Data Structures for Range Aggregation by Categories

Saladi Rahul* Prosenjit Gupta† K. S. Rajan*

Abstract

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 R^d, $d \geq 1$ such that given a query orthogonal range q, we can report efficiently for each distinct color c of the points in $S \cap q$, the tuple $< c, \mathcal{F}(c) >$ where $\mathcal{F}(c)$ is a function (e.g. weighted sum, bounding box etc.) of the objects of color c in q.

1 Introduction

In many applications like on-line analytical processing (OLAP), geographic information systems (GIS) and information retrieval (IR), aggregation plays an important role in summarizing information [9] and hence large number of algorithms and storage schemes have been proposed to support such queries. In *range-aggregate query* problems [9] 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 all of them as in a range reporting query.

In this work, we consider instances of a general class of problems defined as follows: Preprocess a set S of possibly weighted colored geometric objects in R^d, $d \geq 1$, such that given a query orthogonal range q, we can report efficiently for each distinct color c of the points in $S \cap q$, the tuple $< c, \mathcal{F}(c) >$ where $\mathcal{F}(c)$ is a function of the objects of color c in q. If S is a set of colored points and $\mathcal{F}(c) = NULL$, the unweighted variant of the problem is the generalized orthogonal range reporting problem [5].

Lai et al. [6] studied this class of problems for approximate queries for functions like *min*, *max*, *sum*, *count*, *report* and *heavy*. Special cases were studied in [3, 1]. These problems have been studied in the database community as "*GROUP-BY*" queries, a class of common basic operations in databases applied to the categorical attributes [8].

*Lab for Spatial Informatics, International Institute of Information Technology, Gachibowli, Hyderabad, Andhra Pradesh 500 032, India. Email: saladi.rahul@gmail.com, rajan@iiit.ac.in.

†Mentor Graphics, Hyderabad, Andhra Pradesh 500 082, India. Email: prosenjit_gupta@acm.org.

2 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 query box q in $\mathbb{R}^d$, we can report efficiently for each distinct color c of the points in q, the tuple $< c, s_c >$ where s_c is the sum of the weights of the points of color c in q.

The generalized *type*-2 range counting problem is a special case of the above problem where each point has unit weight. For the 1-dimensional static *type*-2 range counting problem, 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 [3]. The space bound was improved to $O(n)$ in [1]. 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 [1]. For the 1-dimensional static *type*-2 point enclosure counting problem, a solution that takes $O(n)$ space and $O(\log n + C)$ query time was given in [3]. To the best of our knowledge, no other results are known for these problems.

2.1 The solution for $d = 1$

Consider the semi-infinite problem. For each color c, we sort the points in S by nondecreasing 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 [7] PST to support the query of reporting points in a quadrant. Given a query $q = [a, \infty)$, we map it to the quadrant $q' = [a, \infty) \times (-\infty, a)$ 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'.

Theorem 1 *A set, $\mathcal{S}$, of n colored weighted points in $I\!R^1$ can be preprocessed into a data structure of size $O(n)$ such that for any query range $q = [a, \infty)$, pairs (c, s_c) where s_c is the sum of the weights of the points of color c in q, can be reported in $O(\log n + C)$ time, where C is the number of distinct colors of points in q.*

To extend the solution to finite ranges, we store the points of $\mathcal{S}$ 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 1 built on $S(\mathit{Left}(v))$, the set of points stored in the leaves of of the left subtree of v. Similarly we store another data structure $DR(v)$ built on $S(\mathit{Right}(v))$, the set of points stored in the leaves of the right subtree of v supporting queries of the form $(-\infty, b]$. To answer a query $q = [a, b]$ we search with a and b in T. This generates paths ℓ and r in T that possibly diverge at some non-leaf node u of T. We query $DL(v)$ (respectively $DR(v)$) with $[a, \infty)$ (respectively $(-\infty, b]$) to retrieve the partial results, which are then composed.

Theorem 2 *A set, $\mathcal{S}$, of n colored weighted points in $I\!R^1$ can be preprocessed into a data structure of size $O(n \log n)$ such that for any query range $q = [a, b]$, pairs (c, s_c) where s_c is the sum of the weights of the points of color c in q, can be reported in $O(\log n + C)$ time, where C is the number of distinct colors of points in q.*

2.2 Adding range Restrictions

In [4] a general technique was proposed to add range restrictions to generalized reporting problems. Here we adapt the technique to add range restrictions to colored range-aggregate. The idea is similar, except that when we combine solutions to two subproblems, instead of taking a 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 subproblems are defined on disjoint partitions of points. To keep our solution output-sensitive, we also need to make sure that we try to add non-zero weights only, since we must report only non-zero weights in the answer.

Similar to [4], let $PR(q, S)$ denote the answer to a generalized weighted sum problem PR with query object q and object set S. Let TPR be the generalized weighted sum problem that is obtained by adding a half-infinite range restriction to PR.

