

The Convex Polyhedra Technique: An Index Structure for High-Dimensional Space

Jiyuan An * Hanxiong Chen † Kazutaka Furuse † Masahiro Ishikawa ‡
Nobuo Ohbo †

*Doctoral Program in Engineering, University of Tsukuba
1-1-1 Tennoudai, Tsukuba City, Ibaraki Japan, 305-8573,
Email: an@dblab.is.tsukuba.ac.jp

†Institute of Information Sciences and Electronics, University of Tsukuba
1-1-1 Tennoudai, Tsukuba City, Ibaraki Japan, 305-8573,
Email: {chx, furuse, ohbo}@is.tsukuba.ac.jp

‡National Institute of Agrobiological Sciences
Email: misi@nias.affrc.go.jp

Abstract

This paper proposes a new dimensionality reduction technique and an indexing mechanism for high dimensional data sets in which data points are not uniformly distributed. The proposed technique decomposes a data space into convex polyhedra, and the dimensionality of each data point is reduced according to which polyhedron includes the data point. One of the advantages of the proposed technique is that it reduces the dimensionality *locally*. This local dimensionality reduction contributes to improve indexing mechanisms for non-uniformly distributed data sets.

To show the applicability and the effectiveness of the proposed technique, this paper describes a new indexing mechanism called CVA-file (Compact VA-File) which is a revised version of the VA-file. With the proposed dimensionality reduction technique, the size of data points stored in index files can be reduced. Furthermore, it can estimate upper and lower bounds of each entry in index files by using geographic properties of convex polyhedra. Results from experimental simulations show that the CVA-file is better than the VA-file for non-uniformly distributed real data sets.

Keywords: High-dimensional Space, Spatial Index, Image Retrieval, Local Dimensionality Reduction, VA-file, CVA-file.

1 Introduction

Constructing efficient indexing mechanisms for high dimensional data has been a challenging topic because of the *dimensionality curse*. Some of recent researches conclude that it is inherently impossible to construct efficient indexing mechanisms under the assumption that data points are uniformly distributed and dimensions are not correlated[Berchtold et al., 1997][Beyer et al., 1999].

We can observe such a kind of problem in the pyramid-tree technique[Berchtold et al., 1998] which maps d -dimensional points into a 1-dimensional space and uses a B⁺-tree to index the 1-dimensional space. Figure 1 illustrates the range query q of length α in pyramids of a unit space. In this figure, if the center of the query q is placed in one of the white triangle areas, the search procedure can be localized to the triangle. That is, we can omit to check data points stored in other triangles. If data points are uniformly distributed, the probability that a query is

localized to a single triangle is proportional to the total amount of the areas of triangles. However, this amount is rapidly decreasing as the dimensionality becomes higher. Actually, the total amount of areas of triangles is $(1 - 2\alpha)^d$, and this amount becomes almost 0 when d is large, even if α is very small.

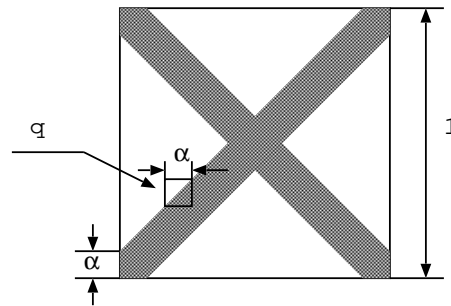


Figure 1: Range Query in Pyramid-tree

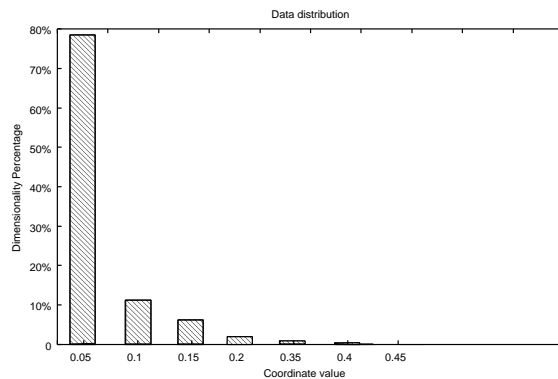


Figure 2: Data Distribution

However, it is very unlikely that data points in real data sets are uniformly distributed and dimensions are independent of each other. Figure 2 is a histogram of the unit color feature vector space of a certain image data set. This figure shows the distribution of coordinate values composing vectors in the data set. In this data set, about 78.5% of coordinate values are in the range (0.0, 0.05), and about 11.2% of coordinate values of all data points are in the range (0.05, 0.1). This means that almost all vector components have values of nearly 0, data points are not uniformly distributed.

By exploiting the non-uniformity of real data sets, we can have room for avoiding the di-

mensionality curse. Such kinds of research work include the projected clustering called PROCLUS[Aggarwal et al., 1999]. This clustering algorithm finds subspaces, in which a subset of data points are close together, from original high dimensional space. In this algorithm, each subspace is composed of a small set of dimensions. The sets of dimensions selected for each subspace may not be identical. That is, dimensions are reduced *locally* for each cluster. This technique is also useful for constructing indexing mechanism.

To scale to higher dimensionality, a commonly used approach is the dimensionality reduction. The FastMap technique proposed in [Faloutsos et al., 1995]

projects data points in an Euclidean space into its subspace. In [Shinohara et al., 2000], another technique applying the FastMap to L_1 distance space is proposed. These techniques reduce the dimensionality *globally*. Therefore, all data points are represented in a common (reduced) subspace.

In this paper, we propose a new dimensionality reduction technique. Unlike the FastMap, the proposed technique reduces the dimensionality *locally*, i.e., the set of selected dimensions for a data point may differ from the ones for other data points. This local dimensionality reduction contributes to improve indexing mechanisms for non-uniformly distributed data sets.

