Efficient Rectangle Indexing Algorithms Based on Point Dominance*

Peter Revesz
Department of Computer Science and Engineering
University of Nebraska-Lincoln, Lincoln, NE 68588, USA

Abstract

An approximate count of the number of (1) k-dimensional rectangles that contain, overlap or are within a query rectangle Q, and (2) linearly moving points that are to the left of a moving query point Q on the x-axis at time t, can be found in (poly)-logarithmic time in the number of rectangles or moving points.

1 Introduction

Let S be a set of k-dimensional rectilinear rectangles, that is, rectangles with sides parallel to the axes, P be a k-dimensional point, and Q be a k-dimensional rectilinear rectangle. Consider the following problems that ask to find the:

Stabbing: Number of rectangles in S that contain P.

Contain: Number of rectangles in S that contain Q.

Overlap: Number of rectangles in S that overlap Q.

Within: Number of rectangles in S that are within Q.

Alternatively, let S be a set of linearly moving points on the x-axis, let t be a time instance, and Q be a moving point, and consider the problem that asks to find the:

Count: Number of points in S to the left of Q at time t.

The above five problems can be reduced to **Dominance**, which for a set S of points and a point P asks to find the:

Dominance: Number of points in S dominated by P.

where *point dominance* is defined as follows:

*This research was supported in part by USA NSF grant EIA-0091530 and a NASA Nebraska Space and EPSCoR grant. Author's email:`revesz@cse.unl.edu`

Definition 1 Point $A = (a_1, \ldots, a_k)$ dominates point $B = (b_1, \ldots, b_k)$, written as $A \succ B$, if and only if $b_i \leq a_i$ for $1 \leq i \leq k$.

Using an ECDF-tree [1] the dominance problem can be solved in logarithmic time in the worst case. The ECDF-tree is a static data structure that does not allow updates; however, it can be extended to an ECDF-B-tree which performs both querying and updates efficiently, that is:

Theorem 1 [Zhang et al. [7]] For any fixed constant size page capacity B, the dominance problem can be solved using an $O(n \log^{k-1} n)$ size ECDF-B-tree in $O(\log^k n)$ time. Further, the ECDF-B-tree allows a sequence of updates in $O(\log^k n)$ amortized time. ■

Main results: The **Stabbing**, **Contain**, **Overlap**, and **Within** problems can be solved approximately in $O(n \log^{k-1} n)$ space and $O(\log^k n)$ time (**Theorem 3**). The **Count** problem can be solved approximately in $O(\log n)$ time (**Theorem 5**).

2 Reductions of the Rectangle Problems

In the following, let $A = (a_1, \ldots, a_k)$, $B = (b_1, \ldots, b_k)$, $C = (c_1, \ldots, c_k)$, and $D = (d_1, \ldots, d_k)$ be k-dimensional points, let $-A$ denote the point $(-a_1, \ldots, -a_k)$ and (A, B) denote the $2k$-dimensional point $(a_1, \ldots, a_k, b_1, \ldots, b_k)$. The following are well-known facts about point dominance.

Lemma 1 $A \succ B \quad \leftrightarrow \quad -B \succ -A.$ ■

Lemma 2 $A \succ B$ and $C \succ D \leftrightarrow (A, C) \succ (B, D)$. ■

Also let R be the rectangle with lower-most corner A and upper-most corner B and Q be the rectangle with lower-most corner C and upper-most corner D. We assume that R and Q are non-empty, that is, $B \succ A$ and $D \succ C$. The following four lemmas are also known [3] or easy to prove.

Lemma 3 R contains $C \leftrightarrow (C, -C) \succ (A, -B)$. ■

Lemma 4 R contains $Q \leftrightarrow (C, -D) \succ (A, -B)$. ■

Lemma 5 R overlaps $Q \leftrightarrow (D, -C) \succ (A, -B)$. ■

Lemma 6 R is within $Q \leftrightarrow (-C, D) \succ (-A, B)$. ■

Let f be the function that maps each rectangle of form R into the point $(A, -B)$.

Let g be the function that maps each rectangle of form R into the point $(-A, B)$.

Theorem 2 k-dimensional **Stabbing**, **Contain**, **Overlap**, and **Within** reduce to $2k$-dimensional **Dominance**.

Proof: First we use f and g to map each k-dimensional rectangle in S into a $2k$-dimensional point. Let $f(S)$ and $g(S)$ denote the set of points obtained by using f and g, respectively. Second, we create an ECDF-B-tree index I_f for $f(S)$ and a separate ECDF-B-tree index I_g for $g(S)$.

