Effective Data Co-Reduction for Multimedia Similarity Search

Zi Huang Heng Tao Shen Jiajun Liu Xiaofang Zhou
School of Information Technology and Electrical Engineering
The University of Queensland, QLD 4072, Australia
{huang,shenht,jiajun,zxf}@itee.uq.edu.au

ABSTRACT

Multimedia similarity search has been playing a critical role in many novel applications. Typically, multimedia objects are described by high-dimensional feature vectors (or points) which are organized in databases for retrieval. Although many high-dimensional indexing methods have been proposed to facilitate the search process, efficient retrieval over large, sparse and extremely high-dimensional databases remains challenging due to the continuous increases in data size and feature dimensionality. In this paper, we propose the first framework for Data Co-Reduction (DCR) on both data size and feature dimensionality. By utilizing recently developed co-clustering methods, DCR simultaneously reduces both size and dimensionality of the original data into a compact subspace, where lower bounds of the actual distances in the original space can be efficiently established to achieve fast and lossless similarity search in the filter-and-refine approach. Particularly, DCR considers the duality between size and dimensionality, and achieves the optimal co-reduction which generates the least number of candidates for actual distance computations. We conduct an extensive experimental study on large and real-life multimedia datasets, with dimensionality ranging from 432 to 1936. Our results demonstrate that DCR outperforms existing methods significantly for lossless retrieval, especially in the presence of extremely high dimensionality.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Multimedia Databases

General Terms

Algorithms, Measurement, Experimentation

Keywords

Data co-reduction, high-dimensional indexing, search

1. INTRODUCTION

Multimedia similarity search automatically retrieves multimedia objects similar to query objects from databases based on their visual features. It has been extensively studied in the last few decades, due to its wide range of applications [9, 17]. Its success mainly relies on the visual features that represent the multimedia objects, either globally for the entire object or locally for a region. Many high-dimensional features have been proposed in the literature, such as commonly used color, texture, shape, and local descriptors [9]. To match two objects, different similarity measures have been introduced to capture the similarity between their high-dimensional feature vectors, including the classic Euclidean distance, histogram intersection, and many other sophisticated ones such as Edit distance [6], Earth Mover's distance [27], and Bregman divergence [29]. The choice of similarity measures is usually determined by the underlying feature space and application requirement.

Driven by the advances in Internet and multimedia technologies, the amount of multimedia content has grown to an unprecedented level. For instance, the number of photos in Facebook, the popular social networking website, has reached 50 billions in July 2010 [19]. Meanwhile, some applications may also desire extremely high-dimensional features for high retrieval accuracy. For example, recent proposed image features for face recognition can reach up to tens of thousands of dimensions for effective recognition [28]. There is a trend that multimedia databases in most existing and emerging multimedia applications are keeping growing to much larger scales. Efficient similarity search is becoming evermore crucial to find multimedia objects of interests.

Towards efficient retrieval, database community has made tremendous contributions in the last few decades. The main goal is to avoid complete access to the original data for full dimensional distance computation, by designing novel tree structures [8, 21], approximating data representation [26, 16], reducing feature dimensionality [23, 27], and so on. Due to the known "dimensionality curse", most tree structures perform rapidly towards sequential scan as feature dimensionality increases [26]. Data approximation like VA-File approximates each dimension of a data point into a smaller bit sequence without changing dimensionality. However, the performance is limited by the ratio between a dimension (typically 32 bits) and its bit sequence. Dimensionality reduction reduces feature dimensionality into a smaller number by removing insignificant dimensions. Because most methods do not preserve complete information, retrieval accuracy may be compromised. Demand of high accuracy often limits the number of dimensions to be removed, thus affects the performance improvement.

In this paper, we initiate the investigation on simultaneous data reduction on both data size and feature dimensionality, named Data Co-Reduction (DCR). Given a large and high-dimensional dataset, it is unlikely that dimensions are well correlated across the whole dataset. However, a subset of dimensions may have very close values for a subset of data points. Similarly, a subset of data points may have very similar values along a subset of dimensions. Considering the dataset as a matrix, where each row represents a data point and each column represents a dimension, we can partition the dataset into blocks of similar values by co-clustering rows and columns and approximate each block with its data range. By establishing a lower bound of the actual distance based on block ranges, lossless similarity search can be achieved. The novelty of DCR lies in the consideration of duality of rows and columns to generate more compact and precise data approximations. In the co-reduced subspace, column reduction contributes to reduce the time complexity of lower bound computation since dimensionality is lower, while row reduction further contributes to reduce the number of lower bound computations since multiple data points are reduced into the same reduced row. With smaller data representations and quicker data filtering, DCR expects substantial performance improvements. To summarize, we make the following main contributions:

- We formalize the problem of data co-reduction and discuss its advantages over existing approaches. A tight lower bound of the actual distance is defined in the co-reduced subspace and can be efficiently computed for fast filtering in query processing.

- We present the general framework for DCR and explain its major components for efficient similarity search.

- We define the optimality of data co-reduction with respect to the number of candidates for actual distance computations. Given the desired numbers of reduced rows and columns, the optimal co-reduction function minimizes the designed objective function to achieve the maximal pruning power in a query-independent fashion. By utilizing the co-clustering technique which induces row and column clustering from each other, we introduce an Expectation Maximization (EM) based algorithm to alternate the optimization process on rows and columns for finding the optimal co-reduction function with guarantee of convergence.

- We demonstrate the space and time superiority of DCR over existing methods by extensive experimental results on large-scale multimedia datasets of different dimensionalities.

The rest of this paper is organized as follows. Section 2 reviews some related work on multimedia similarity search. The problem is formulated in Section 3. Our DCR framework with optimization and query processing is discussed in Section 4. Experimental results are shown in Section 5. Finally, we conclude the paper in Section 6.

2. RELATED WORK

This work is mainly related to multimedia similarity search, with emphasis on high-dimensional indexing. Various categories of high-dimensional indexing methods have been proposed to tackle the "curse of dimensionality", such as tree structures, data approximations, hashing-based methods and dimensionality reduction methods.

Tree structures have achieved notable success in managing multi-dimensional feature vectors, from early R-tree, kd-tree and their variants, to M-tree [8], A-tree [21] and many other trees [4]. The key idea is to prune tree branches as much as possible based on the established bounding distances so that the number of accessed feature vectors (or points) can be reduced significantly. However, their performance rapidly degrades as feature dimensionality increases, and eventually most of them are outperformed by sequence scan when dimensionality reaches high tens due to the massive overlap among different branches [26]. By utilizing the efficient B^+-tree, recent method iDistance [15] partitions high-dimensional data points into clusters and then indexes all the points by their distances to their respective reference points using a single B^+-tree. Its efficiency comes from the localized distances to corresponding reference points and B^+-tree. Its performance is further improved by finding the optimal reference points which can maximize the performance of B^+-tree [22]. Nonetheless, single dimensional distance values become less distinguishable as dimensionality increases. While most tree structures are designed for exact search, the spatial approximation sample hierarchy (SASH) [13] is a multi-level search structure for approximate search in extremely high-dimensional spaces, by sampling the data points and considering samples' neighborhood distances.

Data approximation is another direction to index high-dimensional data. Typical methods include VA-File [26] and Local Digital Coding (LDC) [16]. The key idea is to encode an original data point with a much smaller representation. VA-file approximates each dimension with a small number of bits, by dividing the data space into 2^b rectangular cells where b denotes a user specified number of bits. It allocates a unique bit-string of length b for each cell, and approximates data points that fall into a cell by that bit-string. VA-File itself is simply an array of these compact and geometric approximations. Query process is performed by scanning the entire approximation file and excluding points from the actual distance computation based on the lower and upper bounds established from these approximations. This approach is insensitive to the dimensionality and thus able to outperform sequential scan if a small number of candidates are finally accessed. However, the improvement ratio is rather limited since every single dimension needs to be encoded and the whole VA-File needs to be scanned. Some refined approaches based on VA-file have also been proposed, such as IQ-tree [3] which integrates a tree structure and VA-File. LDC [16] extracts a simple bitmap representation called Digital Code(DC) for each point by encoding a single bit for each dimension. Pruning during search is performed by dynamically selecting only a subset of the bits from the DC based on which subsequent comparisons are performed. A large number of expensive high-dimensional distance computations can be avoided. However, LDC cannot guarantee the correctness of results.