One of the major advantages of the proposed technique is that it exploits geographic properties of convex polyhedra. In general, d -dimensional unit hypercube can be decomposed to convex polyhedra which are clipped by lower dimensional hyperplanes. We reduce the dimensionality of each data point according to which polyhedron includes the data point. This approach is very effective to decrease the loss of information caused by the dimensionality reduction.

The proposed technique is widely applicable to a class of indexing mechanisms. To verify the effectiveness of the technique, we apply it to the VA-file[Weber et al., 1998] and construct another indexing mechanism called CVA-file (Compact VA-file). By the proposed dimensionality reduction technique, we can reduce the size of data points stored in index files. Furthermore, we can estimate upper and lower bounds of each entry in index files by exploiting the properties of convex polyhedra. Results from experimental simulations show the effectiveness of the proposed technique.

The remainder of this paper is organized as follows. Next section explains our idea with a simple 2-dimensionality example. Generalized description of the idea and definition of the convex polyhedra used in the proposed technique are given in Section 3. Section 4 describes the structure of the CVA-file and the mechanism to estimate the lower and upper bounds by the polyhedra. Section 5 confirms the effectiveness of the proposed technique with some experiments using real data sets. Section 6 concludes this paper.

2 Outline of the Approach

This section explains the idea of our approach with a simple 2-dimensional example as illustrated in Figure 3. Data space is normalized into a unit square and thus any data point can be represented by two coordinates on X_1 and X_2 range over $(0, 0)$ to $(1, 1)$. In this figure, lines $x_2 = x_1$ and $x_2 = 1 - x_1$ divide the space into 4 disjoint triangles. These 4 triangles have base lines $x_1 = 0, x_1 = 1, x_2 = 0$, and $x_2 = 1$ respectively. X_2 is called the *surface axis* of the two triangles TOX_2 and TX_1I because it parallels (or is) the base line of each of these two triangles. Similarly, X_1 is called surface axis of the other two triangles. Each data point has shorter distance to the base line

of the triangle which includes the data point than to base lines of other triangles. For example, in Figure 3(a) the data point v_1 has shorter distance to base line $x_1 = 0$ than to $x_2 = 0$, and v_2 has shorter distance to base line $x_2 = 0$ than to $x_1 = 1$. We call this distance the *face-altitude* of the point.

According to which triangle the data point falls in, some certain coordinates of the data point are omitted, hence resulting in a smaller index file. By “omit”, we mean that the coordinate values are not stored in the index entry of the data point. Generally, the coordinates of non-surface axes are omitted. For instance, the X_1 coordinate of v_1 in Figure 3(a) is omitted, and so is that of v_3 . Similarly, for v_2 and v_4 , the coordinate of their non-surface axis X_2 is omitted. That is, the triangle a data point belongs to specifies omitted coordinates of the data point. In next section we give the definition of convex polyhedra generalized triangle for higher dimensional space and formally describe the condition that determines which convex polyhedra a data point belongs to.

Figure 3(b) illustrates how the index entry for the data point $v = (v.x_1, v.x_2)$ is constructed. The $v.x_1$ is omitted since X_1 is not a surface axis of the triangle to which v belongs. In our approach, $p_{x_2}[s]$ is stored in the index entry as approximation of the coordinate $v.x_2$ of surface axis X_2 . For each data point, header area of the index entry records which coordinate values are omitted for that data point.

Generally, $p_{x_i}[\cdot]$ stores the so-called “partition values” on axis X_i . In the case of Figure 3(b), $p_{x_2}[s]$ is taken as the approximation of $v.x_2$ for the reason that $p_{x_2}[s] \leq v.x_2 < p_{x_2}[s + 1]$. Binary partition values are intuitive and simple ones for dividing axis value into 2^b partitions determined by a given parameter b . With binary partition values each $p_{x_i}[\cdot]$ is expressed as a bit pattern, and usually takes much smaller space than storing the coordinate itself.

It is possible to apply our approach of dimensionality reduction to a class of index structures. Here we illustrate an example of the application to VA-file index structure[Weber et al., 1998]. Suppose that an index file is built for all data points as described above. This file is called “*CVA-file*”. Consider a query point whose coordinate values are $q.x_i$ ($i = 1, 2$) is given as illustrated in Figure 3(b). k -nearest-neighbor(k -NN) search with CVA-file is performed as follows. In the first phase, the CVA-file is scanned and the lower and upper bounds are calculated for each entry. If the lower bound of next entry is greater than the k -th biggest upper bound of entries examined so far, then the entry is pruned, or in other words, the real data of the entry needs not to be accessed since k better candidates have already been found. In the second phase, real data are accessed in the increasing order of the lower bounds to determine final answer set. Note that it is not necessary to examine all entries in the second phase when the lower bound of next entry is greater than the distance of k -th nearest neighbor found so far.

With CVA-file, bounds of omitted coordinates can be estimated as follows. Because the query point q is in right-and-upper position of data point v , the lower bound(denoted by l) and upper bound(denoted by u) of v would be $l = \text{dist}(q, (p_{x_1}[t + 1], p_{x_2}[s + 1]))$ and $u = \text{dist}(q, (p_{x_1}[t], p_{x_2}[s]))$, respectively. For the case that distance function is L_p , $l = ((q.x_1 - p_{x_1}[t + 1])^p + (q.x_2 - p_{x_2}[s + 1])^p)^{1/p}$, where $q.x_2 - p_{x_2}[s + 1]$ is the l_2 shown in Figure 3(b). u can be obtained in the similar way.

