Range-Aggregate Proximity Queries

R. Sharathkumar* Prosenjit Gupta†

Abstract

In a *range-aggegate query* problem we wish to preprocess a set S of geometric objects such that given a query orthogonal range q, a certain intersection or proximity query on the objects of S intersected by q can be answered efficiently. Although range-aggregate queries have been widely investigated in the past for aggregation functions like average, count, min, max, sum etc. there is little work on proximity problems. In this paper, we solve two problems. We first consider the problem of determining if any pair of points in a query orthogonal rectangle are within a constant λ of each other and give a solution that takes $O(n \log^{2+\epsilon} n)$ space and $O(\log^2 n)$ query time. Subsequently, we solve the problem of finding the closest pair in a query orthogonal rectangle which takes $O(n \log^3 n)$ space and $O(\log^3 n)$ query time.

1 Introduction

1.1 Range-Aggregate Queries

Range searching is a fairly well-studied problem in Computational Geometry [1]. In such a problem, we are given a set S of n geometric objects. The goal is to efficiently report or count the intersections of the given set of objects with a given query range q. If we are required to perform queries on the geometric data set several times, it is worthwhile to arrange the information into a data structure to facilitate searching.

In a class of problems called *range-aggregate query* problems [12] we deal with some composite queries involving range searching, where we need to do more than just a simple range reporting or counting. In a generic instance of a range-aggregate query problem, we wish to preprocess a set of geometric objects S, such that given a query range q, a certain aggregation function that operates on the objects of $S' = S \cap q$ can be computed efficiently.

*Department of Computer Science, Duke University, Durham, NC 27708 USA. Email: sharath@cs.duke.edu

†Algorithms and Computation Theory Laboratory, 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

In on-line analytical processing (OLAP), geographic information systems (GIS) and other applications range aggregate queries play an important role in summarizing information [12] and hence large number of algorithms and storage schemes have been proposed. However most of these work consider functions like count, sum, min, max, average etc. [12, 8]. In [4], range-aggregate query problems involving geometric aggregation operations like intersection, point enclosure and proximity were studied.

Geometric proximity problems arise in numerous applications and have been widely studied in computational geometry. The closest-pair problem involves finding a pair of points in a set such that the distance between them is minimal. Work on the closest-pair and some related problems are surveyed in [10]. The range-aggregate version was considered in [7] for the variant where the query range is an axes-parallel rectangle and a R-tree based solution was given. In [4] the 1-dimensional (resp. 2-dimensional) range-aggregate closest pair problems with query orthogonal rectangles were solved in $O(n)$ space (resp. $O(n^2 \log^3 n)$ space) and $O(1)$ (resp. $O(\log^3 n)$) query time.

1.2 Motivating Applications

The range-aggregate closest-pair problem considered in [7] can arise in applications like GIS. For instance, consider an example borrowed from [7]. We may be interested in finding the closest pair of post offices in a city. If the City Hall wants to move some post office to a new location, this query may help in decision making.

As another application consider VLSI design rule checking (also called "DRC"[11]). VLSI design rules are often based on the so-called *lambda* (λ) based design rules made popular by Mead and Conway [5]. Design rule checking (DRC) is the process of checking if the layout satisfies the given set of rules. In a VLSI layout editing environment [9], geometric queries commonly arise. However, the user often zooms to a part of the layout and is interested in queries with respect to the portion of the layout on the screen. One problem of interest to the designer is to check whether certain features are apart at least by a required separation. To check for violations in a part of the circuit, we can check if any two points in a query range violate the minimum separation rule. This can be reduced to an instance of the range-aggregate closest-pair query in [7].

2 Contributions and Techniques

We assume that all distances are euclidean distances. Let $d(a, b)$ denote the euclidean distance between points a and b.

In this paper, we first consider variants of the *range-aggregate proximity detection* problem. A generic instance of the problem may be specified as follows:

Problem 2.1 *Preprocess a set S of n points in $\mathcal{R}^2$ into a data structure such that given a query range q, whether or not there exists a pair of points $(v, w), v, w \in S \cap q$ and $d(v, w) < \lambda$ can be reported efficiently, where $\lambda > 0$ is a constant.*

Next we consider range-aggregate closest pair problems:

Problem 2.2 *Preprocess a set S of n points in $\mathcal{R}^2$ into a data structure such that given a query range q, a pair of points $(v, w), v, w \in S \cap q$ and $d(v, w) \leq d(x, y), \forall x, y \in S \cap q$. $(v, w), v, w \in S \cap q$ can be reported efficiently.*

To outline the techniques used, let us first recall the solution in [4] for the range-aggregate closest pair problem for a query orthogonal rectangle. To solve the 2-dimensional problem, we first find all pairwise distances between the points in S. We create $O(n^2)$ tuples $(a_x, a_y, b_x, b_y, d(a, b))$ where $a = (a_x, a_y)$ and $b = (b_x, b_y)$ are points in S and $d(a, b)$ is the euclidean distance between a and b. We preprocess these tuples into a data structure D for the 4-dimensional range minimum query of [3]. An instance of D built on a set of n points occupies $O(n \log^3 n)$ space and can be queried in time $O(\log^3 n)$. Given q, we query D to find a pair (a, b) with the minimum value of $d(a, b)$. This gives us an $O(n^2 \log^3 n)$ space and $O(\log^3 n)$ time solution. Clearly the space bound is too high for the solution to be of any practical use. In this paper we give low space solutions by capturing relevant information without storing all pairwise distances.

Our technique for solving instances of Problem 2.1 is to store pairs of points (a, b) for $d(a, b) < \lambda$ in a data structure DR for range reporting in $\mathcal{R}^4$ [2]. We show that we need to store at most n (respectively $4n$ and $8n$) pairs for the case where the query range is an orthogonal strip (respectively a quadrant and an orthogonal rectangle). The data structures occupy $O(n \log^{2+\epsilon} n)$ space and can be queried in time $O(\log^2 n)$.

Our techniques for solving the range-aggregate closest pair problems are more involved. We make use of the data structures of [3] and some properties of a graph used for storing the solutions to the problem of finding the closest pair in a query horizontal strip. Our final result is that we are able to solve the range-aggregate closest pair problem involving a query orthogonal rectangle in $O(n \log^3 n)$ space and $O(\log^3 n)$ query time thus improving the space bound of $O(n^2 \log^3 n)$ in [4] without sacrificing the query time.

In Section 3, we consider the range-aggregate proximity detection problem involving a query orthogonal rectangle. In Section 4, we solve the range-aggregate closest pair problem for a query orthogonal rectangle with bounded aspect ratio. This leads us to the solution to the problem of finding the closest pair in a query orthogonal rectangle in Section 5. We conclude in Section 6.

3 Proximity checking with a query orthogonal range

In this section we show how to store at most $O(n)$ pairs of points (a, b) for $d(a, b) < \lambda$ in a data structure DR for range reporting in $\mathcal{R}^4$ [2] and use that to solve the range-aggregate proximity detection problem with an orthogonal rectangle as a query range. We first develop the solution for a query strip and a query quadrant and then extend the same to a query rectangle.

3.1 Proximity checking in an orthogonal strip

In this section, we consider a special case of Problem 2.1 where the query q is a vertical strip. The solution for a query horizontal strip is similar.

During preprocessing, we create for each point $p = (p_x, p_y)$ in S, a pair (p, r) where $r = (r_x, r_y)$ is a point with the minimum $|s_x - p_x|$ amongst all points $s = (s_x, s_y)$ satisfying $s_x \geq p_x$ and $d(p, r) < \lambda$. (We do not create a pair for p if no such point r exists.) We preprocess these pairs into a data structure DR for orthogonal range reporting in $\mathcal{R}^4$ and stop after the first report. An instance of DR built on a set of n points occupies $O(n \log^{2+\epsilon} n)$ space and can be queried in time $O(\log^2 n)$ [2]. Given the vertical strip $q = [a, b]$, defined by the vertical lines $x = a$ and $x = b$, we query DR to find a pair (p, r). Then we report YES if such a pair is found and NO otherwise.