Hashing has been demonstrated to be effective in looking for answers from quickly identified buckets. One typical method is Locality Sensitive Hashing (LSH) [10]. The basic idea is to use a family of locality sensitive hash functions composed of linear projection over random directions in the feature space. The intuition behind is that for at lease one of the hash functions, nearby objects have high probability of being hashed into the same state. Improvements to LSH have been made during the past decade by enforcing its projection method [2], by amending the probe of buckets in search stage [18], and by combining with a tree structure [25]. However, hashing-based methods only offer approximate results.

Instead of directly indexing the original high-dimensional data, dimensionality reduction (DR) aims to map the original data into a much lower-dimensional subspace. An index can then be built on the subspace to further facilitate the retrieval [23, 5]. The main idea is to transform data from a high-dimensional space to a lower-dimensional space without losing much information. Many dimensionality reduction methods have been proposed, including global and local methods. Global dimensionality reduction maps the dataset as a whole down to a suitable and lower-dimensional subspace. Local dimensionality reduction typically first divides the whole dataset into correlated clusters, each of which is then reduced into their respective subspaces by classical PCA or other methods. There also exist some global methods which preserve locality information, such as Locality Preserving Projections (LPP) [12] and Locality Condensation (LC) [14]. LPP is an optimal linear approximation to the eigenfunctions of the Laplace-Beltrami operator on the manifold. It preserves the local neighborhood structures of the datasets. As a step further, Locality Condensation (LC) also keeps distant localities separate from each other for more effective similarity search. Differently, in [27] a combining dimensionality reduction method is proposed to achieve fast lossless retrieval, where several original dimensions are combined into a single dimension. In the subspace, a lower bound on the distance is used as a filtering step. The resulting set of candidates is then refined in the original space for exact search. However, this method is specific to Earth Mover's distance only.

In this paper, we frame our study in the context of exact (or lossless) similarity search, where tight bounds established in the co-reduced subspace serve as the effective filtering conditions for the popular similarity measures in high-dimensional spaces. The co-reduced data is typically smaller than VA-File and dimensionality reduced data greatly.

3. PROBLEM FORMULATION

In this section, we formalize the data co-reduction (DCR) problem, followed by the reduced distance in the subspace which serves as a lower bound of the original distance. For easy illustration, we use the L_1 distance and assume that all the dimensional values are normalized into the range of [0,1] for each high-dimensional point. Table 1 shows a list of notations used in this paper.

3.1 Data Co-Reduction

Given a set of n d-dimensional data points, it can be represented as a two-dimensional data matrix X, where the i^{th} row of X represents the i^{th} d-dimensional data point and the j^{th} column contains all the values from all data points on the the j^{th} dimension. A co-reduction of matrix X consists of simultaneous reductions on both rows and columns by combining original rows/columns to form one reduced row/column. Denoting the reduction function for row and column as Θ and Φ respectively, we formalize the data co-reduction below.

Notation	Description
n	number of data points
d	dimensionality of original data space
m	reduced number of rows
l	reduced number of columns
i, j	index of original row and column
i', j'	index of reduced row and column
$R_{i'j'}$	block approximation
$r_{\check{i'}j'}$	minimal value in block $R_{i'j'}$
$r_{\hat{i'}j'}$	maximal value in block $R_{i'j'}$
$\|R_{i'}\|$	number of data points in i'^{th} reduced row
$\|R_{j'}\|$	number of dimensions in j'^{th} reduced column
Θ	row reduction function
Φ	column reduction function

Table 1: Notations

Definition 1 (Data Co-Reduction). *Given a data matrix X, and m and l as the reduced row number and column number respectively, a co-reduction function (Θ,Φ) which reduces X to X', where X' has m rows and l columns, is defined by:*

$$\Theta = \{\theta_{ii'}\}, \theta_{ii'} \in \{0,1\}, i = 1..n, i' = 1..m \quad (1)$$

$$\Phi = \{\phi_{jj'}\}, \phi_{jj'} \in \{0,1\}, j = 1..d, j' = 1..l \quad (2)$$

$$\forall 1 \le i \le n, \sum_{i'=1}^{m} \theta_{ii'} = 1, \forall 1 \le j \le d, \sum_{j'=1}^{l} \phi_{jj'} = 1 \quad (3)$$

$$\forall 1 \le i' \le m, \sum_{i=1}^{n} \theta_{ii'} \ge 1, \forall 1 \le j' \le l, \sum_{j=1}^{d} \phi_{jj'} \ge 1 \quad (4)$$

Restriction (3) constrains that each row/column of X is assigned to one and only one reduced row/column in X', and Restriction (4) constrains that each reduced row/column in X' has at least one original row/column in X.

Given a co-reduction function (Θ,Φ), the rows of X is "re-ordered" such that all the rows mapping into the first reduced row are grouped together, followed by all the rows mapping into the second reduced row, and so on. Simultaneously, this is also applied to column. This row-column reordering re-arranges the original X in size of $n \times d$ into $m \times l$ blocks, each of which is represented as $B_{i'j'} = \{x_{ij}\}$, where $\theta_{ii'} = 1$, $\phi_{jj'} = 1$, $i = 1..n$, $j = 1..d$, $i' = 1..m$, and $j' = 1..l$.

Definition 2 (Block Approximation). *Given a block $B_{i'j'}$, $i' = 1..m$ and $j' = 1..l$, it can be characterized by a lossless approximation $R_{i'j'} = [\check{r}_{i'j'}, \hat{r}_{i'j'}]$, where*

$$\check{r}_{i'j'} = \min\{x_{ij} | \theta_{ii'} = 1, \phi_{jj'} = 1, i = 1..n, j = 1..d\}$$

$$\hat{r}_{i'j'} = \max\{x_{ij} | \theta_{ii'} = 1, \phi_{jj'} = 1, i = 1..n, j = 1..d\}$$

Basically, each block is approximated with a range specified by its minimal and maximal values. This representation is coarse yet lossless since all the possible values for the block are included in the range. By applying a co-reduction function (Θ,Φ), an original matrix X in size of $n \times d$ can be reduced into an $m \times l$ matrix X', each of whose entries is a

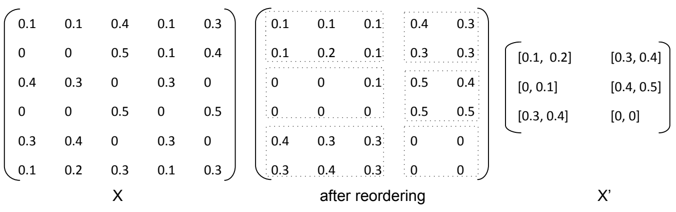


Figure 1: An example of DCR

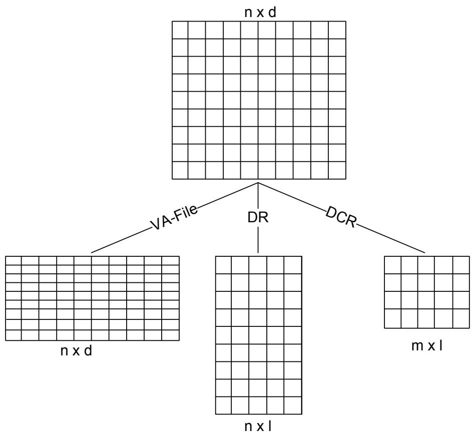


Figure 2: DCR vs. DR vs. VA-file

block approximation. Figure 1 shows an example of DCR on how an original 6×5 data matrix X is co-reduced into a 3×2 matrix X', where Θ combines the first and sixth, second and fourth, third and fifth data points into three reduced rows respectively, and Φ combines the first, second, and fourth dimensions into the first reduced column, and the third and fifth dimensions into the second reduced column.

Compared with dimensionality reduction (DR) which reduces the original space into a lower-dimensional subspace, DCR also reduces the original data size into a much smaller number. Compared with VA-File which reduces each dimension from four bytes to few bits, DCR reduces both size and dimensionality with block approximations. Although a block approximation carries two values, the reduced X' occupies just $2\frac{m \times l}{n \times d}$ of the original storage space. As n and d become larger, the reduction ratio achieved by DCR tends to be greater. It is expected that the reduced data by DCR is significantly smaller than that by VA-File or dimensionality reduction methods for very large-scale datasets in extremely high-dimensional spaces. Figure 2 visualizes the differences among DCR, DR and VA-File.