In the case of Figure 3(b), $v.x_1$ or $p_{x_1}[\cdot]$ is omitted, and hence the bound of coordinate X_1 (lower bound and upper bound is denoted by l_1, u_1 , respectively) have to be estimated. Since v falls into the triangle TOX_2 , it is guaranteed that $v.x_1$ is not greater than

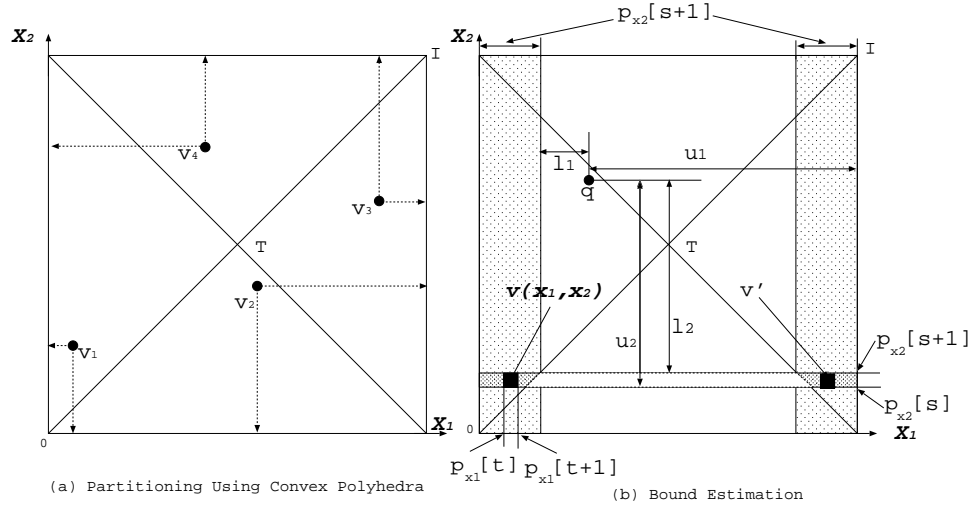


Figure 3: A 2-Dimensional Example

$v.x_2$. Moreover, the width of two vertical shaded rectangles is $p_{x_2}[s+1]$. Thus, $p_{x_2}[\cdot]$ of $v.x_2$ can be used as the upper approximation for $p_{x_1}[\cdot]$ of $v.x_1$. As a consequence, the lower bound for v on X_1 , $|q.x_1 - p_{x_1}[t+1]|$, can be estimated by $l_1 = |q.x_1 - p_{x_2}[s+1]|$. This expression means that the range of $v.x_1$'s value is predictable to be in the left vertical shaded rectangle of Figure 3(b). Because of the symmetry between the two triangles with surface axis X_2 , the opposite side $|q.x_1 - (1 - p_{x_2}[s+1])|$ must be taken into account (see Figure 3(b) where the data point v' has the same partition number s as v). Therefore, $l_1 = \min(|q.x_1 - p_{x_2}[s+1]|, |q.x_1 - (1 - p_{x_2}[s+1])|)$. Similarly, the upper bound is $u_1 = \max(|q.x_1 - 0|, |q.x_1 - 1|)$. If q comes into the two vertical shaded rectangles (that is, $0 \leq q.x_1 \leq p_{x_2}[s+1]$ or $p_{x_2}[s+1] \leq q.x_1 \leq 1$) then l_1 is 0. Consequently, $l = (l_1^p + l_2^p)^{1/p}$ and $u = (u_1^p + u_2^p)^{1/p}$, with u_j and l_j ($j = 1, 2$) shown in the Figure 3(b).

Finally, it is worth noting that when there are more than one non-omitted coordinates like $v.x_2$, then the minimum value of them is used to estimate omitted coordinate value.

3 Convex Polyhedra of High Dimensional Space

A d -dimensional unit hypercube is covered by $(d-1)$ -dimensional hyperplanes (surfaces), and some of the $(d-1)$ -dimensional hyperplanes make $(d-2)$ -dimensional tangents. In general, a d -dimensional hypercube is covered by $d-1, d-2, \dots, 0$ -dimensional (hyper-)planes. For example, a 3-dimensional cube is covered by six 2-dimensional surfaces (planes), twelve 1-dimensional surfaces (lines), and eight 0-dimensional surfaces (points). In general, the number of m -dimensional hyperplanes which a d -dimensional hypercube has is $2^{d-m} \cdot \binom{d}{d-m}$.

Now, we explain how to partition a d -dimensional hypercube into convex polyhedra of equal size by using lower dimensional hyperplanes. We start with the explanation of the Pyramid Technique[Berchtold et al., 1998], since it is a special case of our Convex Polyhedra Technique.

In the Pyramid Technique, a d -dimensional unit hypercube is divided into $2d$ pyramids having center of the hypercube $(0.5, 0.5, \dots, 0.5)$ as their top and

$(d-1)$ -dimensional surfaces of the hypercube as their base. Figure 4(a) illustrates this technique when $d = 3$. Since a 3-dimensional cube has six 2-dimensional planes, the Pyramid Technique partitions a cube into six pyramids.

Obviously, each data point in a pyramid has the shortest distance to the base hyperplane than to other surfaces. In this paper, this is called the *pyramid property*. The pyramid property can be mathematically described as follows. For a pyramid π , assume that X_i is the axis which is orthogonal to the pyramid's base (in this case, the pyramid's base can be defined as $x_i = 0$ or $x_i = 1$). In Figure 4(a), the base is defined by

$$x_2 = 0 \quad (1)$$

Let $p(x_1, x_2, \dots, x_d)$ be a data point in π . Then, the following inequation holds:

$$x_i' \leq x_j' \quad (j \neq i)$$

where

$$x_k' = \begin{cases} x_k & (x_k \leq 0.5) \\ 1.0 - x_k & (x_k > 0.5) \end{cases} \quad (1 \leq k \leq d) \quad (2)$$

Hereafter, we call x_k' the *face-altitude* of the dimension k (of the data point $p(x_1, x_2, \dots, x_d)$).

