A General Cost Model for Dimensionality Reduction in High Dimensional Spaces

Xiang Lian and Lei Chen
Department of Computer Science and Engineering
Hong Kong University of Science and Technology
Clear Water Bay, Kowloon, Hong Kong
{*xlian, leichen*}*@cs.ust.hk*

Abstract

Similarity search usually encounters a serious problem in the high dimensional space, known as the "curse of dimensionality". In order to speed up the retrieval efficiency, previous approaches usually reduce the dimensionality of the entire data set to a fixed lower value before building indexes (referred to as global dimensionality reduction (GDR)). More recent works focus on locally reducing the dimensionality of data to different values (called the local dimensionality reduction (LDR)). However, so far little work has formally evaluated the effectiveness and efficiency of both GDR and LDR for range queries. Motivated by this, in this paper, we propose a general cost model for both GDR and LDR, in light of which we introduce a novel LDR method, PRANS*. It can achieve high retrieval efficiency with the guarantee of optimality given by the formal model. Finally, a B^+-tree index is constructed over the reduced partitions for fast similarity search. Extensive experiments validate the correctness of our cost model on both real and synthetic data sets, and demonstrate the efficiency and effectiveness of the proposed* PRANS *method.*

1 Introduction

Similarity search in *high* dimensional space has many applications such as information retrieval [19], image data analysis [11], time-series matching [1, 9], and the like. One serious problem in achieving efficient and effective similarity search is well-known as the "curse of dimensionality". That is, if we build an index on high dimensional data objects directly, the *similarity query* with the index might have worse performance than that of a linear scan (i.e., without the index). One way to tackle this problem is to design an efficient index that can break the "dimensionality curse" [3, 17, 21, 6]. However, the proposed indexes can only work well in the *low* or *medium* ($\leq$ 15 [13]) dimensional space, and their query performance usually dramatically degrades with the increasing dimensionality [4].

Another approach is to reduce the dimensionality of data objects before indexing them. There are two categories of proposals, *global dimensionality reduction* (GDR) [1, 9, 28, 18, 5] and *local dimensionality reduction* (LDR) [7, 14]. For GDR, the dimensionality of all data objects are reduced to a *fixed* lower value, whereas for LDR the reduction is made to different values. In particular, in GDR, data objects are first transformed to a reasonably lower dimensional space and then inserted into a multidimensional index, such as M-tree [8] or R-tree [10], which can efficiently handle queries in the reduced space. There are many dimensionality reduction techniques in literature [1, 9, 28, 18, 5]. These techniques can guarantee no *false dismissals* for the similarity search, however, the search result may contain *false positives* which have to be further refined in a *post-processing* step. In LDR [7, 14], the entire data set is divided into several partitions, each of which reduces the dimensionality of data to a best value locally, according to the partition characteristics.

Previous work on both GDR and LDR are *heuristic-based*. In particular, GDR requires users to select the value of the reduced dimension, whereas in LDR users have to give some data-dependent thresholds. Since both methods are heuristic-based, no guarantee of optimality is provided on the reduction, which may affect the search performance. To the best of our knowledge, no existing work has formally evaluated the effectiveness and efficiency of GDR and LDR approaches. Motivated by this, we make the following contributions in this paper:

1. We propose a general cost model for both GDR and LDR approaches, taking into consideration the *pruning power* and *computation cost*, which we give the definitions later, of *range queries*[1].
2. The cost model derived for GDR gives a hint of selecting the best value of the reduced dimension, where users can flexibly make the balance between the *pruning power* and *computation cost*.

[1]The result can be easily extended to *k nearest neighbor queries* [25].

1-4244-0803-2/07/$20.00 ©2007 IEEE.

3. In light of the cost model for LDR, we present a partitioning-based LDR approach, PRANS, to divide the data set into several disjoined partitions, each of which locally reduces the dimensionality of data objects[2].

The rest of the paper is organized as follows. Section 2 gives a brief review of high dimensional indexes, followed by previous work on GDR and LDR techniques. Section 3 presents the formal problem definition. Section 4 introduces our general cost model for range queries. Section 5 illustrates the partitioning-based LDR method, PRANS, using the proposed cost model. Section 6 discusses details of the index construction. Section 7 verifies the correctness of our cost model and compares the effectiveness and efficiency of our approach PRANS to the existing LDR method, MMDR, with extensive experiments. Finally, Section 8 concludes this paper.

2 Related Work

In order to perform an efficient similarity search in the high dimensional space, many works focus on designing high dimensional indexes, such as X-tree [3], SR-tree [17], TV-tree [21], Hybrid-tree [6], and so on. Some tree indexes, however, are effective on specific data types. For example, TV-tree [21] works well when feature vectors have coordinates with small diversity, such as categorical or string data. Furthermore, although these indexes work very well in *low* or *medium* dimensional spaces, their query performance always degrades rapidly when the dimensionality increases (e.g. up to 100). This is a well-known problem called the "curse of dimensionality" [4]. In the high dimensional space, the distance from a query object to its nearest neighbor is nearly the same as that to its farthest neighbor, which makes it difficult to prune the search space through indexes. Thus, for most indexes over high dimensional objects, the overhead of a *range query* can be even greater than that of a *linear scan*. The high dimensional indexing problem also motivates previous work on improving the retrieval efficiency of the *linear scan*, for example, VA-file [26].

