Efficient Similarity Search on Vector Sets

Hailing Yu Wei Niu Divyakant Agrawal Amr El Abbadi Ambuj K Singh
University of California at Santa Barbara
Computer Science Department
Santa Barbara, 93106, USA
{hailing,kevinniu,agrawal,amr,ambuj}@cs.ucsb.edu

ABSTRACT

Similarity search in applications such as multimedia databases has been gaining a lot of attention in recent years. Due to the curse of dimensionality, it is hard to improve the query cost of similarity search in a high dimensional space. The problem is getting even worse when objects are represented using sets of local feature vectors, since the distance measurement among vector sets is the minimum matching distance. In the minimum matching distance, the vectors from two sets are paired to minimize the sum of the distance of all pairs while the distance measurements over single feature vectors do not need the process of pairing. In this paper, we extend the minimum matching distance to the Euclidean minimum matching distance and the Manhattan minimum matching distance since Euclidean and Manhattan distances are commonly used in similarity search over single feature vectors. Then we propose a novel filtering technique to reduce the query cost of similarity search on both Euclidean and Manhattan minimum matching distances. The experimental results show that our technique reduces the query cost significantly.

1. INTRODUCTION

Similarity search has been gaining a lot of attention due to its wide application in the areas such as image retrieval [8], multimedia [13], bioinfomatics [3], and spatial databases [9]. A commonly used method for efficient similarity search is to map objects into points in a high dimensional space. For instance, in image databases, an image is represented using a global feature vector, which captures information about color distribution, texture patterns, and shape descriptors of the image. Similarity of two objects is defined as the distance between their respective feature vectors, where the distance is computed using a metric defined in the feature vector space, such as Euclidean distance or Manhattan distance. Since image databases usually contain millions of images and the vector space is a very high dimensional space, finding the images which are similar to a given query image is too expensive if we simply compare the given image to all images in the databases. The high cost is due to the I/O time of retrieving all images and the computation cost of comparing each pair. Hence, numerous methods have been proposed to improve this query cost. These methods can be divided into three classes. Indexing techniques, such as TV-tree [11], M-tree [4], and X-tree [2], are developed to tackle this problem by pruning the search space. However, the index structures are subject to the curse of dimensionality. Dimension reduction techniques have been proposed to map a feature vector to a vector with lower dimensions. The distance metric in the lower-dimensional space is a lower bound on the distance metric in the original high-dimensional space [13, 8]. Therefore images with larger distance can be pruned in the lower-dimensional space. Moreover, indexing and dimension reduction techniques can be combined together to answer queries more efficiently. Some proposals such as VA-file [15] and A-tree [14] use compression to reduce the size of data processed at the filter step, therefore the cost of computing the query results at the refinement step is improved dramatically. Recently, Vries et al. [6] considered the problem of improving query performance as a physical database design problem. They proposed to use the Decomposition Storage Model (DSM), which has been used in the context of relational database systems [5], to store feature vectors. The data of each dimension is maintained in a separate table. Based on this physical design, the authors presented a search technique called Branch-and-bound On Decomposed data (BOND).

In recent years, some applications need an object to be represented as a set of feature vectors instead of a global feature vector. For example, in image databases, since images may be rotated, transformed, slightly changed, or presented in different formats for copyright protection, in order to still recognize them as similar images, one feature vector is not good enough to describe an image. Hence an image is divided into regions, and feature vectors are computed for each region which then form a set of feature vectors. The similarity of two images uses a distance metric which is based on their sets of feature vectors. Another example is in handwriting recognition, the different handwritten versions of one character usually appear clearly different. Therefore it is hard to identify them as similar if their feature vectors are obtained from their entire images. However in general, it is possible that parts of the character are similar, i.e., the distance between the sub-parts of the different versions is small. Hence it is often beneficial to represent characters using several feature vectors. Kriegel et al. [9] proposed to use feature vector sets for similarity search on voxelized CAD objects. The distance measurement on vector sets is defined as the minimal matching distance of these sets. Even though a lot of work have been proposed to improve the similarity search on objects denoted by single feature vectors, they can not be directly applied to the similarity search over feature vector sets. Since the distance metric for objects using sets of feature vectors needs to consider matchings among several local feature vectors, the computation

cost is much higher than the one on single feature vectors. The method proposed in [10, 12] to compute the distance among sets has $O(m^3 + m^2n)$ time complexity (m is the number of vectors in each set, n is the number of dimensions of each vector). Even with relatively small values of m, the computation cost becomes very high, hence in [9], the authors proposed a filter technique, *the extended centroid method*, for fast similarity search over sets of feature vectors. The distance of the extended centroids of two sets of feature vectors is proved to be a lower bound of the minimum matching distance of these two sets, therefore some objects can be filtered out if the distance of their extended centroids is larger than the given bound.

Due to the popularity of the Euclidean and Manhattan distances in the context of similarity search, we define two new minimum matching distance metrics on vector sets which are based on the Euclidean and Manhattan distances respectively, referred to as *Euclidean minimum matching distance* and *Manhattan minimum matching distance*. The query type we consider is the *similarity range query* [1]. We propose a novel filter technique which can be applied to both distance metrics, and show that the similarity query cost can be reduced dramatically. Similar to the other approaches using filter techniques, our query processing algorithm consists of two steps: filter and refinement. At the filter step, we use our proposed filter to filter out most unqualified objects. During the refinement step, the exact distance using the method in [10, 12] is computed to determine the final results. The filter technique we proposed is a multi-step filter. In other words, given a similarity range query, the filter computes the lower bound of the distance of two objects step by step. At each step, if the computed distance of one object to the given object is larger than the given bound, this object is filtered out and will not be considered in subsequent steps.

The rest of the paper is organized as follows. Section 2 gives the definitions of different distance metrics. The proposed filter technique and related optimizations are presented in Section 3. Section 4 shows the experimental results. We conclude the paper with Section 5.

2. DEFINITIONS

In this section, we first define two distance metrics between two objects each represented as a set of feature vectors. Then we give the definition of the type of queries, similarity range queries, which are considered in this paper.

An object x is represented as a set of feature vectors, V_x. Let m be the number of feature vectors in the set, n be the number of dimensions of a single vector, $V_x(i)$ ($1 \leq i \leq m$) be the ith feature vector in V_x, and $v^x_{i,j}$ ($1 \leq j \leq n$) be the value of the jth dimension of the ith feature vector in a set. For the sake of simplicity, we assume all sets have the same number of feature vectors and all feature vectors have the same number of dimensions. If sets contain different numbers of feature vectors, we extend the smaller vector sets by adding some zero vectors [9].