Lemma 3.1 *The query algorithm reports YES, iff there exist points p and r in $S \cap q$ such that $d(p, r) < \lambda$.*

Proof If the algorithm reports YES, it does so after the query on DR returns a pair (p, r), $p \in S$, $q \in S$, such that $d(p, r) < \lambda$. From the correctness of the structure DR it follows that p and r are in $q = [a, b]$. To prove the converse let there exist a pair of points $(p, r), p \in (S \cap q)$, $r \in (S \cap q)$ such that $d(p, r) < \lambda$. Without loss of generality let $p = (p_x, p_y)$ and $r = (r_x, r_y)$ be such that $r_x \geq p_x$. Then DR must contain tuples $(p, s, d(p, s))$ for points $s \in (S \cap q))$ where $p_x \leq s_x \leq r_x$ and $d(p, s) < \lambda$. Then the query on DR will return a pair (p, s) for such a point s.

In Section 5.1, we present a solution to the range-aggregate closest pair problem which can improve the solution given above. However, in the following sections, we show how to extend the solution described above to solutions to the range-aggregate proximity detection problems involving a query orthogonal quadrant and then a query orthogonal rectangle by storing more pairs in DR.

3.2 Proximity checking in a query quadrant

In this section, we consider a special case of Problem 2.1 where the query is an orthogonal quadrant. Without loss of generality, let the query be the northeast quadrant of a point q. Given point $q = (a, b)$, the northeast quadrant of q, denoted by $NE(q)$, is the set of all points (x, y) in $\mathcal{R}^2$ such that $x \geq a$ and $y \geq b$. Analogously, we can define $NW(q)$, $SE(q)$, $SW(q)$ as the northwest, southeast and southwest quadrants respectively.

During preprocessing, we compute for each point $p \in S$, the nearest neighbor $NNE(p)$ (respectively $NNW(p)$, $NSE(p)$ and $NSW(p)$) of p in the northeast (respectively northwest, southeast and southwest) quadrant. We create the pairs (p, $NNE(p)$), (p, $NNW(p)$), (p, $NSE(p)$) and (p, $NSW(p)$), if the distance of the corresponding neighbor from p is less than λ. We preprocess these pairs into a data structure DR for the 4-dimensional orthogonal range reporting problem [2]. An instance of DR built on a set of n points occupies $O(n \log^{2+\epsilon} n)$ space and can be queried in time $O(\log^2 n)$. We also preprocess the points in S into a data

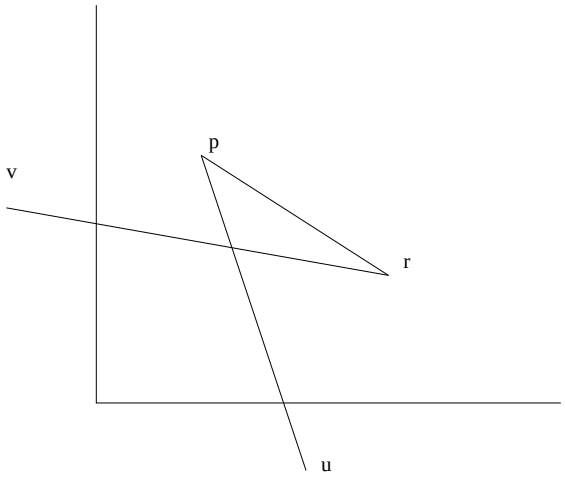

Figure 1: Points r and p are in the query quadrant but the corresponding pair is not stored.

structure DC for the 2-dimensional range counting problem. An instance of DC built on a set of n points occupies $O(n \log n)$ space and can be queried in time $O(\log n)$.

Given a query point $q = (a, b)$, we query DR with the range $NE(q)$ to find a pair (p, r) with $d(p, r) < \lambda$ and report YES if such a pair is found. Else we query DC with the square $[a, a + \lambda] \times [b, b + \lambda]$. If the count returned by the second query is more than four, we report YES. Else we check the distances between all pairs of points within the square and report YES iff we find a pair (p, r) with $d(p, r) < \lambda$.

Lemma 3.2 *Let S be a set of points in $\mathcal{R}^2$, and let for any pair of points (p, r), $p \in S$, $r \in S$, $d(p, r) \geq \lambda$, then a square of side λ contains at most 4 points of S.*

Proof See [10].

Lemma 3.3 *Given a query quadrant $NE(q)$, $q = (a, b)$, the query algorithm reports YES, iff p and r are points in $S \cap q$ and $d(p, r) < \lambda$.*

Proof The only-if part is true since $d(p, r) < \lambda$ for a pair (p, r) that is reported. To prove the if part, let (p, r) be a pair of points in $S \cap q$ and $d(p, r) < \lambda$. If (p, r) is a pair stored in DR, then the query on DR will return a pair of points (s, t) such that $d(s, t) < \lambda$. Hence a YES answer is reported. So assume that the pair (p, r) is not stored in DR. Without loss of generaliy, let p be the point with the higher y-coordinate amongst p and r. Then either $p \in NE(r)$ or $p \in NW(r)$. Let $p \in NE(r)$. Since $r \in NE(q)$, $d(p, r) < \lambda$ and the pair (p, r) is not stored in DR, there exists a point $s = NNE(r)$, $d(r, s) < \lambda$ such that the pair (r, s) is stored in DR. Point $r \in NE(q)$. Since $NE(r) \subseteq NE(q)$, point $s \in NE(q)$. Hence the query on DR returns a witness and the query algorithm returns YES. If $p \in NW(r)$, $r \in SE(p)$. Since the pair (p, r) is not stored in DR, there exist points $u \in S$, $v \in S$ such that $u \notin q$, $v \notin q$, $u = NSE(p)$, $v = NNW(r)$, $d(p, u) < \lambda$ and $d(r, v) < \lambda$. See Figure 1. Consider the square $q' = [a, a + \lambda] \times [b, b + \lambda]$. If $p_y < a + \lambda$ and $r_x > a + \lambda$, $d(r, v) \geq \lambda$ which

contradicts the fact that $d(r, v) < \lambda$. Similarly if $p_y > a + \lambda$ and $r_x < a + \lambda$, $d(p, u) \geq \lambda$, which contradicts the fact that $d(p, u) < \lambda$. If $p_y > a + \lambda$ and $r_x > a + \lambda$, $d(r, v) \geq \lambda$ and $d(p, u) \geq \lambda$, again leading to a contradiction. Hence $p \in q'$ and $r \in q'$. If the number of points in q' is more than four, the query algorithm will return YES. This step is correct by Lemma 3.2. If the number of points in q' is four or less, it checks all pairs of points in q', and since there is at least a pair (p, r) with $d(p, r) < \lambda$, the query algorithm reports YES.

3.3 Extending the solution to query rectangles

In this section we consider an instance of Problem 2.1 where the query range q is a rectangle. During preprocessing, for each point $p \in S$, we compute $NXNE(p)$ (respectively $NXNW(p)$, $NXSE(p)$ and $NXSW(p)$), being the point r in $NE(p)$ (respectively $NW(p)$, $SE(p)$ and $SW(p)$) such that $d(p, r) < \lambda$ and $|r_x - p_x|$ is the minimum for all such points r in $NE(p)$ (respectively $NW(p)$, $SE(p)$ or $SW(p)$). Similarly, we compute $NYNE(p)$ (respectively $NYNW(p)$, $NYSE(p)$ and $NYSW(p)$), being the point s in $NE(p)$ (respectively $NW(p)$, $SE(p)$ and $SW(p)$) such that $d(p, s) < \lambda$ and $|s_y - p_y|$ is the minimum for all such points s in $NE(p)$ (respectively $NW(p)$, $SE(p)$ or $SW(p)$).

