International Journal of Foundations of Computer Science
Vol. 22, No. 7 (2011) 1707–1728

DOI: 10.1142/S0129054111008982

DATA STRUCTURES FOR RANGE-AGGREGATION OVER CATEGORIES*

SALADI RAHUL†
Laboratory for Spatial Informatics
International Institute of Information Technology
Gachibowli, Hyderabad, Andhra Pradesh 500 032, India
saladi.rahul@gmail.com

PROSENJIT GUPTA
Department of Computer Science and Engineering
Heritage Institute of Technology, Kolkata 700107, India
prosenjit_gupta@acm.org

K. S. RAJAN
Laboratory for Spatial Informatics
International Institute of Information Technology
Gachibowli, Hyderabad, Andhra Pradesh 500 032, India
rajan@iiit.ac.in

Received 12 March 2010
Accepted 27 April 2011
Communicated by Samir Khuller

We solve instances of a general class of problems defined as follows: Preprocess a set S of possibly weighted colored geometric objects (e.g. points/orthogonal segments/rectangles) in $\mathbb{R}^d$, $d \geq 1$ such that given a query orthogonal range q (e.g. hyperbox/point), we can report efficiently for each distinct color c of the points in $S \cap q$, the tuple $\langle\ c, \mathcal{F}(c)\ \rangle$ where $\mathcal{F}(c)$ is a function (e.g. weighted sum, bounding box etc.) of the objects of color c in q.

Keywords: Geometric searching; output-sensitive querying; colored or generalized intersection aggregate problems.

1. Introduction

Range searching is a fairly well-studied problem in Computational Geometry [2]. In such a problem, the goal is to preprocess a set S of n geometric objects to report or

*A preliminary 4-page version of this paper appears in the Proceedings of the Canadian Conference on Computational Geometry, 2009.
†Corresponding author.

count the intersections of the objects in S with a given query range q. The paper by Agarwal and Erickson [2] provides a comprehensive survey of geometric range searching.

There arise situations in which it is not sufficient to merely report or count the objects of S intersecting with the query range q. In applications like on-line analytical processing (OLAP), geographic information systems (GIS) and information retrieval (IR), *aggregation* plays an important role in summarizing information [25] and hence large number of algorithms and storage schemes have been proposed to support such queries.

In a class of problems called *range-aggregate query* problems [25] many composite queries involving range searching are considered, wherein one needs to compute the aggregate function of the objects in $S \cap q$ rather than report (or count) all of them as in a range reporting (or counting) query [19, 15].

In *colored intersection searching* (also known in the literature as *generalized intersection searching* and *categorical intersection searching*) [18], a set of objects S comes aggregated in groups and we indicate the group an object belongs to by assigning each group a unique color. Our goal is to preprocess these objects such that given a query q, the distinct colors of the objects in S that intersect q can be reported or counted efficiently [11, 12, 13, 14].

1.1. *Problem statement*

In this work, we consider a general class of problems and the problem statement is as follows:

- We are given a set S of possibly weighted objects in $\mathbb{R}^d$, $d \geq 1$ which comes aggregated in groups and we indicate the group an object belongs to by assigning each group a unique color. The goal is to preprocess S such that given a query orthogonal range q, we can report efficiently for each distinct color c of the objects in $S \cap q$, the tuple $\langle\ c, \mathcal{F}(c)\ \rangle$ where $\mathcal{F}(c)$ is a function of the objects of color c in q. The function $\mathcal{F}$ is assumed to be a commutative semigroup which allows us to combine results of subproblems without losing correctness.

Lai et al. [20] studied this class of problems for approximate queries for functions like *min*, *max*, *sum*, *count*, *report* and *heavy*. If $\mathcal{F}(c) = NULL$, the unweighted variant of the problem is the generalized orthogonal range reporting problem [18]. We assume that the dimension d is a constant relative to n and denote the coordinate axes by $x_1, x_2, \ldots, x_d$.

1.2. *Motivation*

The above class of queries has been well studied in the database community as "*GROUP-BY*" queries. The "*GROUP-BY*" is a common basic operation in databases and is applied to the categorical attributes [24, 3, 10]. As remarked in

[3], they are among the most important class of queries in OLAP ("Online Analytical Processing") applications in decision support systems. As an example of a *GROUP-BY* query, consider a database of mutual funds which has three attributes for each fund: its annual total return, its beta (a real number measuring the fund's volatility) and the fund family the fund belongs to. Whereas the first two attributes are range attributes (on which we can perform range searching), the third one is not. A typical query is to determine the families that offer funds whose total return is between say 15% and 20% and whose beta is between, say, 0.9 and 1.1 and report the number of funds in the given range for each such family. In database terminology, this is a *GROUP-BY* on the fund family attribute with aggregation operation *COUNT*. This is also an instance of the weighted version of the problem above for $d = 2$, wherein each object is a point having unit weight.

1.3. *Related work*

In a 1-dimensional static *type*-2 range counting problem, we wish to preprocess a set S of n colored points, so that for each color intersected by a query interval $q = [a_1, b_1]$, the number of points of that color in q can be reported efficiently. A solution that takes $O(n \log n)$ space and supports queries in time $O(\log n + C)$, C being the number of colors reported, was given in [11]. The space bound was improved to $O(n)$ in [4]. For the 2-dimensional static *type*-2 range counting problem, a solution that takes $O(n \log n)$ space and $O(\log^2 n + C \log n)$ query time was given in [4]. For the 1-dimensional static *type*-2 point enclosure counting problem (here set S is n intervals and query q is a point), a solution that takes $O(n)$ space and $O(\log n + C)$ query time was given in [11]. To the best of our knowledge, no other results are known for these problems.

2. Our Contributions

In this paper, we have come up with output sensitive solutions while using near-linear space. The results obtained in this paper are summarized in Table 1. The flow of the paper is as follows: In Section 2, we consider the 'colored weighted sum problem' ($\mathcal{F}(c)$ is weighted sum) in $\mathbb{R}^d$, $d \geq 1$. The technique of 'adding range restrictions' is suitably adapted in this section for solving some of the problems considered in this paper. In Section 3, the 'colored bounding box problem' ($\mathcal{F}(c)$ is bounding box) in $\mathbb{R}^d$, $d \geq 1$. A variation of this problem is considered in Section 4, wherein each color needs to have at least two points inside the query region. In Section 5 and 6, the 'colored point enclosure weighted sum' problem is considered in $\mathbb{R}^1$ and $\mathbb{R}^2$, respectively. Section 7 deals with 'Colored segment intersection weighted sum' problem in $\mathbb{R}^2$. Finally in Section 8 we conclude our work and mention some open problems.

Table 1. Summary of results. All results are "big-Oh" and worst case. Rectangles are axis-parallel. "three sided rectangle" means an orthogonal box of the form $[a_1, b_1] \times [a_2, \infty)$. The "nontrivial bounding box" function returns a bounding box of all those colors that have at least two points that intersect q. A "bounding box" for a point set in $\mathbb{R}^d$ is the hyper box with the smallest measure within which all the points lie. ϵ is an arbitrarily small positive constant. C denotes the number of colors reported.