The Pyramid Technique uses $(d-1)$ -dimensional hyperplanes as bases. We generalize this technique so that it uses m -dimensional hyperplanes as bases. The Pyramid Technique partitions a unit hypercube into $2d$ pyramids of equal size, while our technique partitions it into $2^{d-m} \cdot \binom{d}{d-m}$ convex polyhedra of equal size. Thus, we call our technique the *Convex Polyhedra Technique*. Of course, this technique is identical to the Pyramid Technique when $m = d-1$.

Figure 4(b) illustrates our technique when $d = 3$ and $m = 1$. In this case, the base is 1-dimensional (i.e., a line) since $m = 1$. In this figure, a convex polyhedron whose base is a line defined by

$$\begin{cases} x_1 = 1 \\ x_2 = 0 \end{cases} \quad (3)$$

is depicted. Like the Pyramid Technique, each data point in a polyhedron has the shortest distance to

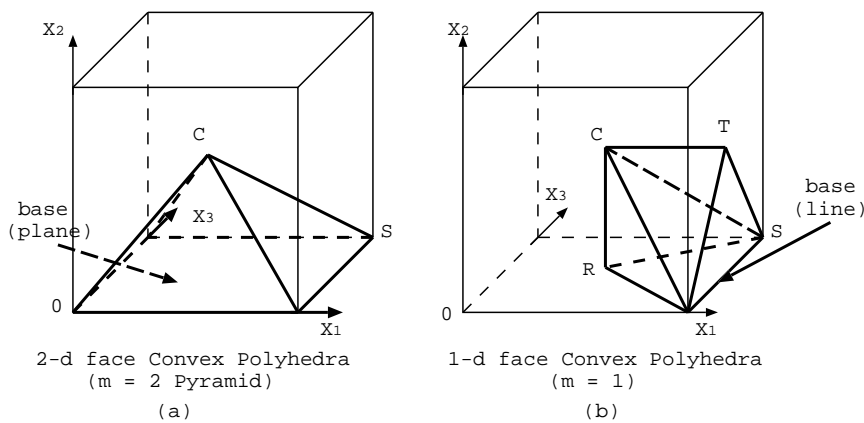


Figure 4: Convex Polyhedra

the base line than to other lines. Thus, the following inequations hold.

$$\begin{cases} x_1' \leq x_3' \\ x_2' \leq x_3' \end{cases}$$

Since a cube has twelve lines, our technique partitions it into twelve convex polyhedra.

In general, a m -dimensional surface can be defined as

$$x_{j_s} = 0 \text{ or } x_{j_s} = 1 \quad (1 \leq s \leq d - m) \quad (4)$$

and the number of such surfaces is $2^{d-m} \cdot \binom{d}{d-m}$, as mentioned above.

Like the pyramid property, we can define the *convex polyhedron property*: for any data point $p(x_1, x_2, \dots, x_d)$, the following inequation holds

$$x_{j_s}' \leq x_{j_t}' \quad (1 \leq s \leq d - m, \quad d - m + 1 \leq t \leq d) \quad (5)$$

where x_{j_i}' and x_{j_j}' are the face-altitudes of the dimension i and j , respectively.

4 An Application: CVA-file

In this section, we show how to apply our approach to other existing indexing techniques. In this paper, a well known indexing technique called “VA-file” is used. Note that our approach is not limited to VA-file.

4.1 VA-file

VA-file ([Weber et al., 1998]) is invented as an index structure which accelerates the sequential scan by the use of data compression. The basic idea of the VA-file is to keep two files: a bit compressed, quantized version of the data points and their exact representation. Both files are unsorted, and the positions of the data points in the two files agree. k -NN queries using VA-file are processed in two phases. In the first phase, the quantized points are loaded into main memory by sequential scanning. The filtering is performed based on the lower and the upper bounds of the distance of the approximative data points from the query point. Better approximations lead to better filtering and less candidates. In the second phase, the candidates which cannot be pruned are refined, that is, their exact coordinates are called in from the data file in the order of their lower bounds of distance. The nearest neighbor algorithm stops when next lower bound is greater than the distance of k -th

actual nearest neighbor found so far. Tighter lower bounds cause the algorithm to stop earlier and visit less number of candidates, and therefore less number of page accesses to the secondary storage caused.

Figure 5 compares the number of page accesses between the first and the second phases for a uniformly 64-dimensional distributed data set of 70,000 data points. It can be seen that in the case of 64-dimensions, page accesses in the first phase is proportional to the number of bits used to partition each dimension. In the second phase, the number of page accesses is tremendous if the prune in the first phase is not enough, the random access used in the second phase is ineffectual.

To make lower and upper bounds tighter, more bits are need for partitioning each dimension. However, using more bits makes the index file larger, and as a consequence, the number of page accesses in the first phase will be bigger.

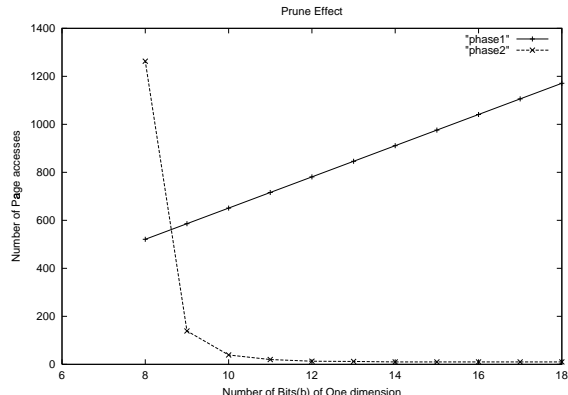


Figure 5: Effect of Pruning

4.2 Dimensionality Reduction by Convex Polyhedra

To address the tradeoff between tightness of bounds and size of index file, we propose a new index file using dimensionality reduction which enables us to use more bits for partitioning while keeping index files small. CVA-file proposed here is constructed in the following way.