We preprocess the $8n$ pairs (p, r) for all $p \in S$ and $r \in \{NXNE(p),\ NXNW(p),\ NXSE(p),\ NXSW(p),\ NYNE(p),\ NYNW(p),\ NYSE(p),\ NYSW(p)\}$ into an instance DR for 4-dimensional orthogonal range reporting [2]. This occupies $O(n \log^{2+\epsilon} n)$ space and can be queried in time $O(\log^2 n)$. We also preprocess the points in S into a data structure DC for 2-dimensional range counting which takes $O(n \log n)$ space and can be queried in time $O(\log^2 n)$.

Given a query rectangle $q = [a, b] \times [c, d]$, we query DR and stop after the first report. If we find a pair (p, r), we report YES. Else if $|b - a| \leq 2\lambda$ and $|d - c| \leq 2\lambda$, we query DC with q. If the number of points in q is 9 or less, we check all pairs of points in q to find if there is a pair (p, r) with $d(p, r) < \lambda$ and report YES if we find such a pair. If the number of points in q are more than 9, we report YES. In all other cases, we report NO.

Lemma 3.4 *Given a query rectangle $q = [a, b] \times [c, d]$ such that $|b - a| > 2\lambda$, the query on DR returns a pair (s, t), $s \in S$, $t \in S$, such that $d(s, t) < \lambda$ if and only if there is a pair (p, r), $p \in (S \cap q)$, $r \in (S \cap q)$, such that $d(p, r) < \lambda$.*

Proof The proof of the only-if part follows from the correctness of the data structure DR. To prove the if part, let there be a pair (p, r), $p \in (S \cap q)$, $r \in (S \cap q)$, such that $d(p, r) < \lambda$. In DR, we must have stored pairs (p, u), $u = NXNE(p)$, $d(p, u) < \lambda$ and (r, v), $v = NXSW(r)$, $d(r, v) < \lambda$. See Figure 2. Suppose the query on DR with q misses both the pairs. Then $u \notin q$ and $v \notin q$. Then, $|b - a| < d(p, u) + d(r, v) < 2\lambda$. This contradicts the fact that $|b - a| > 2\lambda$. Hence the query on DR returns a pair (s, t), $s \in S$, $t \in S$, such that $d(s, t) < \lambda$.

Lemma 3.5 *Given a query rectangle $q = [a, b] \times [c, d]$ such that $|d - c| > 2\lambda$, the query on DR returns a pair (s, t), $s \in S$, $t \in S$, such that $d(s, t) < \lambda$ if and only if there is a pair (p, r), $p \in (S \cap q)$, $r \in (S \cap q)$, such that $d(p, r) < \lambda$.*

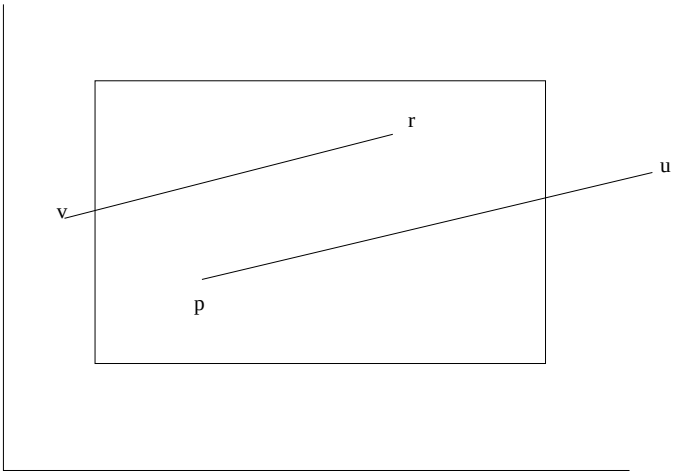

Figure 2: Points r and p are in the query rectangle but the corresponding tuple is not stored.

Proof Similar to that of Lemma 3.4.

Lemma 3.6 *Let S be a set of points in $\mathcal{R}^2$, and let q be an orthogonal rectangle of sides at most 2λ. If for any pair of points (p,r), $p \in (S \cap q)$, $r \in (S \cap q)$, $d(p,r) \geq \lambda$, then q contains at most 9 points of S.*

Proof Let us divide a square of side 2λ containing the rectangle into 9 squares of side $2\lambda/3$ each. Let the square contain more than 9 points. Then one of the smaller squares must have at least two points p and r, $p \in S$, $r \in S$. Then $d(p,r) \leq 2\sqrt{2}/3\lambda < \lambda$ which contradicts that $d(p,r) \geq \lambda$.

Lemma 3.7 *Given a query rectangle $q = [a,b] \times [c,d]$ such that $|b-a| \leq 2\lambda$ and $|d-c| \leq 2\lambda$, the query algorithm reports YES if and only if there is a pair (p,r), $p \in (S \cap q)$, $r \in (S \cap q)$, such that $d(p,r) < \lambda$.*

Proof The proof of the only-if part follows from the fact that we report (p,r) such that $p \in (S \cap q)$, $r \in (S \cap q)$ and $d(p,r) < \lambda$. To prove the if part, let there be a pair (p,r), $p \in (S \cap q)$, $r \in (S \cap q)$, such that $d(p,r) < \lambda$. If the number of points in the rectangle q is 9 or less, the algorithm checks the distance between all possible pairs of points in the rectangle and hence at least some pair (s,t),$s \in S$, $t \in S$, such that $d(s,t) < \lambda$, is detected. If the number of points in the rectangle is more than 9, then by Lemma 3.6, the rectangle must contain a pair of points (p,r), $d(p,r) < \lambda$. Then the query algorithm correctly determines the answer to be YES.

We conclude:

Theorem 3.1 *A set of n points in $\mathcal{R}^2$ can be preprocessed into a data structure of size $O(n \log^{2+\epsilon} n)$ such that given a query rectangle $q = [a,b] \times [c,d]$, whether q contains a pair of points (p,r), $d(r,p) < \lambda$, can be reported in time $O(\log^2 n)$.*

4 Closest pair in a bounded aspect-ratio orthogonal rectangle

In this section, we apply the techniques developed above to solve the range-aggregate closest pair problem in a query rectangle whose aspect ratio (the ratio of length to width) is bounded by a constant. We state the problem below.

Problem 4.1 *Preprocess a set S of n points in $\mathcal{R}^2$ into a data structure such that given a query orthogonal rectangle q whose aspect ratio is bounded by a constant, the closest pair of points (p, r), $p \in (S \cap q)$, $r \in (S \cap q)$, can be reported efficiently.*

We show that solving Problem 4.1 is equivalent to solving two simpler problems. In Section 4.3, we will show how to combine the solutions to Problem 4.2 in Section 4.1 and Problem 4.3 in Section 4.2 to give an algorithm which solves Problem 4.1. The solution to Problem 4.2 in Section 4.1 does not require the bounded aspect ratio assumption. Hence to extend the solution to Problem 4.1 to the case of general orthogonal rectangles, we replace the solution of Problem 4.3 in Section 4.2 with the corresponding solution applicable to general query rectangles in Section 5.

Definition 4.1 *For any query rectangle q, with length l and width w, we define the Critical Distance $C(q)$ as $C(q) = min(l, w)/2$.*

Definition 4.2 *For any query rectangle q, we define the closest pair as the pair of points $(v, w), v, w \in S \cap q$ and $d(v, w) \leq d(x, y), \forall x, y \in S \cap q$. The distance between the closest pair of points is the closest pair distance $CPD(q)$.*

Consider two simpler problems.