Aggregate function	Colored objects in S	Query q	Ambient Space	Space	Query time	Theorem
weighted sum	points	interval	$\mathbb{R}^1$	$O(n \log n)$	$O(\log n + C)$	3
		rectangle	$\mathbb{R}^2$	$O(n^{1+\epsilon})$	$O(\log n + C)$	8
		hyper box	$\mathbb{R}^d$ ($d \geq 3$)	$O(n^{1+\epsilon})$	$O(\log n + C)$	9
bounding box	points	interval	$\mathbb{R}^1$	$O(n)$	$O(\log n + C)$	11
		rectangle	$\mathbb{R}^2$	$O(n \log^2 n)$	$O(\log n + C)$	14
		hyper box	$\mathbb{R}^d$ ($d \geq 3$)	$O(n^{1+\epsilon})$	$O(\log n + C)$	17, 18
nontrivial bounding box	points	quadrant	$\mathbb{R}^2$	$O(n \log n)$	$O(\log n + C \log n)$	25
		three sided rectangle	$\mathbb{R}^2$	$O(n \log^2 n)$	$O(\log^2 n + C \log n)$	27
weighted sum	intervals	point	$\mathbb{R}^1$	$O(n)$	$O(\log n + C)$	28
weighted sum	rectangles	point	$\mathbb{R}^2$	$O(n \log n)$	$O(\log^2 n + C \log n)$	29
				$O(n^{1+\epsilon})$	$O(\log n + C)$	30
weighted sum	orthogonal segments	rectangle	$\mathbb{R}^2$	$O(n^{1+\epsilon})$	$O(\log n + C)$	31

3. The Colored Weighted Sum Problem

Problem: Preprocess a set S of n colored points in $\mathbb{R}^d$, where the points additionally come with a real-valued weight $w(p) \geq 0$, into a data structure such that given a d-dimensional orthogonal query box $q = \Pi_{i=1}^{d}[a_i, b_i]$, the tuples $\langle\ c,\ s_c\ \rangle$ are reported where s_c is sum of the weights of points of color c in q.

We present a solution to the static d-dimensional orthogonal generalized weighted sum problem for $d \geq 2$ that takes $O(n^{1+\epsilon})$ space (for an arbitrarily small positive constant ϵ) and $O(\log n + C)$ time. For $d = 1$, our solution takes $O(n \log n)$ space and $O(\log n + C)$ time.

3.1. *Query composition technique*

The number of colors for a given problem defined on a set of n points, can range from 1 to n. We encode each color as an integer in the range $[1, n]$. This allows us to use colors as array indices. While answering a query for some of the problems in this paper, we may sometimes encounter the same color more than once while querying data structures built on disjoint sets. The count corresponding to these partial results need to be added up efficiently. We can do this by using an array, $A[1:n]$, of counts indexed by colors (encoded as integers) to keep track of the count of each distinct color that is found during a query. While executing a query, we also store the distinct colors found in a linked list. This way we can avoid reading the zero counts in A later. After the query, the counts can be output and A can be reset in time proportional to the output size by scanning the list.

3.2. *The solution for $\mathbb{R}^1$*

First, we consider the semi-infinite problem for $d = 1$. We need to preprocess a set S of n weighted, colored points in $\mathbb{R}^1$ (or the x-axis) into a data structure such that given a query interval $q = [a_1, \infty)$, the tuples $\langle\ c,\ s_c\ \rangle$ are reported where s_c is sum of the weights of color c in q.

For each color c, we sort the points in S by non-decreasing order of their x coordinates. For each point $p \in S$ of color c, let $pred(p)$ be its predecessor in the sorted order, with $pred(p) = -\infty$ for the leftmost point. We then map the point p to the point $p' = (p, pred(p))$ in $\mathbb{R}^2$ and associate with it the color c and weight $w(p')$ set to the cumulative weight of all the points of color c in S whose x-coordinate is greater than or equal to p. Let S' be the set of such points in $\mathbb{R}^2$. We preprocess the points in S' into a priority search tree [23] PST to support the query of reporting points in a quadrant. Given a query $q = [a_1, \infty)$, we map it to the quadrant $q' = [a_1, \infty) \times (-\infty, a_1)$ in $\mathbb{R}^2$ and query PST with q'. For each point p' retrieved, we report $(c', w(p'))$ where c' is the color of p'.

Lemma 1. *The query algorithm reports a pair (c, w) iff the total weight of the points of color c in $S \cap q$ is w. Moreover at most one such pair is reported for each color c.*

Proof.
$\Rightarrow$ Let the query algorithm report a pair (c, w). Then there exists a point $p' \in q'$ of color c and weight w. Hence there exists a point $p \in q$ of color c such that $pred(p) \in (-\infty, a_1)$ and the cumulative weight of all the points of color c in S whose x-coordinate is greater than or equal to p is equal to w.
$\Leftarrow$ Let the total weight of the points of color c in $S \cap q$ be w. Let p be the leftmost point of color c in q. Then there is a point $p' = (p, pred(p))$ of color c and weight w in S'. Since $p \in [a_1, \infty)$ and $pred(p) \in (-\infty, a_1)$, point p' is retrieved by the query algorithm and the pair (c, w) is reported.

To prove uniqueness, let the query algorithm report a pair (c, w_1) corresponding to a point $p'_1 = (p_1, pred(p_1))$ and another pair (c, w_2) corresponding to a point

$p_2' = (p_2, pred(p_2))$. Hence $p_1 \in q$, $p_2 \in q$, $pred(p_1) \in (-\infty, a_1)$ and $pred(p_2) \in (-\infty, a_1)$. Without loss of generality, let p_1 be to the left of p_2. Then, since $p_1 \in q$ and p_1 is of color c, $pred(p_2) \in q$. This contradicts the fact that $pred(p_2) \in (-\infty, a_1)$. Hence for any color c, at most one pair is reported. □

Theorem 2. *The colored weighted sum problem in $\mathbb{R}^1$ can be solved for semi-infinite queries using a structure of size $O(n)$ and query time $O(\log n + C)$.*

3.2.1. *Extending to finite ranges*

We store the points of S at the leaves of a balanced binary search tree T in non-decreasing order of their x coordinates. At each internal node v, we store an instance $DL(v)$ of the data structure of Theorem 2 built on $S(Left(v))$, the set of points stored in the leaves of the left subtree of v. Similarly we store another data structure $DR(v)$ built on $S(Right(v))$, the set of points stored in the leaves of the right subtree of v supporting queries of the form $(-\infty, b_1]$. To answer a query $q = [a_1, b_1]$ we search with a_1 and b_1 in T. This generates paths ℓ and r in T that possibly diverge at some non-leaf node v of T. We query $DL(v)$ (respectively $DR(v)$) with $[a_1, \infty)$ (resp., $(-\infty, b_1]$) to retrieve the partial results. The partial results are then composed using the technique of Section 3.1. We conclude:

Theorem 3. *The colored weighted sum problem in $\mathbb{R}^1$ can be solved for bounded queries (i.e. $[a_1, b_1]$) using a structure of size $O(n \log n)$ and query time $O(\log n + C)$.*

3.3. *Adding range restrictions*

In [13] a general technique was proposed to *add range restrictions* to colored (or generalized) reporting problems. In this section we show how to adapt the technique to add range restrictions to our current problem. *Adding range restrictions* shall be defined in detail in some time. The idea used here is similar to [13], except that when we combine solutions to two sub problems, instead of taking an union of colors reported, we need to add up the weights. Note that this is only possible if we decompose the problem in a way that the sub problems are defined on disjoint partitions of points.

Similar to [13], let $PR(q, S)$ denote the answer to a colored weighted sum problem PR with query object q and object set S. To add a range restriction to PR, we give each object p in S an additional parameter $k_p \in \mathbb{R}$. In the transformed searching problem, we only query objects in S that have their parameter in a given range.

3.3.1. *Adding a semi-infinite range restriction*

Let S be a set of n colored objects, and let $PR(q, S)$ be a colored weighted sum problem for S with query object q. To add a *semi-infinite range restriction*, we associate with each object p of S an additional parameter $k_p \in \mathbb{R}$. Now, let TPR

be the colored weighted sum problem that is obtained by adding a semi-infinite range restriction to PR, i.e., $TPR(q, [a, \infty), S) := PR(q, \{p \in S | a \leq k_p\})$.

Assume we have a data structure DS that stores the set S, such that colored weighted sum queries $PR(q, S)$ can be solved in $O(\log n + C)$ time. Let the size of DS be bounded by $O(n^{1+\epsilon})$, where ϵ is an arbitrarily small positive constant. Also, assume we have a data structure TDS for the set S, such that colored weighted sum queries $TPR(q, [a, \infty), S)$ can be solved in $O(\log n + C)$ time. Let the size of TDS be bounded by $O(n^w)$ for some constant $w > 1$. Extending the idea of [13], we can show how to construct a data structure that solves colored weighted sum queries $TPR(q, [a, \infty), S)$ in $O(\log n + C)$ time, using $O(n^{1+\epsilon})$ space, for an arbitrarily small positive constant ϵ.

