Transforming Range Queries To Equivalent Box Queries To Optimize Page Access

Sakti Pramanik Alok Watve Chad R. Meiners Alex Liu
Department of Computer Science and Engineering
Michigan State University
East Lansing, MI, USA
{pramanik,watvealo,meinersc,alexliu}@cse.msu.edu

ABSTRACT

Range queries based on L_1 distance are a common type of queries in multimedia databases containing feature vectors. We propose a novel approach that transforms the feature space into a new feature space such that range queries in the original space are mapped into equivalent box queries in the transformed space. Since box queries are axes aligned, there are several implementational advantages that can be exploited to speed up the retrieval of query results. For two dimensional data the transformation is precise. For greater than two dimensions we propose a space transformation scheme based on disjoint planer rotation, and along with pruning query box the results are precise. Experimental results with large synthetic databases and some real databases show the effectiveness of the proposed transformation scheme. These experimental results have been corroborated with appropriate mathematical models.

1. INTRODUCTION

Range queries using L_1 distance measure (or its weighted variants) are used widely in commercial multimedia databases [15]. In this paper, we focus on the implementation of range queries in L_1 space and use box queries for optimizing this implementation.

1.1 Background

Let D be the set of d-dimensional records in the database then the range query at point $p = (p_1, p_2, ...p_d)$, denoted by $r@(p_1, p_2 \dots p_d)$, is defined as,

$$r@(p_1, p_2 \dots p_d) = \left\{ q \;\middle|\; \begin{array}{l} q \in D \\ \wedge \\ |q_1 - p_1| + \cdots + |q_d - p_d| \leq r \end{array} \right\}$$

where, $q_i (1 \leq i \leq d)$ is i^{th} dimension of point q.

For efficient execution of queries in very large databases, usually multi-dimensional index is created for these records and queries are implemented using this index. Effectiveness of an index for implementing the range query is determined by the number of pages (amount of IO) accessed in the index tree. We will use R*-tree for optimizing implementation of range queries, but by first transforming them into box queries and then executing box query on R*-tree[2]. A box query with range $r_i = [min_i, max_i]$ in dimension i is defined as follows:

$$b@(r_1, r_2, ..., r_d) = \left\{ q \;\middle|\; \begin{array}{l} q \in D \\ \wedge \\ min_i \leq q_i \leq max_i \text{ for } 1 \leq i \leq d \end{array} \right\}$$

where, $q_i (1 \leq i \leq d)$ is i^{th} dimension of point q.

1.2 Our Approach

Implementing range queries using indexing is a well studied problem. However, to the best of our knowledge, transforming a range query into an equivalent box query and then applying indexes for searching has not been studied before. Our motivation for this transformation is that in L_1 space, range query is a $d-$dimensional hyper-diamond while a box query is an axes aligned hyper-rectangle. This axes alignment provides a simpler and similar interfaces of the query box with the axes aligned bounding boxes of the index. Thus, we propose transforming data space so that the range query becomes axes aligned box query. Once we find such a transformation, we can then apply any existing indexing scheme that uses bounding boxes.

In 2-dimensional case, this transformation is easy and involves simple axis alignment. We can illustrate this using the example in Figure 1. Note that the edges of the range query follow the line vectors $\langle 1, 1\rangle$ and $\langle -1, 1\rangle$. If we align the query space's axes to align with these vectors $\langle 1, 1\rangle$ and $\langle 1, -1\rangle$ instead of the units vectors $\langle 1, 0\rangle$ and $\langle 0, 1\rangle$, our query space become the space shown in Figure 2. It is interesting to see that in the transformed space, the minimal bounding box $([-2, -2], [2, 2])$ precisely define our original range query in Figure 1.

The transformation is challenging in the case where number of dimensions is greater than two. It is generally difficult to align the query space with the axes without blowing up the number of dimensions. We tackle this problem by transforming projections of the space into two dimensions. As an artifact of this, the bounding box no longer models the range query correctly. We propose a novel method to alleviate this problem.

1.3 Key Contributions

The key contributions of the paper are summarized as

Proceedings of the VLDB Endowment, Vol. 3, No. 1

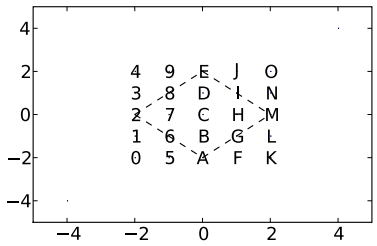

Figure 1: Range query

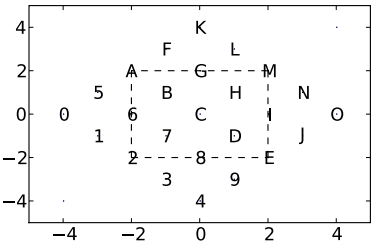

Figure 2: Transformed range query

follows:

- We propose a novel transformation for mapping range queries using L_1 distance to box queries. For 2-dimensional data, the proposed transformation is precise in the sense that the box query retrieves exactly the same results as the range query.
- For high dimensional data, we propose disjoint planar rotation for space transformation along with the pruning box query to give precise query results.
- Based on the experimental results we show that the proposed transformation schemes can be leveraged to improve range query performance.
- We also provide theoretical analysis for the uniformly distributed data which can be used to estimate the performance improvement resulting from the proposed transformation.