Problem 4.2 *Given a set S of n points in R^2, preprocess them into a data structure such that for any query rectangle q, a pair (a, b), $a, b \in S \cap q$, is reported satisfying the following conditions:*

- *if $CPD(q) < C(q)$, then (a, b) is this closest pair.*
- *if $CPD(q) \geq C(q)$, then $d(a, b) \geq C(q)$.*

Problem 4.3 *Given a set S of n points in R^2, preprocess them into a data structure such that for a query rectangle q with bounded aspect ratio and satisfying the condition $CPD(q) \geq C(q)$, the closest pair (a, b), $a, b \in S \cap q$ is reported efficiently.*

4.1 Closest pairs closer than critical distance

During preprocessing, we compute for each point $p \in S$, the nearest neighbor $NNE(p)$ (respectively $NNW(p)$, $NSE(p)$ and $NSW(p)$) of p in the northeast (respectively northwest, southeast and southwest) quadrant. We create the tuples (p, $NNE(p)$, $d(p, NNE(p))$), (p, $NNW(p)$, $d(p, NNW(p))$), (p, $NSE(p)$, $d(p, NSE(p))$), and (p, $NSW(p)$, $d(p, NSW(p))$). We preprocess these tuples into a data structure DM for the 4-dimensional range minimum query of [3]. We define $\Delta(q)$ as the closest-pair distance in q returned by the range minimum query. If the range minimum query fails to return any pair, $\Delta = \infty$. We define $\delta = \min\{\Delta, C(q)\}$. We also store all the points in S in a data structure $DR2$ for 2-dimensional orthogonal range searching.

Lemma 4.1 *Let S be a set of n points in $\mathcal{R}^2$. Given a query rectangle q, let $CPD(q) < C(q)$. If the closest pair (p, r) in the range q is not stored in DM, then p and r must both be contained in a square SQ of side δ in one of the corners of q.*

Proof Let $CPD(q) < C(q)$. Let (p, r) be the closest pair in the range q with $d(p, r) < \delta$. Hence $CPD(q) = d(p, r)$. However suppose the range-minimum query does not report (p, r) since the tuple $(p, r, d(p, r))$ is not stored in DM. Instead in DM, suppose we have stored tuples $(p, u, d(p, u))$, $u = NNE(p)$ and $(r, v, d(r, v))$, $v = NSW(r)$ where $d(p, u) \leq d(p, r)$ and $d(r, v) \leq d(p, r)$.

If points u and v are as in the Figure 2, then

$$2d(p, r) \geq d(u, p) + d(r, v) \geq \ell \geq 2C(q)$$

We get a contradiction since, $(d(p, r) = CPD(q)) < C(q)$.

Similarly in the symmetrical case if v is below q and u is above, we get a contradiction since

$$2d(p, r) \geq d(u, p) + d(r, v) \geq w \geq 2C(q)$$

but, $(d(p, r) = CPD(q)) < C(q)$. Thus u and v cannot be in either of these two configurations.

Let u and v be in one if the four symmetrical configurations in the corners of the query rectangle as shown in Figure 1 and let (p, r) be the closest pair in the range q with $d(p, r) < \delta$. Let $p = (p_x, p_y)$ and $r = (r_x, r_y)$. Without loss of generality, let p be the point with the higher y-coordinate than r. Then in this configuration, $p \in NW(r)$. However suppose the range-minimum query does not report (p, r) since the tuple $(p, r, d(p, r))$ is not stored in DM. Instead in DM, suppose we have stored tuples $(p, u, d(p, u))$, $u = NSE(p)$ and $(r, v, d(r, v))$, $v = NNW(r)$ where $d(p, u) \leq d(p, r)$ and $d(r, v) \leq d(p, r)$.

Consider a square SQ of side δ at the southwest corner of q. Without loss of generality assume that the southwest corner of q is at the origin. If $p_y < \delta$ and $r_x > \delta$, $d(r, v) \geq \delta$. But $d(p, r) \geq d(r, v)$. Hence $d(p, r) \geq \delta$ which contradicts the fact that $d(p, r) < \delta$ is the closest pair in the range q. Similarly if $p_y > \delta$ and $r_x < \delta$, $d(p, u) \geq \delta$. But $d(p, r) \geq d(p, u)$. Hence $d(p, r) \geq \delta$ which contradicts the fact that $d(p, r) < \delta$ is the closest pair in the range q. If

$p_y > \delta$ and $r_x > \delta$, $d(p,r) \geq d(r,v)$ and $d(p,u) \geq \delta$, and we again reach a contradiction. Thus $p_y \leq \delta$ and $r_x \leq \delta$. Since $p \in NW(r)$, now we can conclude that $p \in SQ$ and $r \in SQ$.

In the appendix, we show that the square SQ cannot have more than 4 points (Lemma A.1).

We query $DR2$ with SQ to retrieve the points in SQ. We check all pairs of points inside SQ to find the closest pair. We repeat these for all the corner squares and compare with δ, to determine the closest pair in range q.

Now, let us consider the other case where $CPD(q) \geq C(q)$. The specification of Problem 4.2 requires that if $CPD(q) \geq C(q)$, we must report (a,b) where $d(a,b) \geq C(q)$. Any pair (a,b) in $S \cap q$ satisfies $d(a,b) \geq (CPD)(q) \geq C(q)$. We report one such pair of points.

We conclude:

Theorem 4.1 *A set S of n points in $\mathcal{R}^2$ can be preprocessed into a data structure of size $O(n \log^3 n)$ such that given a query orthogonal rectangle q a pair of points (p,r), $p \in (S \cap q)$, $r \in (S \cap q)$, can be reported such that*

- *if $CPD(q) < C(q)$, then (p,r) is this closest pair.*
- *if $CPD(q) \geq C(q)$, then $d(p,r) \geq C(q)$.*

in time $O(\log^3 n)$.

4.2 Closest pairs at least as far as critical distance

This problem assumes that the given query rectangle q satisfies the condition that $CPD(q) \geq C(q)$. This implies that all pairs of points within the query rectangle has a distance greater than or equal to $C(q)$. Without loss of generality, let length ℓ be greater than width w. Thus, $C(q) = w/2$. Let us consider the case where ℓ is parallel to the x-axis.

If we divide the rectangle q into $\lfloor \ell/w \rfloor$ squares of side w each and a rectangle of width w and length less than w. Each contain at most 9 points by Lemma 3.6. Since ℓ/w is bounded by a constant as well, q contains a constant number of points. Hence the closest pair in q in this case can be determined in constant time, once the points in the range q are determined. To determine the points in the range q, we query the data structure for 2-dimensional range searching $DR2$ with q.

Theorem 4.2 *A set S of n points in $\mathcal{R}^2$ can be preprocessed into a data structure of size $O(n \log n)$ such that given a query orthogonal rectangle q with aspect ratio bounded by a constant and satisfying the condition $CPD(q) \geq C(q)$, the closest pair (a,b), $a,b \in S \cap q$ can be reported in time $O(\log n)$.*

4.3 Putting it all together

Combining the solutions to Problem 4.2 and Problem 4.3, we give an algorithm which solves Problem 4.1.

Given a query rectangle q, we obtain the solution to Problem 4.2. Let the solution to the problem be (e, f). If $d(e, f) < C(q)$ then we report (e, f) as the solution to Problem 4.1. On the other hand, if $d(e, f) \geq C(q)$, then we obtain and report the solution to Problem 4.3 as the solution to Problem 4.1. Thus it is sufficient to solve Problem 4.2 and Problem 4.3 to solve Problem 4.1.