Let $S = \{p_1, p_2, \ldots, p_n\}$, where $k_{p_1} \geq k_{p_2} \geq \ldots \geq k_{p_n}$. Let m be an arbitrary parameter with $1 \leq m \leq n$. We assume for simplicity that n/m is an integer. Let $S_j = \{p_1, p_2, \ldots, p_{jm}\}$ and $S'_j = \{p_{jm+1}, p_{jm+2}, \ldots, p_{(j+1)m}\}$ for $0 \leq j < n/m$.

The transformed data structure consists of the following. For each j with $0 \leq j < n/m$, there is a data structure DS_j (of type DS) storing S_j for solving generalized weighted sum queries of the form $PR(q, S_j)$ and a data structure TDS_j (of type TDS) storing S'_j for solving generalized weighted sum queries of the form $TPR(q, [a, \infty), S'_j)$.

To answer a query $TPR(q, [a, \infty), S)$, we do the following. Compute the index j such that $k_{p_{(j+1)m}} < a \leq k_{p_{jm}}$. Solve the query $PR(q, S_j)$ using DS_j, solve the query $TPR(q, [a, \infty), S'_j)$ using TDS_j, and output the union of the colors reported by these two queries. The correctness of the query algorithm is trivial to observe. Next we state a lemma which leads us to the main theorem.

Lemma 4. *The basic transformation results in a data structure of size* $O(n^{2+\epsilon}/m + nm^{w-1})$ *and answers the colored weighted sum queries in* $O(\log n + C)$ *time.*

Theorem 5. *Let* S, DS *and* TDS *be as defined above. Then, there exists a data structure that solves colored weighted sum queries* $TPR(q, [a, \infty), S)$

(1) *with a query time of* $O(\log n + C)$,
(2) *using* $O(n^{1+\epsilon})$ *space, for an arbitrarily small positive constant* ϵ.

Proof. First assume that $w > 2$, i.e., the data structure TDS uses more than quadratic space. Choosing $m = n^{1/w}$ in Lemma 4 gives a data structure for solving queries $TPR(q, [a, \infty), S)$, with a query time of $O(\log n + C)$ and space $O(n^2)$. By applying Lemma 4 repeatedly, we obtain, for each integer constant $a \geq 1$, a data structure of size $O(n^{1+\epsilon+1/a})$ that answers queries $TPR(q, [a, \infty), S)$ in $O(\log n + C)$ time. This claim follows by induction on a; in the inductive step from a to $a + 1$, we apply Lemma 4 with $m = n^{a/(a+1)}$. □

3.3.2. *Extending to a full range restriction*

Analogous to the result for colored reporting problems in [13], we can build a data structure $TPR(q, [a, b], S)$ that solves colored weighted sum queries. We store the points of S at the leaves of a balanced binary search tree, sorted in non-decreasing order of their parameter k_p. For each non-root node u of this tree, let S_u denote the subset of S that is stored at the leaves of u's subtree. If u is a left (resp., right) child, then we store at u, an instance of the data structure that solves colored weighted sum queries $TPR(q, [a, \infty), S_u)$ (resp., $TPR(q, (-\infty, b], S_u)$). While querying, we search the tree for values a and b. Let u be the node at which the search paths diverge. Let u_ℓ (resp., u_r) be the left (resp., right) child of u. Then we query the secondary structure stored in the left (resp., right) child of u with $TPR(q, [a, \infty), S_{u_\ell})$ (resp., $TPR(q, (-\infty, b], S_{u_r})$). For each color c, weights are retrieved at most once by the query on the secondary structure stored at u_ℓ and at most once by the query on the secondary structure stored at u_r. For any color that is in the answer for both the queries, the total weight of the points to be reported is the sum of the weights reported by the two queries for the same color; for any color which is the answer to exactly one of the queries, the weight for the color is reported as is. Other colors (having no points in q) are not reported at all. The fact that S_{u_ℓ} and S_{u_r} are disjoint sets, ensures that the counts are correct.

Corollary 6. *Let DS and TDS be as defined above. Then, there exists a data structure that solves colored weighted sum queries* $TPR(q, [a, b], S)$

(1) *with a query time of* $O(\log n + C)$,
(2) *using* $O(n^{1+\epsilon})$ *space, for an arbitrarily small positive constant* ϵ.

Theorem 5 and Corollary 6 imply that in order to solve the colored weighted sum problem TPR, it suffices to have (i) a data structure for PR with $O(\log n + C)$ query time and $O(n^{1+\epsilon})$ space, and (ii) a data structure for TPR with $O(\log n + C)$ query time and polynomial space.

3.4. *Colored weighted sum problem for* $d = 2$

In this section, we show how to solve the colored weighted sum problem for a query rectangle, $q = [a_1, b_1] \times [a_2, b_2]$.

We make use of Theorem 5 to solve the problem. Let DS in Theorem 5 be the data structure of Theorem 3 for solving the colored weighted sum problem for $d = 1$ and queries of the form $q' = [a_2, b_2]$. We need a data structure TDS to solve colored weighted sum queries $TPR(q', [a_1, \infty), S)$. Given n colored points in $\mathbb{R}^2$, we sort the points by their x-coordinates and rank the points in left to right order (ties broken arbitrarily) and store their x-coordinates in an auxiliary array AUX. We create data structures DA_i for $1 \leq i \leq n$. Each such data structure is an instance of the data structure of Theorem 3 for the 1-dimensional static colored weighted sum problem which takes $O(n \log n)$ space and supports queries in time $O(\log n + C)$. We

build data structure DA_i on the y-coordinates of the points p whose x-coordinates are at least $AUX[i]$. Given a query $TPR(q', [a_1, \infty), S)$, we first binary search in AUX with a_1 to determine the index i of the leftmost point whose x-coordinate is greater than or equal to a_1. Then we simply query DA_i with q'.

Lemma 7. *The colored weighted sum problem in $\mathbb{R}^2$ can be solved for query $TPR(q', [a_1, \infty), S)$ using $O(n^2 \log n)$ space and $O(\log n + C)$ query time.*

Proof. The correctness follows from the correctness of Theorem 3 and the fact that DA_i is really built on all points in S with x-coordinates in $[a_1, \infty)$. Since each data structure DA_i takes space $O(n \log n)$ and there are $O(n)$ of them, the total space taken is $O(n^2 \log n)$. The query time is $O(\log n + C)$, since it takes $O(\log n)$ time to determine the index i and $O(\log n + C)$ time to query DA_i. □

Now since we have both the structures DS and TDS, we can apply Theorem 5 with $w = 3$, and Corollary 6 to conclude:

Theorem 8. *The colored weighted sum problem in $\mathbb{R}^2$ for orthogonal rectangular queries can be solved using $O(n^{1+\epsilon})$ space and $O(\log n + C)$ query time.*

3.5. *Colored weighted sum problem for $d > 2$*

To solve the problem in dimension $d > 2$ we again make use of Theorem 5. Assume that as DS, we have the data structure to solve the problem PR in dimension $d-1$, which takes $O(n^{1+\epsilon})$ space and $O(\log n + C)$ time. As TDS, we create a structure similar to Lemma 7, by taking $O(n)$ instances of DS, which gives us a data structure with $O(n^{2+\epsilon})$ space and $O(\log n + C)$ query time. Now we apply Theorem 5, with $w = 3$ and Corollary 6 to conclude:

Theorem 9. *The colored weighted sum problem in $\mathbb{R}^d$ can be solved for orthogonal query box using $O(n^{1+\epsilon})$ space and $O(\log n + C)$ query time.*

4. The Colored Bounding Box Problem

Problem: Preprocess a set S of n colored points in $\mathbb{R}^d$ into a data structure such that given an orthogonal query box q, the tuples $\langle\ c, bb_c\ \rangle$ are reported where bb_c is the bounding box of all the points of color c which lie within q. If a color has a single point p inside q, then the bounding box of that color will be the point p which is reported as a degenerate box.

