree can be used (in $\mathbb{R}^3$ since the query with Q becomes a halfspace query in $\mathbb{R}^3$, via the lifting map). This yields a solution using $O(n^{1+\epsilon})$ space, with a query time of $O(n^{1/2+\epsilon})$ in the former case and $O(n^{2/3+\epsilon})$ in the latter case, where $\epsilon > 0$ is a constant. This also generalizes to higher dimensions, with the bounds shown in Table 1.

6 Computing the diameter in a query rectangle

Let S be a set of n points in the plane. Given an axes-parallel rectangle Q, we wish to report the diameter of the set $S \cap Q$.

We choose a parameter k, $1 \leq k \leq n$, and build the following 3-part structure: (1) A range tree storing the points of S. The primary tree of stores the points at its leaves, sorted by their x-coordinates. Every node u in this primary tree stores a pointer to a secondary tree which stores the points of S_u at its leaves, sorted by their y-coordinates. (2) Each node v in each secondary tree in the range tree stores (i) the diameter of the set S_v and (ii) the furthest-point Voronoi diagram of the set S_v. Each furthest-point Voronoi diagram is preprocessed for point location queries; see Kirkpatrick [12]. (3) Let B be the set of all nodes v such that v is a node in some secondary tree and $|S_v| \geq k$. A table $\mathcal{T}$ is stored, giving for each pair v, w of distinct vertices in B, the maximum distance between any point in S_v and any point in S_w.

The space to store the range tree and the furthest-point Voronoi diagrams is $O(n \log^2 n)$. Since B has size $O((n/k)\log n)$, the structure has size $O\left(n \log^2 n + |B|^2\right) = O\left(\left(n + (n/k)^2\right)\log^2 n\right)$.

The query algorithm consists of the following steps: (a) Compute a set C of $O(\log^2 n)$ canonical nodes v in the secondary structures of the range tree such that $S \cap Q = \cup_{v \in C} S_v$. (b) Each node v of C stores the diameter of the set S_v; compute the maximum of these diameters. (c) For each pair v, w of distinct nodes in C, do the following. If both S_v and S_w have size at least k, then the table $\mathcal{T}$ stores the largest distance between any point in S_v and any point in S_w. Otherwise, assume w.l.o.g that $|S_v| < k$. Then, for each point p in S_v, perform a point location query in the furthest-point Voronoi diagram of S_w and, thus, obtain the largest distance between any point in S_v and any point in S_w. (d) Return the largest distance thus found.

Theorem 18 *Let S be a set of n points in the plane and let k be an integer with $1 \leq k \leq n$. The set S can be preprocessed into a structure of size $O((n + (n/k)^2)\log^2 n)$ such that for any axes-parallel query rectangle Q, the diameter of $S \cap Q$ can be reported in $O(k \log^5 n)$ time.*

7 Approximating the diameter in a query rectangle

Let S be a set of n points in the plane and let δ be a real, $0 < \delta < 1$. Given an axes-parallel rectangle Q, we wish to report a pair of points in $S \cap Q$ whose distance is at least $(1 - \delta)$ times the diameter of $S \cap Q$.

We first consider the case where δ is fixed. Let $\beta(\delta) = \left\lceil \frac{2\arcsin(1-\delta)}{\pi - 2\arcsin(1-\delta)} \right\rceil$. It can be shown that $\beta(\delta) = O(1/\sqrt{\delta})$ if δ converges to zero. The following lemma is adapted from Janardan [11].

Lemma 19 *Choose $2(\beta(\delta)+1)$ equally-spaced vectors around the unit-circle. For each such vector d_i, let p_i be the point of S that is extreme in direction d_i, and let q_i be the point of S that is extreme in direction $-d_i$. Let D be the diameter of S and let $\Delta = \max_i d(p_i, q_i)$. Then $1 - \delta \leq \Delta/D \leq 1$.*

Our structure is a range tree. With each node v in each secondary tree, we store the $O(\beta(\delta)) = O(1/\sqrt{\delta})$ point-pairs of Lemma 19 for S_v. The space used is $O((1/\sqrt{\delta})n \log n)$.

Given Q, we compute a set C of $O(\log^2 n)$ canonical nodes v in the secondary structures of the range tree such that $S \cap Q = \cup_{v \in C} S_v$. Consider any of the $O(\beta(\delta))$ direction pairs d_i and $-d_i$. We compute the extreme points of $S \cap Q$ in directions d_i and $-d_i$ by computing the extreme points among those stored with the canonical nodes for this direction pair. By Lemma 19, the farthest pair so computed over all direction pairs is an approximation to the diameter of $S \cap Q$.