For a d -dimensional data set, parameter $m(\leq d)$ is determined, and the data space is partitioned into convex polyhedra based on m -dimensional faces as described in Section 3. $(d - m)$ axes are used to define the bases of convex polyhedra like Equation (1) and (3). Remaining m surface axes are called *effective dimensions*. Moreover, by the number of bits

(b) for approximating each data point v , quantization parameters b_i are determined for each dimension for $i = 1, 2, \dots, d$ as described in [Weber et al., 1998]. Then the index file is constructed by appending the entry of v one by one, in the following steps.

1. Keep a d -bit header.
2. Choose m largest face altitudes of v , and set corresponding bits of the header.
3. Append the entry's approximation of the m -coordinate values corresponding to the bits set in header. Their order is kept as in the coordinates. For each effective dimension i , the approximation is $\lfloor x_i \times 2^{b_i} \rfloor$

Figure 6 shows the structure of VA-file and CVA-file. In CVA-file (b), header fields "dim. inf." preserve effective dimensions for each data point, and the "VA-data" fields preserve data approximation of effective dimensions. Although the header fields are overhead, the size of CVA-file is smaller than VA-file because data approximation of many non-effective dimensions can be omitted.

As an instance, if $d = 4$, $m = 2$, $b_2 = 3$, $b_3 = 2$, and $v = (0.1, 0.3, 0.6, 0.2)$, then the index entry for v is $(0110, 010, 10)_2$, because the second and third dimensions are the effective dimensions thus leads to header "0110", in which the corresponding bits are set. The coordinate value of the second dimension $0.3 \rightarrow \lfloor 0.3 \times 8 \rfloor = 2$ falls into the third partition of the $8 (= 2^{b_2})$ partitions counted from "000", and so forth. In case of two coordinates having equal value during choosing m largest values first larger m -values, priority is set to same as the order of dimensions. Therefore if there is a data point $v = (0, 0.1, 0.6, 0.1)$, then the first coordinate is chosen from the two '0.1' values, and leads to an index entry $(0110, 000, 10)_2$.

4.3 Estimating Bounds in CVA-file

Let m be the number of effective dimensions, N be the number of data points, and d be the number of dimensions. We use $i \in \{1, \dots, N\}$ to range over data points, and the dimensions is divided into two parts, one is non-effective dimensions $X_{j_t} (1 \leq t \leq d - m)$, and the other is effective dimensions $X_{j_t} (d - m + 1 \leq t \leq d)$. v_i denotes an individual data point, and $v_i \cdot j_t$ is v_i 's coordinate value of j_t -th dimension. b is the number of bits required in each approximation, and b_{j_t} indicates the number of bits assigned to dimension X_{j_t} . All notations are summarized in Table 1.

Let us consider a query q and a distance function L_p , for some p . An approximation of v_i determines a lower bound l_i and an upper bound u_i such that:

$$l_i \leq L_p(q, v_i) \leq u_i$$

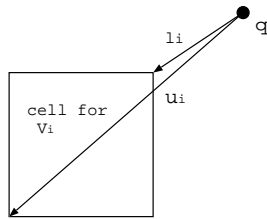


Figure 7: Lower Bound and Upper Bound in VA-file

This is sketched in Figure 7. The lower bound l_i is simply the shortest distance from the query to a point in that cell. Similarly, the upper bound u_i is the longest distance to a point in that cell. Formally, l_i

and u_i are derived as follows. Assume that the dimensionality of hyperface is m , non-effective dimensions are $X_{j_1}, X_{j_2}, \dots, X_{j_{d-m}}$, and the effective dimensions are $X_{j_{d-m+1}}, X_{j_{d-m+2}}, \dots, X_{j_d}$. Then, the bounds l_i and u_i are defined by the equations:

$$l_i = \left(\sum_{t=1}^{d-m} l'_{i \cdot j_t} + \sum_{t=d-m+1}^d l_{i \cdot j_t} \right)^{\frac{1}{p}}$$

$$u_i = \left(\sum_{t=1}^{d-m} u'_{i \cdot j_t} + \sum_{t=d-m+1}^d u_{i \cdot j_t} \right)^{\frac{1}{p}}$$

$l_{i \cdot j_t}$ and $u_{i \cdot j_t}$ are the lower and the upper bounds of effective dimensions $X_{j_t} (d - m + 1 \leq t \leq d)$, respectively. They are computed like VA-file as follows.

$$l_{i \cdot j_t} = \begin{cases} q \cdot j_t - p_{j_t} \lfloor r_{i \cdot j_t} + 1 \rfloor & (r_{i \cdot j_t} < r_q \cdot j_t) \\ 0 & (r_{i \cdot j_t} = r_q \cdot j_t) \\ p_{j_t} \lfloor r_{i \cdot j_t} \rfloor - v_q \cdot j_t & (r_{i \cdot j_t} > r_q \cdot j_t) \end{cases}$$

$$u_{i \cdot j_t} = \begin{cases} q \cdot j_t - p_{j_t} \lfloor r_{i \cdot j_t} \rfloor & \text{if } (r_{i \cdot j_t} < r_q \cdot j_t) \\ \max(q \cdot j_t - p_{j_t} \lfloor r_{i \cdot j_t} \rfloor, p_{j_t} \lfloor r_{i \cdot j_t} + 1 \rfloor - q \cdot j_t) & \text{if } (r_{i \cdot j_t} = r_q \cdot j_t) \\ p_{j_t} \lfloor r_{i \cdot j_t} + 1 \rfloor - q \cdot j_t & \text{if } (r_{i \cdot j_t} > r_q \cdot j_t) \end{cases}$$

As with data points, a query q consists of coordinate values $q \cdot j_t (1 \leq t \leq d)$, and these fall into regions numbered $r_q \cdot j_t$. For effective dimensions, the components $r_{i \cdot j_t} (d - m + 1 \leq t \leq d)$ of v_i are easily extracted.