In subsection 4.1 a fully dynamic solution to this problem is presented on the real line. Then in subsection 4.2 a static solution to the problem is found on a plane. Next we give a solution to the static version of the problem in $\mathbb{R}^d$ for $d \geq 3$ that takes $O(n^{1+\epsilon})$ space (for an arbitrarily small positive constant ϵ) and $O(\log n + C)$ time.

4.1. *A solution in $\mathbb{R}^1$*

On the real line the problem reduces to finding, for each color c that has at least one point inside the query interval q, the interval spanned by all the points of color c lying inside q.

For each color c, sort all the points in S by non-decreasing order of their x-coordinates and build a balanced binary search tree T_c. For each point $p \in S$ of color c, let $pred(p)$ and $succ(p)$ be its predecessor and successor in the sorted order, with $pred(p) = -\infty$ for the leftmost point and $succ(p) = \infty$ for the rightmost point. Then each point p is mapped to a new point $p' = (p, pred(p))$ (and $p'' = (p, succ(p))$) in $\mathbb{R}^2$ and p' (and p'') is assigned the color of point p. Call this set of points in $\mathbb{R}^2$, S' (and S''). We build a dynamic priority search tree D' (and D'') [23] based on the points in S' (and S''). Given a query $q = [a_1, b_1]$, we map it to $q' = [a_1, b_1] \times (-\infty, a_1)$ (and $q'' = [a_1, b_1] \times (b_1, \infty)$) in $\mathbb{R}^2$ and query D' (and D'') with q' (and q''). If a point $p' = (p, pred(p))$ (or $p'' = (p, succ(p))$) gets reported and p' (or p'') has color c, then it can be inferred that among all the points of color c lying inside q, the leftmost point (or the rightmost point) is p. We state this formally in the following lemma.

Lemma 10. *The following statements are observed from the discussion above:*

(1) *A point $p' = (p, pred(p))$ having color c is reported by D' iff p happens to be the leftmost point among all the points of color c that lie inside the query interval q. Also for each color, exactly one point is reported by D'.*
(2) *A point $p'' = (p, succ(p))$ having color c is reported by D'' iff p happens to be the rightmost point among all the points of color c that lie inside the query interval q. Also for each color, exactly one point is reported by D''.*

Proof. Consider the first part of the lemma. Let a point $p' = (p, pred(p))$ of color c be reported. If p is not the leftmost point then $pred(p) \in [a_1, b_1]$. Since D' has reported point p', *pred(p)* should lie in the interval $(-\infty, a_1]$. A contradiction arises.

Let p be the leftmost point among all the points of color c that lie inside $q = [a_1, b_1]$. So, *pred(p)* should lie to the left of the query interval q, i.e., in the interval $(-\infty, a_1]$. Therefore, point p' is reported by D'. From the above discussion it is clear that exactly one point is reported for each color by D'. The second part of the lemma follows a symmetric argument. □

A dynamic priority search tree build on m points takes $O(m)$ space and answers three-sided rectangular queries in $O(\log m + k)$ time. The size of set S' and S'' is n and from the above lemma, it is clear that $k = C$.

Next we show how the solution can be made *dynamic*. Let r be the new point having color c which is to be *inserted*. First we insert r into T_c. Let r_p and r_s be the points in the leaf nodes to the immediate left and to the immediate right of r, respectively. Before the insertion of r, $r_p = pred(r_s)$ and $r_s = succ(r_p)$ were valid relations. After insertion of r, the following new relations arise : $r_p = pred(r)$,

$r = pred(r_s)$, $r = succ(r_p)$ and $r_s = succ(r)$. To represent these changes in our data structures we delete (r_s, r_p) from D' and delete (r_p, r_s) from D''. Then we insert (r, r_p) and (r_s, r) into D', and insert (r_p, r) and (r, r_s) into D''. The total time taken for handling these operations is $O(\log n)$. Deletions are symmetric to insertions. Therefore, insertions and deletions can be handled in $O(\log n)$ time.

Theorem 11. *The colored bounding box problem in $\mathbb{R}^1$ can be solved using a structure of size $O(n)$ and query time $O(\log n + C)$. Also, insertion or deletion of a point can be handled in $O(\log n)$ time.*

4.2. *Extending to $\mathbb{R}^2$*

In this subsection, the solution developed for one-dimensional case is combined with the technique of *persistence* to build a solution in $\mathbb{R}^2$.

For a general rectangle r=$[x, x'] \times [y, y']$, $[x, x']$ is defined to be the *x-projection* and $[y, y']$ is defined to be the *y-projection* of r. Now given a query q, each color c having at least one point in q is defined as a *valid color*. Reporting of the bounding box, BB_c, for each *valid color* c is done by first finding out the *x-projection* of BB_c and then the *y-projection* of BB_c.

First the *x-projection*'s of all the *valid colors* are found out. We do this by initially considering a query region of the form $q = [a_1, b_1] \times [a_2, \infty)$. Using the technique of persistence described in [8], a partially persistent version of the data structure of Lemma 11 is built, by treating the y-coordinate as time and inserting the points by non-increasing y-coordinate into an initially empty data structure. In fact only D' and D'' needs to be made persistent. The trees T_c are only needed to do updates efficiently in the current version. While querying they are not needed and can be discarded once the persistent version of D' and D'' have been built. To answer the query $q = [a_1, b_1] \times [a_2, \infty)$, we access the version corresponding to the smallest y-coordinate greater than or equal to a_2 and query it with $[a_1, b_1]$.

Lemma 12. *A set S of n colored points in $\mathbb{R}^2$ can be preprocessed into a data structure of size $O(n \log n)$, such that given a query $q = [a_1, b_1] \times [a_2, \infty)$, the x-projection of each valid color is reported in $O(\log n + C)$ time.*

Proof. The structures of D' and D'' of Lemma 11 have a constant in-degree and so the results of [8] apply. The correctness of the query algorithm follows from the fact that since the version accessed does not contain any point having y-coordinate less than a_2, querying D' and D'' with $q = [a_1, b_1] \times [a_2, \infty)$ is same as querying with $[a_1, b_1]$. The query time follows from Lemma 11. To build the persistent structure we do n insertions, each of which does $O(\log n)$ memory modifications. Thus the persistent structure uses $O(n \log n)$ space. □

To extend the solution in the above lemma for query boxes, $q = [a_1, b_1] \times [a_2, b_2]$, we follow an approach similar to that described previously : The points are stored

in a balanced binary tree sorted by y-coordinates. Each internal node v in the tree is associated with the auxiliary structure of Lemma 12 for answering queries of the form $[a_1, b_1]\times[a_2, \infty)$ (resp. $[a_1, b_1]\times(-\infty, b_2]$) on the points in the leaves of v's left (resp. right) subtree. To perform a query this tree is searched using the interval $[a_2, b_2]$. The query time remains $O(\log n + C)$ but the space increases by a logarithmic factor. The y-projections can be found in a similar way by switching the role of x and y. Next we summarize the above discussion in the following lemma which leads us to the final result.

Lemma 13. *A set S of n colored points in $\mathbb{R}^2$ can be preprocessed into a data structure of size $O(n \log^2 n)$, such that given a query $q = [a_1, b_1]\times[a_2, b_2]$, the x-projection (or y-projection) of each valid color is reported in $O(\log n + C)$ time.*

Theorem 14. *The colored bounding box problem in $\mathbb{R}^2$ can be solved using a structure of size $O(n \log^2 n)$ and query time $O(\log n + C)$.*

4.3. *A solution for $\mathbb{R}^3$*

In this section we consider points in a 3-dimensional space (XYZ) and solve the colored bounding box problem for a query cuboid $q = [a_1, b_1]\times[a_2, b_2]\times[a_3, b_3]$. The solution is obtained by applying Theorem 5. Let DS in Theorem 5 be a data structure of Theorem 14 for colored bounding box query in the XY-plane. A data structure TDS needs to be built for finding the x-projection's and the y-projection's of all the valid colors for queries of the form $TPR(q', [a_3, \infty), S)$, where $q' = [a_1, b_1]\times[a_2, b_2]$. Note that z-projection's *won't* be found out by TDS. Given n colored points in $\mathbb{R}^3$, we sort the points by their z-coordinates and rank the points in left to right order (ties broken arbitrarily) and store the z-coordinates in an auxiliary array AUX. Data structures DA_i for $1 \leq i \leq n$ are created. Each DA_i is an instance of the data structure of Theorem 14 for the 2-dimensional static colored bounding box problem which takes $O(n \log^2 n)$ space and answers queries in time $O(\log n + C)$. We build data structure DA_i on the x and y coordinates of the points in S whose z-coordinates are at least $AUX[i]$. Given a query $TPR(q', [a_3, \infty), S)$, we first binary search in AUX with a_3 to determine the index i of the leftmost point whose z-coordinate is greater than or equal to a_3. Then DA_i is simply queried with q'.