The rest of the paper is organized as follows: In section 2, we present the prior work related to this paper. Section 3 proposes a transformation method for 2-D data. These concepts are extended for higher dimensional data in section 4. Experimental evaluation of the proposed transformation is presented in section 5. Concluding remarks follow in the last section.

2. RELATED WORK

The idea of transforming one data-space into some other data-space for efficient query processing has been around for some time. Most of these methods try to reduce the dimensionality of the data using transformations such as Principal Component Analysis(PCA) or Singular Value Decomposition(SVD) [1, 18, 4, 12]. Increasing data dimensionality through transformations such as the ones used for SVMs [17] is a possible way of eliminating false positives however, increasing dimensions negatively affects the performance of any index. Linear transformation is a well known technique in linear algebra [11]. However, their applications to database queries have not been studied much. The proposed work focuses on using linear transformation for improving page accesses for range queries.

There has been a lot of work on executing range queries and nearest neighbor queries using database indexes. A detailed discussion on these topics can be found in [8, 7]. Most of the existing work executes range queries by measuring distance of the query center from the minimum bounding rectangle (MBR) at each subtree and expanding a subtree only when certain distance criterion is satisfied [6, 3, 19, 10]. Some schemes try to incorporate distance metric (or other distance statistics) into the index structure [5, 20, 9, 14]. For each node, they first use the distance metric to get an estimate of maximum distance of any data record in the subtree rooted at the node. Any node that fails the distance criterion is pruned. Box queries are executed by testing if the query box overlaps with the minimum bounding rectangle of a subtree in the index [13].

We use R*-tree [2]for our experiments. However, some of the other indexing schemes may also be used for these experiments. R*-tree index nodes contain minimum bounding rectangles for the child nodes and both range and box queries can be implemented using this index structure.

3. 2-D TRANSFORMATIONS

Mapping range queries in to box queries requires transformations of both data space and user queries. The space transformation is performed offline on the data, before building the index. For simplicity, we call a database built with a transformed data a "transformed database". The query transformation is performed online on each query. The transformations need to satisfy the property that the result of the transformed query over the transformed database is equal to, or at least an approximation of, the result of the original query over the original database. When the two query results are equal, we call such transformations *precise transformations*; otherwise, we call them *approximate transformations*. Approximate transformations may introduce false positives (*i.e.*, the points that do not satisfy the original query but do satisfy the transformed query) or false negatives (*i.e.*, the points that are in the original query but are not in the transformed query). Precise transformations have neither false positives nor false negatives. In this section, we present transformations from range queries to box queries for 2 dimensional (2-D) databases. We show that our transformations are precise.

3.1 From Range Queries to Box Queries

Space Transformation: For 2 dimensional (2-D) databases, our transformations from range queries to box queries are accomplished by mapping each point (x, y) in the original 2-D space to the point $(x + y, x - y)$ in the transformed space, which is essentially a change of axis as shown in Figure 1. Formally, our transformation function $T\colon \mathbb{R}^2 \to \mathbb{R}^2$ is defined as

$$T(x, y) = (x + y, x - y) \tag{1}$$

And the inverse transformation function $T^{-1}\colon \mathbb{R}^2 \to \mathbb{R}^2$ is

defined as,

$$T^{-1}(x,y) = (\frac{x+y}{2}, \frac{x-y}{2}) \quad (2)$$

This function essentially changes the two axes so that they align perfectly with the geometric faces of range queries in the original space. By such transformations, a range query in the original space based on L1 distance becomes precisely a box query in the transformed space. For any 2-D database D, which is a set of 2-D points in the original space, the transformed database $T(D)$ is defined by $T(D) = \{T(x,y)|(x,y) \in D\}$. Note that we do not need to store $T(D)$ as a separate database; rather, we build an index for $T(D)$, which points back to the original points in D.

Query Transformation: We use $r@(a,b)$ to denote the range query $\{(x,y)||x-a| + |y-b| \leq r\}$. Mathematically, $r@(a,b)$ denotes the set of all the points that are within range r based on L1 distance from point (a,b). Geometrically, all points in $r@(a,b)$ form a diamond with four end points: $(a+r,b)$, $(a-r,b)$, $(a,b+r)$, $(a,b-r)$. We use $([a_1,b_1],[a_2,b_2])$ to denote the box query $\{(x,y)|a_1 \leq x \leq a_2, b_1 \leq y \leq b_2\}$. Geometrically, all points in $([a_1,b_1],[a_2,b_2])$ form a rectangle with four end points: (a_1,b_1), (a_2,b_1), (a_2,b_2), (a_1,b_2). After space transformation, geometrically, these transformed points $T(r@(a,b))$ form a square with four end points: $(a+r+b, a+r-b)$, $(a-r+b, a-r-b)$, $(a+r+b, a-r-b)$, $(a-r+b, a+r-b)$. Thus, these transformed points are precisely the representation of a box query $([a-r+b, a-r-b],[a+r+b, a+r-b])$ in the transformed space. Geometrically, our 2-D transformation from range queries to box queries converts a diamond in the original space to a square in the transformed space. For example, in Figure 1, range query $2@(0,0)$ in the original space is equivalent to the box query $([-2,-2],[2,2])$ in the transformed space.

3.2 Transformation Properties

Next, we present several important properties of our transformation function T defined in formula 2.

Precision Property: Our Theorem 3.1 shows that both our range to box query transformation and box to range query transformation are precise.