More importantly, different from VA-File which examines every single data point's approximation to get its bounding distances, DCR provides an opportunity to have one-off pruning of multiple data points mapped into the same reduced row by efficiently computing a lower bound in the subspace. The pruning power of a filter is affected by the tightness of lower bounds. Naturally, the next question is how to compute a reduced distance in the subspace to serve as a tight lower bound of the original distance.

3.2 Reduced Distance

A reduced distance between a query point q and a database point x is defined in the subspace mapped by a co-reduction

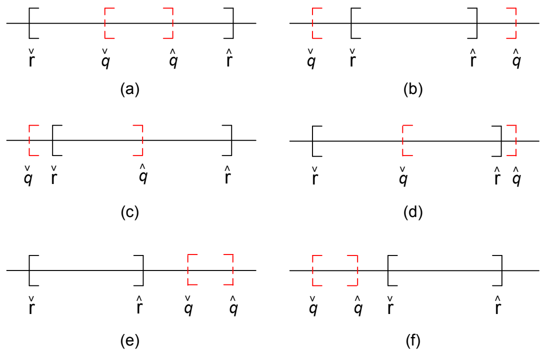


Figure 3: Six relationships between two blocks

function (Θ, Φ). For a database point x that is mapped into the i'^{th} reduced row, it can be represented as $x' = \{R_{i'j'} | R_{i'j'} = [\check{r}_{i'j'}, \hat{r}_{i'j'}], j' = 1..l\}$ in the subspace. Based on the column reduction function Φ, a query q can be reduced and represented in the subspace as $q' = \{R^q_{j'} | R^q_{j'} = [\check{r}^q_{j'}, \hat{r}^q_{j'}], j' = 1..l\}$, where

$$\check{r}^q_{j'} = \min\{q_j | \phi_{jj'} = 1, j = 1..d\}$$
$$\hat{r}^q_{j'} = \max\{q_j | \phi_{jj'} = 1, j = 1..d\}$$

As illustrated in Figure 3 (a-f), there are six possible relationships between a query block and a database block, from which the block distance is defined.

Definition 3 (Block Distance). *Given the j'^{th} query block approximation $R^q_{j'}$ of q', its block distance to $R_{i'j'}$ of x' is defined as:*

$$BD(R^q_{j'}, R_{i'j'}) = \begin{cases} 0 & \text{if } \check{r}_{i'j'} \le \check{r}^q_{j'} \le \hat{r}^q_{j'} \le \hat{r}_{i'j'} \\ \check{r}_{i'j'} - \check{r}^q_{j'} + \hat{r}^q_{j'} - \hat{r}_{i'j'} & \text{if } \check{r}^q_{j'} \le \check{r}_{i'j'} \le \hat{r}_{i'j'} \le \hat{r}^q_{j'} \\ \check{r}_{i'j'} - \check{r}^q_{j'} & \text{if } \check{r}^q_{j'} \le \check{r}_{i'j'} \le \hat{r}^q_{j'} \le \hat{r}_{i'j'} \\ \hat{r}^q_{j'} - \hat{r}_{i'j'} & \text{if } \check{r}_{i'j'} \le \check{r}^q_{j'} \le \hat{r}_{i'j'} \le \hat{r}^q_{j'} \\ (\check{r}^q_{j'} - \hat{r}_{i'j'}) \cdot (|R_{j'}| - 1) + \hat{r}^q_{j'} - \hat{r}_{i'j'} & \text{if } \hat{r}_{i'j'} \le \check{r}^q_{j'} \\ (\check{r}_{i'j'} - \hat{r}^q_{j'}) \cdot (|R_{j'}| - 1) + \check{r}_{i'j'} - \check{r}^q_{j'} & \text{if } \hat{r}^q_{j'} \le \check{r}_{i'j'} \end{cases} \tag{5}$$

where $|R_{j'}|$ is the number of dimensions mapped into the j'^{th} reduced column.

Obviously, the above block distance only considers the minimal and maximal values in two blocks, resulting in a constant time complexity. By further considering the actual value of q on each dimension, we get an expanded block distance as defined below.

Definition 4 (Expanded Block Distance). *Given the j'^{th} query block approximation $R^q_{j'}$ of q', the expanded block distance from $R^q_{j'}$ to $R_{i'j'}$ of x' is defined as:*

$$EBD(R^q_{j'}, R_{i'j'}) = \sum_{j | \phi_{jj'} = 1} D(q_j, R_{i'j'}) \tag{6}$$

where

$$D(q_j, R_{i'j'}) = \begin{cases} 0 & \check{r}_{i'j'} \leq q_j \leq \hat{r}_{i'j'} \\ min\{|q_j - \check{r}_{i'j'}|, |q_j - \hat{r}_{i'j'}|\} & otherwise \end{cases}$$

The expanded block distance (EBD) uses the actual values of those dimensions in the query block. It is straightforward to show that the expanded block distance is always greater than or equal to the block distance, since any value of the query in the j'^{th} block is always within the range of $[\check{r}^q_{j'}, \hat{r}^q_{j'}]$. It is also obvious that the EBD is a lower bound of the actual distance in the original space. Although EBD computation shows a linear time complexity in $|R_{j'}|$ (Equation 6), it can be easily reduced in real implementation. Note that zero occurs only if the query value $q_j|\phi_{jj'} = 1$ lies in the range of $[\check{r_{i'j'}}, \hat{r_{i'j'}}]$. If the range of $R^q_{j'}$ is a subrange of $R_{i'j'}$ (i.e., Figure 3 (a)), EBD is zero. For the relationship when $\check{r}^q_{j'} \geq \hat{r}_{i'j'}$ (i.e., Figure 3 (e)), Equation 6 can be rewritten as:

$$EBD(R^q_{j'}, R_{i'j'}) = \sum_{j|\phi_{jj'}=1} (q_j - \hat{r}_{i'j'}) = \sum_{j|\phi_{jj'}=1} q_j - \hat{r}_{i'j'} \cdot |R_{j'}|$$

Since the column reduction function Φ is known, $\sum_{j|\phi_{jj'}=1} q_j$ can be computed in advance. This results in a constant time complexity for EBD computation. Similarly, for the relationship when $\hat{r}^q_{j'} \leq \check{r}_{i'j'}$ (i.e., Figure 3 (f)), EBD can also be computed in constant time. For the rest relationships (i.e., Figure 3 (b-d)), we can pre-order all the values in the query block and maintain a corresponding list of sums of q_j for each ranked value. The query subrange which intersects with $[\check{r}_{i'j'}, \hat{r}_{i'j'}]$ can be ignored since it contributes zero to the EBD. For the query subrange that is out of $[\check{r}_{i'j'}, \hat{r}_{i'j'}]$, we can look up the sum list with a binary search and get the sum of q_j within the query subranges in $O(log|R_{j'}|)$ time. Therefore, EBD computation's time complexity can be reduced to $O(log|R_{j'}|)$ in the worst case.

Next, we prove the monotony of the EBD, stating that the EBD increases as the minimal and maximal values of a block approximation get closer.

THEOREM 1 (MONOTONY OF THE EBD). *Given two block approximations represented as $R1$ and $R2$ respectively. For a query block approximation R^q, it holds:*

$$R1 \subseteq R2 \Rightarrow \forall q : EBD(R^q, R1) \geq EBD(R^q, R2) \qquad (7)$$

where $R1 \subseteq R2 \Rightarrow \check{r1} \geq \check{r2} \ \wedge \ \hat{r1} \leq \hat{r2}$

PROOF. For any q_j in the query block, four cases need to be considered:

1. $q_j \in [\check{r1}, \hat{r1}] \subseteq [\check{r2}, \hat{r2}]$:
$$D(q_j, R1) = 0 = D(q_j, R2)$$
2. $q_j \in [\check{r2}, \hat{r2}] \wedge q_j \notin [\check{r1}, \hat{r1}]$:
$$D(q_j, R1) > 0 = D(q_j, R2)$$
3. $q_j < \check{r2} \leq \check{r1}$:
$$D(q_j, R1) = \check{r1} - q_j \geq \check{r2} - q_j = D(q_j, R2)$$
4. $q_j > \hat{r2} \geq \hat{r1}$:
$$D(q_j, R1) = q_j - \hat{r1} \geq q_j - \hat{r2} = D(q_j, R2)$$