There are several distance metrics defined over two feature vector sets. In [9], the authors argued that similarity can be defined as the distance metric of the minimum weight perfect matching of two feature vectors. Hence we define our two distance measurements of vector sets based on the minimum matching distance. Since the Euclidean and Manhattan distances are commonly used distance measurements to define similarity of objects, we define *the Euclidean minimum matching distance* and *the Manhattan minimum matching distance* respectively. Before giving the formal definitions, we first define *the Euclidean sum* and *the Manhattan sum* as follows.

DEFINITION 1. *Given a set $S = \{d_1, d_2, \ldots, d_n\}$, the Euclidean sum S_{sum} of set S is defined using the following formula:*

$$S_{sum} = (\sum_{i=1}^{n} d_i^2)^{1/2}.$$

DEFINITION 2. *Given a set $S = \{d_1, d_2, \ldots, d_n\}$, the Manhattan sum S_{sum} of set S is defined using the following formula:*

$$S_{sum} = \sum_{i=1}^{n} d_i.$$

Based on the definitions of the Euclidean sum and the Manhattan sum, we define the Euclidean minimum matching distance and the Manhattan minimum matching distance as follows.

DEFINITION 3. *The Euclidean minimum matching distance of two objects is a distance function: Dom $R^m \times$Dom $R^m \rightarrow$ Dom R. Let $V_x = \{V_x(1), V_x(2), \ldots, V_x(m)\}$ and $V_y = \{V_y(1), V_y(2), \ldots, V_y(m)\}$ be sets of feature vectors for objects x and y respectively. The distance between x and y is defined as follows:*

$$d_{mm} = min((\sum_{i=1}^{m} (\parallel V_x(i), V_y(p(i)) \parallel_2)^2)^{1/2}).$$

Where $p(i)$ is a permutation of the vectors in V_y that minimizes d_{mm}, and $\|\|_2$ denotes the Euclidean distance.

In this definition, vector $V_x(i)$ from the vector set of object x matches one and only one vector $V_y(p(i))$ from the vector set of object y, and the distance between $V_x(i)$ and $V_y(p(i))$ is computed using the Euclidean distance. Finally, the Euclidean minimum matching distance is the one minimizing the Euclidean sum of the distances of all vector pairs. Similarly, the Manhattan minimum matching distance can be defined except that the distance of vector pairs is computed using the Manhattan distance and the Manhattan minimum matching distance is the minimal Manhattan sum of the distance of all vector pairs.

DEFINITION 4. *The Manhattan minimum matching distance of two objects is a distance function: Dom $R^m \times$Dom $R^m \rightarrow$ Dom R. Let $V_x = \{V_x(1), V_x(2), \ldots, V_x(m)\}$ and $V_y = \{V_y(1), V_y(2), \ldots, V_y(m)\}$ be two sets of feature vectors of objects x and y respectively. Then the Manhattan minimum matching distance is defined as follows:*

$$d_{mm} = min(\sum_{i=1}^{m} \parallel V_x(i), V_y(p(i)) \parallel_1).$$

Where $p(i)$ is a permutation of vectors in V_y minimizing d_{mm}, and $\|\|_1$ denotes the Manhattan distance.

The type of queries we consider in this paper is *similarity range query* [1]. Given a query object o and a range value ε, we want to find all objects in the database which have distances to the query object o no larger than ε.

DEFINITION 5. *Similarity range query: For a query object $o \in R^n$, and a query range ε, the query reports all objects x in the database satisfying:*

$$d_{mm}(o, x) \leq \varepsilon.$$

After having defined the distance metrics and query types, we now define some terms which are used in our filter techniques. Each object is represented using a set of feature vectors, which can be viewed as a $m \times n$ matrix: each row is a feature vector, and each column is composed of feature values of all vectors in one dimension (we will use both dimension and column alternately). For an object x, $C(j)$ denotes the jth column. Therefore, $C_x(j) = \{v^x_{1,j}, v^x_{2,j}, \ldots, v^x_{m,j}\}^T$, where m is the number of vectors in a set. Two columns of two sets are called a *column pair* if they are in the same dimension. For example, $C_x(1)$ and $C_y(1)$ are the first columns of sets V_x and V_y respectively, and they form a column pair. If the number of dimensions is n, then the number of column pairs of two vector sets is n. A column-pair minimum matching distance is defined as follows:

DEFINITION 6. *A column-pair minimum matching is a distance function f: Dom$R^m \times$Dom$R^m \rightarrow$ Dom$R$. Let $C_x(j) = \{v^x_{1,j}, v^x_{2,j}, \ldots, v^x_{m,j}\}^T$ and $C_y(j) = \{v^y_{1,j}, v^y_{2,j}, \ldots, v^y_{m,j}\}^T$ be the two jth columns of vector sets for objects x and y respectively. Then the minimum matching distance of this column pair is defined as follows:*

$$d^c_{mm} = min(\sum_{i=1}^{m} dist(v^x_{i,j}, v^y_{p_j(i),j})).$$

Where $p_j(i)$ is a permutation of the elements in $C_y(j)$ minimizing d^c_{mm}.

Note that a column-pair minimum matching distance is a special case of the minimum matching distance, since it can be treated as vector sets where each vector is one-dimensional. In Definition 6, the distance between $v^x_{i,j}$ and $v^y_{p_j(i),j}$ is either computed using the Euclidean distance or the Manhattan distance, we refer to them as *the Euclidean-based column-pair minimum matching distance* and *the Manhattan-based column-pair minimum matching distance* respectively.

3. DIVIDE-AND CONQUER FILTER (DCF)

In this section, we present the algorithms to compute the filter values for both Euclidean minimum matching distance and Manhattan minimum matching distance. Then we give the theoretical analysis and discuss some optimization issues.

3.1 DCF Algorithm for the Euclidean Minimum Matching Distance

We first describe Divide-and-conquer Filter for the Euclidean minimum matching distance, then we present the algorithm for processing similarity range queries where the Euclidean minimum matching distance is considered.