THEOREM 3.1. *For any point $(x,y) \in D$, (x,y) satisfies the range query $r@(a,b)$ if and only if (iff) $T(x,y)$ satisfies the box query $([a+b-r, a-b-r],[a+b+r, a-b+r])$.*

PROOF. Our proof is based on the fact that for any two numbers u and v, $|u|+|v| \leq r$ iff $|u+v| \leq r$ and $|u-v| \leq r$. This fact can be easily proved by assuming $u > v$ without loss of generality and then considering the following three cases: (1) $u > v > 0$, (2) $u > 0 > v$, and (3) $0 > u > v$. We omit the proof of this fact.

Based on this fact, $|x-a| + |y-b| \leq r$ holds iff both $|(x+y)-(a+b)| = |(x-a)+(y-b)| \leq r$ and $|(x-y)-(a-b)| = |(x-a)-(y-b)| \leq r$ hold. Note that (x,y) satisfies the range query $r@(a,b)$ iff $|x-a|+|y-b| \leq r$ holds, and $T(x,y)$ satisfies the box query $([a+b-r, a-b-r],[a+b+r, a-b+r])$ iff $|(x+y)-(a+b)| \leq r$ and $|(x-y)-(a-b)| \leq r$ holds. □

Distance Property: Theorem 3.2 shows that our transformation function T does not preserve L1 distance, even though it is precise. For any two points (x_1,y_1) and (x_2,y_2), we use $|(x_1,y_1)-(x_2,y_2)|$ to denote their L1 distance.

THEOREM 3.2. *For any two points (x_1,y_1) and (x_2,y_2), $|T(x_1,y_1)-T(x_2,y_2)| = |(x_1,y_1)-(x_2,y_2)| + ||x_1-x_2| - |y_1-y_2||$.*

PROOF. Our proof is based on the fact that for any two numbers u and v, $|u+v|+|u-v| = |u|+|v|+||u|-|v||$. This fact can be proved easily proved by assuming $u > v$ without loss of generality and then considering the following three cases: (1) $u > v > 0$, (2) $u > 0 > v$, and (3) $0 > u > v$. We omit the proof of this fact.

Based on this fact, we have $|T(x_1,y_1)-T(x_2,y_2)| = |(x_1+y_1, x_1-y_1)-(x_2+y_2, x_2-y_2)| = |(x_1+y_1)-(x_2+y_2)| + |(x_1-y_1)-(x_2-y_2)| = |(x_1-x_2)+(y_1-y_2)| + |(x_1-x_2)-(y_1-y_2)| = |x_1-x_2| + |y_1-y_2| + ||x_1-x_2| - |y_1-y_2|| = |(x_1,y_1)-(x_2,y_2)| + ||x_1-x_2| - |y_1-y_2||$. □

We can prove a similar property for T^{-1}.

COROLLARY 3.1. *For any two points (x_1,y_1) and (x_2,y_2), $|T^{-1}((x_1,y_1))-T^{-1}((x_2,y_2))| = (|(x_1,y_1)-(x_2,y_2)|+||x_1-x_2| - |y_1-y_2||)/2$.*

Inequality Property: Although transformation function T does not preserve L1 distance, Theorem 3.3 show an important special case where T preserves distance inequality.

THEOREM 3.3. *Given a point (x_1,y_1), an MBR represented as a rectangle B, and a point (x_2,y_2) on the edge of the rectangle, if among all the points in B, (x_2,y_2) is the point that is closest to (x_1,y_1), then $T((x_2,y_2))$ is the closest point in $T(B)$ to $T((x_1,y_1))$ and $T^{-1}((x_2,y_2))$ is the closest point in $T^{-1}(B)$ to $T^{-1}((x_1,y_1))$*

PROOF. Our proof is based on the fact that for any four non-negative numbers u, v, w, and z, if $u+v \leq w+z$, $u \leq w$, and $v < z$, then $u+v+|u-v| \leq w+z+|w-z|$. This fact can be easily proved by considering the following three cases: (1) $u \leq w \leq v \leq z$, $u \leq v \leq w \leq z$, and $u \leq v \leq z \leq w$. We omit the proof of this fact.

Considering any point (x_3,y_3) in the rectangle, because (x_2,y_2) is closer to (x_1,y_1) than (x_3,y_3), we have $|(x_2,y_2)-(x_1,y_1)| \leq |(x_3,y_3)-(x_1,y_1)|$, $|x_2-x_1| \leq |x_3-x_1|$ and $|y_2-y_1| \leq |y_3-y_1|$. Now we need to prove $|T((x_2,y_2))-T((x_1,y_1))| \leq |T((x_3,y_3))-T((x_1,y_1))|$. By Theorem 3.2, we have $|T((x_2,y_2))-T((x_1,y_1))| = |(x_1,y_1)-(x_2,y_2)| + ||x_1-x_2|-|y_1-y_2|| = |x_1-x_2|+|y_1-y_2|+||x_1-x_2|-|y_1-y_2||$ and $|T((x_3,y_3))-T((x_1,y_1))| = |(x_1,y_1)-(x_3,y_3)| + ||x_1-x_3|-|y_1-y_3|| = |x_1-x_3|+|y_1-y_3|+||x_1-x_3|-|y_1-y_3||$. By the above fact, we have $|x_1-x_2|+|y_1-y_2|+||x_1-x_2|-|y_1-y_2|| \leq |x_1-x_3|+|y_1-y_3|+||x_1-x_3|-|y_1-y_3||$. □

Figure 3 shows an example scenario for points (x_1,y_1), (x_2,y_2), (x_3,y_3) and the rectangle.