Theorem 20 *Let S be a set of n points in the plane and let δ be a fixed real, $0 < \delta < 1$. S can be preprocessed into a structure of size $O((1/\sqrt{\delta})n \log n)$ so that for any axes-parallel query rectangle Q, a $(1-\delta)$-approximation to the diameter of $S \cap Q$ can be reported in $O((1/\sqrt{\delta}) \log^2 n)$ time.*

The above structure depends on δ. We now give a structure that is independent of δ and can answer queries where δ comes as an input parameter along with the query rectangle Q.

We again use a range tree for the points of S. Now, every secondary node v stores the convex hull of the point set S_v. The size of this structure is $O(n \log^2 n)$.

Given Q, we find a set C of $O(\log^2 n)$ canonical nodes. By merging the $O(\log^2 n)$ convex hulls of the sets S_v, with $v \in C$, we obtain the convex hull of the set $S \cap Q$. We use this hull to compute the $O(1/\delta)$ extremal pairs of $S \cap Q$. We can, in logarithmic time, compute (i) the convex hull of two convex polygons and (ii) the extreme point of a convex polygon for a given direction; see [16]. It follows that the query time is $O(\log^3 n + (1/\sqrt{\delta}) \log n)$.

Theorem 21 *Let S be a set of n points in the plane. S can be preprocessed into a structure of size $O(n \log^2 n)$ such that for any axes-parallel query rectangle Q and any real number δ, $0 < \delta < 1$, that is specified as part of the query, a $(1-\delta)$-approximation to the diameter of $S \cap Q$ can be reported in $O(\log^3 n + (1/\sqrt{\delta}) \log n)$ time.*

8 Approximating the width in a query rectangle

Recall that the *width* of a point-set is the width of a narrowest strip that encloses the point-set. Let S be a set of n points in the plane and let δ be a real number, $0 < \delta < 1$. Given an axes-parallel query rectangle Q, we wish to report a strip enclosing the points in $S \cap Q$ whose width is at most $1 + \delta$ times the width of $S \cap Q$.

Let δ be fixed. Define $\gamma(\delta) = \left\lceil \frac{\pi}{2 \arccos(1/(1+\delta))} \right\rceil$. It can be shown that $\gamma(\delta) = O(1/\sqrt{\delta})$ if δ converges to zero. Our approach is based on the following lemma, adapted from Janardan [11].

Lemma 22 *Let $S_0 = S$ and for $1 \le i \le \gamma(\delta)$, let S_i be a copy of S that is rotated clockwise around the origin from S_{i-1} by an angle of $\pi/\gamma(\delta)$. For $0 \le i \le \gamma(\delta)$, let L_i be the downward convex chain that is dual to the upper hull of the convex hull of S_i, and let R_i be the upward convex chain that is dual to the lower hull of the convex hull of S_i. (L_i is strictly above R_i.) Let ω_i^L be the minimum distance between any vertex of L_i and any point vertically below it on R_i, and let ω_i^R be the minimum distance between any vertex of R_i and any point vertically above it on L_i. Let $\Omega = \min_i\{\omega_i^L, \omega_i^R\}$ and let W be the width of S. Then $1 \le \Omega/W \le 1 + \delta$.*

As discussed in [11], the distances ω_i^L and ω_i^R can be computed in $O(\log^2 n)$ time, via binary search, if the chains L_i and R_i are stored in balanced binary search trees.

Our structure is a range tree for S. Each node v in each secondary tree stores $1+\gamma(\delta)$ instances of the data structure underlying Lemma 22 for the set S_v; specifically, the i-th instance is a pair of balanced binary search trees built on the dual chains $L_i(v)$ and $R_i(v)$ associated with the i-th rotated copy of S_v. The space used is $O((1/\sqrt{\delta})n \log^2 n)$.

Given the query Q, we compute, in $O(\log^2 n)$ time, a set C of $O(\log^2 n)$ canonical nodes v in the secondary structures of the range tree such that $S \cap Q = \cup_{v \in C} S_v$. Then, for each i, taken in turn, we merge the $L_i(v)$'s for all $v \in C$ into a single chain. Similarly, for the $R_i(v)$'s. This takes $O((1/\sqrt{\delta}) \log^3 n)$ time. From the resulting $O(1/\sqrt{\delta})$ pairs of chains, we compute the minimum vertical distance between each pair and obtain the smallest of these distances as Ω.

Theorem 23 *Let S be a set of n points in the plane and let δ be a real number, $0 < \delta < 1$. S can be preprocessed into a data structure of size $O((1/\sqrt{\delta}) n \log^2 n)$ such that for any axes-parallel query rectangle Q, a $(1+\delta)$-approximation to the width of $S \cap Q$ can be reported in $O((1/\sqrt{\delta}) \log^3 n)$ time.*

References

[1] 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*, volume 223 of *Contemporary Mathematics*, pages 1–56. American Mathematical Society, Providence, RI, 1999.