Theorem 3 *Let DS be a data structure that stores a set S of n colored objects, such that generalized 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. Let TDS be a data structure for the set S, such that generalized 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 $w > 1$.*

There exists a data structure that solves generalized queries $TPR(q, [a : \infty), S)$, with a query time of $O(\log n + C)$ using $O(n^{1+\epsilon})$ space, for an arbitrarily small positive constant ϵ.

Corollary 2.1 *Let DS and TDS be as defined in Theorem 3. There exists a data structure that solves generalized weighted sum queries $TPR(q, [a : b], S)$, with a query time of $O(\log n + C)$, using $O(n^{1+\epsilon})$ space, for an arbitrarily small positive constant ϵ.*

To solve the problem for $d = 2$, let DS in Theorem 3 be the data structure of Theorem 2. We need a data structure TDS to solve generalized weighted sum queries $TPR(q, [a : \infty), S)$.

Lemma 4 *A set of n colored weighted points in $I\!R^2$ can be preprocessed into a data structure of size $O(n^2)$ such that a generalized weighted sum query $TPR(q, [a : \infty), S)$ can be answered in $O(\log n + C)$ time where C is the output size.*

Theorem 5 *A set of n colored weighted points in $I\!R^2$ 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 rectangle $[a, b] \times [c, d]$, a generalized weighted sum query can be answered in $O(\log n + C)$ time where C is the output size.*

To solve the problem in dimension $d > 2$, 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 the one for Lemma 4, 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 3, with $w = 3$ and Corollary 2.1.

Theorem 6 *A set of n colored weighted points in $I\!R^d$ 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 d-dimensional orthogonal box, a generalized weighted sum query can be answered in $O(\log n + C)$ time where C is the output size.*

3 Point enclosure counting for $d = 1$

Problem: Preprocess a set S of n colored intervals on the x-axis, such that given a query point q, we can report for each color c such that there is an interval of color c

stabbed by q, the number of intervals of color c stabbed by q.

Consider a color c and let S_c be the set of n_c intervals of color c. Let $p_1, p_2, \ldots, p_m$ be the list of distinct interval endpoints, sorted from left to right. These endpoints induce partitions on the real line and the regions in this partitioning are called 'elementary intervals'. Therefore, the elementary intervals, I_c, from left to right are: $(-\infty, p_1), [p_1, p_1], (p_1, p_2), [p_2, p_2], \ldots, (p_{m-1}, p_m), [p_m, p_m]$, (p_m, ∞). With each interval $i \in I_c$, we shall maintain the count of the number of intervals in S_c which have an overlap with i. With the elementary intervals in I_c, for all colors c, we build an interval tree, IT. Given a query point q, we search IT, and report the counts associated with intervals stabbed by q.

Theorem 7 *A set of n colored intervals on the x-axis can be preprocessed into a data structure of size $O(n)$, such that given a query point q, we can report in $O(\log n + C)$ time, for every color c with at least one interval of its color stabbed by q, the number of intervals of color c which are stabbed by q.*

4 Point enclosure counting for $d = 2$

Problem: Preprocess a set S of n colored rectangles in the plane, such that given a query point q, we can report for each color c such that there is a rectangle of color c stabbed by q, the number of rectangles of color c stabbed by q.

A segment tree T is created based on the distinct x-coordinates of the vertical sides of the rectangles in S. Consider a rectangle, $r = [x_1, x_2] \times [y_1, y_2]$, of S. Let v be a node of T such that the range of v is contained in $[x_1, x_2]$, but the range of v's parent is not. Then the interval $[y_1, y_2]$ is *associated* with node v. At each node v, using the intervals associated with v we build an instance of the data structure of Theorem 7. Given a query point $q = (a, b)$, we search in T for a and query the auxillary structure of each node v visited, with b. For each reported color c, the count obtained from each node is added up. We can obtain an alternative solution by reducing the point enclosure problem in $\mathbb{R}^2$ to a range search problem in $\mathbb{R}^4$.

Theorem 8 *A set of n colored rectangles in the plane can be preprocessed into a data structure of size $O(n \log n)$ (resp. $O(n^{1+\epsilon})$) such that given a query point q, for every color c with at least one rectangle of its color stabbed by q, the number of rectangles of color c which are stabbed by q can be reported in $O(\log^2 n + C \log n)$ (resp. $O(\log n + C)$) time.*

5 Segment Intersection Counting

Problem: Preprocess a set S of colored orthogonal segments in $\mathbb{R}^2$ into a data structure such that given a query orthogonal rectangle q , we can report for each color c such that there is at least one line segment of color c intersected by q, the number of such segments of color c intersected by q.

Consider one of the vertical segments, say L. Its lower end point is (x, y_l) and the upper end point is (x, y_u). Given a query rectangle, $q = [a, b] \times [c, d]$, L will intersect with q, if the following conditions are satisfied: 1) $a \leq x \leq b$, 2) $y_u \geq c$ and 3) $y_l \leq d$. Each vertical segment in $\mathbb{R}^2$ is transformed into a point in $\mathbb{R}^3$, such that the segment L is mapped to (x, y_l, y_u). Using these transformed points, we build an instance D of the data structure of Theorem 6. Thus, for all colors having at least one vertical segment intersecting q, D will report the number of vertical segments of these colors intersecting q. We build a similar data structure to handle horizontal segments.