There are many studies on various dimensionality reduction techniques to be used as GDR, such as *Singular Value Decomposition* (SVD) [16], *Discrete Fourier Transform* (DFT) [1, 9], *Discrete Wavelet Transform* (DWT) [23], *Piecewise Aggregate Approximation* (PAA) [28], *Adaptive Piecewise Constant Approximation* (APCA) [18], and *Chebyshev Polynomial* [5]. In order to guarantee no *false dismissals* (actual answers that are absent in the final result) during a query, any dimensionality reduction technique must follow a *lower bounding lemma*, that is, the *Euclidean distance* between any two transformed objects is never greater than that in the original space. For any given query object, we first transform it to a point in the lower dimensional space over which we have already build an index, and then search over the index. Since the resulting candidate set may contain *false positives* (objects that do not belong in the actual answer set), the final *post-processing* step refines all the candidates. Other GDR approaches map all the high dimensional objects into a 1-dimensional space, such as *pyramid* [2], *Hilbert curve* [15], *iMinMax* [29], and use usual 1-dimensional indexes to search on this space.

LDR converts data objects into different dimensional spaces according to data characteristics. Chakrabarti and Mehrotra [7] explored local correlations in the data set using the existing clustering algorithm and apply *Principal Component Analysis* (PCA) on each correlated cluster individually. In this work, one important clustering criterion is that the *reconstruction distance* of any data object must be never greater than a user-specified bound. However, it allows a small fraction of objects to violate this condition, which are added to an outlier set. Each cluster is indexed independently with a multidimensional tree whose root is collected in a directory. The resulting hybrid structure is not guaranteed to be balanced and, thus, may incur high I/O cost for clusters with skewed sizes. Jin et al. [14] proposed another LDR method, MMDR, which can identify ellipse shaped clusters in the subspace using the *Mahalanobis distance*. Moreover, one of the clustering criteria requires that the average *representation error* in clusters not exceed a user-specified threshold. After all clusters are obtained, each data object in the reduced subspace is hashed to a unique 1-dimensional key and then inserted into a B^+ tree following the framework of $iDistance$ [13].

All known techniques following both GDR and LDR are *heuristic-based*. In particular, GDR globally reduces dimensionality of data to a specific dimension value, however, no methodology has been proposed to select this value. Furthermore, although previous LDR approaches aim at minimizing the lossy information caused by the dimensionality reduction, the clustering results highly depend on the user-specified parameters. To the best of our knowledge, no existing work has provided a concrete cost model to formalize the analysis of the query performance of both GDR and LDR. Therefore, in this paper, we give a general cost model to estimate the query cost, based on which a novel LDR approach is proposed.

3 Problem Definition

Given a very large database $\mathcal{D}$ containing N data objects of high dimension d (e.g. $d = 100$), GDR reduces the dimensionality of all data objects from d to a *fixed* lower one, $d_G \ll d$. On the other hand, since data objects in $\mathcal{D}$ may be correlated in different subspaces, LDR divides $\mathcal{D}$ into m

[2]Throughout this paper, we assume that all partitions are applied the same reduction technique. However, the proposed approach can be easily extended to the case where different reduction techniques are used for different partitions.

disjoint partitions $\mathcal{P}_1$, $\mathcal{P}_2$, ..., and $\mathcal{P}_m$, and locally reduces the dimensionality from d to d_i ($1 \leq d_i \ll d$) for each partition $\mathcal{P}_i$.

In order to facilitate the similarity search, a number of pivots are selected in the transformed data set. Without loss of generality, we assume that GDR randomly selects one pivot $p \in \mathcal{D}$, whereas LDR randomly selects one pivot $p_i \in \mathcal{P}_i$ for each partition $\mathcal{P}_i$, where $1 \leq i \leq m$. We will discuss later how the case of selecting one pivot within each partition can be easily extended to that of choosing multiple ones. By pre-computing the distance $dist(p_i, o_j)$ between any object o_j in a partition $\mathcal{P}_i$ and its corresponding pivot p_i, the *triangle inequality* can help prune the search space of queries. In particular, given a query q, the triangle inequality holds that $dist(q, o_j) \geq |dist(q, p_i) - dist(p_i, o_j)|$, where $dist$ is a distance function. For a range query with a radius r_q, as long as $|dist(q, p_i) - dist(p_i, o_j)| > r_q$, object o_j can be safely pruned, since $dist(q, o_j) \geq r_q$ by the inequality transition. Similarly, the solution can be easily extended to handle kNN queries [25]. In the sequel, we mainly focus on the range query q with a *fixed* radius r_q. To address the general cases, we later extend it to range queries that have radii with any distribution.

Given a query q, we define two measures to characterize the performance of the dimensionality reduction techniques, the *pruning power* (PP) and *computation cost* (CC). Specifically, PP is given by the number of data objects that can be pruned by the triangle inequality without introducing false dismissals, whereas CC is measured by the number of real distance computations (i.e. in terms of the distance computation between two d-dimensional objects) during query processing. With these two measures, we formally present two problems of deriving a general cost model for both GDR and LDR.

Problem 3.1 *(Cost Model for GDR) Given a database $\mathcal{D}$ with d-dimensional data objects, we reduce the dimensionality of all data objects to a fixed value d_G ($\ll d$). The problem is to derive a cost model for GDR in terms of PP and CC, such that the best value of d_G is selected to either maximize PP or minimize CC.*

Problem 3.2 *(Cost Model for LDR) Given a database $\mathcal{D}$ with d-dimensional data objects, we divide $\mathcal{D}$ into m disjoint partitions $\mathcal{P}_1$, $\mathcal{P}_2$, ..., and $\mathcal{P}_m$, and reduce the dimensionality of each data object in partition $\mathcal{P}_i$ from d to d_i ($1 \leq d_i \ll d$). The problem is to design a cost model for LDR in terms of PP and CC, in light of which partitions of $\mathcal{D}$ are obtained such that either PP is maximized or CC is minimized.*

According to the two problems above, our fundamental goal is to formalize PP and CC for query processing with either GDR or LDR. Note however, that the cost model for LDR in Problem 3.1 is more general and can be reduced to that of GDR in Problem 3.2, by letting $m = 1$. Therefore, in the sequel, we first derive a cost model for LDR, and then extend it to the GDR. Furthermore, in LDR, since the converted data in each partition $\mathcal{P}_i$ are of different dimensions d_i, we are also interested in building up an index structure for these partitions to achieve an efficient similarity search. Figure 1 summarizes the commonly-used symbols in this paper.