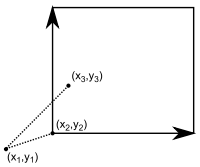

Figure 3: Illustration for Theorem 3.3

4. MULTI-DIMENSIONAL TRANSFORMATIONS

For $d = 2$, range queries can be precisely transformed into box queries. However, to the best of our knowledge, for $d > 2$ we have not found any heuristics in the literature on precise transformations. We conjuncture that such a precise transformation may not exist because the number of faces in a range query exceeds the number of faces in the box query. For example, for $d = 3$, the range query takes the form of a octahedron; whereas, the corresponding box query takes the form of a cube. Since we cannot define such a precise transformation, we develop a new type of box query that uses the range value from the original range query to prune the false positives in the transformed box query while preventing the occurrence of false negatives.

In this section, we first use the 2-D transformation of Section 3 to define an approximate transformation call disjoint planar rotations (DPR). Second, we describe how to use the disjoint planar rotations to map a range query in to a precise box query. Finally, we discuss why the proposed new type of box query out performs the original range query.

4.1 Disjoint Planar Rotations

DPR is a transformation that is derived from our two dimensional transformation function. We transform a d dimensional space via this technique by transforming disjoint planes in the database. For example, a four dimensional point (x, y, z, w) can be transformed into $(T(x, y), T(z, w))$. That is, this transformation can be visualized as a rotation of each disjoint plane in the database's space.

More formally, we define a d dimensional transformation $T^d(p)$ as follows:

$$T^d(p) = \begin{pmatrix} T(p_1, p_2), \\ \cdots, \\ T(p_{d-1}, p_d) \end{pmatrix} \quad \text{when } d \text{ is even} \tag{3}$$

$$T^d(p) = \begin{pmatrix} T(p_1, p_2), \\ \cdots, \\ T(p_{d-2}, p_{d-1}), d \end{pmatrix} \quad \text{when } d \text{ is odd} \tag{4}$$

Note that in the odd case we choose to preserve the last dimension because it does not significantly affect the performance of the proposed box query.

Our modification to box query is based on the observation that if we can estimate distance between the query center and an MBR of the index tree, we can prune the branches of the tree that do not contain any true positives. We first prove the result proposed in theorem 3.3 for $d-$dimensional data.

THEOREM 4.1. *Given a point c and a set of points B representing the points from an MBR. If p is the closest point in B to c, then $T^d(p)$ is the closest point in $T^d(B)$ to $T^d(c)$ and $(T^{-1})^d(p)$ is the closest point in $(T^{-1})^d(B)$ to $(T^{-1})^d(c)$.*

PROOF. The proof is similar to that of theorem 3.3 and is omitted. □

Based on the theorem we propose following heuristic to eliminate all the false positives:

Heuristic: If an MBR M overlaps with the query box, we find the closest point $T^d(p)$ in M to query center $T^d(c)$. Using the inverse transformation we then calculate distance between p and c; if it is greater than the query range then the MBR is pruned.

Let u be the point in MBR M that is nearest to center of the box query b. Using the above heuristic, we now formally define pruning box query (PBQ) as,

$$pbq@(r_1, r_2, ..., r_d) = \left\{ q \;\middle|\; \begin{array}{l} q \in D \\ \wedge \\ min_i \le q_i \le max_i \text{ for } 1 \le i \le d \wedge \\ \text{distance between } (T^{-1})^d(q) \text{ and} \\ (T^{-1})^d)(u) \text{is less than the range} \end{array} \right\}$$

This approach not only eliminates all the false positives but it also provides more efficient query execution. It is important to note that we do not miss any data record with this pruning strategy. Theorem 4.2 states this.

THEOREM 4.2. *For every point p that satisfies the range query $r@(p_1, p_2 \dots p_d)$, p is also contained in the result of the PBQ.*

PROOF. Let, if possible, there be a point p such that, $p \in r@(c_1, c_2 \dots c_d)$ but p' is not contained in the box query, where $p' = T^d(p)$. This is possible, only if the MBR M containing p' was pruned by the box query at some point, i.e. the estimated distance between M and p' was less than r. Let $u' = T^d(u)$ be the closest point in M to c' ($c' = T^d(c)$). This implies that while u' was the closest point to c' in the transformed domain, u was not the closest point to c in the original data domain. This contradicts theorem 4.1. Hence, such a point p does not exist. In other words, resultset returned by the PBQ contains all the points from the one returned by the original range query. □

4.2 Analysis of performance improvement

Using DPR and pruning box queries improves the performance of indexed queries because the transformation aligns the index's minimum bounding boxes' faces with faces of the query. In this section we provide a detailed analysis of the range query and the PBQ performances. We first present it for 2-dimensional queries and then generalize it for higher dimensions.

4.2.1 Model Basics

Without loss of generality, we fix the size of all MBRs so that we can calculate the area of MBR centroids whose corresponding MBRs intersect with a query. From this area, we can calculate the probability that an MBR of a certain size will intersect with a query. For example, for 2D query spaces, we fix an MBRs length to be $2l$ and breadth to be $2b$. We must calculate probability that a random MBR of certain size intersects the query space (a diamond in case of the range query and a square for the PBQ). We analyze only one quadrant of the 2-D plane. Analysis for other quadrants is similar and is omitted.