Lemma 4.2 *The above algorithm correctly reports the closest pair in q.*

Proof If $CPD(q) \geq C(q)$, since $d(e, f) \geq CPD(q)$, $d(e, f) \geq C(q)$. If $d(e, f) \geq C(q)$, then we obtain and report the solution to Problem 4.3 as the solution to Problem 4.1, which is correct. If $CPD(q) < C(q)$, then $CPD(q) = d(e, f)$ and (e, f) is correctly reported.

We conclude:

Theorem 4.3 *A set S of n points in $\mathcal{R}^2$ can be preprocessed into a data structure of size $O(n \log^3 n)$ such that given a query orthogonal rectangle q whose aspect ratio is bounded by a constant, the closest pair of points (p, r), $p \in (S \cap q)$, $r \in (S \cap q)$, can be reported in time $O(\log^3 n)$.*

5 Closest pair in an orthogonal rectangle

In this section, we show how to remove the restriction on aspect ratios to give a solution for general orthogonal rectangles.

5.1 Closest pair in an orthogonal strip

First we provide an efficient solution to solve the range-aggregate closest pair problem in a query strip. This forms a building block for the closest pair solution we present in the next section.

We state the problem below:

Problem 5.1 *Preprocess a set S of n points in $\mathcal{R}^2$ into a data structure such that given a query horizontal strip $q = [y_1, y_2]$, the closest pair of points (p, r), $p \in (S \cap q)$, $r \in (S \cap q)$, can be reported efficiently.*

Lemma 5.1 *Given a query horizontal strip $q : [y_a, y_b]$, if (p, r) (where $p \in (S \cap q)$ and $r \in (S \cap q)$) is the closest pair of points in q, then (p, r) is the closest pair of points in all horizontal strips $q' \subseteq q$ where $p \in (S \cap q')$ and $r \in (S \cap q')$.*

To solve Problem 5.1, we define an undirected graph $G = (V, E)$ with points in S forming the vertices of G, as follows:

Definition 5.1 *A "Horizontal Strip Neighborhood Graph" HSNG(S)= G = (V,E), is defined over a set of points S in $\mathcal{R}^2$ with points in S forming the vertices of G. An edge between two vertices (u, v), $u : (u_x, u_y)$ and $v : (v_x, v_y)$ exists if and only if the pair (u, v) is the closest pair of points in the horizontal strip represented by the range $[u_y, v_y]$.*

Let $p : (p_x, p_y)$ and $r : (r_x, r_y)$.

Lemma 5.2 *If the pair (u, v) is the closest pair of points in the horizontal strip represented by some vertical range $[a, b]$, the edge $e = (u, v)$ is in E.*

Proof Since $[u_y, v_y] \subseteq [a, b]$, by Lemma 5.1, (u, v) is the closest pair of points in the horizontal strip represented by the vertical range $[u_y, v_y]$. Then by the definition of G, the edge $e = (u, v)$ is in E.

5.1.1 Upper bound on the size of the $HSNG$

Now, we show an $O(n \log n)$ upper bound on the number of edges of the graph $HSNG(S) = G = (V, E)$. Let $G^+ = (V, E^+)$ (respectively $G^- = (V, E^-)$) be the subgraph of G formed by all edges of G with non-negative (respectively negative) slope. Clearly $G = G^+ \cup G^-$. Consider a horizontal line $y = k$. Let $G_k^+ = (V, E_k^+)$ (respectively $G_k^- = (V, E_k^-)$) be a subgraph of G^+ (respectively G^-) formed by all edges of G^+ (respectively G^-) which intersect the horizontal line $y = k$. Let $G_k = G_k^+ \cup G_k^-$. In order to compute an upper-bound on the size of G, we first compute an upper-bound on the size of G_k.

Lemma 5.3 *The size of $G_k = (V, E_k)$ is $O(|V|)$.*

Proof In the proof of Lemma A.3, we show that G_k^+ is acyclic. Since there are no cycles in G_k^+, every connected component of G is a tree. Hence $|E| = O(|V|)$. The proof for G_k^- is analogous.

Theorem 5.1 *The size of G is $O(n \log n)$ where $n = |V|$.*

Proof We define a minimum height binary search tree T with n leaves, whose ith leftmost leaf corresponds to the point of S with the ith smallest y-coordinate of the endpoints of edges in G^+. To each internal node v of T, we associate a canonical subset $C_v \subseteq S$ containing the points stored at leaves in the subtree rooted at v. For any node v, let $Y(v)$ be the average of the y-coordinate corresponding to the rightmost leaf in v's left subtree and the y-coordinate corresponding to the leftmost leaf in v's right subtree; for a leaf v, we take $Y(v)$ to be the y-coordinate corresponding to v. For an internal node v let v' and v'' be the right and left child of v. Clearly, the line $y = Y(v)$ acts as a separator between the sets of points $C_{v'}$ and $C_{v''}$. Let $I(v)$ be the set of the edges of the form (a, b), $a \in C_{v'}$ and $b \in C_{v''}$ and $(a, b) \in G^+$ which intersect this line $y = Y_v$ at v. Each edge $(a, b) \in G^+$ is associated with I_v for the highest node v in T such that (a, b) intersects Y_v. From Lemma 5.3, it is clear that $|I(v)|$ is $O(|C_v|)$. Thus, at each level of the binary search tree, $O(n)$ edges are stored. Since there are $O(\log n)$ levels, the total number of edges stored is $O(n \log n)$. Similarly, we can show that the size of G^- is $O(n \log n)$. Thus the size of $G = G^+ \cup G^-$ will be $O(n \log n)$.

5.1.2 The data structure

For every edge (u, v) in G, we create a tuple $(u_y, v_y, d(u, v))$. We preprocess these tuples into a data structure DM for the 2-dimensional range minimum query of [3]. An instance of DM built on a set of n tuples takes $O(n \log n)$ space and can be queried in time $O(\log n)$. Hence the total space requirement of this data structure is $O(n \log^2 n)$ for $O(n \log n)$ tuples. This data structure can be queried with a horizontal strip $q : [y_1, y_2]$ in $O(\log n)$.

Theorem 5.2 *A set S of n points in $\mathcal{R}^2$ can be preprocessed into a data structure of size $O(n \log^2 n)$ such that given a query horizontal strip $q = [y_1, y_2]$, the closest pair of points (p, r), $p \in (S \cap q)$, $r \in (S \cap q)$, can be reported in $O(\log n)$ time.*

5.2 Solving the general case

Next we consider the general problem:

Problem 5.2 *Preprocess a set S of n points in $\mathcal{R}^2$ into a data structure such that given a query orthogonal rectangle q, the closest pair of points (p, r), $p \in (S \cap q)$, $r \in (S \cap q)$, can be reported efficiently.*

In view of the results of Theorem 4.1 available to us, for this problem we can assume that the given query rectangle q satisfies the condition $CPD(q) \geq C(q)$. Without loss of generality, let length ℓ be greater than width w. Thus, $C(q) = w/2$. Let us consider the case where ℓ is parallel to the x-axis. (The symmetrical case wherein ℓ is parallel to the y-axis can be solved analogously). Combining the result of Theorem 4.1 with the solution we get with the restriction $CPD(q) \geq C(q)$, we can solve Problem 5.2 matching the bounds of Theorem 4.3.