Lemma 15. *A set of n colored points in $\mathbb{R}^3$ can be preprocessed into a data structure of size $O(n^2 \log^4 n)$ such that given a query $TPR(q', [a_3, \infty), S)$ the x-projection and the y-projection of each valid color is reported in $O(\log n + C)$ time. Here $q' = [a_1, b_1]\times[a_2, b_2]$.*

Proof. The correctness follows from the correctness of the solution to the 2-dimensional problem in Theorem 14 and the fact that DA_i is really built on all points in S with z-coordinates in $[a_3, \infty)$. Since each data structure DA_i takes space $O(n \log^2 n)$ and there are $O(n)$ of them, the total space taken is $O(n^2 \log^4 n)$.

The query time is $O(\log n + C)$, since it takes $O(\log n)$ time to determine the index i and $O(\log n + C)$ time to query DA_i. □

Now since we have both the structures DS and TDS, we can apply Theorem 5 with $w = 3$, and Corollary 6 to reach the following lemma :

Lemma 16. *A set of n colored points in $\mathbb{R}^3$ can be preprocessed into a data structure of size $O(n^{1+\epsilon})$, for an arbitrarily small positive constant ϵ, such that given a query cuboid $q' = [a_1, b_1] \times [a_2, b_2] \times [a_3, b_3]$, the x-projection and the y-projection of each valid color is reported in $O(\log n + C)$ time where C is the output size.*

The z-projection can be found by switching its role with x or y. This leads us to the following result.

Theorem 17. *The colored bounding box problem in $\mathbb{R}^3$ can be solved using a structure of size $O(n^{1+\epsilon})$ and query time $O(\log n + C)$.*

4.4. *A solution for $d \geq 3$*

In order to solve the problem in dimension $d \geq 3$, first consider a $(d-1)$-dimensional subspace of $\mathbb{R}^d$. Call it Δ and let d_i be the dimension excluded from Δ. Let DS be a data structure to solve the problem PR in the dimensional subspace Δ which takes $O(n^{1+\epsilon})$ space and $O(\log n + C)$ time. As TDS, we create a structure similar to Lemma 15, by sorting points based on their values in dimension d_i and then taking $O(n)$ instances of DS, which gives us a data structure with $O(n^{2+\epsilon})$ space and $O(\log n + C)$ query time. Next another $(d-1)$-dimensional subspace, Δ' of $\mathbb{R}^d$ is considered, such that $\Delta' \neq \Delta$. Again DS is built to solve PR in subspace Δ'. TDS is also built on similar lines. Using the structures for these two distinct subspaces we can find out for each valid color, the projection of its bounding box in each dimension. Now we apply Theorem 5, with $w = 3$ and Corollary 6 to conclude:

Theorem 18. *The colored bounding box problem in $\mathbb{R}^d$ can be solved using a structure of size $O(n^{1+\epsilon})$ and query time $O(\log n + C)$.*

5. Nontrivial Colored Bounding Box Problem

Problem: Preprocess a set S of n colored points in $\mathbb{R}^2$ into a data structure such that given an orthogonal query box q, the tuples $\langle\ c,\ bb_c\ \rangle$ are reported where bb_c is the bounding box of all the points of color c which lie within q and there is an additional constraint that color c should have *at least two points lying within q*.

In section 5.1 the problem is solved for the case when the query region q is a quadrant. In section 5.2, a query of the form $[a_1, b_1] \times [a_2, \infty)$ is considered.

5.1. *Querying with quadrants in $\mathbb{R}^2$*

Given a set, S, of n colored points in $\mathbb{R}^2$, the nontrivial colored bounding box query is solved for a north-east quadrant defined by a query point $q = (a_1, a_2)$. The

north-east quadrant, $NE(q)$, is defined as the set of all points (x, y) such that $x \geq a_1$ and $y \geq a_2$.

The solution to this problem is found in two stages. In the first stage, all the colors c that have at least two points of color c in $NE(q)$ are found out. Then for each such color c, the bounding box of all the points of color c which lie inside q is reported.

The details of the first stage are stated next. Let S_c be the set of points of color c and M_c be the set of maximal points of S_c, i.e, $M_c \subseteq S_c$ consists of points which do not contain any other point of S_c in their north-east quadrant. Define $S'_c = S_c \setminus M_c$ and MM_c to be the set of maximal points of S'_c. Next we state a lemma.

Lemma 19. *There are at least two points of color c in $NE(q)$, if and only if, either a) at least two points of M_c and none of the points of MM_c lie in $NE(q)$, or b) at least one point of M_c and at least one point of MM_c lie in $NE(q)$.*

Proof. We classify set S_c into three categories : M_c, MM_c and $R_c = S'_c \setminus MM_c$. Let there be at least two points of color c in $NE(q)$. Two possibilities arise :

(1) If any of these points belong to R_c, then by definition of a maximal point, at least one point of MM_c and one point of M_c should also lie in the $NE(q)$. This satisfies condition b.
(2) If none of these points which lie in $NE(q)$ are from R_c, then they need to belong to M_c and MM_c only. None of the points belonging to M_c and all the points belonging to MM_c is an impossible condition (fundamental property of maximal points). The remaining valid conditions are, both M_c and MM_c having at least one point (condition b) or M_c having all the points (condition a).

The converse is quite trivial to prove. □

Based on the above lemma, two data structures are built. The first data structure, D_1, reports all those colors of S which *partially* satisfy condition (a) of the above lemma (Lemma 19). D_1 is built on the points in M_c, for each color c, such that given a query point q, we shall report all colors c which have at least two points of color c in $NE(q)$. Therefore, colors reported by D_1 might have some points of MM_c lying in $NE(q)$ but that will not concern us, as each color which gets reported will still be having at least two points in $NE(q)$.

The second one, D_2, is a data structure which is built on the points in MM_c, for each color c, so that given a query point q, the distinct colors of all the points lying in $NE(q)$ are reported. It must be noted that if a color c has at least one point of MM_c in $NE(q)$, then it will also have at least one point of M_c in $NE(q)$. So, indirectly D_2 reports all the colors that satisfy condition b) of Lemma 19. The answers obtained by querying D_1 and D_2 are combined to find out all the colors which have at least two points in $NE(q)$. A color will get reported at most twice (once by D_1 and once by D_2).

5.1.1. *Description of data structure D_2*

Define $S' = \bigcup MM_c$, for all colors c. Based on the points S' lying in the plane, a data structure D_2 is built on these points such that all the distinct colors of the points that lie in the north-east quadrant get reported.

For a point $p(p_1, p_2) \in MM_c$, denote by $Q_p = (-\infty, p_1] \times (-\infty, p_2] \subseteq \mathbb{R}^2$, the region within which query point q should lie for point p to lie in the $NE(q)$. Let $R(MM_c) = \bigcup_{p \in MM_c} Q_p$. R_c is the set of pairwise disjoint axis-parallel rectangles obtained by decomposing $R(MM_c)$. If a color c has at least one point of MM_c in the north-east quadrant of q, then q should lie inside exactly one of the rectangle in R_c. For each color c, rectangles R_c are built and a standard data structure for reporting all the rectangles containing a query point $q \in \mathbb{R}^2$ is built. For m axis-parallel rectangles in a plane, this data structure takes $O(m)$ space and given a query point q the query is reported in $O(\log m + k)$ time [5]. In [17] it is proved that $R(MM_c)$ built over points in MM_c can be decomposed into $O(|MM_c|)$ pairwise disjoint axis-parallel rectangles. So, the total number of rectangles obtained on decomposing $R(MM_c)$, for each color c, will be $O(n)$. Also, each color c is reported only once, i.e., $k = C$. Therefore, D_2 will take $O(n)$ space and $O(\log n + C)$ time.