Figure 4 shows the space in which an MBR of size $2l \times 2b$ must lie in order to intersect with the query space. We can see from this visualization that the query faces align with the MBR faces after transformation, and we conjuncture that this alignment improves query performance for two reasons: MBRs are less likely to intersect with the PBQ than the range query, and MBRs that do intersect with PBQ have a higher likelihood of containing a point within the query than MBRs that intersect the range query.

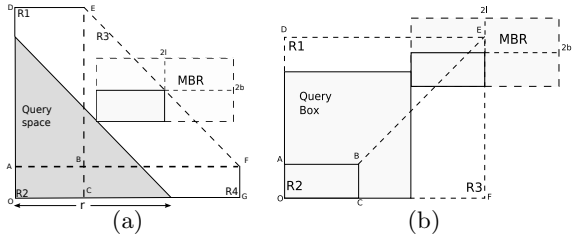

Figure 4: The MBR intersection areas in a quadrant for range and transformed range queries

4.2.2 Analysis of Range Query

To calculate the probability of intersection for a range query $P_{RQ(r,l,b)}$ with an MBR, we determine the area in which the centroid of an MBR with length $2l$ and breadth $2b$ must lie for it to intersect with the query. As shown in Figure 4(a), the intersection space can be divided into four regions $R_1(\square ADEB), R_2(\square OABC), R_3(\triangle BEF)$ and $R_4(\square CBFG)$. The area of the intersection space can then be calculated as,

$$\begin{aligned} A_{RQ(r,l,b)} =& \text{Area of R1, R2, R3, and R4} \\ =& \int_0^r b\ dx + lb \\ &+ \int_0^{\frac{r}{\sqrt{2}}} 2x\ dx + \int_0^r l\ dx \\ =& \frac{r^2}{2} + lb + r(l+b) \end{aligned} \tag{5}$$

Hence, given the area of the data space, A, the probability that an MBR will overlap with a range query is,

$$P_{RQ(r,l,b)} = \frac{A_{RQ(r,l,b)}}{A} \tag{6}$$

4.2.3 Analysis of PBQ

To calculate the probability $P_{B(r,l,w)}$ of intersection of a PBQ with an MBR, we determine the area in which the centroid of an MBR with length $2l$ and breadth $2b$ must lie for it to intersect with the query. As shown in Figure 4(b), the intersection space is an extended box query with length of $2l + r\sqrt{2}$ and breadth of $2b + r\sqrt{2}$. We divide the space in to three regions
$R_1(\square ADEB), R_2(\square OABC)$ and $R_3(\square CBEF)$. The area of the intersection space can then be calculated as,

$$\begin{aligned} A_{B(r,l,b)} =& \text{Area of R1, R2, and R3} \\ =& \int_0^{\frac{r\sqrt{2}}{2}} (l+x)\ dx + lb \\ &+ \int_0^{\frac{r\sqrt{2}}{2}} (b+x)\ dx \\ =& \frac{r^2}{2} + lb + \frac{r}{\sqrt{2}}(l+b) \end{aligned} \tag{7}$$

Hence, given a the area of the data space, A, the probability that a random MBR intersects the transformed range query is,

$$P_{B(r,l,b)} = \frac{A_{B(r,l,b)}}{A} \tag{8}$$

4.2.4 Hyperdimensional range queries

Figure 5 represents the four dimensional space as a two dimensional grid of two dimensional slices through the four dimensional space. In this case, the grid is a coarse grain visualization of the effect of the wz plane on the xy planes so each panel in the wz represents the xy plane that is fixed at the wz panel's coordinate. While this visualization is coarse grained in that it does not show every point in the range query, it illustrates how DPR transforms the range query into a PBQ as shown in Figure 5(b), which shows Figure 5(a)'s range query as a PBQ in a DPR transformed space. Note that this visualization can be generalized to visualize any hyperspace as a nested series of two dimensional grids. The above equations can be generalized to any even

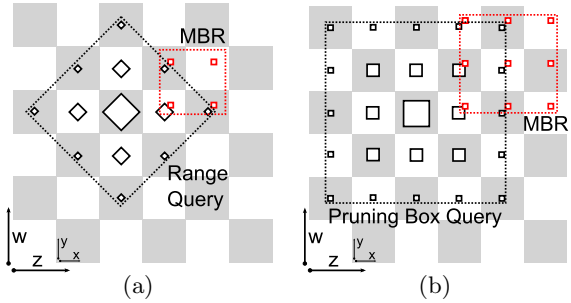

Figure 5: Visualizations of range and pruning box queries relationship with MBRs in hyperspace

number of dimensions via a density function. We use this nesting concept to find the area of centroids for any even dimensional query. As we move away from the center of the query in xy plane, density of points (or query space) in wz plane decreases. We define density function as the area in which center of an MBR must lie in order to intersect with query space. The d-dimensional density function for range query is denoted as $A_{RQ(r,W,d)}$ and that for the box query is denoted as $A_{PBQ(r,W,d)}$.

Consider the MBR in Figure 5(a); we first examine the intersections of the wz projections of the MBR and query, which is shown by the dotted lines. Note that if these projections did not intersect there would be no intersection of query and MBR; however, since there is an intersection we can determine if query and MBR do intersect by looking for an intersection in the xy projection that is closest to the origin of the range query.