For non-effective dimensions, $r_{i \cdot j_t} (1 \leq t \leq d - m + 1)$ has not been stored in CVA-file, and thus such components must be estimated. Equation (7) and (8) give the estimation. From the convex polyhedron property mentioned in Section 3, for a data point v_i , the bounds of non-effective dimensions are estimated by Equation (5) in Section 3.

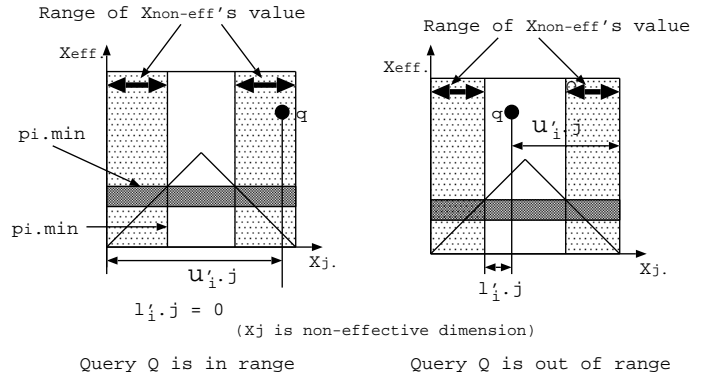


Figure 8: Estimating Bounds of Face Axes

Two lightly shaded rectangles in Figure 8 indicate the bounds of all non-effective dimensions for v_i . The width $pi.min$ of lightly rectangles is computed by selecting an effective dimension which has the smallest face-altitude among effective dimensions. In Figure 8, $pi.min$ is denoted by "Range of $X_{non-eff}$'s Value".

$$pi.min = \min(p'_{j_t} \lfloor r_{i \cdot j_t} \rfloor) \quad (1 \leq t \leq m)$$

$$p'_{j_t} \lfloor r_{i \cdot j_t} \rfloor = \begin{cases} p_{j_t} \lfloor r_{i \cdot j_t} + 1 \rfloor & (p_{j_t} \lfloor r_{i \cdot j_t} \rfloor < 0.5) \\ 1.0 - p_{j_t} \lfloor r_{i \cdot j_t} \rfloor & (p_{j_t} \lfloor r_{i \cdot j_t} \rfloor \geq 0.5) \end{cases} \quad (6)$$

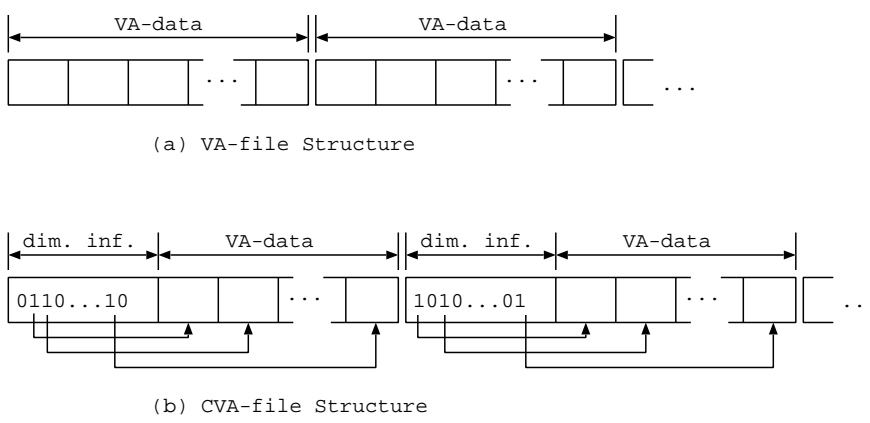


Figure 6: CVA-file Structure

Table 1: Notations and Basic Definitions

m	number of effective dimensions
d	number of dimensions
N	number of data points
i	range over data point, $i \in \{1, \dots, N\}$
v_i	i -th data point
b	number of bits per approximation
$p_{j_t}[\beta]$	β -th partition point in j_t -th dimension
q	a query
l_i, u_i	bounds: $l_i \leq L_p(q, v_i) \leq u_i$
$v_i \cdot j_t$	v_i 's coordinate value of j_t -th dimension
b_{j_t}	bits per approximation in dimension j_t
$r_i \cdot j_t$	region into which v_i falls in dimension j_t
n	number of data points in result set
L_p	distance function $L_p(q, v_i)$
$l_i \cdot j_t', u_i \cdot j_t'$	contribution to l_i, u_i for non-effective dimension j_t ($1 \leq t \leq d - m$)
$l_i \cdot j_t, u_i \cdot j_t$	contribution to l_i, u_i for effective dimension j_t ($d - m + 1 \leq t \leq d$)

Any non-effective dimension's coordinate value of v_i must be in the two lightly shaded rectangles of the figure, so we can compute the lower and the upper bounds of v_i from query point q , according to whether q is inside or outside of the range.