Given a query object o with vector set V_o, for any object x with vector set V_x in the search database DB, the Euclidean sum of all Euclidean-based column-pair minimum matching distances is a lower bound of the Euclidean minimum matching distance between sets V_o and V_x. Since the distance of each column pair can be computed separately, our filter technique computes the lower bound column by column. During the computation, if the partial Euclidean sum of the considered column-pairs in V_o and V_x is larger than the given bound ε, the Euclidean minimum matching distance between object o and object x will be larger than ε, hence object x cannot be in the result set and does not need to be considered further. Since our filter technique relies on the Euclidean-based column-pair minimum matching distance of a column pair, we first give the algorithm on how to compute it, which is shown in Algorithm 1

Algorithm 1 Compute the Euclidean-based column-pair minimum matching distance

Input:
x and y are two objects with vector sets V_x and V_y;
$C_x(j)$ and $C_y(j)$ are the two jth columns of V_x and V_y;
//j denotes one dimension;
Procedure:
Sorted $C_x(j)$ in ascending order and results are stored in $C^s_x(j)$;
Sorted $C_y(j)$ in ascending order and results are stored in $C^s_y(j)$;
$sum = 0$;
for i from 1 to m **do**
  $v^{xs}_{i,j}$ from column $C^s_x(j)$;
  $v^{ys}_{i,j}$ from column $C^s_y(j)$;
  $sum+ = (v^{xs}_{i,j} - v^{ys}_{i,j})^2$;
end for
$sum = sum^{1/2}$;
Output:
Return sum.

In Algorithm 1, we first sort the two columns, then pair the elements in these two column based on their sorted order. The entire process is shown in Figure 1. Algorithm 1 returns the Euclidean-based column-pair minimum matching distance of two columns. Theorem 1 guarantees the correctness of this algorithm.

THEOREM 1. *The Euclidean-based column-pair minimum matching distance of a column pair is the Euclidean distance of these two columns where the elements in these two columns are paired up based on their sorted order.*

Proof: We use induction to prove Theorem 1. Figure 2(a) shows the basic case: columns C_1 and C_2 contain two elements. Assume they are ordered, i.e., $a_1 \leq a_2$ and $b_1 \leq b_2$. the column pair of minimum matching distance between C_1 and C_2 is $((a_1 - b_1)^2 + (a_2 - b_2)^2)^{1/2}$. Since in this case, there are only two options of pairing: $a_1 \leftrightarrow b_1$ and $a_2 \leftrightarrow b_2$, or $a_1 \leftrightarrow b_2$ and $a_2 \leftrightarrow b_1$. We need to prove that

$$((a_1 - b_1)^2 + (a_2 - b_2)^2)^{1/2} \leq ((a_1 - b_2)^2 + (a_2 - b_1)^2)^{1/2}.$$

Since,

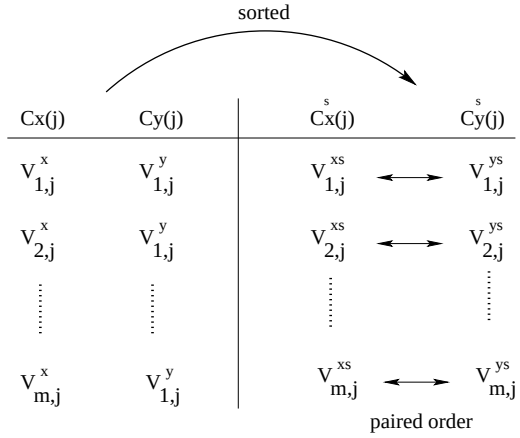

Figure 1: The demonstration of Algorithm 1

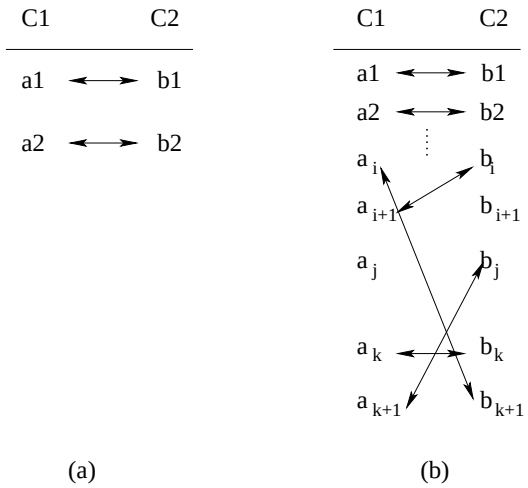

Figure 2: Proof of Theorem 1

$$e = ((a_1 - b_1)^2 + (a_2 - b_2)^2) - ((a_1 - b_2)^2 + (a_2 - b_1)^2)$$

$$= -2a_1b_1 - 2a_2b_2 + 2a_1b_2 + 2a_2b_1$$

$$= 2(b_2 - b_1)(a_1 - a_2).$$

We have $a_1 \leq a_2$ and $b_1 \leq b_2$, therefore $e \leq 0$, i.e., $a_1 \leftrightarrow b_1$ and $a_2 \leftrightarrow b_2$ are the pairs which achieve the column-pair minimum matching distance of columns C_1 and C_2.

In the induction hypothesis step, we assume Theorem 1 is correct when the number of elements in two columns is k. We need to prove that the statement holds for $k+1$. The two columns with $k+1$ elements are shown in Figure 2(b), where $a_1 \leq a_2 \leq \ldots \leq a_{k+1}$ and $b_1 \leq b_2 \leq \ldots \leq b_{k+1}$. If a_{k+1} is paired with b_{k+1}, then a_i will be paired with b_i according to the induction hypothesis since $1 \leq i \leq k$. In this case, the statement in Theorem 1 holds since $a_1 \leq a_2 \leq \ldots \leq a_{k+1}$ and $b_1 \leq b_2 \leq \ldots \leq b_{k+1}$. Now we need to prove that a_{k+1} has to be paired with b_{k+1} in order to achieve the column-pair minimum matching distance between C_1 and C_2. Now we assume a_{k+1} is not paired with b_{k+1}, i.e., a_{k+1} is paired with b_j and b_{k+1} is paired with a_i without loss of generality, as shown in Figure 2(b). Similar to the basic case, since $a_i \leq a_{k+1}$ and $b_j \leq b_{k+1}$, if the pairs are $a_i \leftrightarrow b_{k+1}$ and $a_{k+1} \leftrightarrow b_j$, we have the result that $((a_i - b_{k+1})^2 + (a_{k+1} - b_j)^2)^{1/2} \geq ((a_i - b_j)^2 + (a_{k+1} - b_{k+1})^2)^{1/2}$. It means that in order to achieve the column-pair minimum matching distance, the pairs should be $a_i \leftrightarrow b_j$ and $a_{k+1} \leftrightarrow b_{k+1}$ instead of $a_i \leftrightarrow b_{k+1}$ and $a_{k+1} \leftrightarrow b_j$. After deciding that a_{k+1} is paired with b_{k+1}, there are k elements in each column left. Based on the assumption, they have to be paired according to the sorted order. Hence, Theorem 1 holds when the number of elements in each column is $k + 1$.□

Algorithm 2 DCF algorithm for the Euclidean minimum matching distance

Input:
q is the query object with vector set V_q;
ε is the given bound;
Procedure:
$partial\_sum = 0$;
sum_j is the jth column-pair minimum matching distance;
for Each column j in vector sets **do**
  Sorted $C_q(j)$ and results are stored in $C_q^s(j)$;
  for Each object x in the search database **do**
    Sorted $C_x(j)$ and results are stored in $C_x^s(j)$;
    compute sum_j using Algorithm 1;
    $partial\_sum = (partial\_sum^2 + {sum_j}^2)^{1/2}$;
    if $partial\_sum \geq \varepsilon$ **then**
      object x does not need further consideration.
    end if
  end for
end for
Output:
Return all the objects left.