By Lemmas 3, 4, and 5 we can use I_f and the $2k$-dimensional query points $(C, -C)$, $(C, -D)$, and $(D, -C)$, respectively, to answer the first three problems. By Lemma 6 we can use I_g and the query point $(-C, D)$ to answer the **Within** problem. ■

3 Border Point and Window Queries

An **upper (lower) bound dominance** query has the following form:

> Does S have less (more) than s rectangles that contain C, or contain, overlap, or are within Q?

Let us create separate indices I_A and I_B for the lower-most and the upper-most corner vertices, respectively, of the rectangles in S, and also let us create indices I_{-A} and I_{-B} for their negatives. Let $\#(P, I)$ be the number of rectangles in index I dominated by point P, and let $\min$ be the minimum function. Then:

Lemma 7

$$\begin{array}{rcl}
\#((C,-C),I_f) & \leq & \min(\#(C,I_A),\ \#(-C,I_{-B})) \\
\#((C,-D),I_f) & \leq & \min(\#(C,I_A),\ \#(-D,I_{-B})) \\
\#((D,-C),I_f) & \leq & \min(\#(D,I_A),\ \#(-C,I_{-B})) \\
\#((-C,D),I_g) & \leq & \min(\#(-C,I_{-A}),\ \#(D,I_B))
\end{array}$$

Proof: By Lemma 3, $\#((C, -C), I_f)$ is the count of the rectangles that contain C, while $\#(C, I_A)$ (or $\#(-C, I_{-B})$) clearly is the count of the rectangles whose lower-most (resp. negative upper-most) corner point is dominated by C (resp. $-C$). Since each R that contains C has its lower-most (negative upper-most) corner dominated by C (reps. $-C$), but not all rectangles whose lower-most (negative upper-most) corner is dominated by C (resp. $-C$) actually contain C, the first condition must hold. The others cases are similar. ■

Lemma 7 is particularly useful for *border points and rectangles* (the latter also called *border windows*), which are located close to the border of the space in which all the rectangles in S lie.

Example 1 Suppose that in the 2-dimensional case, all rectangles in S lie within the rectangular space $0 \leq x, y \leq 100,000$. Also suppose that we need to find the number of rectangles that contain the point $C = (25, 47)$, which clearly is a border point. Hence, unless there is an unusual distribution of the rectangles, we expect $(25, 47)$ to dominate few or no lower-left corner points of the rectangles in S. Hence we also expect $\#(C, I_A)$ to be zero or a small non-negative integer and a good upper bound approximation for $\#((C, -C), I_f)$. We can find that upper bound more efficiently by searching index I_A with point C than we can find the exact value by searching index I_f with $(C, -C)$.

For k-dimensional rectangles, the upper (lower) bound dominance query can be answered using Theorems 1 and 2 in $O(\log^{2k} n)$ time. Here we have:

Theorem 3 The approximate algorithm based on Lemma 7 requires $O(n \log^{k-1} n)$ space and returns an upper bound u in $O(\log^k n)$ time. When $u < s$, then the **upper bound dominance** query is "yes" and the **lower bound dominance** query is "no." ■

Since in general for border point and window queries $u < s$, Theorem 3 is particularly useful for them.

4 Sequences of Updates

Theorems 1 and 2 imply that I_f and I_g allow a sequence of updates in $O(\log^{2k} n)$ amortized time. In some cases only a finite number of insertion updates are possible.

Definition 2 Rectangle R with lower-most corner A and upper-most corner B *dominates* rectangle Q with lower-most corner C and upper-most corner D, if and only if $A \succ C$ and $B \succ D$.

By Theorem 2 and Dixon's Lemma ([4], p. 123):

Theorem 4 Let c be any fixed constant. If in a sequence of k-dimensional rectangles $R_1, R_2, \ldots$ no rectangle dominates any earlier rectangle, and every rectangle has integer coordinate values greater than or equal to c, then the sequence must be finite. ■

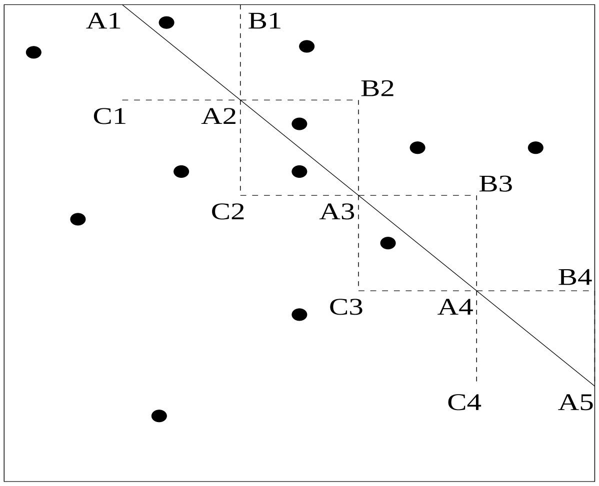

Figure 1. Approximating points below line.