Given a hyper-rectangle with widths $W = \langle w_1, \cdots, w_d \rangle$, the density function for the range query is recursively defined as,

$$\begin{aligned} A_{RQ(r,W,0)} =& 1 \\ A_{RQ(r,W,d)} =& \int_0^r A_{R(r-x,W,d-2)} w_d\ dx \\ &+ A_{RQ(r,W,d-2)} w_{d-1} w_d \\ &+ \int_0^{\frac{r}{\sqrt{2}}} A_{R(r-x\sqrt{2},W,d-2)} 2x\ dx \\ &+ \int_0^r A_{R(r-x,W,d-2)} w_{d-1}\ dx \end{aligned} \tag{9}$$

$$P_{RQ(r,W,d)} = \frac{A_{RQ(r,W,d)}}{A} \tag{10}$$

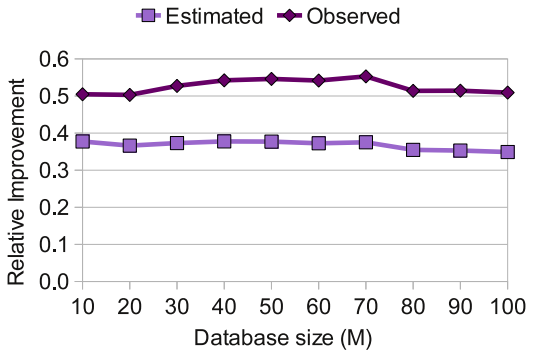

Figure 6: Comparison of estimated and observed improvement for 10-dimensional data

in the original space and

$$A_{PBQ(r,W,d)} = 1$$

$$A_{PBQ(r,W,d)} = \int_0^{\frac{r\sqrt{2}}{2}} A_{B(r-x\sqrt{2},W,d-2)}(w_{d-1}+x)\,dx + A_{PBQ(r,W,d-2)} w_{d-1} w_d + \int_0^{\frac{r\sqrt{2}}{2}} A_{B(r-x\sqrt{2},W,d-2)}(w_d+x)\,dx \quad (11)$$

$$P_{PBQ(r,W,d)} = \frac{A_{PBQ(r,W,d)}}{A} \quad (12)$$

in the transformed space. It can be seen that area analyses for two dimensional cases are in fact special cases of equations 9 through 11

We can see that because the density function for the PBQ is less than the density function for the range query, when we increase the number of dimensions, the range query has a larger number of valid centroids that intersect it than the PBQ.

4.2.5 Improvement estimation

Based on the equations, 9 and 11, we can estimate expected relative improvement due to the transformation as,

$$I(r,W,d) = P_{PBQ(r,W,d)} / P_{RQ(r,W,d)} \quad (13)$$

and the difference as,

$$D(r,W,d) = P_{RQ(r,W,d)} - P_{PBQ(r,W,d)} \quad (14)$$

When range r is considerably large compared to size of the MBRs, the first term in equations 9 and 11 will dominate and we won't have significant improvement. However, in most of the real world systems, query range is usually small so that it retrieves few hundreds of records. Hence, in general we can expect a considerable performance gain.

It should be noted that this analysis does not provide an exact modeling of the actual trees because, first, MBRs are not completely random in any index tree (due to the use of heuristics) , and second, the analysis holds for MBRs of uniform size which is generally not the case.

Figure 6 compares the estimated values of relative improvements (equation 13) with the observed values for a fixed query for a ten dimensional database as the size of the database increase. We can see that although there is not an exact match, the trends in ratio are similar.

PBQ	$D=2$ $R=0.005$		
Level	Total	Empty	Non-empty
Top	1.34	0.00	1.34
Middle	4.85	0.04	4.81
Bottom	144.42	0.10	144.32
RQ	$D=2$ $R=0.005$		
Level	Total	Empty	Non-empty
Top	1.00	0.00	1.00
Middle	5.54	0.13	5.41
Bottom	153.29	3.29	150.00
PBQ	$D=10$ $R=0.07$		
Level	Total	Empty	Non-empty
Top	36.25	22.79	13.46
Mid-h	604.14	545.11	59.03
Mid-l	6008.50	5863.13	145.37
Bottom	23627.35	23368.56	258.79
RQ	$D=10$ $R=0.07$		
Level	Total	Empty	Non-empty
Top	52.81	33.64	19.17
Mid-h	940.16	873.98	66.18
Mid-l	9847.35	9699.43	147.92
Bottom	46404.61	46142.31	262.30

Table 1: Break down of page accesses

4.2.6 Avoiding Empty Pages

Experimentally, we can see that the reduction in area that our model predicts for pruning box queries translates to fewer page accesses in the index tree. Table 1 shows the performance break down at each level of the index tree for both range and pruning box queries for two and ten dimensions and database of 100 million. The break down shows the average number of page accesses, the average number of empty page accesses, and the average number of non-empty page accesses. An *empty* page has no children that satisfy the query, while a *non-empty* page has at least one child that satisfies the query.

We observe that both range and pruning box queries have a similar number of non-empty page accesses, which corresponds in our model to the shared centroid area that with both queries. The number of non-empty pages access should be similar between both query types because our transformation does not change the relative distribution of the records in the space.

We observe that the performance improvements are best gained by reducing the number of empty page accesses. For example, when $D = 2$, there are very few empty pages, and we see a small difference in performance between queries.However, when $D = 10$, empty pages make up the majority of page accesses, and we see that the pruning box query load approximately half the number of empty pages than the range query, which results in a much larger performance improvement.