In Algorithm 2, we give the filtering algorithm. The returned results are the objects whose $partial\_sum$ values are smaller than the bound ε. Theorem 2 guarantees that the result set returned by Algorithm 2 is a superset of the final result set for the similarity range query with query object q and bound ε.

THEOREM 2. *The Euclidean sum of the Euclidean-based column-pair minimum matching distance of all column pairs of two sets is a lower bound of their Euclidean minimum matching distance.*

Proof: For two objects x and y with vector sets V_x and V_y, $\mid V_x \mid = \mid V_y \mid = m$, and each vector has n dimensions. The jth columns are denoted as $C_x(j)$ and $C_y(j)$, and their Euclidean-based column-pair minimum matching distance is $sum_j = (\sum_{i=1}^{m}(v_{i,j}^x - v_{p_j(i),j}^y)^2)^{1/2}$, where $p_j(i)$ is the permutation minimizing sum_j. We need to prove that $(\sum_{j=1}^{n}({sum_j}^2))^{1/2}$ is smaller than their Euclidean minimum matching distance $\sum_{i=1}^{m}(\sum_{j=1}^{n}(v_{i,j}^x - v_{p(i),j}^y)^2)^{1/2}$, where $p(i)$ is the permutation in their Euclidean minimum matching distance. Since $p_j(i)$ is the permutation optimizing each Euclidean-based column-pair minimum matching distance while $p(i)$ is the permutation to minimize their Euclidean minimum matching distance, for the jth column, the column-pair matching distance using $p(i)$ is no smaller than the one using $p_j(i)$. Therefore we have:

$$(\sum_{j=1}^{n}({sum_j}^2))^{1/2} = (\sum_{j=1}^{n}(\sum_{i=1}^{m}(v_{i,j}^x - v_{p_j(i),j}^y)^2))^{1/2}$$

$$\leq (\sum_{j=1}^{n}(\sum_{i=1}^{m}(v_{i,j}^x - v_{p(i),j}^y)^2))^{1/2}$$

$$= (\sum_{i=1}^{m}(\sum_{j=1}^{n}(v_{i,j}^x - v_{p(i),j}^y)^2))^{1/2}.\square$$

In the DCF algorithm, we compute filter values column-by-column. Theorem 2 guarantees that all objects filtered will not be in the result sets. After computing the minimum matching distance using algorithm in [10, 12] at the refinement step, we will answer similarity range queries correctly, i.e., all objects which distances to the query object is no larger than the given bound ε are in the query result set.

3.2 DCF Algorithm for the Manhattan Minimum Matching Distance

The algorithm for computing the Manhattan-based column-pair minimum matching distance is similar to Algorithm 1. The differences are that we replace the Euclidean distance with the Manhattan distance in compute a column-pair minimum matching distance, and replace the Euclidean sum with the Manhattan sum. The algorithm is shown in Algorithm 3. Now we prove that Algorithm 3 returns the Manhattan-based column-pair minimum matching distance of a column pair.

Algorithm 3 Compute the Manhattan-based column-pair minimum matching distance

Input:
x and y are two objects with vector sets V_x and V_y;
$C_x(j)$ and $C_y(j)$ are the two lth columns of V_x and V_y;
//l denotes one dimension;
Procedure:
Sorted $C_x(j)$ in ascending order and results are stored in $C_x^s(l)$;
Sorted $C_y(j)$ in ascending order and results are stored in $C_y^s(l)$;
$sum = 0$;
for i from 1 to m **do**
  $v_{i,j}^{xs}$ from column $C_x^s(j)$;
  $v_{i,j}^{ys}$ from column $C_y^s(j)$;
  $sum+ =\mid v_{i,j}^{xs} - v_{i,j}^{ys} \mid$;
end for
Output:
Return sum.

THEOREM 3. *The Manhattan-based column-pair minimum matching distance of a column pair is the Manhattan distance of these two columns where the elements in these two columns are paired up based on their sorted order.*

Proof: We use induction to prove Theorem 3. Figure 2(a) shows the basic case: columns C_1 and C_2 contain two elements. Assume they are ordered, i.e., $a_1 \leq a_2$ and $b_1 \leq b_2$. the column pair of minimum matching distance between C_1 and C_2 is $\mid a_1 - b_1 \mid + \mid a_2 - b_2 \mid$. Since in this case, there are only two options of pairing: $a_1 \leftrightarrow b_1$ and $a_2 \leftrightarrow b_2$, or $a_1 \leftrightarrow b_2$ and $a_2 \leftrightarrow b_1$. We need to prove that $(\mid a_1 - b_1 \mid + \mid a_2 - b_2 \mid) \leq (\mid a_1 - b_2 \mid + \mid a_2 - b_1 \mid)$. There are four cases to consider:

- Case 1: $a_1 \leq a_2 \leq b_1 \leq b_2$. We have $\mid a_1 - b_1 \mid + \mid a_2 - b_2 \mid = b_2 - a_2 + b_1 - a_1$ and $\mid a_1 - b_2 \mid + \mid a_2 - b_1 \mid = b_2 - a_1 + b_1 - a_2$. Then we get $(\mid a_1 - b_1 \mid + \mid a_2 - b_2 \mid) = (\mid a_1 - b_2 \mid + \mid a_2 - b_1 \mid)$, i.e., the inequality holds.
- Case 2: $a_1 \leq b_1 \leq a_2 \leq b_2$. We have $\mid a_1 - b_1 \mid + \mid a_2 - b_2 \mid = b_2 - a_1 + b_1 - a_2$ and $\mid a_1 - b_2 \mid + \mid a_2 - b_1 \mid = b_2 - a_1 + a_2 - b_1 \geq b_2 - a_1 + b_1 - a_2$ since $a_2 \leq b_1$. Therefore $(\mid a_1 - b_1 \mid + \mid a_2 - b_2 \mid) \leq (\mid a_1 - b_2 \mid + \mid a_2 - b_1 \mid)$.
- Case 3: $b_1 \leq b_2 \leq a_2 \leq a_1$. This case is similar to Case 1, and we can obtain that $(\mid a_1 - b_1 \mid + \mid a_2 - b_2 \mid) = (\mid a_1 - b_2 \mid + \mid a_2 - b_1 \mid)$.
- Case 4: $b_1 \leq a_1 \leq b_2 \leq a_2$. The analysis for this case is similar to Case 2, we can get $(\mid a_1 - b_1 \mid + \mid a_2 - b_2 \mid) \leq (\mid a_1 - b_2 \mid + \mid a_2 - b_1 \mid)$.