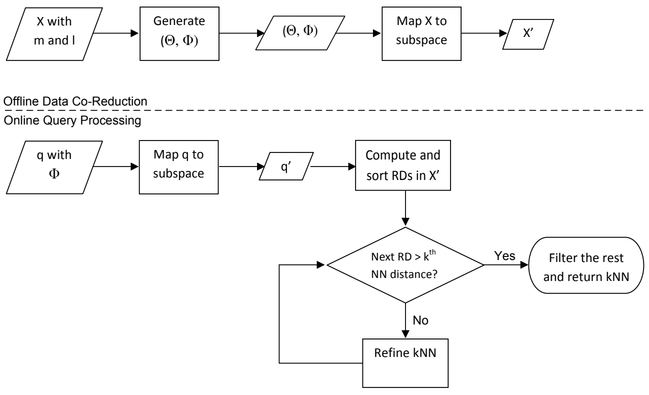


Figure 4: Framework for DCR

Since $D(q_j, R1) \geq D(q_j, R2)$ holds in all the above cases, according to Equation 6, $EBD(R^q, R1) \geq EBD(R^q, R2)$. Therefore, the EBD is monotonic. □

The monotony of the EBD indicates that a block approximation with a tighter range yields a larger EBD. Based on the EBD, the reduced distance from q to x in the subspace is computed by summing up the corresponding l expanded block distances.

DEFINITION 5 (REDUCED DISTANCE). *Given a query point q and a database point x represented by $q' = \{R^q_{j'}|j' = 1..l\}$ and $x' = \{R_{i'j'}|j' = 1..l\}$ respectively in the subspace reduced by a co-reduction function (Θ, Φ), the reduced distance between q and x is defined as:*

$$RD(q, x) = \sum_{j'=1}^{l} EBD(R^q_{j'}, R_{i'j'}) \qquad (8)$$

It has the time complexity of $O(l)$ $(l < d)$. All the data points which are mapped into the same reduced row have the same reduced distance.

THEOREM 2 (LOWER BOUND). *Given a query point q and a database point x represented by $q' = \{R^q_{j'}|j' = 1..l\}$ and $x' = \{R_{i'j'}|j' = 1..l\}$ respectively in the subspace reduced by a co-reduction function (Θ, Φ), it holds that*

$$RD(q, x) \leq \sum_{j=1}^{d} |q_j - x_j|$$

PROOF. Omitted for brevity. □

Note that the lower bound also holds for other L_p norms.

4. DCR FRAMEWORK

In the last section, we have discussed how DCR is formalized and how the reduced distance is defined in the subspace as a lower bound of the distance in the original space. In this section, we present the general framework for similarity search based on DCR, as depicted in Figure 4.

Our DCR framework has an offline data co-reduction component and an online query processing component. For offline indexing, a co-reduction function (Θ, Φ) has to be firstly generated for given m and l. Based on (Θ, Φ), the original data X is then reduced into X', as explained in Section 3.1.

For online query processing, given a query q, it is firstly mapped into the subspace based on the column reduction function Φ. In the subspace, the reduced query q' is then compared with all reduced rows in X' to compute their reduced distances, according to Definition 5. For more efficient filtering, all the reduced distances are sorted in ascending order. Data points mapped into the reduced rows in X' with smallest reduced distances are first accessed to initialize the k-Nearest Neighbors (kNN). Note that one reduced row may contain multiple original data points. If the next reduced distance is already greater than the current k^{th} nearest neighbor's distance, all the data points mapped into the rest reduced rows can be safely filtered according to Theorem 2, and the found kNN are returned. Otherwise, all the data points which are mapped into the next reduced row are accessed and their actual distances to the query are computed and used to refine the kNN. The iteration continues until either the kNN are guaranteed or all the reduced rows have been processed, which is the worst case. Range search is just a special case of kNN search, by replacing the k^{th} nearest neighbor's distance with the query range.

Recall the example shown in Figure 1. Given a query q=(0.4, 0.4, 0, 0.2, 0), based on the column reduction function Φ which maps the dimensions 1, 2 and 4 into the first reduced column, and the dimensions 3 and 5 into the second reduced column, q is reduced into q'=([0.2, 0.4], [0,0]). Next is to compute the reduced distances. Based on Equation 8, the reduced distances between q and three reduced row are 0.8, 1.3 and 0 respectively. Assume we search for top-2 nearest neighbors. Two data points mapped into the third reduced row with the smallest reduced distance are first accessed and compared with q in the original space. Both points have the actual distance of 0.2 to the query, which is updated as the current second nearest neighbor's distance. Since the first and second reduced rows have larger reduced distances than the second nearest neighbor's actual distance with respect to the query, those data points mapped into both reduced rows can be filtered straightaway.

As we can see, in query processing, we compute m ($m < n$) reduced distances which are calculated more efficiently with the $O(l)$ time complexity ($l < d$). DCR is used to generate a set of data point candidates in the co-reduced subspace. Those candidates are then accessed and refined by their actual distances. Smaller candidate sets induce fewer data accesses and distance computations in refinement.

In this paper we define the optimality of data co-reduction with respect to the number of candidates for refinement. Given m and l, mapping the data into a subspace by the optimal co-reduction function which generates the least number of candidates for refinement is the ultimate goal.

4.1 Defining Optimal (Θ, Φ)

During query processing, the reduced distances between the query and the reduced rows in X' are used as the filter. Clearly, more reduced rows can be pruned for a larger reduced distance (i.e., lower bound). Since the reduced distance is computed based on the EBD, the tightness of lower bounds is affected by block approximations. The co-reduction function (Θ,Φ) determines how the original data is co-reduced in both rows and columns. Different co-reduction functions lead to different mappings which result in different block approximations. According to the monotony of the EBD (Theorem 1), block approximations with smaller ranges yields tighter lower bounds. Therefore, one objective is to minimize the block ranges, which are co-determined by both row reduction and column reduction.

In our DCR framework, multiple data points are usually mapped into the same reduced rows due to the row reduction. Besides the tightness of lower bounds, the pruning power measured by the number of data points being pruned is also affected by the number of points mapped into those reduced rows being filtered. The number of data points being pruned is actually the total number of data points in those reduced rows being filtered. Data points falling in the blocks with larger ranges are more likely to be accessed as candidates due to the loose lower bound. Thus, the other objective is to reduce the number of data points mapped into the blocks with larger ranges so that less data points are required to be accessed.

We attempt a query-independent objective function to achieve an optimal co-reduction function (Θ,Φ), which co-reduces the original data into the subspace where the smallest candidate sets can be generated in the filter step. Without prior knowledge of queries, we formally define the "optimal" co-reduction as below, for given m and l.

Definition 6 (Optimal (Θ, Φ)). *Given a data matrix X, and m and l as the reduced row number and column number respectively, the optimal co-reduction function, denoted as (Θ^*, Φ^*), achieves the following objective function:*

$$\arg\min_{\Theta,\Phi} SPR(\Theta, \Phi) \tag{9}$$

where $SPR(\Theta, \Phi)$, the sum of penalized ranges, is defined as

$$SPR(\Theta, \Phi) = \sum_{i'=1..m, j'=1..l} (\hat{r}_{i'j'} - \check{r}_{i'j'}) \cdot |R_{i'}| \tag{10}$$

and $|R_{i'}|$ is the number of data points mapped into the i'^{th} reduced row.

By multiplying the number of data points assigned into the block $B_{i'j'}$, the objective function incurs a heavy penalty (i.e., $|R_{i'}|$) if the range of the block $B_{i'j'}$ (i.e., $\hat{r}_{i'j'} - \check{r}_{i'j'}$) is large. Therefore, minimizing the sum of penalized ranges (SPR) is an attempt to encourage smaller block ranges and ensure that less data points are mapped into the blocks with larger ranges which produce looser lower bounds. In other words, more data points can be mapped into the blocks of smaller ranges, resulting in higher pruning power.

4.2 Finding (Θ^*, Φ^*)

A co-reduction function corresponds to a combinatorial transformation of the data matrix. Globally optimizing the data co-reduction according to the objective function in Equation 9 is obviously intractable. Recently, co-clustering which simultaneously partitions both rows and columns of the data matrix has attracted much interests for its quality improvements over traditional single-sided data clustering, especially in gene expression data analysis [7], image classification [11] and text mining [24]. An intuitive explanation is that simultaneous clustering of row and column can reenforce each other. Typically, co-clustering is formulated as an optimization problem which is NP-hard [1].