Symbol	Description
$\mathcal{D}$	a data set of size N
$\mathcal{P}_i$	the i-th partition of the data set $\mathcal{D}$
$\lvert \cdot \rvert$	the data size in $\cdot$
m	the number of partitions
d	the original high dimension of data objects
d_G	the reduced dimension of GDR
d_i	the reduced dimension of partition $\mathcal{P}_i$
p_i	a pivot in the partition $\mathcal{P}_i$
q	a query point
r_q	the radius of a range query centered at q
$\phi(z)$	the *cumulative distribution function* (CDF) following a *normal* distribution

Figure 1. Meanings of Symbols Used

4 Cost Model for Range Queries

4.1 Cost Model for LDR

We first focus on the definition of the pruning power $PP_{LDR}(q, r_q)$ for LDR. From the probabilistic point of view, $PP_{LDR}(q, r_q)$ can be obtained by summing up the probability, with which each data object is pruned by the triangle inequality. That is,

$$\begin{aligned} & PP_{LDR}(q, r_q) \\ = & \sum_{i=1}^{m} (\text{the number of pruned objects in } \mathcal{P}_i) \\ = & \sum_{i=1}^{m} \sum_{j=1}^{|\mathcal{P}_i|} Pr\{o_j \text{ can be pruned, for } o_j \in \mathcal{P}_i\} \\ = & \sum_{i=1}^{m} \sum_{j=1}^{|\mathcal{P}_i|} Pr\{|dist^{(d_i)}(q, p_i) - dist^{(d_i)}(p_i, o_j)| > r_q\} \\ = & \sum_{i=1}^{m} \sum_{j=1}^{|\mathcal{P}_i|} (1 - Pr\{|dist^{(d_i)}(q, p_i) - dist^{(d_i)}(p_i, o_j)| \leq r_q\}) \end{aligned} \quad (1)$$

where $dist^{(d_i)}(x, y)$ is the distance between objects x and y in their reduced d_i-dimensional space.

Let $\Pr\{|dist^{(d_i)}(q, p_i) - dist^{(d_i)}(p_i, o_j)| \leq r_q\}$ be $Prob_i$, that is, the probability that any object $o_j \in \mathcal{P}_i$ cannot be pruned by pivot p_i. We have

$$\begin{aligned} PP_{LDR}(q, r_q) &= \sum_{i=1}^{m} (|\mathcal{P}_i| - |\mathcal{P}_i| \cdot Prob_i) \\ &= |\mathcal{D}| - \sum_{i=1}^{m} (|\mathcal{P}_i| \cdot Prob_i) \end{aligned} \quad (2)$$

Based on Equation (2), the pruning power of range queries largely depends on the partitioning strategy, which is related to the data size of each partition $\mathcal{P}_i$ and the probability $Prob_i$ of that partition that we discuss later.

As a second step, we consider the computation cost of range queries with LDR. Obviously, for each partition

$\mathcal{P}_i$, as long as the data size of partition $\mathcal{P}_i$ is not zero, we have to compute the distance $dist^{(d_i)}(q, p_i)$ between query q and pivot p_i in the d_i-dimensional space, whose computation cost is $O(d_i)$. Then, for each partition $\mathcal{P}_i$, we sequentially scan $|\mathcal{P}_i|$ data objects. If an object o_j in $\mathcal{P}_i$ cannot be pruned by the triangle inequality, the real distance $dist^{(d)}(q, o_j)$ between query q and object o_j is computed, which has the computation cost $O(d)$. Note that, the probability that a data object in $\mathcal{P}_i$ cannot be pruned is exactly $Prob_i$. Therefore, the computation cost $CC_{LDR}(q, r_q)$ is given as follows:

$$\begin{aligned} & CC_{LDR}(q, r_q) \\ &= \sum_{i=1}^{m}(d_i \cdot sign(|\mathcal{P}_i|) + d \cdot \sum_{j=1}^{|\mathcal{P}_i|} Prob_i) \\ &= \sum_{i=1}^{m}(d_i \cdot sign(|\mathcal{P}_i|) + d \cdot |\mathcal{P}_i| \cdot Prob_i) \end{aligned} \quad (3)$$

where $sign(|x|) = 1$, if $|x| \neq 0$; 0, otherwise.

In the above equations, although we assume only one random pivot selected from each partition, it can be easily extended to the case of n randomly chosen pivots by replacing $Prob_i$ with ${Prob_i}^n$, and d_i with nd_i. Since we have derived the cost model for LDR in terms of the pruning power PP_{LDR} and computation cost CC_{LDR}, we can partition the data set based on this model to either maximize PP_{LDR} or minimize CC_{LDR}. In other words, we can partition the data set such that either $\sum_{i=1}^{m}(|\mathcal{P}_i| \cdot Prob_i)$ or $\sum_{i=1}^{m}(d_i \cdot sign(|\mathcal{P}_i|) + d \cdot |\mathcal{P}_i| \cdot Prob_i)$ is minimized.