In conclusion, we have $(\mid a_1 - b_1 \mid + \mid a_2 - b_2 \mid) \leq (\mid a_1 - b_2 \mid + \mid a_2 - b_1 \mid)$, which means that $a_1 \leftrightarrow b_1$ and $a_2 \leftrightarrow b_2$ are the right pairs to achieve the column-pair minimum matching distance of columns C_1 and C_2.

The following part of the proof is similar to that for Theorem 1 with some slight changes. In the induction hypothesis, we assume Theorem 3 is correct when the number of elements in two columns is k. We need to prove that the statement holds for $k + 1$. The two columns with $k + 1$ elements are shown in Figure 2(b), where $a_1 \leq a_2 \leq \ldots \leq a_{k+1}$ and $b_1 \leq b_2 \leq \ldots \leq b_{k+1}$. If a_{k+1} is paired with b_{k+1}, then a_i will be paired with b_i according to the assumption in the assumption step since $1 \leq i \leq k$. In this case, the statement in Theorem 3 holds since $a_1 \leq a_2 \leq \ldots \leq a_{k+1}$ and $b_1 \leq b_2 \leq \ldots \leq b_{k+1}$. Now we need to prove that a_{k+1} has to be paired with b_{k+1} in order to achieve the column-pair minimum matching distance between C_1 and C_2. Now without loss of generality, we assume a_{k+1} is not paired with b_{k+1}, i.e., a_{k+1} is paired with b_j and b_{k+1} is paired with a_i, as shown in Figure 2(b). Similar to the basic case, since $a_i \leq a_{k+1}$ and $b_j \leq b_{k+1}$, if the pairs are $a_i \leftrightarrow b_{k+1}$ and $a_{k+1} \leftrightarrow b_j$, we have the result that $\mid a_i - b_{k+1} \mid + \mid a_{k+1} - b_j \mid \geq \mid a_i - b_j \mid + \mid a_{k+1} - b_{k+1} \mid$. It means that in order to achieve the column-pair minimum matching distance, the pairs should be $a_i \leftrightarrow b_j$ and $a_{k+1} \leftrightarrow b_{k+1}$ instead of $a_i \leftrightarrow b_{k+1}$ and $a_{k+1} \leftrightarrow b_j$. After deciding that a_{k+1} is paired with b_{k+1}, there are k elements in each column left. Based on the assumption, they have to be paired according to the sorted order. Hence, Theorem 3 holds when the number of elements in each column is $k + 1$.□

The DCF algorithm for the Manhattan minimum matching distance is slightly different from the algorithm for the Euclidean minimum matching distance given in Algorithm 2. Instead of using the Euclidean sum of column-pair minimum matching distance, we use the Manhattan sum of the Manhattan-based column-pair minimum matching distance of all pairs. The algorithm is given is Algorithm 4. The following theorem guarantees that there are no false positives.

Algorithm 4 DCF algorithm for the Manhattan minimum matching distance

Input:
q is the query object with vector set V_q;
ε is the given bound;
Procedure:
$partial\_sum = 0$;
sum_j is the jth column-pair minimum matching distance;
for Each column j in vector sets **do**
  Sorted $C_q(j)$ and results are stored in $C_q^s(j)$;
  for Each object x in the search database **do**
    Sorted $C_x(j)$ and results are stored in $C_x^s(j)$;
    compute sum_j using Algorithm 1;
    $partial\_sum = partial\_sum + sum_j$;
    if $partial\_sum \geq \varepsilon$ **then**
      object x does not need further consideration.
    end if
  end for
end for
Output:
Return all the objects left.

THEOREM 4. *The Manhattan sum of the Manhattan-based column-pair minimum matching distance of all column pairs of two vector sets is a lower bound of the Manhattan minimum matching distance of these two vector sets.*

Proof: For two objects x and y with vector sets V_x and V_y, $\mid V_x \mid = \mid V_y \mid = m$, and each vector has n dimensions. The jth columns are denoted as $C_x(j)$ and $C_y(j)$, and their column-pair minimum matching distance is $sum_j = \sum_{i=1}^{m} \mid v_{i,j}^x - v_{p_j(i),j}^y \mid$, where $p_j(i)$ is the permutation minimizing sum_j. We need to prove that $\sum_{j=1}^{n} sum_j$ is smaller than their Manhattan minimum matching distance $\sum_{i=1}^{m}\sum_{j=1}^{n} \mid v_{i,j}^x - v_{p(i),j}^y \mid$, where $p(i)$ is the permutation in the minimum matching distance. Since $p_j(i)$ is the permutation optimizing each column-pair minimum matching distance while $p(i)$ is the permutation to minimize the minimum matching distance, for jth column, the column-pair matching distance using $p(i)$ is larger than the one using $p_j(i)$. Therefore we have:

$$\sum_{j=1}^{n} sum_j = \sum_{j=1}^{n}(\sum_{i=1}^{m} \mid v_{i,j}^x - v_{p_j(i),j}^y \mid)$$

$$\leq \sum_{j=1}^{n}(\sum_{i=1}^{m} \mid v_{i,j}^x - v_{p(i),j}^y \mid)$$

$$= \sum_{i=1}^{m}(\sum_{j=1}^{n} \mid v_{i,j}^x - v_{p(i),j}^y \mid). \square$$

3.3 Theoretical analysis

The minimum matching distance using the methods proposed in [10, 12] has time complexity $O(m^3 + m^2 n)$ for each pair of vector sets, where m is the number of vectors in one set, and n is the number of dimensions of a single vector. Assume the number of vector sets in the image database is N, hence the time complexity of a similarity range query is $O(Nm^3 + Nm^2 n)$. In our processing algorithm for similarity range search, the time complexity consists of two parts: the filter value computation cost, and the refinement cost. The filter value computation cost is the time of the DCF algorithm. Since the time complexity of computing a column-pair minimum matching distance of two vector sets is $O(mlog(m))$, the time complexity of computing the filter value between the query object and an object x in the database in the worst case is $O(nmlog(m))$. Assume the filtering effectiveness of the DCF algorithm is f ($0 \leq f \leq 1$), the number of objects filtered after the filter step is fN. Then the number of objects needs to be processed at the refinement step is $(1-f)N$. Hence the time complexity is $(Nnmlog(m)+(1-f)N(m^3+m^2n))$. In other words, the time complexity depends on the effectiveness of the filter and the cost of the refinement step, i.e., the larger the value of f is, the less time is spent on the refinement step, hence the faster the query processing is.