Theorem 9 *A set S of colored orthogonal line segments in $\mathbb{R}^2$ can be preprocessed into a data structure of size $O(n^{1+\epsilon})$ such that given a query orthogonal rectangle q , we can report in $O(\log n + C)$ time, for every color c such that there is a line segment of color c intersected by q, the number of segments of color c intersected by q.*

6 The Colored Bounding Box Problem

Problem: Preprocess a set S of n colored points in $\mathbb{R}^d$ such that given an orthogonal query box q, for every color c having *at least one point* in q, report the bounding box of all points of color c inside 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 rectangle.

First let us consider the case $d = 1$. For each color c, sort all the points in S by non-decreasing order of their x-coordinates and build a balanced binary 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)* = -∞ for the leftmost point and *succ(p)* = ∞ for the rightmost point. Then each point p is mapped to a new point $p' = (p, pred(p))$ (resp. $p'' = (p, succ(p))$) in $\mathbb{R}^2$, which is assigned the color of point p. Call these set of new points in $\mathbb{R}^2$, S' (resp. S''). We build a dynamic priority search tree D_1 (resp. D_2) based on the points in S' (resp. S''). Given a query $q = [x, x']$, we map it to $q' = [x, x'] \times (-\infty, x)$ (resp. $q'' = [x, x'] \times [x', \infty)$) in $\mathbb{R}^2$ and query D_1 (resp. D_2) with q' (resp. q'').

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. We delete (r_s, r_p) from D_1 and delete (r_p, r_s) from D_2. Then we insert (r, r_p) and (r_s, r) into D_1, and insert (r_p, r) and (r, r_s) into D_2. The deletion process is symmetric. The total time taken for handling these operations is $O(\log n)$.

Lemma 10 *A set S of n colored points in $I\!R^1$ can be preprocessed into a data structure of size $O(n)$, such that given a query $q = [x, x']$, a generalized bounding box query can be answered in $O(\log n + C)$ time. Also, insertion of a point into S or deletion of a point from S can be handled in $O(\log n)$ time.*

Now consider $d = 2$. Given a query q, we denote a color c as *valid* iff at least one point in q is of color c. Given a query q, the 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 query region of the form $q = [x, x'] \times [y, \infty)$ is considered. Then the *x-projection*'s of all the *valid colors* are found out as follows : Using the technique of persistence described in [2], a partially persistent version of the data structure of Lemma 10 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_1 and D_2 (and not T_c) needs to be made persistent. To answer the query $q = [a, b] \times [c, \infty)$, we access the version corresponding to the smallest y-coordinate greater than or equal to c and query it with $[a, b]$. We can extend the solution to query boxes $q = [a, b] \times [c, d]$ with a $O(\log n)$ overhead on space. The y-projections can be similarly found.

Theorem 11 *A set S of n colored points in $I\!R^2$ can be preprocessed into a data structure of size $O(n \log^2 n)$, such that given a query $q = [a, b] \times [c, d]$, a generalized bounding box query can be answered in $O(\log n + C)$ time.*

Finally, let us extend the solution to $d \geq 3$. Let DS in Theorem 3 be a data structure of Theorem 11 for solving the generalized 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 colours for queries of the form $TPR(q, [z, \infty), S)$. Given n colored points in $\mathbb{R}^3$, we sort the points by their z-coordinates and store the z-coordinates in an auxillary array AUX. Data structures DA_i for $1 \leq i \leq n$ are created. Each such data structure is an instance of the data structure of Theorem 11 for the 2-dimensional static generalized 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 $TPR(q, [z, \infty), S)$, we first binary search in AUX with z to determine the index i of the leftmost point whose z-coordinate is greater than or equal to z. Then DA_i is simply queried with q.

Lemma 12 *A set of n colored points in $I\!R^3$ can be preprocessed into a data structure of size $O(n^2 \log^4 n)$ such that given a query $TPR(q, [z, \infty), S)$ the x-projection's and the y-projection's of all the valid colors can be found in $O(\log n + C)$ time.*

Applying Theorem 3, with $w = 3$ and Corollary 2.1, we can find the x-projection's and the y-projection's of all the valid colors in $O(\log n + C)$ time, using $O(n^{1+\epsilon})$ space. Similarly, we can find the x-projections and z-projections of all valid colors to solve the problem for $d = 3$. A similar technique can be applied for extending the solution to $d > 3$.

Theorem 13 *A set of n colored points in $I\!R^d$ 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 d-dimensional orthogonal box, the generalized bounding box query can be answered in $O(\log n + C)$ time where C is the output size.*

References

[1] 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.

[2] 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.

[3] 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.

[4] 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.

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

[6] 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.

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

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

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