Our data co-reduction shares similar spirits with co-clustering by taking the duality of both data size and feature dimensionality into consideration. The basic idea is to use row clustering to induce column clustering and use column clustering to induce row clustering as well. The process is intertwined and iterated until it converges to a local optimum. Data co-reduction aims to find an optimal co-reduction function, which is then used to map the data into much more smaller representations to facilitate lossless similarity search.

```
   Input  : $X$, $m$, $l$
   Output: $\Theta^*$, $\Phi^*$
1  $\Theta \leftarrow \Theta^0$;
2  $\Phi \leftarrow \Phi^0$;
3  $SPR \leftarrow$ CompSPR($X$,$\Theta$, $\Phi$);
4  $improved \leftarrow$ true;
5  while $improved$ do
6      $row\_improved \leftarrow$ false;
       // E-step of size-reduction
7      $\Theta_{temp} \leftarrow \Theta$;
8      for $i \leftarrow 1$ to $n$ do
9          for $i' \leftarrow 1$ to $m$ do
10             $\Theta' \leftarrow \Theta$.reassign($i$,$i'$);
11             $NewSPR \leftarrow$ CompSPR($X$,$\Theta'$, $\Phi$);
12             if $NewSPR<SPR$ then
13                 $SPR \leftarrow NewSPR$;
14                 $\Theta_{temp} \leftarrow \Theta_{temp}$.reassign($i$,$i'$);
15                 $row\_improved \leftarrow$ true;
       // M-step of size-reduction
16     if $row\_improved$ then
17         $\Theta \leftarrow \Theta_{temp}$;
18     $col\_improved \leftarrow$ false;
       // E-step of dimensionality-reduction
19     $\Phi_{temp} \leftarrow \Phi$;
20     for $j \leftarrow 1$ to $d$ do
21         for $j' \leftarrow 1$ to $l$ do
22             $\Phi' \leftarrow \Phi$.reassign($j$,$j'$);
23             $NewSPR \leftarrow$CompSPR($X$,$\Theta$, $\Phi'$);
24             if $NewSPR<SPR$ then
25                 $SPR \leftarrow NewSPR$;
26                 $\Phi_{temp} \leftarrow \Phi_{temp}$.reassign($j$,$j'$);
27                 $col\_improved \leftarrow true$;
       // M-step of dimensionality-reduction
28     if $col\_improved$ then
29         $\Phi \leftarrow \Phi_{temp}$;
30     $improved \leftarrow row\_improved \vee col\_improved$
31 $\Theta^* \leftarrow \Theta$;
32 $\Phi^* \leftarrow \Phi$;
33 Return ($\Theta^*$,$\Phi^*$);
```

Algorithm 1: Finding (Θ^*, Φ^*)

We utilize the recently developed Expectation Maximization (EM) algorithm in co-clustering [24] to find the optimal co-reduction function (denoted as (Θ^*, Φ^*)), as outlined in Algorithm 1. The algorithm is to alternate the optimization process on data size and feature dimensionality. First, the algorithm fixes Φ and minimizes the objective function in Definition 6 with respect to Θ, which is called size-reduction. Then it fixes Θ and minimizes the objective function in Definition 6 with respect to Φ, which is called dimensionality-reduction. The process continues until convergence is achieved.

The general EM algorithm is applied in both size-reduction and dimensionality-reduction to find an estimation. There are two steps in the EM algorithm: the Expectation step (E-step) and the Maximization step (M-step). The algorithm starts with an initialization of the co-reduction function (Θ, Φ) and computes its corresponding SPR in Equation 10 (lines 1-3).

In the E-step (lines 7-15) of size-reduction, we update the row mapping based on the fixed co-reduction functions (Θ, Φ) from the last iteration. More precisely, for each row, we find its most appropriate "row-subspace" under the current reduction generated from the last iteration. The objective function is greedily solved by updating one row at a time, and keeping all the other rows fixed. In the M-step (lines 16-17), we update Θ by reassigning the original rows for obtaining the smallest SPR. Similarly, the E-step and the M-step of dimensionality-reduction are conducted in Algorithm 1 (lines 19-29). The finally generated (Θ, Φ) is regarded as the optimal co-reduction function and returned (lines 31-33).

It is important to show the convergence property of this algorithm, as proved in the following Theorem.

THEOREM 3 (MONOTONY OF FINDING (Θ^*, Φ^*)). *The algorithm of finding (Θ^*, Φ^*) monotonically decreases the objective function given in Definition 6.*

PROOF.

1. After the t^{th} iteration, we get $\Theta^{(t)}, \Phi^{(t)}$ and the corresponding sum of penalized ranges $SPR(\Theta^{(t)}, \Phi^{(t)})$.

2. Following the E-step of size-reduction, we fix $\Phi = \Phi^{(t)}$ and find the new assignment for the i^{th} row for the smallest sum of penalized ranges in the current iteration.

3. Following the M-step of size-reduction, we update $\Theta^{(t+1)}$ based on Θ_{temp} which stores the new assignments of all the original rows. Thus, we have $SPR(\Theta^{(t+1)}, \Phi^{(t)}) \leq SPR(\Theta^{(t)}, \Phi^{(t)})$.

4. Following the E-step of dimensionality-reduction, we fix $\Theta = \Theta^{(t+1)}$ and find the new assignment for the j^{th} column for the smallest sum of penalized ranges in the current iteration.

5. Following the M-step of dimensionality-reduction, we update $\Phi^{(t+1)}$ based on Φ_{temp} which stores the new assignment of all the original column. Thus we have $SPR(\Theta^{(t+1)}, \Phi^{(t+1)}) \leq SPR(\Theta^{(t+1)}, \Phi^{(t)})$.

6. From 3 and 5, we have

$$SPR(\Theta^{(t+1)}, \Phi^{(t+1)}) \leq SPR(\Theta^{(t)}, \Phi^{(t)}).$$

□

Therefore, Algorithm 1 is monotonic. It decreases the sum of penalized ranges to a local optimum, given an initial co-reduction function. Note that although the algorithm converges, it does not guarantee the global optimal solution. The time complexity of algorithm 1 is about $O(Iter \cdot (nm +$

$dl)$), where $Iter$ is the number of iterations of the EM steps. In real implementation, $Iter$ can be controlled by checking the degree of changes between iterations. If the changes are extremely small, the algorithm can be stopped without sacrificing much quality. To handle large datasets, sample-based approaches [20] can also be applied for efficiently finding (Θ^*, Φ^*).

So finally, after the optimal co-reduction function is found, the original data matrix X is then co-reduced into X' which has $m \times l$ blocks, each of which is approximated by its minimal and maximal values. Different m and l may lead to different optimal co-reduction functions. For different datasets, different values of m and l may be needed for optimal performance. We will further discuss them in the experiments.

5. EXPERIMENTS

In this section, we report the results of an extensive performance study conducted to evaluate our DCR framework on large and real-life multimedia datasets.

5.1 Set Up

5.1.1 Datasets

Two real-life multimedia datasets are used in our experiments. The original dimensionality of their feature spaces ranges from 432 to 1936.

- YouTube dataset: we extract 250,000 frames from YouTube videos which contain various categories such as news, entertainment, sports, and so on. Each image is divided into 12x12 regions, each of which is represented by a 3-dimensional YUV color moment. That is, each image is represented by a concatenated 432-dimensional local color moment feature vector.

- FACE dataset: we use 50,000 images collected from face-rec.com[1] which consists of face images taken for various people at their different ages. The dataset, named FACE, is processed to derive a 1936-dimensional LDP [28] feature for each image, where each image is divided into 11x11 regions and each region is represented by a 16-dimensional regional LDP histogram.

5.1.2 Performance indicators

Two measures are used for performance indication.

- Pruning power (PP): PP is defined as the percentage of data points being pruned in the filtering step of query processing. In DCR framework, by storing the data points for the same reduced row into consecutive disk pages, one reduced row with multiple data points for actual distance computations incurs a single random access only. For extremely high-dimensional feature spaces, distance computations dominate the total response time since multiple data point candidates consumes one random access. In the filter-and-refine approach, PP shows the capability in filtering data points. It is an objective measure independent of implementations and settings.

