Range-Aggregate Proximity Detection for Design Rule Checking in VLSI Layouts

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 consider the problem of determining if any pair of points in a query range are within a constant λ of each other. We consider variants with ranges being a vertical strip, a quadrant and a rectangle. All our solutions work in close to linear space and polylogarithmic 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. Since 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 [11] 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.

In on-line analytical processing (OLAP), geographic information systems (GIS) and other applications range aggregate queries play an important role in summarizing information [11] 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. [11, 7]. In [4], range-aggregate query problems involving geometric aggregation operations like intersection, point enclosure and proximity were studied. Other than the work of [6] on the range-aggregate closest-pair problem, and the 1-dimensional (resp. 2-dimensional) range-aggregate closest pair problems with query orthogonal rectangles which were solved in [4], little or no work has been done on range-aggregate queries with proximity related aggregation functions.

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 [9]. The range-aggregate version was considered in [6] 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 [6] can arise in applications like GIS. For instance, consider an example borrowed from [6]. 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"[10]). 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 [8], 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

*Algorithms and Computation Theory Laboratory, International Institute of Information Technology, Gachibowli, Hyderabad 500032, INDIA. This research is 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: `sharath@students.iiit.ac.in`, `pgupta@iiit.ac.in` the range-aggregate closest-pair query in [6].

2 Our Contributions

Unless otherwise mentioned, we assume that all distances are euclidean distances. Let $d(a, b)$ denote the euclidean distance between a and b. Let us consider the special case of the above problems that occurs in VLSI design rule checking. 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.*

Of course, we can solve this problem by using the solution to the range-aggregate closest pair problem of [4]. Let us first recall the solution in [4]. 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 show how we can solve Problem 2.1 by storing at most n (respectively $4n$ and $8n$) tuples for the case where the query range is a vertical strip (respectively a quadrant and an orthogonal rectangle). All our solutions work in close to linear space and polylogarithmic query time. Due to lack of space, we omit some proofs and many details.

3 Proximity checking in a vertical strip

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

During preprocessing, we create for each point $p = (p_x, p_y)$ in S, a tuple $(p, r, d(p, r))$ 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 tuple for p if no such point r exists.) 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 the vertical strip $q = [a, b]$, defined by the vertical lines $x = a$ and $x = b$, we query D to find a pair (p, r) with the minimum value of $d(p, r)$. Then we report YES if such a pair is found and NO otherwise.

Lemma 1 *The query algorithm reports YES, iff p and r are points in $S \cap q$ and $d(p, r) < \lambda$.*

Proof: If the algorithm reports YES, it does so after the query on D returns a pair (p, r) such that $d(p, r) < \lambda$. From the from the correctness of the structure D 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 D 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 D will return a pair (p, s) for such a point s.

4 Proximity checking in a query quadrant

In this section, we consider a special case of Problem 2.1 where the query is 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 neighbour $NNE(p)$ (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 $(p, NSW(p), d(p, NSW(p)))$, if the distance of the corresponding neighbour from p is less than λ. We preprocess these tuples into a data structure DM for the 4-dimensional range minimum query of [3]. An instance of DM built on a set of n points occupies $O(n \log^3 n)$ space and can be queried in time $O(\log^3 n)$. We also preprocess the points in S into a data 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 DM with the range $NE(q)$ to find a pair (p, r) with the minimum value of $d(p, r)$. Then we report YES if $d(p, r) < \lambda$. 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 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.*

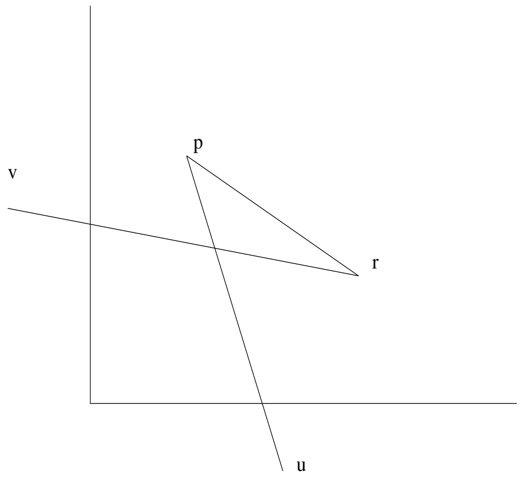

Figure 1: Points r and p are in the query quadrant but the corresponding tuple is not stored.