$$l_i \cdot j_t = \begin{cases} 0 & \text{if } (1 - p_i \cdot \text{min} < q \cdot j_t) \\ \min(q \cdot j_t - p_i \cdot \text{min}, (1 - p_i \cdot \text{min}) - q \cdot j_t) & \text{if } (p_i \cdot \text{min} \leq q \cdot j_t \leq 1 - p_i \cdot \text{min}) \\ 0 & \text{if } (v_q \cdot j_t < p_i \cdot \text{min}) \end{cases} \quad (7)$$

$$u_i \cdot j_t = \begin{cases} 1 - v_q \cdot j_t & \text{if } (1 - p_i \cdot \text{min} < v_q \cdot j_t) \\ \max(1 - v_q \cdot j_t, v_q \cdot j_t) & \text{if } (p_i \cdot \text{min} \leq v_q \cdot j_t \leq 1 - p_i \cdot \text{min}) \\ q \cdot j_t & \text{if } (v_q \cdot j_t < p_i \cdot \text{min}) \end{cases} \quad (8)$$

5 Performance Evaluation

We evaluate the performance of the proposed mechanism with a real data set of color images available from the Corel Database Color Database (<http://corel.digitalriver.com/>) and color histograms provided online at the UCI KDD Archive web site (<http://kdd.ics.uci.edu/databases/CorelFeatures>). The size of the dataset is 70,000. 4, 8, 16, ..., 64-dimensional data points are extracted from the data set. The size of page is set to 8K bytes. The distance function is of the L_2 metric (Euclidean distance).

As mentioned above, the VA-file approach sequentially scans all data points in index to filter candidates in the first phase. This contributes to the gain of the performance because the cost of sequential access is significantly lower than random access which rises in the second phase. A factor of 5 to 10, which means that random access is five times or more slower than sequential access, is assumed in VA-file [Weber et al., 1998]. Having no hint on this factor, yet to compare with VA-file fairly, we evaluate the performance in different way.

Let N be the size of the data set, b_i be the number of bits of one dimension of an index entry. Let \bar{b}_i be the average of b_i , then the size of VA-file is $\bar{b}_i d N$. On the other hand, in CVA-file, the number of bits b_j are assigned to each dimension. Let m be the number of effective dimensions and \bar{b}_j be the average of b_j , then the size of CVA-file is $(\bar{b}_j m + d) N$. Thus, the size of CVA-file compared to VA-file can be expressed as follows.

$$(\bar{b}_j m + d) N / \bar{b}_i d N = (\bar{b}_j m + d) / \bar{b}_i d \quad (9)$$

In the simplest case when we assign to each dimension the same number of bits as in VA-file, namely $\bar{b}_j \doteq \bar{b}_i$, this expression is simplified to

$$\begin{aligned} (\bar{b}_j m + d) N / \bar{b}_i d N &= (\bar{b}_j m + d) / \bar{b}_i d \\ &= m/d + 1/\bar{b}_i \end{aligned} \quad (10)$$

Since \bar{b}_i is about 7 or 8 [Weber et al., 1998], that is $1/\bar{b}_i \doteq 0$, the size of CVA-file is smaller than VA-file at the rate of about m/d . Since the number of page accesses in phase one is in proportion to size of VA-file (or CVA-file), CVA-file index mechanism vastly

cuts down the number of page accesses in phase one. This compensates excess page accesses in phase two because of less tight bounds than VA-file.

Because the factor of random access and sequential access are not stable, we consider two extreme situation.

Firstly, we control CVA-File to have the same number of page accesses as VA-File in the second phase by tuning the parameter m . Figure 9 shows effective dimensions of CVA-file comparing with that of VA-file which has the same number of dimensions as original dimensionality. In Figure 10, the number of sequentially accessed pages in the first phase is compared, and CVA-File outperforms VA-File. The effect becomes significant as the dimensionality increases.

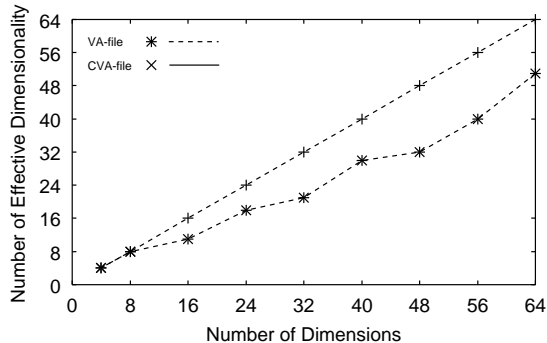


Figure 9: The Effect of Dimensionality Reduction(phase one only)

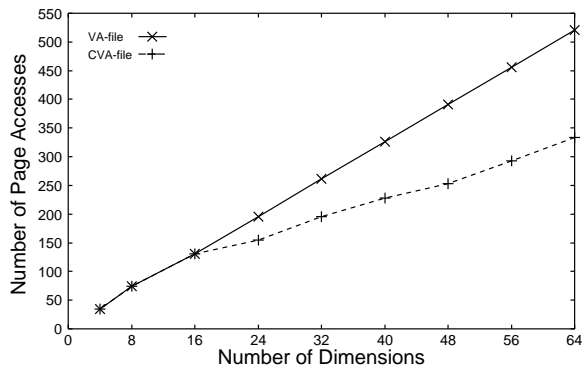


Figure 10: Comparison of Number of Page Accesses

Secondly, we compare the total number of page accesses among SR-tree, VA-File and CVA-File. Certainly, a tradeoff is that CVA-file gives loose bounds than VA-file, and hence requires more page accesses in the second phase. In order to investigate the efficiency in total, we compare the number of page accesses of our approach to other methods on the above-mentioned real data set. Figure 12 shows the results. For each of dimensionality of 4, 8, 16, 24, ..., 64, we perform 10-nearest neighbor queries. For each dimensionality, the number of bits assigned to each dimension for VA-file is chosen to 8 bits for 4 to 24 dimensionality and 7 bits for 32 to 64 dimensionality. These decisions are made according to [Weber et al., 1998] which concludes that this choice gives them the best approximation file.