Theorem 20. *For all colors c, let MM_c be the second layer of maximal set of points of color c in the plane. Then there exists a data structure D_2 of size $O(n)$ such that given a query point q, it reports the C distinct colors of the points that lie in northeast quadrant of q in $(\log n + C)$ time.*

Note: Incidentally, Theorem 20 can be used to solve the following problem: A set S of n colored points lie in a plane. Given a query quadrant q=$[a_1, \infty) \times [a_2, \infty)$, report the distinct colors of the points that lie in q. For each color find out its maximal points in the plane. Then apply the above theorem can be applied on the maximal points obained for each color to solve the problem. Unlike the previously known optimal solution for this problem [14], our solution is easier to implement.

5.1.2. *Description of data structure D_1*

Based on the set of points M_c, for all colors c, we build our data structure D_1, which will report all the colors c having at least two points of M_c in the $NE(q)$.

Fix a color c. Let the points in M_c be sorted in decreasing order based on their y-coordinates. With $p_i(x_i, y_i) \in M_c$, $\forall\ 1 \leq i \leq |M_c|$, we associate an axis-parallel rectangle $R(p_i) \subset \mathbb{R}^2$ which might be unbounded on a few sides. Define $R(p_1) = (-\infty, x_1) \times (-\infty, y_2)$; $R(p_i) = (x_{i-1}, x_i) \times (-\infty, y_{i+1}), \forall\ 2 \leq i \leq |M_c| - 1$ and $R(p_{|M_c|}) = \emptyset$. The union of $R(p_i), \forall 1 \leq i \leq |M_c|$, is denoted by $R(M_c)$ and all the $R(p_i)'s$ are disjoint to each other.

Lemma 21. *At least two points of M_c will lie in the NE quadrant of q if and only if the query point lies within $R(M_c)$.*

Proof. Suppose that there are at least two points of M_c which lie in the *NE* quadrant of $q(a_1, a_2)$. Without loss of generality, let the x-coordinate of q (i.e a_1) lie in the interval (x_{i-1}, x_i). If $a_2 \geq y_i$, then none of the points of M_c can lie in the $NE(q)$. If $y_{i+1} < a_2 \leq y_i$, then only p_i lies in the $NE(q)$. So, b must lie in the interval $(-\infty, y_{i+1}]$ for at least two points of M_c to get reported. Thus in this case q will have to lie within the region $R(p_i)$. As $R(M_c)$ is the union of $R(p_i), \forall$ $1 \leq i \leq |M_c|$, q has to lie within $R(M_c)$.

Conversely, suppose that $q(a_1, a_2)$ lies within $R(M_c)$ and let a_1 lie in the interval (x_{i-1}, x_i). If a_2 lies in the interval (y_{i+2}, y_{i+1}), then points p_i and p_{i+1} get reported. If a_2 lies in the interval (y_{i+3}, y_{i+2}), then points p_i, p_{i+1} and p_{i+2} get reported and so on. Therefore, at least two points of M_c will lie in the $NE(q)$. □

Now, we need to solve the standard point enclosure problem of reporting all the axis-parallel rectangles $R(p_i) \in M_c$, $\forall$ $1 \leq i \leq |M_c|$, for all colors c, which contain the query point $q(a_1, a_2)$. Note that for each color at most one rectangle will get reported. The total size of $R(p_i) \in M_c, \forall$ $1 \leq i \leq |M_c|$, for all colors c is $O(n)$. For m axis-parallel rectangles in a plane, the data structure which solves point enclosure problem takes $O(m)$ space and given a query point q the query is reported in $O(\log m + k)$ time [5]. Therefore, data structure D_1 takes $O(n)$ space and $O(\log n + C)$ time since $k = C$.

Theorem 22. *For all colors c, let M_c be a maximal set of points of color c in the plane. Then there exists a data structure D_1 of size $O(n)$ such that given a query point q, it reports C distinct colors, such that each reported color c has at least two points of color c in northeast quadrant of q, in $(\log n + C)$ time.*

Based on the two conditions stated in Lemma 19, data structures D_1 (Theorem 22) and D_2 (Theorem 20) have been built. By combining the results obtained by querying these two data structures, the following result is obtained.

Theorem 23. *Let S be a set of colored points in the plane. Then there exists a data structure of size $O(n)$ such that given a query point q, it reports C distinct colors, such that each reported color c has at least two points of color c in northeast quadrant of q, in $(\log n + C)$ time.*

5.1.3. *Finding the bounding box*

Let S_c be the set of points of color c and the query region q be a rectangle. Given that $|S_c \cap q| \geq 2$, the problem of the finding the bounding box (BB_c) of the points in $S_c \cap q$ is discussed in this section. Note that a quadrant query is a special case of a query rectangle.

Let S_c be the set of points of color c. Two data structures T^c_{xy} and T^c_{yx} are constructed based on the points in S_c. Given that $|S_c \cap q| \geq 2$, T^c_{xy} reports two points from $S_c \cap q$, one with the minimum y-coordinate and one with the maximum y-coordinate; T^c_{yx} reports two points from $S_c \cap q$, one with the minimum x-coordinate

and one with the maximum x-coordinate. The four values reported by these two data structures are used to find the bounding box of the points in $S_c \cap q$.

T^c_{xy} is actually a 2-dimensional range tree where the primary structure is built on the x-coordinates of the points in S_c and the associated structure is built on the y-coordinates of the points in S_c. The technique of fractional cascading [6] is used to build the associated structures of T^c_{xy}. T^c_{yx} is the same as T^c_{xy} except that the primary structure here is built on the y-coordinates of the points in S_c and the associated structure is built on the x-coordinates of the points in S_c. Given a query rectangle $q = [a_1, b_1] \times [a_2, b_2]$, the primary structure of T^c_{xy} is searched with a_1 and b_1 to determine set V of nodes whose canonical subsets together contain the points with x-coordinate in the range $[a_1, b_1]$. For each node $v \in V$, $A(v)$ is the set of points associated with the node v and is sorted based on the y-coordinates of the points. Let $L(v) \subseteq A(v)$ be the set of points which lie in the interval $[a_2, b_2]$. v_{min} and v_{max}, the points with the minimum and the maximum y-coordinate values in $L(v)$ are picked. Define $y_c = \min\{v_{min}, \forall v \in V\}$ and $y'_c = \max\{v_{max}, \forall v \in V\}$. In the same manner T^c_{yx} is queried and the variables x_c and x'_c are found out analogously. The bounding box of color c, BB_c, then turns out to be $[x_c, x'_c] \times [y_c, y'_c]$.

Theorem 24. *Let S_c be a set of n_c points of color c and the query region q be a rectangle. Given that $|S_c \cap q| \geq 2$, the data structure for finding the bounding box (BB_c) of the points in $S_c \cap q$ takes $O(n_c \log n_c)$ space and answers the query in $O(\log n_c)$ time.*

Putting together Theorem 23 and Theorem 24 we obtain the following result.

Theorem 25. *The nontrivial colored bounding box problem in $\mathbb{R}^2$ can be solved for a query quadrant (i.e. $[a_1, \infty) \times [a_2, \infty)$) using a structure of size $O(n \log n)$ and query time $O(\log n + C \log n)$.*

5.2. *Querying with three sided rectangle in $\mathbb{R}^2$*

Given a set, S, of n colored points in $\mathbb{R}^2$, the colored bounding box query is solved for the query range $q = [a_1, b_1] \times [a_2, \infty)$. As done before, the solution is found in two stages. In the first stage all the colors c which have at least two points inside q are reported. Then for each color c, the corresponding bounding box (BB_c) is found out.