In the cost model of LDR, one remaining issue that needs to be addressed is the probability $Prob_i$ (= $\Pr\{|dist^{(d_i)}(q, p_i) - dist^{(d_i)}(p_i, o_j)| \leq r_q\}$). We formally rewrite the formula of $Prob_i$ as follows.

$$\begin{aligned} & \forall p_i, o_j \in \mathcal{P}_i, \forall q \in \mathcal{D} \\ & Prob_i = Pr\{|dist^{(d_i)}(q, p_i) - dist^{(d_i)}(p_i, o_j)| \leq r_q\} \\ & = Pr\{-r_q \leq dist^{(d_i)}(q, p_i) - dist^{(d_i)}(p_i, o_j) \leq r_q\} \end{aligned} \quad (4)$$

Here, we obtain $Prob_i$ by utilizing the *Central Limit Theorem* [27]. In Equation (4), $dist^{(d_i)}(q, p_i)$ and $dist^{(d_i)}(p_i, o_j)$ can be viewed as two independently generated values from random variables $X_{\mathcal{P}_i}$ and $Y_{\mathcal{P}_i}$ with means $\mu_{X_{\mathcal{P}_i}}$ and $\mu_{Y_{\mathcal{P}_i}}$, and variances $\sigma_{X_{\mathcal{P}_i}}$ and $\sigma_{Y_{\mathcal{P}_i}}$, respectively. According to the *Central Limit Theorem*, if a random variable v is drawn from $V = X - Y$, then $\frac{v-(\mu_{X_{\mathcal{P}_i}} - \mu_{Y_{\mathcal{P}_i}})}{\sqrt{\sigma^2_{X_{\mathcal{P}_i}} + \sigma^2_{Y_{\mathcal{P}_i}}}} = \frac{dist^{(d_i)}(q,p_i) - dist^{(d_i)}(p_i,o_j) - (\mu_{X_{\mathcal{P}_i}} - \mu_{Y_{\mathcal{P}_i}})}{\sqrt{\sigma^2_{X_{\mathcal{P}_i}} + \sigma^2_{Y_{\mathcal{P}_i}}}}$ follows the cumulative distribution function $\phi(z)$ of a *normal distribution*. Thus, we have:

$$\begin{aligned} & Prob_i \\ &= Pr\{\frac{-r_q - (\mu_{X_{\mathcal{P}_i}} - \mu_{Y_{\mathcal{P}_i}})}{\sqrt{\sigma^2_{X_{\mathcal{P}_i}} + \sigma^2_{Y_{\mathcal{P}_i}}}} \leq \\ & \frac{dist^{(d_i)}(q,p_i) - dist^{(d_i)}(p_i,o_j) - (\mu_{X_{\mathcal{P}_i}} - \mu_{Y_{\mathcal{P}_i}})}{\sqrt{\sigma^2_{X_{\mathcal{P}_i}} + \sigma^2_{Y_{\mathcal{P}_i}}}} \leq \frac{r_q - (\mu_{X_{\mathcal{P}_i}} - \mu_{Y_{\mathcal{P}_i}})}{\sqrt{\sigma^2_{X_{\mathcal{P}_i}} + \sigma^2_{Y_{\mathcal{P}_i}}}}\} \\ &= \phi(\frac{r_q - (\mu_{X_{\mathcal{P}_i}} - \mu_{Y_{\mathcal{P}_i}})}{\sqrt{\sigma^2_{X_{\mathcal{P}_i}} + \sigma^2_{Y_{\mathcal{P}_i}}}}) - \phi(\frac{-r_q - (\mu_{X_{\mathcal{P}_i}} - \mu_{Y_{\mathcal{P}_i}})}{\sqrt{\sigma^2_{X_{\mathcal{P}_i}} + \sigma^2_{Y_{\mathcal{P}_i}}}}). \end{aligned} \quad (5)$$

4.2 Cost Model for GDR

After obtaining the cost model for LDR (Problem 3.2), we discuss its special case, Problem 3.1, in detail. Recall that, in Problem 3.1, we reduce the dimension of all data objects to a *fixed* value d_G. We can obtain the formulas of PP_{GDR} and CC_{GDR} for GDR by letting $|\mathcal{P}_j| = |\mathcal{D}|$ for $d_j = d_G$ and $|\mathcal{P}_i| = 0$ for all $i \neq j$ in equations of PP_{LDR} and CC_{LDR}.

$$PP_{GDR}(q, r_q) = |\mathcal{D}| \cdot (1 - Prob_j), \text{and} \quad (6)$$

$$CC_{GDR}(q, r_q) = (d_G + |\mathcal{D}| \cdot d \cdot Prob_j). \quad (7)$$

In order to make the trade-off between PP_{GDR} and CC_{GDR}, users can specify a weight $w \in [0, 1]$ such that the overall cost $TotalCost_{GDR}(d_G, q, r_q)$ of a query can be scored as:

$$\begin{aligned} & TotalCost_{GDR}(d_G, q, r_q) \\ & = w \cdot Prob_j + (1 - w) \cdot \frac{d_G + |\mathcal{D}| \cdot d \cdot Prob_j}{d \cdot (|\mathcal{D}| + 1)}. \end{aligned} \quad (8)$$

Therefore, the best dimension value d_G for Problem 3.1 is the one that minimizes $TotalCost_{GDR}(d_G, q, r_q)$, for $1 \leq d_G \leq d$. From Equation (8), we can see that the total cost of GDR mainly depends on two terms d_G and $Prob_j$. In order to minimize this cost, we can either minimize d_G or $Prob_j$. In particular, d_G is related to the value of $Prob_j$ in Equation (8), that is, the probability that an object can be pruned using d_G reduced dimensions. So an appropriate selection of d_G leads to small $Prob_j$ and thus low total cost. Note that the reduced dimension d_G is usually restricted by the performance of the index, therefore it cannot be too large in practice. Furthermore, according to the definition, $Prob_j$ also depends on the underlying data characteristics, which is fixed once the data set is given and pivots selected. In this paper, we choose only one pivot in the entire data set. The problem of choosing multiple pivots in a data set that can potentially decrease Equation (8) is left as future work. Furthermore, the overall cost by weighting PP and CC can be also introduced to the case of LDR, which is omitted due to the space limit.