[2] P. K. Agarwal, S. Har-Peled, and K. R. Varadarajan. Approximating extent measures of points. *J. ACM*, 51:606–635, 2004.

[3] P. Bose, E. Kranakis, P. Morin, and Y. Tang. Approximate range mode and range median queries. In *Proceedings of the 22nd Symposium on Theoretical Aspects of Computer Science*, volume 3404 of *Lecture Notes in Computer Science*, pages 377–388, Berlin, 2005. Springer-Verlag.

[4] D. Datta and S. Soundaralakshmi. An efficient algorithm for computing the maximum empty rectangle in three dimensions. *Information Sciences*, 128:43–65, 2000.

[5] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, Berlin, 2nd edition, 2000.

[6] S. Felsner. Empty rectangles and graph dimension. arXiv:math/0601767v1, 2006. http://arxiv.org/abs/math/0601767v1.

[7] H. Gabow, J.Bentley, and R. Tarjan. Scaling and related techniques for geometry problems. In *Proceedings of the 16th ACM Symposium on Theory of Computing*, pages 135–142, 1984.

[8] S. Govindarajan, P. K. Agarwal, and L. Arge. CRB-tree: An efficient indexing scheme for range aggregate queries. In *Proceedings of the 9th International Conference on Database Theory*, pages 143–157, 2003.

[9] P. Gupta. Algorithms for range-aggregate query problems involving geometric aggregation operations. In *Proceedings of the 16th Annual International Symposium on Algorithms and Computation*, volume 3827 of *Lecture Notes in Computer Science*, pages 892–901, Berlin, 2005. Springer-Verlag.

[10] S. Hong, B. Song, and S. Lee. Efficient execution of range-aggregate queries in data warehouse environments. In *Proceedings of the 20th International Conference on Conceptual Modeling*, volume 2224 of *Lecture Notes in Computer Science*, pages 299–310, Berlin, 2001. Springer-Verlag.

[11] R. Janardan. On maintaining the width and diameter of a planar point-set online. *International Journal of Computational Geometry & Applications*, 3:331–344, 1993.

[12] D. G. Kirkpatrick. Optimal search in planar subdivisions. *SIAM Journal on Computing*, 12:28–35, 1983.

[13] R. Klein. Direct dominance of points. *International Journal of Computer Mathematics*, 19:225–244, 1987.

[14] D. Krizanc, P. Morin, and M. Smid. Range mode and range median queries on lists and trees. *Nordic Journal of Computing*, 12:1–17, 2005.

[15] M. Mitzenmacher and E. Upfal. *Probability and Computing.* Cambridge University Press, Cambridge, UK, 2005.

[16] F. P. Preparata and M. I. Shamos. *Computational Geometry: An Introduction.* Springer-Verlag, Berlin, 1988.

[17] J. Shan, D. Zhang, and B. Salzberg. On spatial-range closest-pair query. In *Proceedings of the 8th International Symposium on Spatial and Temporal Databases*, volume 2750 of *Lecture Notes in Computer Science*, pages 252–269, Berlin, 2003. Springer-Verlag.

[18] R. Sharathkumar and P. Gupta. Range-aggregate proximity detection for design rule checking in VLSI layouts. In *Proceedings of the 18th Canadian Conference on Computational Geometry*, pages 151–154, 2006.

[19] R. Sharathkumar and P. Gupta. Range-aggregate proximity queries. Technical Report IIIT/TR/2007/80, International Institute of Information Technology Hyderabad, `http://www.iiit.net/techreports/2007_80.pdf`, 2007.

[20] S. Shekhar and S. Chawla. *Spatial Databases: A Tour.* Prentice Hall, 2002.

[21] T. G. Szymanski and C. J. van Wyk. Layout analysis and verification. In B. Preas and M. Lorenzetti, editors, *Physical Design Automation of VLSI Systems*, pages 347–407. Benjamin/Cummins, 1988.

[22] Y. Tao and D. Papadias. Range aggregate processing in spatial databases. *IEEE Transactions on Knowledge and Data Engineering*, 16:1555–1570, 2004.

[23] D. Zhang, A. Markowetz, V. Tsotras, D. Gunopulos, and B. Seeger. Efficient computation of temporal aggregates with range predicates. In *Proceedings of the 20th Symposium on Principles of Database Systems*, pages 237–245, 2001.

[24] D. Zhang and V. J. Tsotras. Improving min/max aggregation over spatial objects. *VLDB Journal*, 14:170–181, 2005.

[25] D. Zhang, V. J. Tsotras, and D. Gunopulos. Efficient aggregation over objects with extent. In *Proceedings of the 21st Symposium on Principles of Database Systems*, pages 121–132, 2002.