5 Moving Points

The position of any point P moving linearly along the x-axis can can be represented by a function $a_P \cdot t + b_P$. Alternatively, it can be represented as a point (a_P, b_P) in a *dual plane*. This dual representation is attractive because of the following well-known lemma (see [5]):

Lemma 8 Let $P = a_P \cdot t + b_P$ and $Q = a_Q \cdot t + b_Q$ be two moving points in one dimensional space, and $P' = (a_P, b_P)$ and $Q' = (a_Q, b_Q)$ be their corresponding points in the dual plane. Suppose P overtakes Q or vice versa at time instance t, then

$$-t = \frac{b_P - b_Q}{a_P - a_Q}$$

that is, $-t$ is the slope of the line $P'Q'$. Hence, the **Count** problem reduces to the problem of finding how many points are below l, where l is a line crossing Q' with slope $-t$ in the dual plane. ■

As an approximate solution, we first find the rectangle that contains all the points in the dual plane. Then we cut the line within the rectangle into m number of equal pieces by horizontal and vertical line segments. For example, Figure 1 shows a set of points within a rectangle and a line that crosses the rectangle. The crossing line is cut into $m = 4$ pieces horizontally by the line segments C_i and B_{i+1} for $1 \leq i \leq 3$ and vertically by the line segments B_j and C_{j+1} for $1 \leq j \leq 3$.

Let I be the ECDF-B-tree that stores the dual representations of the moving points. The following are upper and lower bounds for $\#Below$, the number of points below the crossing line:

$$\#Below \leq \#(A_{m+1}, I) + \sum_{i=1}^{m} \#(B_i, I) - \#(A_{i+1}, I)$$

$$\#Below \geq \#(A_{m+1}, I) + \sum_{i=1}^{m} \#(A_i, I) - \#(C_i, I)$$

An approximation of $\#Below$ is their average:

$$\frac{\#(A_1, I) + \#(A_{m+1}, I) + \sum_{i=1}^{m} \#(B_i, I) - \#(C_i, I)}{2}$$

Example 2 In Figure 1 the lower bound is 5 and the upper bound is 9, and the average of these is 7, which is exactly the number of points below the line.

In general m can be considered to be a constant that effects the accuracy of the approximation.

Theorem 5 The approximation uses $O(n \log n)$ space and answers **Count** queries in $O(m \log n)$ time where the crossing line in the dual plane is cut into m pieces. ■

The above approximation method can be extended to **Count** queries with arbitrary k-dimensional moving points. Hence it contrasts well with earlier precise algorithms for **Count** queries that require $O(\sqrt{n})$ time and $O(n)$ space with 1-dimensional and $O(\log n)$ time and $O(n^2)$ space with k-dimensional moving points [5] and earlier approximation methods [2, 6] that use "buckets" that cannot be efficiently updated.

References

[1] J. L. Bentley. Multidimensional divide-and-conquer. *Communications of the ACM*, 23(4), 1980.

[2] Y.-J. Choi and C.-W. Chung. Selectivity estimation for spatio-temporal queries to moving objects. In *Proc. ACM SIGMOD*, 2002.

[3] F. P. Preparata and M. I. Shamos. *Computational Geometry: An Introduction*. Springer-Verlag, 1985.

[4] P. Revesz. *Introduction to Constraint Databases*. Springer-Verlag, 2001.

[5] P. Revesz and Y. Chen. Efficient aggregation on moving objects. In *TIME-ICTL*, 2003.

[6] Y. Tao, J. Sun, and D. Papadias. Selectivity estimation for predictive spatio-temporal queries. In *ICDE*, 2003.

[7] D. Zhang, V. J. Tsotras, and D. Gunopulos. Efficient aggregation over objects with extent. In *Symposium on Principles of Database Systems*, 2002.