4.3 Range Queries with Radius Distribution

In the above cost models, we always assume that the radii r_q of range queries are fixed. However, there are circumstances where this is not the case. That is, users may issue range queries with different radii. In the sequel, we discuss incorporating the radius distribution of range queries into our cost model such that, given a set of queries with the radius distribution, we can still estimate the cost accurately in terms of the *pruning power* and *computation cost*. Note that, in practical applications, this radius distribution can be determined from a workload of queries [24]. Without loss

of generality, suppose the radius r_q of range query q follows the distribution of a random variable R, with the mean μ_R and variance σ_R.

We consider the cost model for *local dimensionality reduction*, LDR, taking into account the distribution of the query radius. Recall that, both PP_{LDR} and CC_{LDR} of LDR involve $Prob_i$. Thus, we only need to obtain a revised equation of $Prob_i$ which considers the query radius distribution. In particular, we rewrite $Prob_i$ as:

$$Prob_i = \Pr\{(dist^{(d_i)}(q,p_i) - dist^{(d_i)}(p_i,o_j) \geq -r_q) \cap (dist^{(d_i)}(q,p_i) - dist^{(d_i)}(p_i,o_j) \leq r_q)\}. \quad (9)$$

which is simplified as:

$$Prob_i = \Pr\{A \bigcup B\} \quad (10)$$

where $A = dist^{(d_i)}(q,p_i) - dist^{(d_i)}(p_i,o_j) + r_q \geq 0$ and $B = dist^{(d_i)}(q,p_i) - dist^{(d_i)}(p_i,o_j) - r_q \leq 0$.

Since it holds that $\Pr\{A \cup B\} = \Pr\{A\} + \Pr\{B\} - \Pr\{A \cap B\}$, we have $\Pr\{A \cap B\} = (\Pr\{A\} + \Pr\{B\} - \Pr\{A \cup B\})$. Moreover, the probability $\Pr\{A \cup B\}$ in our case is 1. Thus, we can obtain:

$$Prob_i = \Pr\{A \cap B\} = \Pr\{A\} + \Pr\{B\} - 1. \quad (11)$$

Substituting $Prob_i$ in the formula of the pruning power $PP_{LDR}(q, r_q, R)$ for LDR, we obtain:

$$PP_{LDR}(q, r_q, R) = |\mathcal{D}| - \sum_{i=1}^{m}(|\mathcal{P}_i| \cdot (\Pr\{A\} + \Pr\{B\} - 1)). \quad (12)$$

Similarly, the computation cost $CC_{LDR}(q, r_q, R)$ of LDR is given by:

$$CC_{LDR}(q, r_q, R) = \sum_{i=1}^{m}(d_i \cdot sign(|\mathcal{P}_i|) + d \cdot |\mathcal{P}_i| \cdot (\Pr\{A\} + \Pr\{B\} - 1)). \quad (13)$$

The remaining problem is to calculate $\Pr\{A\}$ and $\Pr\{B\}$. Similar to the assumption in Section 4.1, suppose that $dist^{(d_i)}(q,p_i)$ and $dist^{(d_i)}(p_i,o_j)$ follow two random variables $X_{\mathcal{P}_i}$ and $Y_{\mathcal{P}_i}$, respectively. In particular, variable $X_{\mathcal{P}_i}$ ($Y_{\mathcal{P}_i}$) has the mean $\mu_{X_{\mathcal{P}_i}}$ ($\mu_{Y_{\mathcal{P}_i}}$) and variance $\sigma_{X_{\mathcal{P}_i}}$ ($\sigma_{Y_{\mathcal{P}_i}}$). Based on the *Central Limit Theorem*, both $\frac{(dist^{(d_i)}(q,p_i) - dist^{(d_i)}(p_i,o_j) + r_q) - (\mu_{X_{\mathcal{P}_i}} - \mu_{Y_{\mathcal{P}_i}} + \mu_R)}{\sqrt{\sigma^2_{X_{\mathcal{P}_i}} + \sigma^2_{Y_{\mathcal{P}_i}} + \sigma^2_R}}$ and $\frac{(dist^{(d_i)}(q,p_i) - dist^{(d_i)}(p_i,o_j) - r_q) - (\mu_{X_{\mathcal{P}_i}} - \mu_{Y_{\mathcal{P}_i}} - \mu_R)}{\sqrt{\sigma^2_{X_{\mathcal{P}_i}} + \sigma^2_{Y_{\mathcal{P}_i}} + \sigma^2_R}}$ follow the *normal distribution* $\phi(z)$. Therefore, we obtain $\Pr\{A\}$ and $\Pr\{B\}$ as follows:

$$\Pr\{A\} = 1 - \phi(\frac{-(\mu_{X_{\mathcal{P}_i}} - \mu_{Y_{\mathcal{P}_i}} + \mu_R)}{\sqrt{\sigma^2_{X_{\mathcal{P}_i}} + \sigma^2_{Y_{\mathcal{P}_i}} + \sigma^2_R}}), \text{ and} \quad (14)$$

$$\Pr\{B\} = \phi(\frac{-(\mu_{X_{\mathcal{P}_i}} - \mu_{Y_{\mathcal{P}_i}} - \mu_R)}{\sqrt{\sigma^2_{X_{\mathcal{P}_i}} + \sigma^2_{Y_{\mathcal{P}_i}} + \sigma^2_R}}). \quad (15)$$