Lemma 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 $(p, r, d(p, r))$ is a tuple stored in DM, then the range-minimum query will return a pair of points (s, t) such that $d(s, t) \leq d(p, r) < \lambda$. Hence a YES answer is reported. So assume that the tuple $(p, r, d(p, r))$ is not stored in DM. 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 tuple $(p, r, d(p, r))$ is not stored in DM, there exists a point $s = NNE(r)$, $d(r, s) < \lambda$ such that the tuple $(r, s, d(r, s))$ is stored in DM. Point $r \in NE(q)$. Since $NE(r) \subseteq NE(q)$, point $s \in NE(q)$. Hence the range-minimum query on DM returns a witness and the query algorithm returns YES. If $p \in NW(r)$, $r \in SE(p)$. Since the tuple $(p, r, d(p, r))$ is not stored in DM, there exist points $u \in S$, $v \in S$ such that $u \notin q$, $v \notin q$, $u = NSE(p)$ and $v = NNW(r)$. See Figure 1. For $q' = [a, a + \lambda] \times [b, b + \lambda]$, if $p \notin q'$, $d(p, u) > \lambda$ which is impossible since $u = NSE(p)$ and $d(p, u) < \lambda$. Hence $p \in q'$. Similarly $r \in q'$. If the number of points in q' is more than four, the query algorithm will return YES. This step is correct by Lemma 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.

5 Proximity checking in a rectangle

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 $8n$ tuples $(p, r, d(p, r))$ 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 DM of the data structure of [3] for the 4-dimensional range-minimum query. This occupies $O(n \log^3 n)$ space and can be queried in time $O(\log^3 n)$. We also preprocess the points in S into a data structure DC for 2-dimensional range counting.

Given a query rectangle $q = [a, b] \times [c, d]$, we query DM. 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.

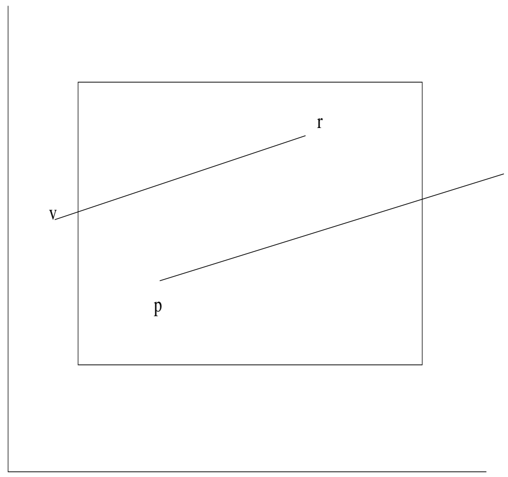

Figure 2: Points r and p are in the query rectangle but the corresponding tuple is not stored.

Lemma 4 *Given a query rectangle $q = [a, b] \times [c, d]$ such that $|b-a| > 2\lambda$, the range-minimum query on DM 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 DM. 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 DM, we must have stored tuples $(p, u, d(p,u))$, $u = NXNE(p)$, $d(p,u) < \lambda$ and $(r, v, d(r,v))$, $v = NXSW(r)$, $d(r,v) < \lambda$. See Figure 2. Suppose the range minimum query on DM with q misses both the tuples. 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 range-minimum query on DM returns a pair (s,t), $s \in S$, $t \in S$, such that $d(s,t) < \lambda$.

Lemma 5 *Given a query rectangle $q = [a,b] \times [c,d]$ such that $|d-c| > 2\lambda$, the range-minimum query on DM 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: Similar to that of Lemma 4.

Lemma 6 *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 rectangle of sides at most 2λ contains at most 9 points of S.*

Lemma 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 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 DM. 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 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 the contrapositive of Lemma 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.

Note that since all tuples stored are ones for which the point pairs are within λ, we can replace the range-minimum query data structure with a data structure for orthogonal range reporting in $\mathcal{R}^4$ and stop after the first report. This can be done in $O(n \log^{2+\epsilon})$ space and $O(\log^2 n)$ time [2].

Theorem 8 *A set of n points in $\mathcal{R}^2$ can be preprocessed into a data structure of size $O(n \log^3 n)$ (respectively $O(n \log^{2+\epsilon})$) 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 $O(\log^3 n)$ (respectively $O(\log^2 n)$).*

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 FOCS*, 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 ISAAC 05*, 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] 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.

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

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

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

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

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