- Total response time (TRT): TRT is the total clock time for a query to get kNN results. In DCR framework, it mainly includes the computations of reduced distances and refinement of kNN results. PP only indicates the number of candidates for actual distance computations in refinement. When the original data is less reduced, the computations of reduced distances become more expensive since more reduced rows and columns are involved. Although different implementations and settings may affect TRT, it shows the practical response in answering queries.

[1]http://www.face-rec.org/databases/

5.1.3 Compared methods

For performance comparisons in lossless retrieval, we choose VA-File [26] and iDistance [15] since most dimensionality reduction and hashing methods provide approximate results. Furthermore, VA-File is independent of dimensionality and iDistance shows good performance because of the high efficiency of B$^+$-tree. The number of bits per dimension used in VA-File is 4 and the number of clusters used in iDistance is 10. They are carefully chosen for their good performance.

All experiments are conducted on a desktop with 2.93GHz Intel CPU and 8GB RAM. By default, we set k=10 for kNN search using L_1 distance. Our results on L_2 distance show similar performance. All reported results are the average over 20 randomly selected image queries.

5.2 Tuning m and l

The number of reduced rows m and the number of reduced columns l are the only two parameters to be tuned for in our DCR framework. The smaller the m and l are, the smaller the reduced subspace is. For given m and l, a defined optimal co-reduction function to maximize the pruning power can be found, according to Algorithm 1. Different sets of m and l correspond to different optimal co-reduction functions. Obviously, m and l co-affect the performance of our method.

Figure 5(a) shows the effect of m and l on PP for YouTube dataset. $\frac{n}{m}$ and $\frac{d}{l}$ indicate the reduction ratio on data size and feature dimensionality respectively. As we can see, for the same $\frac{n}{m}$ (or m), PP increases as $\frac{d}{l}$ decreases. A larger l leads to a larger number of reduced columns. That is, less number of dimensions are mapped into the same reduced column. This results in smaller block ranges which can produce tighter lower bounds. Therefore, more data points can be pruned. Similarly, for the same $\frac{d}{l}$ (or l), PP increases as $\frac{n}{m}$ decreases, since a larger m generates more accurate block representations.

Figure 5(b) shows the effect of m and l on TRT for YouTube dataset. Different from the results on PP, TRT lines are concave. When considering $\frac{n}{m}$ and $\frac{d}{l}$ together, although smaller $\frac{n}{m}$ and $\frac{d}{l}$ generate more accurate block representations with smaller ranges (i.e., higher pruning power), they also lead to a larger number of reduced distance computations in higher dimensional subspaces. In the extremely case when $m = n$ and $l = d$, it performs exactly the same as sequential scan. Therefore, a trade-off on the cost between the filtering step and the refinement step needs to be achieved. Based on the search time shown in Figure 5(b), we set $\frac{n}{m} = 30$ and $\frac{d}{l} = 10$ as the default values for YouTube dataset in the rest experiments. The subspace is just $\frac{2}{30\times 10} = \frac{1}{150}$ of the original space, comparing with VA-File which is $\frac{1}{8}$ of the original space. Note that in our dataset, data size is far greater than feature dimensionality. When n and d are comparable, they are expected to have similar impact on the performance.

Figures 6(a) and (b) depict the effect of m and l on PP and TRT for FACE dataset respectively. This dataset has a much higher dimensionality than YouTube dataset. Hence the performance becomes noticeably less sensitive to dimensionality reduction. Meanwhile, the data size is smaller to YouTube, making it more sensitive to feature grouping. This explains why the TRT varies little with $\frac{d}{l}$ but relates more to the changes on $\frac{n}{m}$. From Figure 6(b) we can conclude that when $\frac{n}{m} = 15$ and $\frac{d}{l} = 35$ it reaches the optimal performance. So we set 15 and 35 as the default values for $\frac{n}{m}$ and $\frac{d}{l}$ respectively for FACE dataset.

5.3 Effect of Size

In this experiment, we test the effect of data size on the performance with comparisons.

Figure 7(a) shows the results of PP for different methods on YouTube dataset of different data sizes. We start with 100K data points and add 50K every time. The first observation is that VA-File achieves the best pruning power, followed by DCR which outperforms iDistance greatly. Since VA-File does not have reduction on either size or dimensionality, it computes the lower bound for every data point in full dimensionality. It is expected that its lower bound is tightest. DCR achieves comparable pruning power to VA-File since it can perform optimal co-reduction for given m and l. iDistance has the worst pruning power since mapping the 432-dimensional feature space into a one-dimensional space causes heavy information loss and many data points have very similar distances to the same reference point. The second observation is that while VA-File and iDistance have very stable pruning power for different data sizes, DCR's pruning power grows as data size increases. This can be explained as follows. Given the fixed $\frac{n}{m}$, the average number of data points in each block remains the same. For a data point in a database, its nearest neighbor's distance is likely smaller when more data points are added into the database. Inductively, the same-sized blocks are likely to contain data points that are closer to each other in a larger dataset than those in a smaller dataset. As a result, block ranges can be reduced and tighter lower bounds can be computed for a higher pruning power.

Figure 7(b) shows the results of TRT for different methods on YouTube dataset of different data sizes. In terms of TRT, DCR outperforms iDistance which in turn improves VA-File. VA-File has the highest pruning power. However, its lower bound computations are expensive because every data point's lower bound still needs to be calculated in very high dimensionality. Although iDistance has the worst pruning power, its filtering step is fastest since range search in B^+-tree is extremely efficient. DCR utilizes the duality of size and dimensionality to integrate the advantages of fast pruning and tight lower bounds. As data size increases, TRT for DCR increases in a much slower pace than other methods, mainly contributed by the improved PP with larger data sizes.

Figures 8(a) and (b) shows the the results of PP and TRT on FACE dataset of different data sizes, where similar observations can be seen.

5.4 Effect of Dimensionality

In this experiment, we test the effect of dimensionality on the performance with comparisons.

Here we only use YouTube dataset since its dimensionality is high enough. Every time we extract the first d-dimensional values in the original 432-dimensional space and vary d from 250 to 432. Figure 9(a) shows the results of PP for different methods on YouTube dataset of different dimensionalities. VA-File has a relatively stable pruning power since it is not very sensitive to the dimensionality. As dimensionality increases, PP for DCR goes up while PP for iDistance goes down slightly. In a higher-dimensional space, since $\frac{d}{l}$ is fixed, block ranges in DCR tend to get smaller for the same reason in a larger-sized dataset. However, iDistance's pruning power gets worse since more information is lost during one-dimensional transformation. From Figure 9(b) which shows the results of TRT, the performance of VA-File and iDistance deteriorates quickly as dimensionality grows. DCR's performance is slightly changed since the increase in the reduced distance computations is largely offset by more effective dimensionality reduction. This experiment confirms that DCR can achieve larger performance gains over existing methods in higher-dimensional spaces.

5.5 Effect of k

The effect of k in kNN search is also tested. From Figures 10 and 11 for YouTube and FACE dataset respectively, we can see that the pruning power decreases and the total response time increases as k goes up for all methods. This is reasonable since the k^{th} nearest neighbor's distance becomes larger as k increases. In the filter-and-refine approach, it means that the filtering condition is harder to be satisfied and less data points can be pruned. If the increase of k causes huge increase in the k^{th} nearest neighbor's distance, the performance may be affected significantly. Nonetheless, DCR is able to outperforms VA-File and iDistance for different k values on total response time.

6. CONCLUSION

In this paper, we introduce a framework for Data Co-Reduction (DCR) on both data size and feature dimensionality. DCR simultaneously reduces both size and dimensionality of the original data into a compact subspace, where lower bounds of the actual distances in the original space can be efficiently established to achieve fast and lossless similarity search. It achieves the optimal co-reduction which generates the least number of candidates for refinement in query processing. From the experimental results performed on real-life datasets, DCR outperforms existing methods significantly in both time and space efficiency. Since the current lossless block representation is quite loose, in future we plan to investigate tighter block representations based on statistical information to test the performance of approximate search.