Substituting Equations (14) and (15) into Equations (12) and (13), we have

$$PP_{LDR}(q, r_q, R) = |\mathcal{D}| - \sum_{i=1}^{m}(|\mathcal{P}_i| \cdot (\phi(\frac{-(\mu_{X_{\mathcal{P}_i}} - \mu_{Y_{\mathcal{P}_i}} - \mu_R)}{\sqrt{\sigma^2_{X_{\mathcal{P}_i}} + \sigma^2_{Y_{\mathcal{P}_i}} + \sigma^2_R}}) - \phi(\frac{-(\mu_{X_{\mathcal{P}_i}} - \mu_{Y_{\mathcal{P}_i}} + \mu_R)}{\sqrt{\sigma^2_{X_{\mathcal{P}_i}} + \sigma^2_{Y_{\mathcal{P}_i}} + \sigma^2_R}}))), \text{ and} \quad (16)$$

$$CC_{LDR}(q, r_q, R) = \sum_{i=1}^{m}(d_i \cdot sign(|\mathcal{P}_i|) + d \cdot |\mathcal{P}_i| \cdot (\phi(\frac{-(\mu_{X_{\mathcal{P}_i}} - \mu_{Y_{\mathcal{P}_i}} - \mu_R)}{\sqrt{\sigma^2_{X_{\mathcal{P}_i}} + \sigma^2_{Y_{\mathcal{P}_i}} + \sigma^2_R}}) - \phi(\frac{-(\mu_{X_{\mathcal{P}_i}} - \mu_{Y_{\mathcal{P}_i}} + \mu_R)}{\sqrt{\sigma^2_{X_{\mathcal{P}_i}} + \sigma^2_{Y_{\mathcal{P}_i}} + \sigma^2_R}}))). \quad (17)$$

Statistics Extraction. Finally, we discuss some implementation details of our cost model. In particular, either the pruning power or the computation cost for range queries involves the distance distribution between two objects, that is, statistics of the data characteristics. More specifically, in order to estimate $Prob_i$, we need to obtain means and variances of the distance distributions $X_{\mathcal{P}_i}$ and $Y_{\mathcal{P}_i}$ for $dist^{(d_i)}(q,p_i)$ and $dist^{(d_i)}(p_i,o_j)$, respectively, where query object q can follow any *location distribution* Q, pivot p_i is an object selected in partition $\mathcal{P}_i$, and data object o_j is any object in $\mathcal{P}_i$. Let the mean and variance of $X_{\mathcal{P}_i}$ ($Y_{\mathcal{P}_i}$) be $\mu_{X_{\mathcal{P}_i}}$ and $\sigma_{X_{\mathcal{P}_i}}$ ($\mu_{Y_{\mathcal{P}_i}}$ and $\sigma_{Y_{\mathcal{P}_i}}$), respectively.

We first consider $dist^{(d_i)}(q,p_i)$ following the distribution $X_{\mathcal{P}_i}$. Note that, here the query point q follows the location distribution Q in the space, which is either known in advance or observed based on query history. Since our estimation is performed offline, we can compute the pairwise distances between the pivot p_i and a number of (either artificially generated or historical) query points following the distribution Q in the reduced d_i-dimensional space. Then, we average these pairwise distances to obtain the mean value $\mu_{X_{\mathcal{P}_i}}$, together with the (squared) variance $\sigma^2_{X_{\mathcal{P}_i}}$, the summation of squared errors between distances and the mean value.

Similarly, for distance $dist^{(d_i)}(p_i,o_j)$ with distribution $Y_{\mathcal{P}_i}$. For a selected pivot $p_i \in \mathcal{P}_i$, we obtain all the distances from p_i to any data object o_j in $\mathcal{P}_i$ considering d_i dimensions, average them and obtain the mean $\mu_{Y_{\mathcal{P}_i}}$ as well as their (squared) variance $\sigma^2_{Y_{\mathcal{P}_i}}$.

5 A Partitioning-Based LDR Approach

As mentioned in Section 1, previous approaches of GDR and LDR are based on heuristics, which cannot guarantee optimal partitions for search operations later on. In this section, we propose a novel partitioning-based LDR approach, PRANS (*Partitioning based on RANdomized Search*), in light of the proposed cost model. In particular, PRANS divides the data set into several partitions and within each partition, locally performs the dimensionality reduction. The resulting partitions can achieve high pruning power, as indicated by our cost model, and adapt to the inherent data characteristics.

5.1 PRANS

Given a data set $\mathcal{D}$, PRANS divides $\mathcal{D}$ into m partitions, each with its own reduced dimension d_i, such that the cost of range queries is as low as possible. The entire partitioning procedure is monitored by our cost model discussed in previous sections. Note that, although the cost

model involves two parts, the *pruning power* and *computation cost*, we focus on maximizing the *pruning power* of LDR for range queries. This assumption, however, does not conflict with our goal of minimizing the *computation cost*, since the *computation cost* is the summation of two terms, $\sum_{i=1}^{m}(d_i \cdot sign(|\mathcal{P}_i|))$ and $\sum_{i=1}^{m}(d \cdot |\mathcal{P}_i| \cdot Prob_i)$. Maximizing the *pruning power* exactly minimizes the latter term, while the former term is usually much less significant and, thus, can be ignored. Therefore, in the sequel, when we mention the total cost $TotalCost_{LDR}$ of a range query, we always mean the *pruning power*.

PRANS (*Partitioning based on RANdomized Search*) follows the framework of a popular clustering algorithm CLARANS (*Clustering Large Applications based on RANdomized Search*) [22]. In order to obtain effective partitions, the effectiveness evaluation of partitions in PRANS is based on the cost model in Section 4. In particular, given two parameters num_iter and $max\_neighbor$, procedure PRANS aims at finding "good" partitions and choosing their best reduced dimensions, such that the pruning power of range queries can be maximized. Specifically, the randomized search can be treated as a transition between vertices in a graph [22], and the parameter $max\_neighbor$ is the maximum number of transitions from a vertex during the search. In order to avoid the local optimum, the procedure runs for num_iter passes and finally selects the one with the highest pruning power as the best partition strategy. Note that, the value selection of num_iter and $max\_neighbor$ in PRANS follows the same settings as that in CLARANS [22].