5. RESULTS

In this section we present the results of applying the proposed transformation on various databases. Effectiveness of the proposed transformation is measured by comparing the IO cost (i.e. number of index page accesses) for the proposed pruning box queries with that of range queries. For performance comparison purposes, we create R*-tree index for the range query in the original space and R*-tree index for the pruning box query in the transformed space.

Uniformly distributed synthetic data sets as well as a real data set were used for the experiments. Data records were

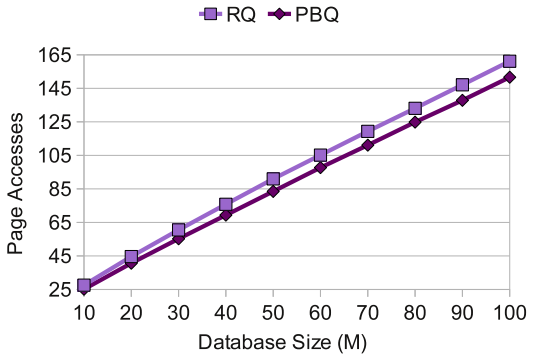

Figure 7: Effect on database size on page accesses

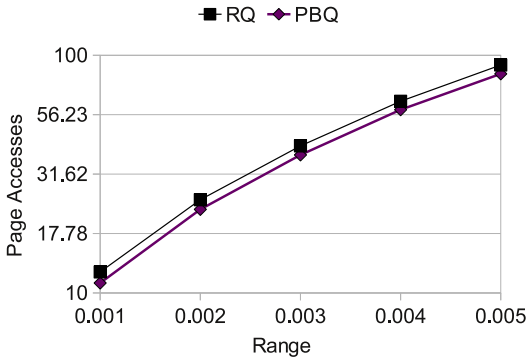

Figure 8: Effect of query range on page accesses

normalized to unit (hyper)cube. Page size of 4K bytes was used for the index nodes. All results presented here are based on averaging the I/O of one hundred random queries. All the experiments were run on AMD Opteron 2.2.GHz systems running GNU/Linux. The labels used for various methods in the figures and tables are as follows: RQ - traditional range query on R*-Tree, PBQ - Pruning Box Query on R*-Tree.

We begin with results on 2-dimensional data and then proceed with the results for higher dimensional data. The source code for all the programs is available at [16]

5.1 2D transformations

As explained earlier, transformation in 2-dimensional databases is perfect, i.e., the transformed query does not lose any useful results nor does it gather any unwanted results.

5.1.1 *Effect of database size*

Figure 7 shows the effect of database size on the number of page accesses. As seen from the figure, as the database size increases, number of page accesses for both range and pruning box queries increases, as expected. However, rate of increase for range query is higher than that of the pruning box query. The performance improvement increases with increasing database size. The relatively small improvement is consistent with our analysis in section 4.2.6.

5.1.2 *Effect of query ranges*

We experimented with various query ranges keeping database size constant (50 million records). The performance comparison of pruning box query with range query is shown in figure 8. Ranges in the figure are a normalized distances. It can be seen from the figure that pruning-box queries perform consistently better than range queries.

5.2 Higher dimensional transformation

The main challenge in transforming high dimensional queries is that the DPR transformation tends to retrieve a lot of false positives. We use the pruning box query to eliminate false positives and reduce the number of page accesses for execution of the query. In the following subsections, we present the results for multi-dimensional synthetic data.

5.2.1 *Effect of database size*

Figure 9 shows effect of database size on the query cost. We used a database with 10 dimensional vectors. Query range was kept constant. As can be seen from the figure, as the database size increases, cost for both range and box queries increases, as expected, but the rate of increase is much slower for pruning box queries than for range queries. We get more than 40% reduction in the cost for a database size of 100 million vectors.

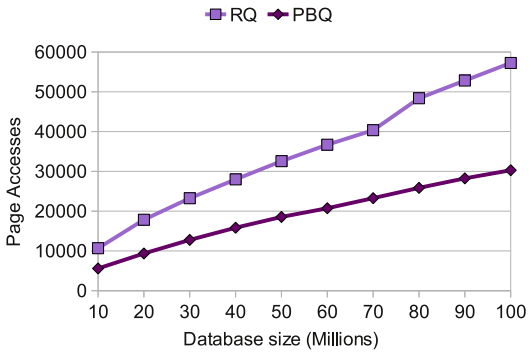

Figure 9: Effect of database size on page accesses

5.2.2 *Effect of query ranges*

Figure 10 gives the comparative performance of range queries versus pruning box queries with increasing query range. As seen from the figure, performance of pruning box queries is consistently better than range queries, and the performance difference gets wider with increasing query ranges. This is because hyper-diamonds of the range queries tend to intersect more with the bounding boxes than the pruning box queries.

5.3 Performance results for real data

The experimental results described so far were carried out on synthetic data. In this section we describe effectiveness of the proposed approach on real data. We used GIS data for our experiments. The data has two dimensions (coordinates of points obtained through GPS) and there are totally 108779 records (obtained from a GIS company). We randomly selected 100 points from the database as range query centers. For each query center, range was changed from 0.01 to 0.05. Figure 11 shows that even for this small database pruning box query has better performance than the traditional range query on the R* tree.