When comparing the effectiveness of the DCF algorithm and the extended centroid method [9], the extended centroid is not applicable to the Euclidean minimum matching distance. since the extended centroid method was proposed for a minimum matching distance which is different from our Euclidean minimum matching distance. We do not compare DCF for the Euclidean minimum matching distance with the extended centroid method. In the case of the Manhattan minimum matching distance, we can adapt the extended centroid method. Now we introduce the extended centroid method for the Manhattan minimum matching distance. Assume all vector sets contain the same number of vectors, m, the extended centroid of a vector set V_x is defined as follows.

DEFINITION 7. *The extended centroid of a vector set V_x is a vector $\bar{V_x} = \bar{V_x}(1), \bar{V_x}(2), \ldots, \bar{V_x}(n)$, and*

$$\bar{V_x}(j) = \frac{\sum_{i=1}^{m} v_{i,j}^x}{m}.$$

Where $1 \leq j \leq n$, n is the number of dimensions of a vector in V_x, m is the number of vectors in V_x, and $v_{i,j}^x$ is the value at the ith row and jth column.

In the extended centroid method, given two vector sets V_x and V_y, we first compute the extended centroids for both vector sets, $\bar{V_x}$ and $\bar{V_y}$, then the Manhattan distance of $\bar{V_x}$ and $\bar{V_y}$, $\bar{d_{x,y}}$, can be computed. For a given bound ε, if $m \times \bar{d_{x,y}} \geq \varepsilon$, the Manhattan minimum matching distance between V_x and V_y must be larger than ε. In other words, $m \times \bar{d_{x,y}}$ is a lower bound of the Manhattan minimum matching distance V_x and V_y, which is given in the following theorem.

THEOREM 5. *Given two objects x and y, assume their vector sets be V_x and V_y and their extended centroids be $\bar{V_x}$ and $\bar{V_y}$. The Manhattan distance between $\bar{V_x}$ and $\bar{V_y}$ multiplied by m is a lower bound of the Manhattan minimum matching distance between x and y.*

Proof: The Manhattan distance between $\bar{V_x}$ and $\bar{V_y}$ is

$$\bar{d_{x,y}} = \sum_{j=1}^{n} \mid \bar{V_x}(j) - \bar{V_y}(j) \mid$$

$$= \sum_{j=1}^{n} \mid \frac{\sum_{i=1}^{m} v_{i,j}^x}{m} - \frac{\sum_{i=1}^{m} v_{i,j}^y}{m} \mid.$$

Then we have

$$m \times \bar{d_{x,y}} = \sum_{j=1}^{n} \mid \sum_{i=1}^{m} v_{i,j}^x - \sum_{i=1}^{m} v_{i,j}^y \mid$$

$$= \sum_{j=1}^{n} \mid \sum_{i=1}^{m} v_{i,j}^x - \sum_{i=1}^{m} v_{p(i),j}^y \mid$$

$$\leq \sum_{j=1}^{n}\sum_{i=1}^{m} \mid v_{i,j}^x - v_{p(i),j}^y \mid$$

$$= \sum_{i=1}^{m}\sum_{j=1}^{n} \mid v_{i,j}^x - v_{p(i),j}^y \mid$$

$$=$$ the Manhattan minimum matching distance between x and y.

Where $p(i)$ is the permutation in the Manhattan minimum matching distance.□

However we can show that the bound computed using the extended centroid method is smaller than the one given by the DCF filter, i.e., they satisfy $d_{centroid} \leq d_{DCF} \leq d_{mm}$, where $d_{centroid}$ and d_{DCF} are the filter values of the extended centroid method and DCF respectively. This relation can be proved as follows. Given objects x and y, we have

$$m \times \bar{d_{x,y}} = \sum_{j=1}^{n} \mid \sum_{i=1}^{m} v_{i,j}^x - \sum_{i=1}^{m} v_{i,j}^y \mid$$

$$= \sum_{j=1}^{n} \mid \sum_{i=1}^{m} v_{i,j}^x - \sum_{i=1}^{m} v_{p_j(i),j}^y \mid$$

$$= \sum_{j=1}^{n} \mid \sum_{i=1}^{m} (v_{i,j}^x - v_{p_j(i),j}^y) \mid$$

$$\leq \sum_{j=1}^{n}\sum_{i=1}^{m} \mid v_{i,j}^x - v_{p_j(i),j}^y \mid$$

$=$ the filter value of the DCF algorithm.

Hence in the Manhattan minimum matching distance, DCF has better filtering performance than the extended centroid method.

3.4 Optimization Issues

Even though Algorithms 2 and 4 can be directly used as a filter for the Euclidean and Manhattan minimum matching distances respectively, we consider several optimization issues that may enhance the filter efficiency. Although the order of dimensions in which we compute the column-pair minimum matching distance does not affect the filtering effectiveness, since DCF algorithm is based on step-by-step computation of the filter values, we do want to find a good order of dimensions so that the filter values can accumulate fast and more objects can be filtered early. As a result, the computation cost at the filtering step is reduced.

According to the definitions of the Euclidean and Manhattan minimum matching distances, both of them depend on the column-pair minimum matching distances and larger column-pair minimum matching distances result in larger filter values. Hence we want to order the dimensions based on the decreasing order of the column-pair minimum matching distances. Since it is impossible to know a good ordering in advance without computing the column-pair minimum matching distances, we have to use statistical information along with each dimension. However the data distribution in each dimension is changed when we compute the column-pair minimum matching distances, since the data is reordered and paired using their sorted orders. It is difficult to estimate the column-pair minimum matching distance using some statistical information such as the data distribution, and histograms. Hence we propose to use simple statistics, the mean values of each dimension, which are not changed after reordering. Given two objects x and y, we compute their mean values along each dimension, then order the dimensions in the decreasing order of their mean differences. The filter value is computed according to this order. The motivation is that the larger the difference of the mean values of a column pair, the more likely that their column-pair minimum distance is larger. The experimental results in the next section show the effectiveness of this optimization technique.

Another optimization is on deciding the number of dimensions considered in the DCF algorithms. This is also a problem of tradeoff between the computation cost of the filter step and the refinement step. When we compute filter values, after considering certain number of dimensions, the number of objects filtered may barely increase. Thus under this condition, we should terminate the filtering step and initiate the refinement step. There will be a time saving if the time saved from the filter step is larger than that wasted in the refinement step. The problem of determining the number of dimensions which should be considered in the filter step is hard since it is data-dependent, we therefore leave this problem as future work.