As illustrated in Figure 2, for each pass (lines 2-13), PRANS first chooses m random objects p_1, p_2, ..., and p_m from data set $\mathcal{D}$, which form the initial representative set (line 4). Then, it incrementally obtains good partitions by swapping representative objects with non-representative ones (lines 6-11), so that higher pruning power can be achieved. In particular, each time a random object $p \in \mathcal{D}\backslash\{p_1, p_2, ..., p_m\}$ is selected and swapped with one random object $p_i \in \{p_1, p_2, ..., p_m\}$ (lines 6-7). The resulting representative set divides the space into m partitions, $\mathcal{P}_1, \mathcal{P}_2, ...,$ and $\mathcal{P}_m$, by assigning any object $o \in \mathcal{D}$ to partition $\mathcal{P}_i$ only if object o is closest to p_i other than p_j ($\neq p_i$) (line 8). Next (line 9), we estimate the total cost of queries on new partitions for all possible values of the reduced dimensions, and choose dimension d_i for each partition $\mathcal{P}_i$ that minimizes $TotalCost_{LDR}$. If $TotalCost_{LDR}$ is lower than the minimum cost $local\_min\_cost$ encountered so far (lines 10-11), we set $local\_min\_cost$ to $TotalCost_{LDR}$, and record the new partition set $\{\mathcal{P}_1, \mathcal{P}_2, ..., \mathcal{P}_m\}$ as the current best strategy $local\_\mathcal{P}$, together with their selected dimension set $local\_DIM$. The decreasing value of $local\_min\_cost$ each time bounds random searches to a maximum $max\_neighbor$. In other words, if the swapping process between a representative object and a non-representative one repeats for $max\_neighbor$ times, during which the costs of new partitions are always higher than $local\_min\_cost$, then the search terminates and the partitioning strategy associated with $local\_min\_cost$ is one of the local optima with the partition set $local\_\mathcal{P}$ and its corresponding dimension set $local\_DIM$.

As a second step, we invoke the procedure MergePartitions which tries to merge any two partitions into one such that the resulting new partition has a lower cost than the summed cost of the original two partitions and moreover the cost difference is the most significant (line 12). The procedure terminates when no partitions can be merged. Thus, parameters $local\_\mathcal{P}$, $local\_DIM$, $local\_min\_cost$, and m are updated correspondingly. Let $\mathcal{P}$ and DIM correspond to the partition and dimension sets, respectively, with the lowest cost $local\_min\_cost$ among all the local optima (lines 13-14). Finally, we output $\mathcal{P}$ and DIM as the result.

```
Procedure PRANS {
  Input: parameters num_iter and max_neighbor
  Output: m partitions P_1, P_2, . . ., and P_m
          as well as their selected best dimensions d_1, d_2, . . ., and d_m, respectively
  (1)  global_min_cost = +∞; P = ∅; DIM = ∅; // initialization
  (2)  for i = 1 to num_iter // multiple iterations to avoid local optimum
  (3)    local_min_cost = +∞; local_P = ∅; local_DIM = ∅
  (4)    randomly select m objects p_i ∈ D where i ∈ [1, m]
  (5)    for j = 1 to max_neighbor
  (6)      randomly select an object p ∈ (D − {p_1, p_2, ..., p_m})
  (7)      replace a random object p_i ∈ {p_1, p_2, ..., p_m} with p
  (8)      assign any object o ∈ D that has object p_i as it NN to partition P_i
  (9)      obtain the minimum total cost TotalCost_LDR for partitions {P_1,
             P_2, ..., P_m} as well as their corresponding {d_1, d_2, ..., d_m}, for all
             possible dimensions in each partition // based on the cost model
  (10)     if TotalCost_LDR < local_min_cost
  (11)       set local_min_cost = TotalCost_LDR, local_P = {P_1,
               P_2, ..., P_m}, local_DIM = {d_1, d_2, ..., d_m} and j = 1
  (12)   MergePartitions(local_P, local_DIM, local_min_cost, m);
  (13)   if global_min_cost > local_min_cost
  (14)     global_min_cost = local_min_cost; P = local_P;
             DIM = local_DIM;
  (15) return P and DIM.
}
```

Figure 2. Partitioning Based on Randomized Search

Selection of m. The number of partitions m highly depends on data characteristics. Specifically, we want to obtain partitions that have the lowest cost. One natural way is to check all possible cases by invoking procedure PRANS with different parameters $m \in [1, |\mathcal{D}|]$. However, this approach is not efficient in terms of the computation cost. To avoid invoking procedure PRANS several times, we can choose a sufficiently large value of m. In line 12 of procedure PRANS, we can combine any two partitions if the estimated pruning power can be decreased after the merge. Therefore, the value of m can be adjusted automatically according to data characteristics.

Since the estimation accuracy of each partition is related to statistics extracted from data, we require that the size of each partition be large enough to obtain useful statistics. Assuming this required minimum size of partitions is

$Card$, we set the initial value of m to $|\mathcal{D}|/Card$ and only need to invoke procedure PRANS once, during which the $|\mathcal{D}|/Card$ partitions are further merged based on our cost model. In this way, we can obtain m ($\leq |\mathcal{D}|/Card$) partitions with the lowest cost among all random searches.