Our experiments measured the number of the most effective dimensions of CVA structure. The average of the number of effective dimensions is shown in Figure 11. The results indicates that there is a slight increase of the number of effective dimensions in high dimensionality. Comparing to VA-file which is linear

to dimensionality of the dataset, our approach reduce the dimensionality to the number of effective dimensions, which is significantly smaller than the original dimensionality. Furthermore, the effect of the reduction becomes more remarkable as the dimensionality becomes higher.

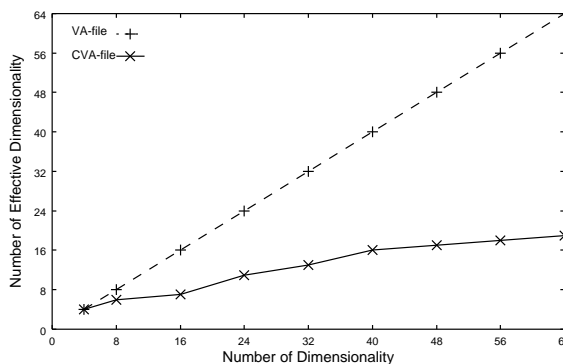


Figure 11: The Effect of Dimensionality Reduction(total)

Figure 12 shows the number of page accesses in various dimensions. We can observe that CVA-file always outperforms VA-files, and it outperforms SR-tree[Katayama et al., 1997] for high-dimensional dataset. The margin increases as the number of dimensionality increases. For the case of 64-dimension, the number of page accesses of CVA-file is almost 2/3 of SR-tree ([Katayama et al., 1997]), and 1/2 of VA-file.

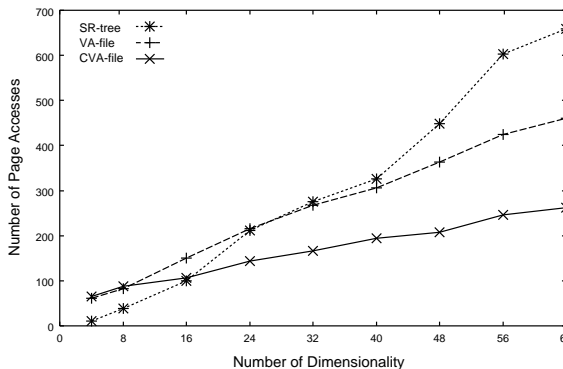


Figure 12: Comparison of Three Approaches Mechanisms

6 Conclusions

In this paper, we observed the skewed distribution of high dimensional real data set, and proposed a new dimensionality reduction technique. An application of our technique to VA-file implements another index file CVA-file. The performance evaluation shows significant improvements on the I/O cost of queries over original VA-file approach for several real datasets. The proposed approach of dimensionality reduction can be used for any applications in which similarity, distance and nearest neighbor search are treated.

As future work, we are planning to study the effect of different data patterns of coordinates in high dimensionality, by varying effective dimensions or by dynamically assigning effective dimensions. We are also considering tighter bounds to enhance the filtering effect of CVA-file.

References

- [Aggarwal et al., 1999] Aggarwal C., Procopiuc C., Wolf J., Yu P, Park J. (1999) Fast Algorithms for Projected Clustering. In *ACM SIGMOD International Conference on Management of Data*, pages 61-72.
- [Aggarwal et al., 2000] Aggarwal Charu C., Yu Philip S. (2000) Finding Generalized Projected Clusters in High Dimensional Spaces. In *ACM SIGMOD International Conference on Management of Data*, pages 70-81.
- [Berchtold et al., 1997] Berchtold S., Bohm C., Keim D., Kriegel H.-P. (1997) A Cost Model For Nearest Neighbor Search in High-Dimensional Data Space. In *ACM PODS Symposium on Principles of Database Systems*, pages 78-86.
- [Berchtold et al., 1998] Berchtold S., Keim D., Kriegel H.-P.(1998) The pyramid-Technique: Towards Breaking the Curse of Dimensional Data Spaces. In *ACM SIGMOD International Conference on Management of Data*, pages 142-153.
- [Beckmann et al., 1990] Beckmann N., Kriegel H.-P., Schneider R., Seeger B.(1990) The R*-tree: An efficient and Robust Access Method for Points and Rectangles. In *ACM SIGMOD International Conference on Management of Data*, pages 322-331.
- [Beyer et al., 1999] Beyer K. S., Goldstein J., Ramakrishnan R., Shaft U.(1999) When Is "Nearest Neighbor" Meaningful. In *Proc. of the 7th Int. Conf. on Database Theory Jerusalem, Israel* pages 217-235.
- [Fukunaga., 1990] Fukunaga K.(1990) Statistical Pattern Recognition. Academic Press.
- [Faloutsos et al., 1995] Faloutsos C., Lin K. I.(1995) FastMap: A Fast Algorithm for Indexing, Data Mining and Visualization of Traditional and Multimedia Datasets. In *ACM SIGMOD International Conference on Management of Data*, pages 163-174. [Beckmann et al., 1990]
- [Katayama et al., 1997] Katayama N., Satoh S.(1997) The SR-tree: An Index Structure for High-Dimensional Nearest Neighbor Queries. In *ACM SIGMOD International Conference on Management of Data*, pages 369-380.
- [Shinohara et al., 2000] Shinohara T., An J., Ishizaka H.(2000) Approximate Retrieval of High-dimensional Data with L1 Metric by Spatial Indexing. *journal of New Generation Computing*, Vol. 18, No. 1(2000), pages. 39-47.
- [Weber et al., 1998] Weber R., Schek H. J., Blott. S.: A Quantitative Analysis and Performance Study for Similarity-Search Methods in high-Dimensional Spaces. In *Proc. of the VLDB conference, New York*, pages 194-205.