6. CONCLUSION

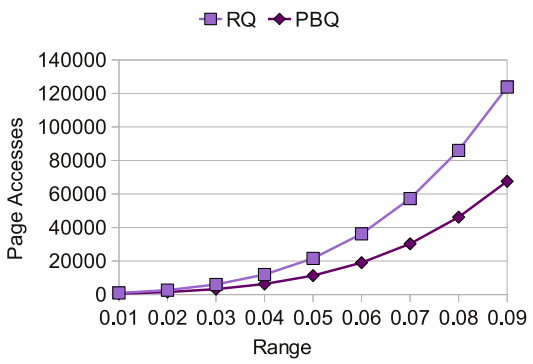

Figure 10: Effect of range on page accesses

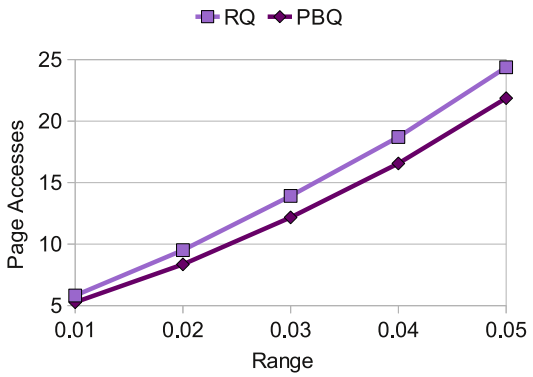

Figure 11: Range query on GIS data

In this paper, we present a novel space transformation technique to improve the performance of the range queries based on L_1 distance. The proposed transformation is computationally easy to implement. Our theoretical analysis suggests that performance improvement is dependent on the relative sizes of the query ranges and the bounding box sizes of the index. Based on the bounding box sizes of R*-tree indexing we see performance improvement increases with database sizes and dimensions. Our experiments with synthetic as well as real data support these results.

7. REFERENCES

[1] C. C. Aggarwal. On the effects of dimensionality reduction on high dimensional similarity search. In *PODS '01: Proceedings of the twentieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 256–266, New York, NY, USA, 2001. ACM.

[2] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger. The R*-tree: an efficient and robust access method for points and rectangles. *Proceedings of ACM SIGMOD*, pages 322–331, 1990.

[3] S. Berchtold, D. Keim, and H.-P. Kriegel. The X-tree: an index structure for high-dimensional data. *Proceedings of the 22nd International Conference on VLDB*, pages 28–39, 1996.

[4] K. Chakrabarti, E. Keogh, S. Mehrotra, and M. Pazzani. Locally adaptive dimensionality reduction for indexing large time series databases. *ACM Trans. Database Syst.*, 27(2):188–228, 2002.

[5] P. Ciaccia, M. Patella, and P. Zezula. M-tree: An efficient access method for similarity search in metric spaces. In *VLDB '97: Proceedings of the 23rd International Conference on Very Large Data Bases*, pages 426–435, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc.

[6] A. Guttman. R-trees: a dynamic index structure for spatial searching. *Proceedings of ACM SIGMOD*, pages 47–57, 1984.

[7] G. R. Hjaltason and H. Samet. Index-driven similarity search in metric spaces (survey article). *ACM Trans. Database Syst.*, 28(4):517–580, 2003.

[8] I. F. Ilyas, G. Beskales, and M. A. Soliman. A survey of top-k query processing techniques in relational database systems. *ACM Comput. Surv.*, 40(4):1–58, 2008.

[9] N. Katayama and S. Satoh. The SR-tree: an index structure for high-dimensional nearest neighbor queries. In *SIGMOD '97: Proceedings of the 1997 ACM SIGMOD international conference on Management of data*, pages 369–380, New York, NY, USA, 1997. ACM.

[10] A. Kumar. G-tree: A new data structure for organizing multidimensional data. *IEEE Trans. on Knowl. and Data Eng.*, 6(2):341–347, 1994.

[11] S. Lang. *Linear Algebra*. New York: Springer-Verlag, 1987.

[12] K. V. Ravi Kanth, D. Agrawal, and A. Singh. Dimensionality reduction for similarity searching in dynamic databases. *SIGMOD Rec.*, 27(2):166–176, 1998.

[13] J. Robinson. The K-D-B-tree: a search structure for large multidimensional dynamic indexes. *Proceedings of ACM SIGMOD*, pages 10 –18, 1981.

[14] J. K. Uhlmann. Satisfying general proximity/similarity queries with metric trees. *Inf. Process. Lett.*, 40(4):175–179, 1991.

[15] `http://download.oracle.com/docs/html/B10829\_01/toc.htm`. Oracle intermedia reference - 10g release 1(10.1).

[16] `http://www.cse.msu.edu/~watvealo/mysoftware.html`. Web url for source code.

[17] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer, New York, 1995.

[18] K. Vu, K. A. Hua, H. Cheng, and S.-D. Lang. A non-linear dimensionality-reduction technique for fast similarity search in large databases. In *SIGMOD '06: Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, pages 527–538, New York, NY, USA, 2006. ACM.

[19] R. Weber, H.-J. Schek, and S. Blott. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In *Proceedings of the 24rd International Conference on Very Large Data Bases*, pages 194–205, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

[20] D. A. White and R. Jain. Similarity indexing with the ss-tree. In *Proceedings of the 12th International Conference on Data Engineering*, pages 516–523, Washington, DC, USA, 1996. IEEE Computer Society.