5.2.1 The data structure

We sort the points in S by non-decreasing order of their x-coordinates and store them at the leaves of a a balanced binary search tree T. At any internal node v of T, let $S(v)$ be the set of points whose x-coordinates are stored in the leaves of the subtree rooted at v. At v, we build a secondary data structure $D(v)$ which is an instance of the data structure of Theorem 5.2, built on the points in $S(v)$. We also store at v, two other structures $PST_1(v)$ and $PST_2(v)$ which are instances of a balanced priority search tree [6]. $PST_1(v)$ (respectively $PST_2(v)$) is essentially a binary search tree on y-coordinates and a max-heap (respectively min-heap) on the x-coordinates. If $|S(v)| = m$, $PST_1(v)$ and $PST_2(v)$ occupy $O(m)$ space each. Given a query horizontal strip q, we can determine in $O(\log m)$ time, the $four$ points with the highest (respectively lowest) x-coordinates present within the query range q by querying $PST_1(v)$ (respectively $PST_2(v)$).

5.2.2 A sparseness property for the query rectangle

The query rectangle $q : [x_1, x_2] \times [y_1, y_2]$ satisfies $CPD(q) \geq C(q)$. Consider the closest pair (p, r) in q. Without loss of generality, let p have a larger x-coordinate than r. Let $p : (p_x, p_y)$ and $r : (r_x, r_y)$. If t is the rectangle $[p_x - CPD(q), p_x] \times [y_1, y_2]$, then $r \in t$.

Lemma 5.4 *Rectangle t contains no more than* four *points.*

Proof Let us divide t into four symmetrical rectangles with their length equal to $CPD(q)/2$ and width equal to $w/2$. Since, $CPD(q) \geq C(q) = w/2$, the maximum distance between any two points within this rectangle is $\sqrt{(CPD(q)/2)^2 + (w/2)^2} < CPD(q)$ (This is a contradiction, since $CPD(q)$ is the distance between the closest pair of points in q. Thus these four rectangles contain at most one point each and hence t contains no more than four points.

5.2.3 The query algorithm

Lemma 5.5 *Given a query orthogonal rectangle $q : [x_1, x_2] \times [y_1, y_2]$ with length $\ell = x_2 - x_1$ and width $w = y_2 - y_1$, $\ell > w$, length parallel to x-axis and satisfying the property $CPD(q) \geq C(q)$ and given an ordering $\mathcal{L}$ of all the points in q in the non-decreasing order of their x-coordinates, the closest pair of points (p, r), $p_x > r_x$, satisfies the property that r is one of the four points which immediately precede p in $\mathcal{L}$.*

Proof Let us assume that r is not one of the four points preceeding p in $\mathcal{L}$. Since $r \in t = [p_x - CPD(q), p_x] \times [y_1, y_2]$, t has more than four points, violating Lemma 5.4.

Lemma 5.6 *Let $q : [x_1, x_2] \times [y_1, y_2]$ be a query orthogonal rectangle with length $\ell = x_2 - x_1$ and width $w = y_2 - y_1$, $\ell > w$, length parallel to x-axis and satisfying the property $CPD(q) \geq C(q)$. Consider a partition of q into k cells $C_1 = [a_0, a_1] \times [y_1, y_2]$, $C_2 = [a_1, a_2] \times [y_1, y_2]$, ..., $C_k = [a_{k-1}, a_k] \times [y_1, y_2]$, where $x_1 = a_0 \leq a_1 \leq a_2 \ldots \leq a_k = x_2$. Let (p, r) be the closest pair of points in the range q such that $p_x > r_x$, $r \in C_i$, $p \in C_j$, $i < j$. Then r is one of the four points with the largest x-coordinates in C_i and p is one of the four points with the smallest x-coordinates in C_j.*

Proof If r is not one of the four points with the largest x-coordinates in C_i or p is not one of the four points with the smallest x-coordinates in C_j, Lemma 5.5 is violated.

Given a query rectangle $q : [x_1, x_2] \times [y_1, y_2]$, we query the primary data structure T with the x-range $[x_1, x_2]$ and obtain the $O(\log n)$ canonical nodes. For each canonical node v we query the secondary data structure $D(v)$ with $q_1 : [y_1, y_2]$ to obtain the closest pair amongst the points in the canonical subset $S(v)$. If the closest pair of points is within any of these sets, it will be detected in this step.

Let us consider the case where the closest pair of points (p, r) are present in different canonical subsets, $p \in S(u)$ and $r \in S(v)$. From Lemma 5.6, p (respectively r) is one of the four points with the lowest (respectively highest) x-coordinates in the canonical subset $S(u)$ (respectively $S(v)$). We obtain all the $O(\log n)$ candidates for r (respectively p) by querying

$PST_1(v)$ (respectively $PST_2(v)$) for each canonical node v. We can obtain the closest pair of points from these $O(\log n)$ points in $O(\log^2 n)$ time.

The solution for the case wherein $\ell > w$ and ℓ is parallel to y-axis is similar. We conclude:

Theorem 5.3 *A set S of n points in R^2 can be preprocessed into a data structure of size $O(n \log^3 n)$ such that for a query rectangle q satisfying the condition $CPD(q) \geq C(q)$, the closest pair (a, b), $a, b \in S \cap q$ can be found in $O(\log^2 n)$ time.*

Proof At each level of the primary data structure, the total size of the secondary data structures is $O(n \log^2 n)$. Hence the overall space requirement is $O(n \log^3 n)$. At each of the $O(\log n)$ canonical nodes v, we execute queries on $D(v)$, $PST_1(v)$ and $PST_2(v)$ each requiring $O(\log n)$ time and $O(\log^2 n)$ overall. The check for the closest pairs in different canonical subsets also takes up $O(\log^2 n)$ time. Thus the total query time is $O(\log^2 n)$.

Combining the results of Theorem 4.1 and Theorem 5.3 we get:

Theorem 5.4 *A set S of n points in $\mathcal{R}^2$ can be preprocessed into a data structure of size $O(n \log^3 n)$ such that given a query orthogonal rectangle q the closest pair of points (p, r), $p \in (S \cap q)$, $r \in (S \cap q)$, can be reported in time $O(\log^3 n)$.*

6 Conclusions

We have considered the problem of detecting whether amongst a set of points in $\mathcal{R}^2$ there exist a pair of points within a distance of λ of each other in an orthogonal query rectangle and the problem of finding the closest pair in an orthogonal query rectangle. Improvement of the space and query time for the above problems, efficient dynamic solutions for these problems and extensions to higher dimensions remain important open problems.

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*, Contemporary Mathematics, 23, 1999, 1–56, American Mathematical Society Press.

[2] S. Alstrup, G. Brodal and T. Rauhe. New data structures for orthogonal range searching. *Proceedings, IEEE Symposium on Foundations of Computer Science*, 2000, 198–207.

[3] H.N. Gabow, J.L. Bentley, and R.E. Tarjan. Scaling and related techniques for geometry problems. *Proceedings, ACM Symposium on Theory of Computing*, 1984, 135–142.

[4] P. Gupta. Algorithms for range-aggregate query problems involving geomteric aggregation operations. *Proceedings, International Symposium on Algorithms and Computation*, Springer Verlag LNCS, Vol. 3827, 2005, 892–901.

[5] C.A. Mead, and L.A. Conway. *Introduction to VLSI Systems*, Addison Wesley, USA, 1980.

[6] E.M. McCreight. Priority search trees, *SIAM Journal of Computing*, 14(2), 1985, 257–276.

[7] J. Shan, D. Zhang, and B. Salzberg. On spatial-range closest-pair query. *Proceedings, Symposium on Spatial and Temporal Databases*, Springer Verlag LNCS, Vol. 2750, 2003, 252–269.

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

[9] N. Sherwani. *Algorithms for VLSI Physical Design Automation*, Kluwer Academic, 1998.

[10] M. Smid. Closest point problems in computational geometry. *Handbook of Computational Geometry*, J. Sack and J. Urrutia editors, Elsevier, 2000, 877–935.

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

[12] Y. Tao and D. Papadias. Range aggregate processing in spatial databases. *IEEE Transactions on Knowledge and Data Engineering*, 16(12), 2004, 1555–1570.