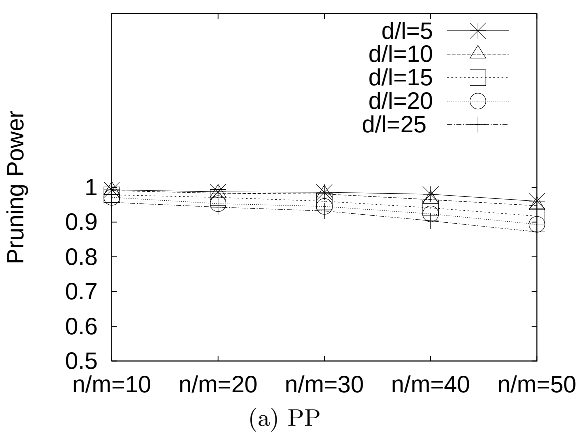
7. REFERENCES

[1] A. Anagnostopoulos, A. Dasgupta, and R. Kumar. Approximation algorithms for co-clustering. In *PODS*, pages 201–210, 2008.

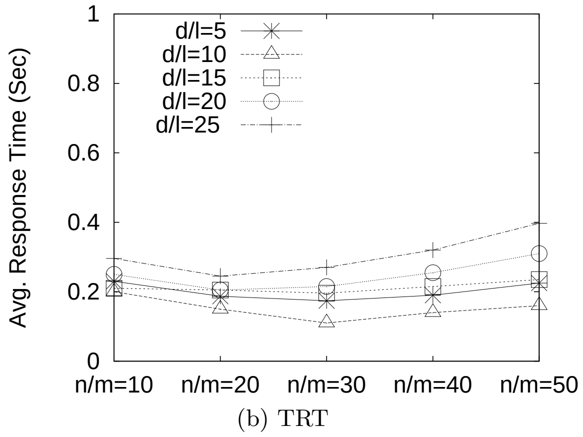
[2] A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *CACM*, 51(1):117–122, 2008.

[3] S. Berchtold, C. Böhm, H. V. Jagadish, H.-P. Kriegel, and J. Sander. Independent quantization: An index compression technique for high-dimensional data spaces. In *ICDE*, pages 577–588, 2000.

[4] C. Böhm, S. Berchtold, and D. A. Keim. Searching in high-dimensional spaces: Index structures for