4. EXPERIMENTAL RESULTS

In this section we evaluate the effectiveness of DCF for the Euclidean minimum matching distance and the Manhattan minimum matching distance. The dataset we used in the experiments is from UCI KDD Archive [7]. The dataset contains 68040 color histograms of 32 dimensions. We organize them into 4252 vector sets each containing 16 color histograms. Therefore our dataset comprises of 4252 feature vector sets, where each vector set is composed of 16 feature vector and each vector has 32 dimensions. We compare DCF with the extended centroid method, and the algorithm in [10, 12]. We refer to the algorithm in [10, 12] as *the naive algorithm*, because similarity search queries are processed only by computing the minimum matching distances among objects without considering the techniques such as pruning, indexing, and dimension reduction.

The first experiment varies the number of column pairs considered in DCF for the Euclidean minimum matching distance, where the bound ε in similarity range query is 1.3. We compare the performance of the DCF and the DCF with optimization. The optimization technique is ordering the dimensions according the mean differences, as we discussed earlier. Figure 3 shows the results for DCF and DCF with optimization. The Y axis is the number of objects filtered. From the curve of DCF, we observe that the number of objects filtered dramatically increases when around half of the column pairs are considered. This is due to the fact that the filter value is accumulated in DCF. Figure 3 also shows the effectiveness of DCF with optimization. When the number of dimensions is varied from 1 to 20, the DCF with optimization filters much more objects than the DCF. It is not surprising to see that both of them filter out the same number of objects when all dimensions are considered, since the optimization of reordering dimensions does not affect the filtering effectiveness. The DCF with optimization filters more objects earlier than the DCF without optimization. We also observe that after about 25 dimensions, the number of objects filtered marginally increases, which confirms our observation that the filtering step should be terminated with the marginal increase in the filtered objects.

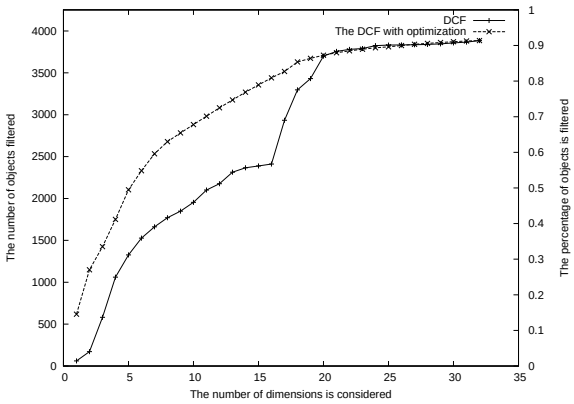

Figure 3: The impact of number of column pairs considered in the Euclidean distance

Figure 4 shows the experimental results of DCF for the Euclidean minimum matching distance where the bound ε is varied. Since both DCF and the DCF with optimization have the same final filter values after all dimensions are considered, we do not need to differentiate them in Figure 4. It is not surprising to see that the number of objects filtered decreases when the value of ε is increased, as shown in Figure 4. The results for the total runtime with the different ε are shown in Figure 5. The performance of DCF is significantly better than the naive algorithm. For different ε, the naive algorithm has similar performance for all values of ε, since all objects are considered. In contrast, the total runtime in our scheme is longer when the value of ε is larger, since the number of objects filtered reduces as ε is increased. Figure 5 shows that the performance of the DCF with optimization has better performance than that of DCF when the similarity threshold ε is small. However, when the similarity threshold ε is larger,the DCF with optimization has simi-

lar performance to that of DCF. This is due to the fact that although the DCF with optimization algorithm can filter more objects earlier to improve the cost of the filtering step, the computation cost at the refinement step is much larger than that of the filtering step. Hence the performance does not improve significantly.

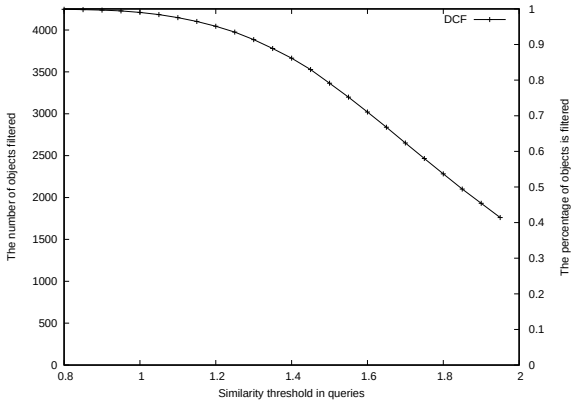

Figure 4: The experimental results for different query bounds in the Euclidean distance

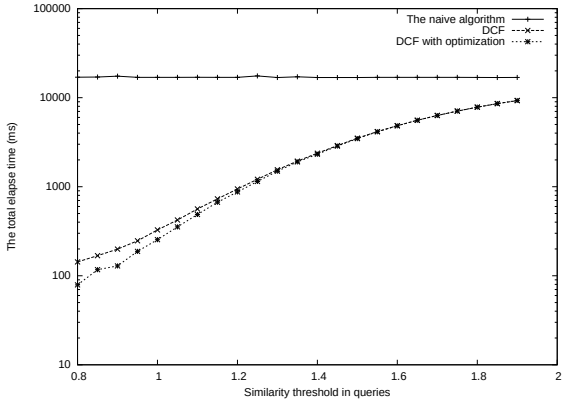

Figure 5: The total runtime for different query bounds in the Euclidean distance

We conducted similar experiments for DCF under the Manhattan minimum matching distance and compared it with the naive algorithm, and the extended centroid method. The similarity threshold values we use in the experiments for the Manhattan minimum matching distance are larger than the ones used in the Euclidean minimum matching distance. This is because the Euclidean minimum matching distance between two vector sets is much smaller than the Manhattan minimum matching distance between these two sets. Figure 6 shows the results for DCF and the DCF with optimization when the number of column pairs considered varies. The behavior is similar to the one given in Figure 3. Since the number of dimensions considered has no impact on the extended centroid filter, the number of objects filtered is a constant value, as shown in Figure 6. When we consider all dimensions in DCF, DCF has much better performance than the extended centroid filter.