A Appendix

Lemma A.1 *Let S be set of points in $\mathcal{R}^2$. Let DM be a data structure that supports range minimum queries in $\mathcal{R}^4$. Let us store for each point $p \in S$, the tuples $(p, NNE(p), d(p, NNE(p)))$, $(p, NNW(p), d(p, NNW(p)))$, $(p, NSE(p), d(p, NSE(p)))$, $(p, NSW(p), d(p, NSW(p)))$ in DM. Let Δ be the distance between the closest pair of points retrieved by the range-minimum query on DM with an orthogonal query rectangle q ($\Delta = \infty$ if no points are retrieved). Let $\delta = \min\{\Delta, C(q)\}$. Then if SQ is a square of size δ in the southwest corner of q, SQ cannot have more than four points.*

Proof Consider a pair of points (a, b). Let $b = NE(a)$. Also, consider a circle $C^a_{a,b}$ centered at a and with radius $d(a, b)$. We define a region $R_{a,b}(a) = C^a_{a,b} \cap NE(a)$. Similarly $R_{a,b}(b) = C^b_{a,b} \cap SW(b)$. In the other case, where $b = NW(a)$, $R_{a,b}(a)$ is defined as $C^a_{a,b} \cap NW(a)$ and $R_{a,b}(b)$ is defined as $C^b_{a,b} \cap SE(b)$.

In order to prove that there cannot be more than 4 points within SQ consider the two regions formed by the diagonal of SQ originating from the south-west corner of q. Let the regions be T_1 and T_2 as shown in Figure 3. Also, let the south-west corner of SQ be the origin. We show that if there are two points,(a, b), within any of these right-angled triangles such that $d(a, b) < \delta$, then there will be some pair (s, t), stored in DM such that $d(s, t) < \delta$.

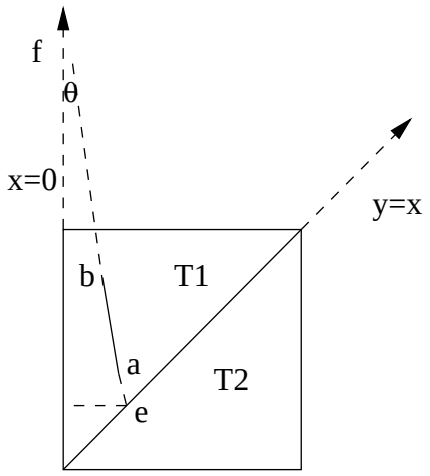

Figure 3: Proof of Lemma A.1: Case: b in $NW(a)$.

Let $a \in SQ$, $b \in SQ$, $d(a,b) < \delta$. The pair of points (a,b) can exist in two configurations. Consider the case where b is in the $NE(a)$. We show that $R_{a,b}(a)$ is completely within the rectangle q. To show this we need to show that the points $s : (a_x + d(a,b), a_y)$ and $t : (a_x, a_y + d(a,b))$ should lie within the query rectangle q. Since $a \in SQ$, $a_x < \delta$ and $a_y < \delta$. Also, $d(a,b) < \delta$. This would mean that both the coordinates of s and t are less than $2\delta \leq 2C(q) \leq min(l,w)$. This would imply that p and r lie within q. Thus $R_{a,b}(a)$ is in q which in turn implies that the closest point to a in the $NE(a)$ is within q and has a distance less than δ. Thus the corresponding pair should have been stored in DM. This leads to a contradiction since if there is a pair of points closer than δ which is in q and is stored in DM, then it would be detected in the range-minimum query.

Consider the case where $b \in NW(a)$. First we show that if the pair (a,b) is in T_1, then $R_{a,b}(b)$ is in q. Thus, we need to show that $p : (b_x + d(a,b), b_y)$ and $r : (b_x, b_y - d(a,b))$ lie within q. Since $b_x < \delta$, $b_y < \delta$ and $d(a,b) < \delta$, both p_x and p_y are less than $2\delta \leq 2C(q)$ $\leq min(l,b)$. Thus, p is in q. To show that r is in q consider the line L passing through a and b. Let the intersection of this line with lines $y = x$ (representing the diagonal of SQ) and $x = 0$ (representing one of the sides of SQ) be e and f as shown in Figure 3. Clearly, $R_{a,b}(b) \subseteq R_{e,f}(f)$ and hence $r_y > f_y - d(e,f)$. But $f_y = d(e,f)(sin\theta + cos\theta)$ where θ is the acute angle formed between lines L and $x = 0$. But, $sin\theta + cos\theta \geq 1$ for $0 < \theta < \pi/2$. Hence $f_y > d(e,f)$ and $r_y > 0$. This would mean that $0 < r_y < \delta$ and $r_x = b_x < \delta$. Hence, r is in q. Thus $R_{a,b}(b)$ is in q which in turn implies that the closest point to b in the quadrant $NW(b)$ is within q and has a distance less than δ. This leads to a contradiction since if there is a pair of points closer than δ which is in q and is stored in DM, then it would be detected in the range-minimum query.

Next we show that if $b \in NW(a)$ and if the pair (a,b) is in T_2, then $R_{a,b}(a)$ is in q. Thus, we need to show that $p : (a_x - d(a,b), a_y)$ and $r : (b_x, b_y + d(a,b))$ lie within q. Since $a_x < \delta$, $a_y < \delta$ and $d(a,b) < \delta$, both r_x and r_y are less than $2\delta \leq 2C(q) \leq min(l,b)$. Thus, r is in q. To show that p is in q consider the line L passing through a and b. Let the intersection

of this line with lines $y = x$ (representing the diagonal of SQ) and $y = 0$ (representing one of the sides of SQ) be e and f. Clearly, $R_{a,b}(a) \subseteq R_{e,f}(f)$ and hence $p_x > f_x - d(e, f)$. But $f_x = d(e, f)(sin\theta + cos\theta)$ where θ is the acute angle formed between lines L and $y = 0$. But, $sin\theta + cos\theta \geq 1$ for $0 < \theta < \pi/2$. Hence $f_x > d(e, f)$ and $p_x > 0$. This would mean that $0 < p_x < \delta$ and $p_y = a_y < \delta$. Hence, p is in q. Thus $R_{a,b}(b)$ is in q which in turn implies that the closest point to b in the quadrant $NW(b)$ is within q and has a distance less than δ. Thus the corresponding pair should have been stored in DM. This leads to a contradiction since if there is a pair of points closer than δ which is in q and is stored in DM, then it would be detected in the range-minimum query.

Neither T_1 nor T_2 can have a pair which have a distance less than δ. If T_1 (respectively T_2) has three or more points, there exists a pair of points $(a, b) \in T_1$ (respectively $(a, b) \in T_2$) such that $d(a, b) < \delta$. Hence T_1 and T_2 each have at most two points. Thus the square SQ cannot have more than 4 points.

A.1 Properties of the $HSNG$

Lemma A.2 *Let $G = (V, E)$ be a $HSNG$ as in Definition 5.1 defined on a set S of n points in $\mathcal{R}^2$. If there are two points a and b, both in the NE (or SE) quadrant of point p, such that $a_x < b_x$ and (p, a) and (p, b) are edges in G, then $a_x < (b_x + p_x)/2$. Similarly, if there are two points a and b in the quadrant NW (or SW) of point p, such that $a_x > b_x$ and (p, a) and (p, b) are edges in G, then $a_x > (p_x + b_x)/2$*

Proof Let $a : (a_x, a_y)$, $b : (b_x, b_y)$ and $p : (p_x, p_y)$ be three vertices in G such that $a = NE(p)$, $b = NE(p)$ and $a_x < b_x$. Since $a_x < b_x$, a_y has to be greater than b_y. Otherwise, edge (p, b) will not exist in G. Let us assume that $a_x \geq (p_x + b_x)/2$. Now, consider the perpendicular bisector to segment (p, b). Also, let $c : (c_x, c_y)$ be the mid-point of (p, b). Thus $c_x = (b_x + p_x)/2$. For segment $\overline{xy}$, let H^x_{xy} (respectively H^y_{xy}) be the halfplane defined by the perpendicular bisector of xy, containing x (respectively y). Since $a_x \geq c_x$, $a_y > c_y$, $a \in H^b_{pb}$. as shown in Figure 4. This leads to a contradiction because if point a lies in this region, it will be closer to b than to p and hence (p, a) will not be an edge in G. Thus $a_x < (p_x + b_x)/2$. A similar proof can be constructed for SE quadrant of p.