The details of the first stage is discussed next. The points of S are stored in sorted order by x-coordinate at the leaves of a height balanced binary tree T. At each internal node v, an instance of the structure of Theorem 23 for NE-queries (resp., NW-queries) built on the points in the leaves of v's left (resp., right) subtree. Call them T_{ne} and T_{nw}. Let $X(v)$ denote the average of the x-coordinate in the rightmost leaf in v's left subtree and the x-coordinate in the leftmost leaf in v's right subtree; for a leaf v, we take $X(v)$ to be the x-coordinate of the point stored at v. Let $L(v)$ and $R(v)$ be the regions to the left and the right of the line $x = X(v)$.

Two more structures T_{vl} and T_{vr} are built at node v, which together report all the colors c which have point(s) in the intersection of q and $L(v)$, and also have point(s) in the intersection of q and $R(v)$. Fix a color c. For the points having color c in v's left subtree, M_l^c is the set of maximal points. Here a point p becomes a maximal point if there is no other point in the set lying in the NE-quadrant of p. Similarly, for the points having color c in v's right subtree, M_r^c is the set of maximal points. Here a point p becomes a maximal point if there is no other point that lies in the NW-quadrant of p. M_l^c and M_r^c are sorted in decreasing order of their y-coordinates. For every point $p(p_x, p_y) \in M_l^c$ (or M_r^c), p'_y is the y-coordinate of the next point to p in M_l^c (or M_r^c); and p'_x is the x-coordinate of the point in M_r^c (or M_l^c) which has the least y-coordinate among the points having their y-coordinate values greater than p_y. Now each point $p \in M_l^c$ in $\mathbb{R}^2$ is transformed to a point $p'(p_x, p_y, p'_x, p'_y)$ in $\mathbb{R}^4$. Let M'_{lc} be the set of transformed points. T_{vl} is a data structure which can handle 4-dimensional dominance query and is built over points in M'_{lc}, for all colors c. Similarly, we transform each point $p \in M_r^c$ to build T_{vr}.

To perform a query q, we do a binary search down T, using $[a_1, b_1]$, until either the search runs of T or a (highest) node v is reached such that $[a_1, b_1]$ intersects $X(v)$. In the former case, we stop. In the latter case, if v is a leaf then we stop. If v is a non-leaf, then we query the structures T_{ne} and T_{nw} at v using the NE-quadrant and the NW-quadrant derived from q (i.e, the quadrants with corners at (a_1, a_2) and (b_1, a_2), respectively). Structure T_{vl} is queried with $q_1 = [a_1, \infty) \times [a_2, \infty) \times (-\infty, b_1] \times (-\infty, a_2]$ and T_{vr} with $q_2 = (-\infty, b_1] \times [a_2, \infty) \times [a_1, \infty) \times (-\infty, a_2]$. The answers obtained are combined to find out all the colors c which have at least two points inside q.

Structure T_{ne} (and T_{nw}) reports all the colors which have at least two points in the NE-quadrant of $[a_1, a_2]$ (and NW-quadrant of $[b_1, a_2]$). All the colors which have at least one point in the NE-quadrant of $[a_1, a_2]$ and NW-quadrant of $[b_1, a_2]$ are reported by T_{vl} and T_{vr}. It must be observed that T_{vl} (and T_{vr}) report each color only once. Also, it is clear that no color is missed, although a single color might get reported a constant number of times.

A recent result by Afshani [1] solves 4-dimensional dominance queries using $O(n \log n)$ space and $O(\log^2 n + k)$ time. So, the total space occupied by our data structure will be $O(n \log^2 n)$ and the time to answer a query will be $O(\log^2 n + C)$. The result is summarized below.

Theorem 26. *Let S be a set of colored points in the plane. Then there exists a data structure of size $O(n \log^2 n)$ such that given a query $q = [a_1, b_1] \times [a_2, \infty)$, it reports C distinct colors, such that each reported color c has at least two points of color c inside q, in $(\log^2 n + C)$ time.*

In the case of a query quadrant once the appropriate colors were known, next the bounding boxes were found out. The same procedure is followed here. This leads to the following theorem.

Theorem 27. *The nontrivial colored bounding box problem in* $\mathbb{R}^2$ *can be solved for a three sided rectangle query (i.e.* $[a_1, b_1] \times [a_2, \infty)$*) using a structure of size* $O(n \log^2 n)$ *and query time* $O(\log^2 n + C \log n)$.

6. Colored Point Enclosure Weighted Sum for $d = 1$

Problem: Preprocess a set S of n colored intervals on the real line, where the intervals additionally come with a real-valued weight, such that given a query point q on the real line, the tuples $\langle\ c,\ s_c\ \rangle$ are reported where s_c is the sum of the weights of intervals of color c stabbed by q.

Consider a color c and let S_c be the set of intervals of color c, s.t., $|S_c| = n_c$. Let the list of distinct interval endpoints of S_c be sorted from left to right. These endpoints induce partitions on the real line and the regions in this partitioning shall be called "elementary intervals". Let I_c be the set of these intervals. With each interval $i \in I_c$, we shall maintain an attribute w_i which holds the sum of the weights of intervals in S_c which intersect i. Based on the elementary intervals in I_c, for all colors c, an interval tree, IT, is built. Given a query point q, a standard search on the interval tree, IT, is carried out and all the intervals stabbing q are found out and the w_i attribute associated with each of them is reported. Note that if a color c has at least one interval of S_c stabbed by q, then exactly one elementary interval in I_c will be reported and the appropriate weight is reported.

Theorem 28. *The colored point enclosure weighted sum problem can be solved in* $\mathbb{R}^1$ *using a structure of size* $O(n)$ *and query time* $O(\log n + C)$.

Proof. The correctness of the query algorithm is quite trivial to observe. Now, if a color c has n_c points, then the size of the set I_c will be $O(n_c)$. Therefore, the total size of all the intervals in set I_c, for all colors c, is $O(n)$. An interval tree built on n intervals on the real line uses $O(n)$ space and given a query point q on the real line, reports the intervals stabbed by q, in $O(\log n + k)$ time, where k is the number of intervals stabbed. Our structure IT will report each color at most once. Therefore, the size of IT will be $O(n)$ and the query time will be $O(\log n + C)$, where C is the number of colors which have at least one interval stabbed by q. □

7. Colored Point Enclosure Weighted Sum for $d = 2$

Problem: Preprocess a set S of n colored orthogonal rectangles in the plane, where the rectangles additionally come with a real-valued weight, such that given a query point q, the tuples $\langle\ c,\ s_c\ \rangle$ are reported where s_c is the sum of the weights of rectangles of color c stabbed by q.

A rectangle $r \in S$ is represented as $r{=}i_x{\times}i_y$, where i_x and i_y are the projection of r on the x-axis and y-axis, respectively. A segment tree T is built based on the projections of rectangles in S on the x-axis. At a particular internal node v of T, if an interval i_x of some rectangle r gets assigned, then we *associate* i_y with node v.

Based on the intervals *associated* with node v, we build an auxiliary data structure of Theorem 28.

Given a query point $q = (a, b)$, we start by querying T with a. If a is not in the range of the root, then stop; else proceed. Let V be the set of nodes of T visited by querying with a. At each node $v \in V$, we perform a secondary query on the auxiliary data structure built at v by querying it with b. For each reported color c, the weight obtained from each relevant secondary query is added up.

Theorem 29. *The colored point enclosure weighted sum problem can be solved in* $I\!R^2$ *using a structure of size* $O(n \log n)$ *and query time* $O(\log^2 n + C \log n)$.

Proof. Consider an orthogonal rectangle $r{=}i_x{\times}i_y$ of set S having color c and say a query point $q{=}(a, b)$. Let $q \in r$. When the primary structure of T is queried with a, then by the property of a segment tree the node v containing interval i_x will be selected. Querying the auxiliary structure at v, the weight of color c reported at this node will contain i_y as well (from Theorem 28). Hence, no rectangle stabbed by q is missed out. Similarly, all rectangles not stabbed by q are also discarded.