Figure 7 shows the comparison results of DCF and the extended centroid method when the similarity threshold ε is varied. The performance of both DCF and the extended centroid gets worse when the values of ε are larger. DCF significantly outperforms the

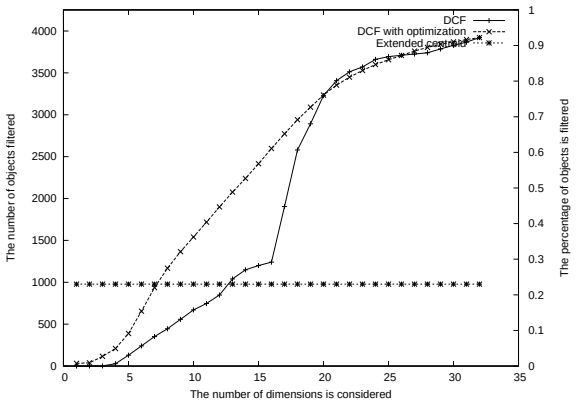

Figure 6: The impact of number of column pairs considered in the Manhattan distance

extended centroid method in terms of filter effectiveness. When ε is no larger than 10, most of the objects are filtered in DCF. Similar to the results in Figure 5, the elapsed time of DCF and the DCF with optimization have similar performance, as shown in Figure 8. This figure also shows the comparison results of total runtime among these four methods. We observe that DCF and the DCF with optimization have superior performance than the extended centroid method and the naive algorithm, while the extended centroid method outperforms the naive algorithm.

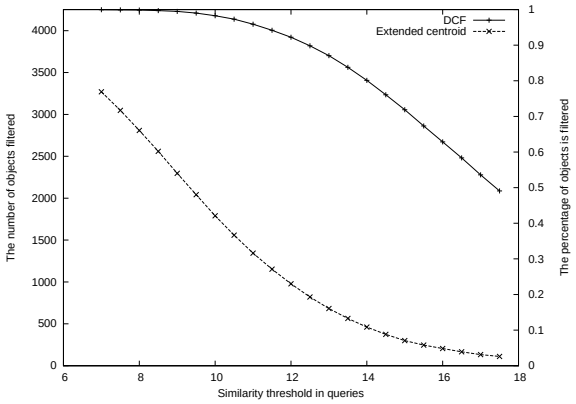

Figure 7: The experimental results for different query bounds in the Manhattan distance

5. CONCLUSION

Similarity search is an important class of queries and plays an important role in a lot of applications, such as multimedia databases, medical imaging, and CAD applications. A lot of studies have been mainly focused on improving the query cost of similarity search over objects represented by feature vectors. Recently, similarity search on objects which are presented using feature vector sets is gaining much attention, since the results are more meaningful. However the existing techniques proposed for similarity search over feature vectors can not be adapted to this context due to the different distance metrics. In this paper, we define two distance metrics based on the Euclidean and Manhattan distances due to the popularity of the Euclidean distance and Manhattan distance. Then we propose filtering techniques to accelerate the query processing. The novel filter technique proposed is Divide-and-Conquer filter (DCF).

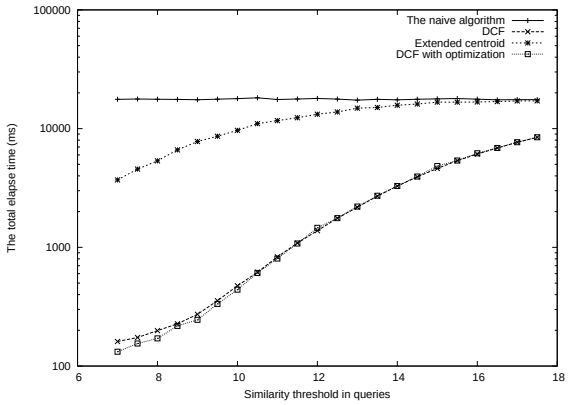

Figure 8: The total runtime for different query bounds in the Manhattan distance

Then we present filtering algorithms for both the Euclidean and Manhattan minimum matching distances based on this technique. In DCF, filter values are computed dimension-by-dimension. The objects are filtered whenever their partial filter values are larger than the given bound. The experimental results show the superior performance of our filter technique.

6. REFERENCES

[1] Mihael Ankerst, Bernhard Braunmüller, Hans-Peter Kriegel, and Thomas Seidl. Improving adaptable similarity query processing by using approximations. In *VLDB'98, Proceedings of 24rd International Conference on Very Large Data Bases*, pages 206–217, 1998.

[2] Stefan Berchtold, Daniel A. Keim, and Hans-Peter Kriegel. The X-tree: An index structure for high-dimensional data. In *Proceedings of the 22nd International Conference on Very Large Databases*, pages 28–39, 1996.

[3] Orhan Camoglu, Tamer Kahveci, and Ambuj Singh. Psi: Indexing protein structures for fast similarity search, 2003.

[4] Paolo Ciaccia, Marco Patella, and Pavel Zezula. M-tree: An efficient access method for similarity search in metric spaces. *Proceedings of the 23rd International Conference on Very Large Data Bases*, pages 426–435, 1997.

[5] G.P. Copeland and S. F. Khoshafian. A decomposition storage model. *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 268–279, May 1985.

[6] Arjen P. de Vries, Nikos Mamoulis, Niels Nes, and Martin Kersten. Efficient k-nn search on vertically decomposed data. In *Proceedings of the 2002 ACM SIGMOD international conference on Management of data*, pages 322–333, 2002.

[7] S. Hettich and S. D. Bay. The UCI KDD archive, 1999. http://kdd.ics.uci.edu.

[8] Flip Korn, Nikolaos Sidiropoulos, Christos Faloutsos, Eliot Siegel, and Zenon Protopapas. Fast nearest neighbor search in medical image databases. In *Proceedings of the 22th International Conference on Very Large Data Bases*, pages 215–226, 1996.

[9] Hans-Peter Kriegel, Stefan Brecheisen, Peer Kroger, Martin Pfeifle, and Matthias Schubert. Using sets of feature vectors for similarity search on voxelized cad objects. In *Proceedings of the 2003 ACM SIGMOD international conference on on Management of data*, pages 587–598, 2003.

[10] H. W. Kuhn. The hungarian method for the assignment problem. *Naval Res. Logist. Quart*, pages 83–98, 1955.

[11] K. Lin, H. V. Jagadish, and C. Faloutsos. The TV-tree: An index structure for high-dimensional data. *VLDB Journal*, 3:517–542, 1995.

[12] J. Munkres. Algorithms for the assignment and transportation problems. *J. SIAM*, pages 32–38, 1957.

[13] Nick Roussopoulos, Stephen Kelley, and Frederic Vincent. Nearest neighbor queries. In *Proceedings of the 1995 ACM SIGMOD international conference on Management of data*, pages 71–79, 1995.

[14] Yasushi Sakurai, Masatoshi Yoshikawa, Shunsuke Uemura, and Haruhiko Kojima. The A-tree: An index structure for high-dimensional spaces using relative approximation. In *Proc. of the 26th International Conference on Very Large Data Bases (VLDB)*, pages 516–526, Cairo, September 2000.

[15] Roger Weber, Hans-J. Schek, and Stephen 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, 1998.