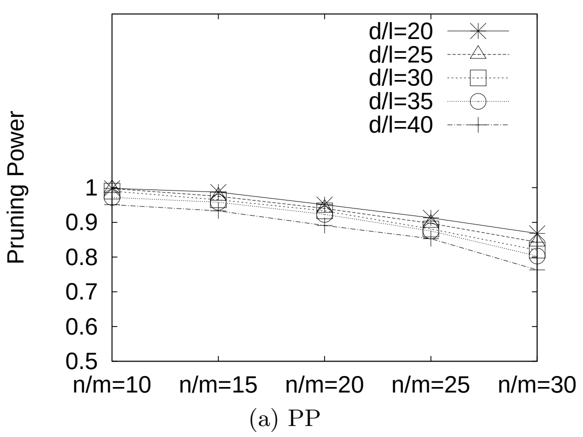


(a) PP

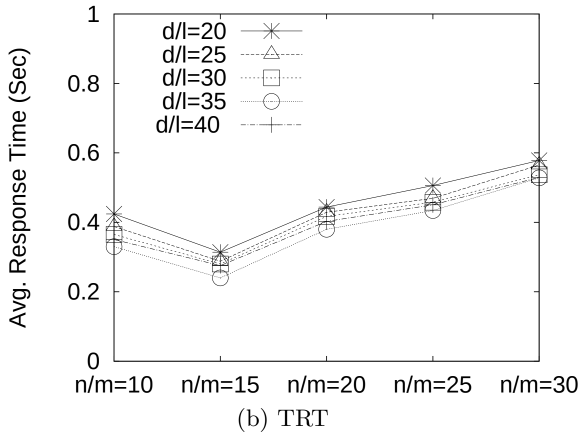


(b) TRT

Figure 5: Tuning m and l on YouTube dataset

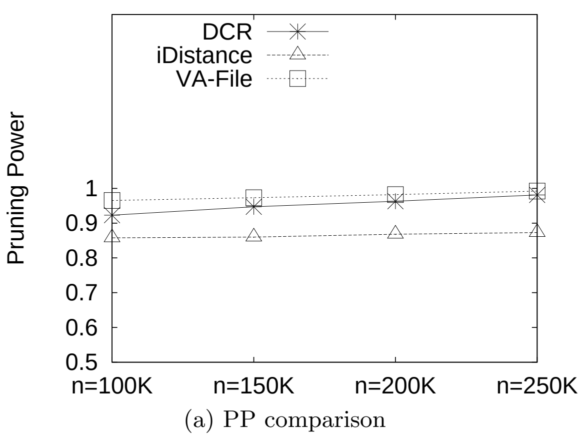


(a) PP

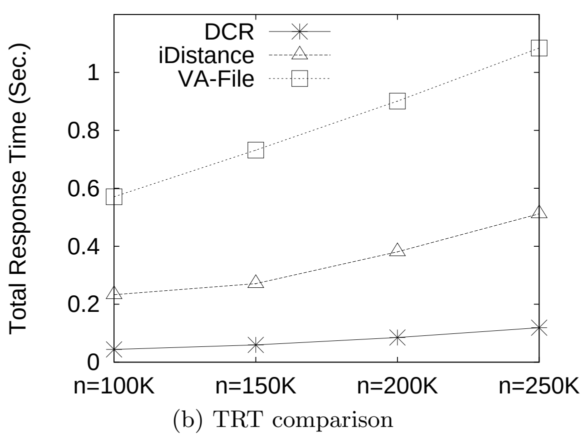


(b) TRT

Figure 6: Tuning m and l on FACE dataset

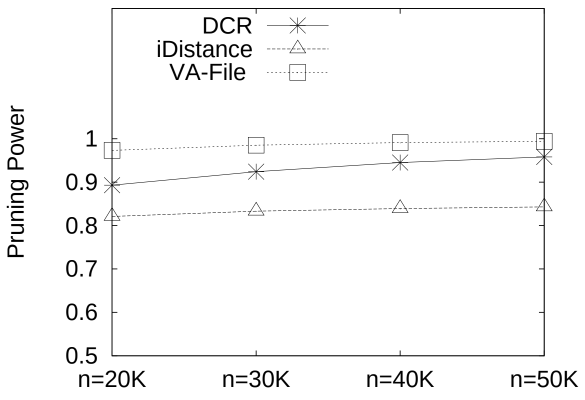


(a) PP comparison

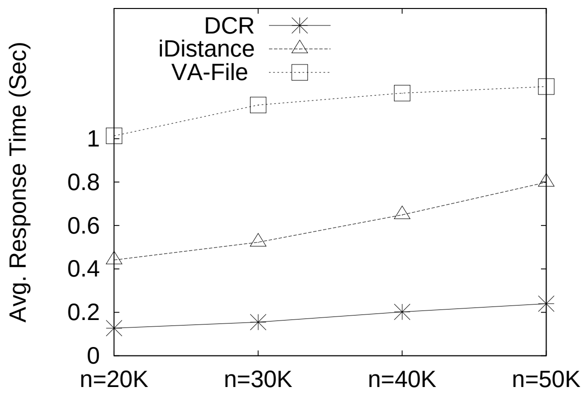


(b) TRT comparison

Figure 7: Comparison w.r.t size on YouTube dataset

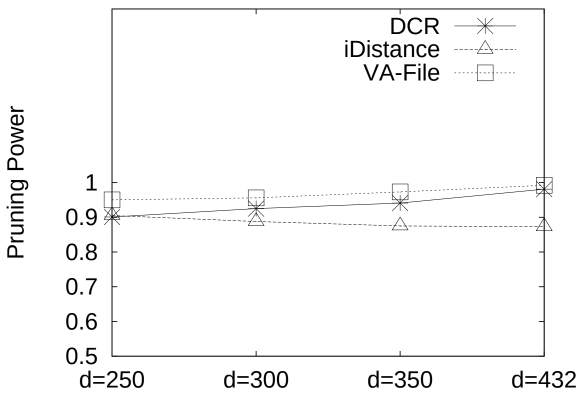


(a) PP comparison

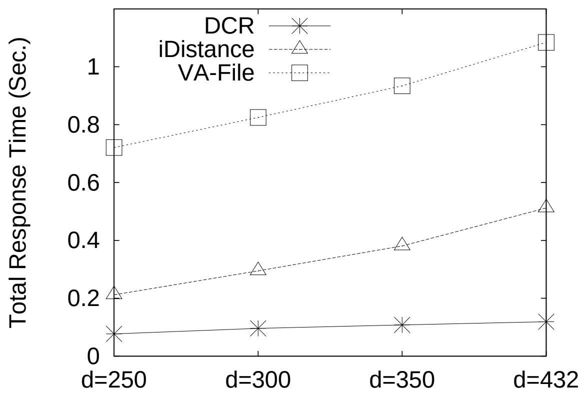


(b) TRT comparison

Figure 8: Comparison w.r.t size on FACE dataset

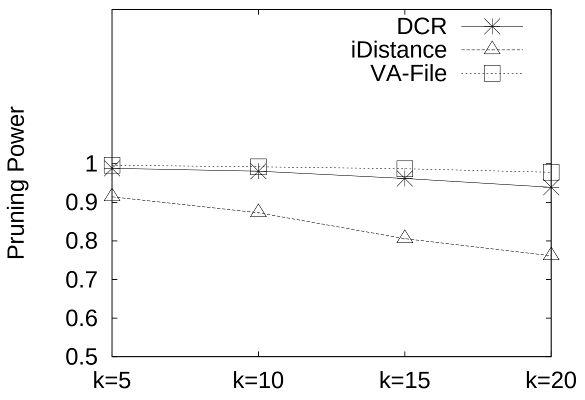


(a) PP comparison

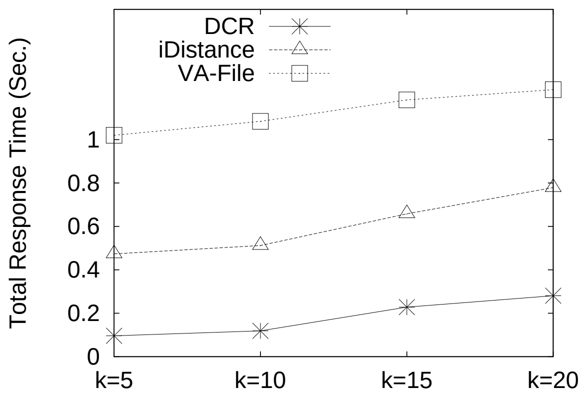


(b) TRT comparison

Figure 9: Comparison w.r.t dimensionality on YouTube dataset

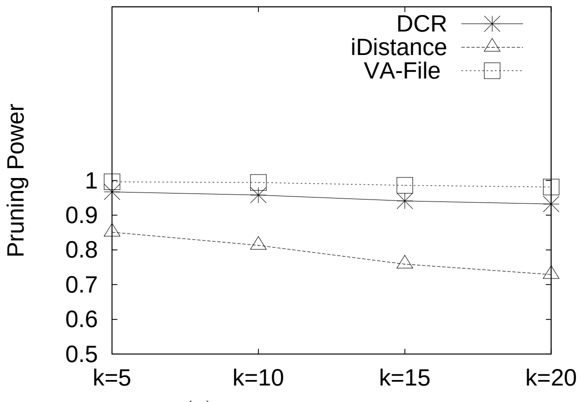


(a) PP comparison

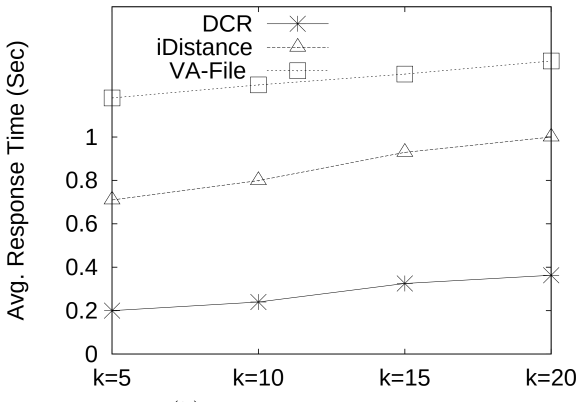


(b) TRT comparison

Figure 10: Comparison w.r.t k on YouTube dataset



(a) PP comparison



(b) TRT comparison

Figure 11: Comparison w.r.t k on FACE dataset

improving the performance of multimedia databases. *ACM Computing Survey*, 33(3):322–373, 2001.

[5] K. Chakrabarti and S. Mehrotra. Local dimensionality reduction: A new approach to indexing high dimensional spaces. In *VLDB*, pages 89–100, 2000.

[6] L. Chen and R. T. Ng. On the marriage of lp-norms and edit distance. In *VLDB*, pages 792–803, 2004.

[7] Y. Cheng and G. M. Church. Biclustering of expression data. In *ISMB*, pages 93–103, 2000.

[8] P. Ciaccia, M. Patella, and P. Zezula. M-tree: An efficient access method for similarity search in metric spaces. In *VLDB*, pages 426–435, 1997.

[9] R. Datta, D. Joshi, J. Li, and J. Z. Wang. Image retrieval: Ideas, influences, and trends of the new age. *ACM Computing Survey*, 40(2), 2008.

[10] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *VLDB*, pages 518–529, 1999.

[11] Q. Gu and J. Zhou. Co-clustering on manifolds. In *KDD*, pages 359–368, 2009.

[12] X. He and P. Niyogi. Locality preserving projections. In *NIPS*, 2003.

[13] M. E. Houle and J. Sakuma. Fast approximate similarity search in extremely high-dimensional data sets. In *ICDE*, pages 619–630, 2005.

[14] Z. Huang, H. T. Shen, J. Shao, S. M. Rüger, and X. Zhou. Locality condensation: a new dimensionality reduction method for image retrieval. In *ACM Multimedia*, pages 219–228, 2008.

[15] H. V. Jagadish, B. C. Ooi, K.-L. Tan, C. Yu, and R. Zhang. iDistance: An adaptive B^+-tree based indexing method for nearest neighbor search. *TODS*, 30(2):364–397, 2005.

[16] N. Koudas, B. C. Ooi, H. T. Shen, and A. K. H. Tung. LDC: Enabling search by partial distance in a hyper-dimensional space. In *ICDE*, pages 6–17, 2004.

[17] M. S. Lew, N. Sebe, C. Djeraba, and R. Jain. Content-based multimedia information retrieval: State of the art and challenges. *TOMCCAP*, 2(1):1–19, 2006.

[18] Q. Lv, W. Josephson, Z. Wang, M. Charikar, and K. Li. Multi-probe LSH: Efficient indexing for high-dimensional similarity search. In *VLDB*, pages 950–961, 2007.

[19] K. McGrath. *Status update: Facebook logs 500 million members*. USA Today, 2010-07-22.

[20] F. Pan, X. Zhang, and W. Wang. Crd: fast co-clustering on large datasets utilizing sampling-based matrix decomposition. In *SIGMOD*, pages 173–184, 2008.

[21] Y. Sakurai, M. Yoshikawa, S. Uemura, and H. Kojima. The A-tree: An index structure for high-dimensional spaces using relative approximation. In *VLDB*, pages 516–526, 2000.

[22] H. T. Shen, B. C. Ooi, X. Zhou, and Z. Huang. Towards effective indexing for very large video sequence database. In *SIGMOD*, pages 730–741, 2005.

[23] H. T. Shen, X. Zhou, and A. Zhou. An adaptive and dynamic dimensionality reduction method for high-dimensional indexing. *VLDB Journal*, 16(2):219–234, 2007.

[24] Y. Song, S. Pan, S. Liu, F. Wei, M. X. Zhou, and W. Qian. Constrained co-clustering for textual documents. In *AAAI*, 2010.

[25] Y. Tao, K. Yi, C. Sheng, and P. Kalnis. Quality and efficiency in high dimensional nearest neighbor search. In *SIGMOD*, pages 563–576, 2009.

[26] R. Weber, H.-J. Schek, and S. Blott. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In *VLDB*, pages 194–205, 1998.

[27] M. Wichterich, I. Assent, P. Kranen, and T. Seidl. Efficient EMD-based similarity search in multimedia databases via flexible dimensionality reduction. In *SIGMOD*, pages 199–212, 2008.

[28] B. Zhang, Y. Gao, S. Zhao, and J. Liu. Local derivative pattern versus local binary pattern: Face recognition with high-order local pattern descriptor. *IEEE TIP*, 19(2):533–544, 2010.

[29] Z. Zhang, B. C. Ooi, S. Parthasarathy, and A. K. H. Tung. Similarity search on bregman divergence: Towards non-metric indexing. *PVLDB*, 2(1):13–24, 2009.