A segment tree built on n intervals takes $O(n \log n)$ space. Also, the auxiliary structure built at each internal node v takes up linear space (from Theorem 28). Hence, the total space complexity of the structure is $O(n \log n)$.

Given a query point q, $O(\log n)$ nodes of T are selected. The time taken to query the auxiliary structure at each node of T is $O(\log n + k)$, where k is the number of colors at that node having at least one interval stabbed by b. Also, a color c can get reported at $O(\log n)$ nodes. Therefore, the query time is $O(\log^2 n + C \log n)$. □

In the above solution there is a penalty of $O(\log n)$ associated with each color that is reported. We can overcome this limitation (at the cost of increasing the space) by reducing the point enclosure problem in $\mathbb{R}^2$ to a range search problem in $\mathbb{R}^4$. Each rectangle $r([x_1, x_2] \times [y_1, y_2]) \in S$, is reduced to a 4-dimensional point (x_1, x_2, y_1, y_2) and the query point $q(a, b) \in \mathbb{R}^2$ is reduced to $(-\infty, a]{\times}[a, \infty){\times}(-\infty, b]{\times}[b, \infty)$. The original problem has been transformed into a "colored weighted sum problem" in $\mathbb{R}^4$.

Theorem 30. *The colored point enclosure weighted sum problem can be solved in* $I\!R^2$ *using a structure of size* $O(n^{1+\epsilon})$ *and query time* $O(\log n + C)$.

8. Colored Segment Intersection Weighted Sum

Problem: Preprocess a set S of n colored orthogonal line segments in the plane, where the segments additionally come with a real-valued weight, such that given a query orthogonal rectangle q, the tuples $\langle\ c,\ s_c\ \rangle$ are reported where s_c is the sum of the weights of segments of color c stabbed by q.

Consider one of the vertical segments, say s. Let it's lower end point be (s_x, s_l) and the upper end point be (s_x, s_u). Given a query rectangle, $q{=}\ [a_1, b_1] \times [a_2, b_2]$,

s will intersect with q, if the following conditions are satisfied: 1) $a_1 \leq s_x \leq b_1$, 2) $s_u \geq a_2$ and 3) $s_l \leq b_2$. Each vertical segment having weight w in $\mathbb{R}^2$ is transformed into a point in $\mathbb{R}^3$, such that the segment s is mapped to (s_x, s_l, s_u) and weight w is associated. Based on these transformed points, we shall build a data structure D of Theorem 9. Thus, for all colors having at least one vertical segment intersecting q, D will report the sum of the weight of vertical segments of these colors intersecting q. We shall build a similar data structure to handle horizontal segments. The two solutions can be trivially merged without affecting output sensitivity.

Theorem 31. *The colored segment intersection weighted sum problem can be solved in $\mathbb{R}^2$ using a structure of size $O(n^{1+\epsilon})$ and query time $O(\log n + C)$.*

9. Conclusions and Open Problems

We considered several range-aggregate queries for colored objects and provided efficient solutions for them. Our techniques have been based mainly on persistent data structures, geometric transformation and on the concept of 'adding range restrictions'.

Several open problems remain. For some of the problems, the space occupied by the structures is $O(n^{1+\epsilon})$. It would be interesting to see if it can be reduced to $O(n \log^c n)$ (for $c > 0$) while keeping the query time poly logarithmic. It would be desirable to improve the query time of some of the solutions from $O(\log n + C \log n)$ or $O(\log^2 n + C \log n)$ to $O(polylog(n) + C)$. Considering non-trivial bounding box problem for cases wherein each color has some c objects ($c > 2$) intersecting the query region will be a challenging open problem. [9] is a work closely related to this open problem. Finally, extensions to problems involving multiple categorical attribute per object would be an interesting future work.

References

[1] Peyman Afshani, On Dominance Reporting in 3D, *Proceedings of the 16th annual European symposium on Algorithms*, 2008, Karlsruhe, Germany, pp. 41–51.

[2] 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.

[3] S. Acharya, P. Gibbons, and V. Poosala. Congressional Samples for Approximate Answering of Group By Queries, *Proceedings SIGMOD (2000)*, pp. 487–497.

[4] P. Bozanis, N. Kitsios, C. Makris, and A. Tsakalidis. New Upper Bounds for Generalized Intersection Searching Problems. *Proceedings 22nd ICALP, Lecture Notes in Computer Science, Vol. 944, Springer-Verlag, Berlin*, 1995, pp. 464–475.

[5] B. M. Chazelle, Filtering search: A new approach to query answering, *SIAM Journal of Computing*, 15, 703–724, 1986.

[6] B. M. Chazelle and L.J. Guibas. Fractional cascading I. A data structure technique, *Algorithmica*, 1 (1986), pp. 133–162.

[7] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein. *Introduction to Algorithms*, 2nd edition, MIT Press, 2000.

[8] J.R. Driscoll, N. Sarnak, D.D. Sleator, and R.E. Tarjan. Making data structures persistent. *Journal of Computer and System Sciences*, 38:86–124, 1989.

[9] Mark de Berg, Herman J. Haverkort. Significant-Presence Range Queries in Categorical Data. *In Proceedings of WADS '2003*, pp. 462–473

[10] J. Gray, S. Chaudhuri, A. Bosworth, A. Layman, D. Recichart, M. VenkatRao, F. Pellow, H. Pirahesh. Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Totals. *Data Mining and Knowledge Discovery*, **1(1)**, pp. 29–53.

[11] P. Gupta, R. Janardan, and M. Smid. Further results on generalized intersection searching problems: counting, reporting, and dynamization.*Journal of Algorithms*, **19** (1995), pp. 282–317.

[12] P. Gupta, R. Janardan, and M. Smid. Algorithms for generalized halfspace range searching and other intersection searching problems. *Computational Geometry: Theory and Applications*, **5** (1996), pp. 321–340.

[13] P. Gupta, R. Janardan, and M. Smid. A technique for adding range restrictions to generalized searching problems. *Information Processing Letters*, **64** (1997), pp. 263–269.

[14] P. Gupta, R. Janardan and M. Smid. Computational geometry: Generalized intersection searching, Chapter 64, *Handbook of Data Structures and Applications, D. Mehta and S. Sahni (editors)*, Chapman & Hall/CRC, Boca Raton, FL, 64-1–64-17, 2005.

[15] P. Gupta. Range-Aggregate Query Problems Involving Geometric Aggregation Operations. *Nordic journal of Computing*. 13(4): 294-308, 2006.

[16] J. L. Bentley, H. T. Kung, M. Schkolnick, and C. D. Thompson. On the average number of maxima in a set of vectors and applications. *Journal of the ACM*, 25(4):536-543, October 1978.

[17] H. Kaplan, N. Rubin, M. Sharir and E. Verbin. Counting colors in boxes. *Proceedings of Symposium on Discrete Algorithms (SODA)*, 785-794, 2007.

[18] R. Janardan and M. Lopez. Generalized intersection searching problems. *International Journal on Computational Geometry & Applications*, **3** (1993), pp. 39–69.

[19] R. Janardan, P. Gupta, Y. Kumar and M. Smid. Data Structures for Range-Aggregate Extent Queries. *20th Canadian Conference on Computational Geometry (CCCG 2008)*, pages 7-10.

[20] Y.K. Lai, C.K. Poon, and B. Shi. Approximate colored range queries, *Proceedings, 16th International Symposium on Algorithms and Computation, (ISAAC 05), Springer Verlag Lecture Notes on Computer Science, Vol. 3827*, 2005, 360–369.

[21] D. Krizanc, P. Morin, and M. Smid. Range Mode and Range Median Queries on Lists and Trees. *Nordic Journal of Computing*, 12(1): 1-17, 2005.

[22] C. Li, K.C. Chang, and I. Ilyas. Supporting Ad-hoc Ranking Aggregates. *Proceedings, ACM SIGMOD International Conference on Management of Data*, 61–72, 2006.

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

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

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