Now consider the a and b to be in the NW quadrant of p. Let $a_x < b_x$. Since, $a_x > b_x$, $a_y > b_y$. Otherwise edge (p, b) will not exist in G. Let us assume, $a_x \leq (p_x + b_x)/2$ Consider the perpendicular bisector of (p, b). Also, let $c : (c_x, c_y)$ be the mid-point of the segment (p, b). Thus $c_x = (b_x + p_x)/2$. Since $a_x \leq c_x$ and $a_y > c_y$, $a \in H^b_{pb}$. But this leads to a contradiction because if point a lies in this region it will be closer to b than to p and hence (p, a) will not be an edge in G. Thus $a_x > (b_x + p_x)/2$.

A similar proof can be constructed for the SW quadrant of p.

Lemma A.3 *Consider a horizontal line $y = k$. Let G^+_k be a subgraph formed by all edges of G^+ which intersect this line. Graph G^+_k does not have any cycles.*

Proof Consider the horizontal line $y = k$. Let S^+_k (respectively S^-_k) be the set of vertices of G^+_k above (respectively below) the line $y = k$ $y > k$ be S_{k+} and the set of points with $y < k$ be

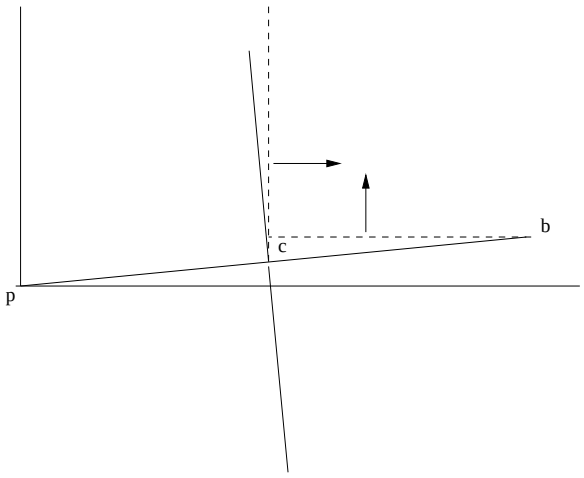

Figure 4: If $a_x > c_x$ and $a_y > b_y$, a will lie in the region indicated above.

S_{k-}. Clearly, G_k^+ is a bipartite graph with S_{k+} and S_{k-} being the two partitions. To prove the above lemma, let us assume that there is a cycle $C : u_1, v_1, u_2, v_2, u_3, v_3 \ldots u_k, v_k, u_1$, $\forall\ i\ u_i \in S_{k-}$, $v_i \in S_{k+}$ in the subgraph G_k^+. For any point u_i, let u_i^x be its x-coordinate and let u_i^y be its y-coordinate. Let $u_a \in S_{k-}$ be the point with the highest y-coordinate among all points in $C \cap S_{k-}$. Vertex u_a contributes two edges to C. These two edges are (v_{a-1}, u_a) and (u_a, v_a). Without loss of generality, let us assume $v_{a-1}^x > v_a^x$. From Lemma A.2, we derive the properties of u_{a-1} and u_{a+1}

- $u_{a+1}^x < v_a^x$: This is because, $u_{a+1}^y < v_a^y$ and the slope of the edge (v_a, u_{a+1}) should be positive.
- $u_{a-1}^x > v_a^x$: v_a and v_{a-1} are in the $NE(u_a)$ and have edges to u_a. Hence, from Lemma A.2, $v_a^x < (v_{a-1}^x + u_a^x)/2$. Also, u_a and u_{a-1} are in $SE(v_{a-1})$ and have edges to v_{a-1}. Hence, from Lemma A.2, $u_{a-1}^x > (u_a^x + v_{a-1})/2$. Thus, $u_{a-1}^x > v_a^x$.

Now consider the remaining part of the cycle C,

$$C' \ : \ (u_{a+1}, v_{a+1}u_{a+2}, v_{a+2} \ldots u_k, v_k, u_1, v_1 \ldots u_{a-1}).$$

Since the x-coordinate of v_a lies between the x-coordinate of the first point of C': u_{a+1} and the last point of C': u_{a-1}, there have to be two consecutive points u_z and u_{z+1} whose x-coordinates lie on either sides of v_a. They also share a common neighbor in v_z in G_k^+.

Consider Figure 5. Vertex u_a, which is the point with the highest y-coordinate in $C \cap S_{k-}$ is connected to v_a and v_{a-1}. The dotted vertical line represents $x = v_a^x$. Vertices u_b and u_c are points on either side of this line. Let us assume that v_d is the common neighbor of u_b and u_c. Consider the perpendicular bisector (denoted by slanted dotted line) of the line segment (u_a, v_d). Clearly, v_a should lie on the side of the perpendicular bisector which contains the point u_a. This is because, otherwise $d(v_a, v_d)$ will be smaller than $d(u_a, v_a)$ and

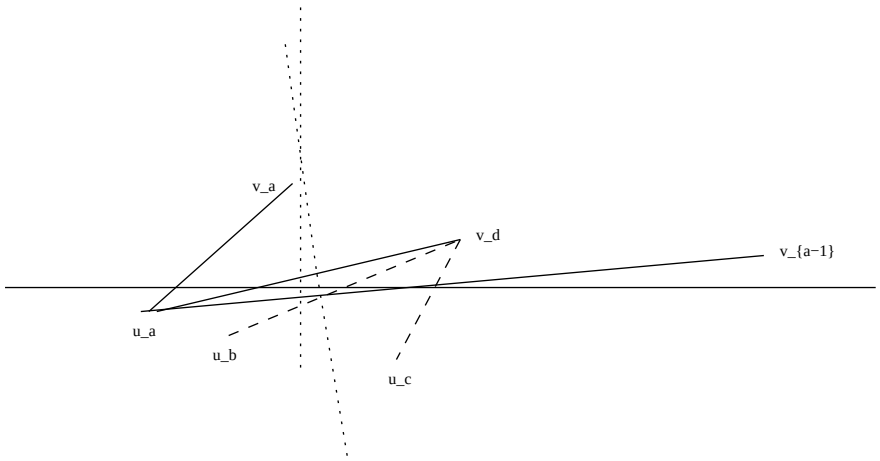

Figure 5: Figure for proof of Lemma A.3

hence (u_a, v_a) won't be an edge in G and hence in G_k^+. Now, since $u_b^x < v_a^x$, u_b will also lie in the same region. But any point in this region is closer to u_a then to v_d. Hence, u_b is closer to u_a then to v_d. Therefore, (u_b, v_d) will not be an edge in G leading to a contradiction of our assumption that v_d is a common neighbor of u_b and u_c.

Hence no two points u_b and u_c in S_k^- with $u_b^x < v_a^x$ and $u_c^x > v_a^x$ can share a common neighbor in G_k^+. In particular u_z and u_{z+1} cannot share a common neighbor in G_k^+. Contradiction. Hence that there are no cycles in G_k^+.