5.2 Discussion on Partitioning Cost

Similar to the CLARANS [22], we view each possible representative set containing distinct m representatives as a vertex g_i in a graph G. So in total there are $H = \binom{N}{m}$ vertices in G. Any edge connects two vertices in G only if the representative sets denoted by the two vertices differ for only one object. Thus, each vertex has $m(N-m)$ neighbors. Let L be the $max\_neighbor$ in procedure PRANS ($L \leq m(N-m)$). Intuitively, the swapping between a representative object and a non-representative one is a traverse from one vertex to one of its neighbors. Assume each vertex g_i is associated with a total cost tc_i, whose computation cost is considered as a unit $O(1)$. Without loss of generality, suppose $tc_i \leq tc_j$ for $i < j$.

First, given the current vertex g_i, we consider the expected times to swap a representative and a non-representative one before traversing to one of its neighbors. We assume that neighbors of g_i are uniformly extracted from the set $\{g_1, ..., g_{i-1}, g_{i+1}, ..., g_H\}$. That is, the probability that a randomly selected neighbor g_j of g_i has lower cost than g_i is $\frac{i-1}{H-1}$. Therefore, the expected number S_i of neighbors for g_i to check before it traverses to another vertex is:

$$\begin{aligned} S_i &= \sum_{j=1}^{L} (j \Pr\{\text{the number of neighbors checked} = j\}) \\ &\quad + L\,(\Pr\{\text{all neighbors have higher cost than } g_i\}) \\ &= \sum_{j=1}^{L} \left(j \left(\frac{H-i}{H-1}\right)^{j-1} \left(\frac{i-1}{H-1}\right)\right) + L\left(\frac{H-i}{H-1}\right)^{L} \\ &= \left(\frac{H-1}{i-1}\right) \left(1 - \left(\frac{H-i}{H-1}\right)^{L}\right). \end{aligned} \quad (18)$$

where $S_1 = L(\frac{H-i}{H-1})^L$.

Therefore, the expected number $E(S_i)$ of S_i for any randomly selected vertex g_i in G is:

$$\begin{aligned} E(S_i) &= \frac{1}{H}\sum_{i=1}^{H} S_i \\ &= \sum_{i=2}^{H} \frac{1}{i-1} - \frac{1}{i-1}\left(\frac{H-i}{H-1}\right)^{L} + \frac{L}{H}\left(\frac{H-i}{H-1}\right)^{L} \end{aligned} \quad (19)$$

As a second step, we consider the average length T_i of the traversal path starting from vertex g_i. Without loss of generality, assume $g_{i_1}, g_{i_2}, ..., g_{i_j}$ is the maximum traversal path, where $i = i_1 > i_2 > ... > i_j$ and thus $tc_i = tc_{i_1} > tc_{i_2} > ... > tc_{i_j}$. Hence, the probability $F(j)$ that the length of the traversal path is no greater than j is given by:

$$F(j) = 1 - \prod_{l=1}^{j} \left(1 - \left(\frac{H-i_l}{H-1}\right)^{L}\right). \quad (20)$$

Let $G(i_l) = 1 - (\frac{H-i_l}{H-1})^L$, where all $G(i_l)$ are independently generated random number from the same $G(i)$.

Then, the probability $f(j)$ that the path length is exactly j is:

$$f(j) = F(j) - F(j-1). \quad (21)$$

Therefore, the expected path length T_i starting from vertex g_i is denoted as:

$$T_i = \sum_{j=1}^{\infty} jf(j) = \sum_{j=1}^{\infty} (jF(j) - jF(j-1)). \quad (22)$$

Considering the expectation $E(T_i)$ of T_i for all possible i, Equation (22) can be simplified and rewritten by substituting Equation (20) as follows:

$$E(T_i) = \frac{2}{E(G(i))^{2L}} - \frac{3}{E(G(i))^{L}}, \quad (23)$$

where $E(G(i))$ is given by

$$E(G(i)) = \frac{1}{H}\sum_{i=1}^{H} \left(1 - \left(\frac{H-i}{H-1}\right)^{L}\right). \quad (24)$$

In summary, the computation cost of PRANS can be approximately measured by $E(S_i) \cdot E(T_i)$, which is the average number of neighbors checked at each vertex times the expected number of the traversal path length.

6 Indexing PRANS Partitions for Range Queries

Up to now, we have obtained m partitions $\mathcal{P}_1, \mathcal{P}_2, ...,$ and $\mathcal{P}_m$, as well as their reduced dimensions $d_1, d_2, ...,$ and d_m, respectively, which can achieve high pruning power for range queries based on the cost model. Without loss of generality, assume that $|\mathcal{P}_i| \neq 0$ and $1 \leq d_i \leq d_j \leq d$ for all $i, j \in [1, m]$ where $i \neq j$. All partitions apply the same type of the dimensionality reduction technique. One natural way to index these m partitions is to construct m separate trees for each of them [7]. However, the resulting partitions might have skewed sizes, so trees with large data sizes are of great height, leading to high I/O cost. Furthermore, for m trees, we have to descend from the root to leaf for at most m times before accessing data. Therefore, it is not an efficient way to construct m separate trees. Instead, for scalability, we index all data objects in a single B$^+$-tree similar to the framework of $iDistance$ [13, 14].

In particular, for each partition $\mathcal{P}_i$, we pre-compute the distance $dist^{(d_i)}(p_i, o_j)$ between the selected pivot p_i and any data object o_j in $\mathcal{P}_i$, along the reduced d_i dimensions. Then, each data object o_j in the partition $\mathcal{P}_i$ is hashed to a 1-dimensional key $key(o_j)$. That is,

$$key(o_j) = i \cdot MAX + dist^{(d_i)}(p_i, o_j) \quad (25)$$

where $p_i \in \mathcal{P}_i$ is the pivot determined by procedure PRANS, o_j is any data object in $\mathcal{P}_i$, and *MAX* is a large positive number that can guarantee key ranges of different partitions do not overlap with each other. Without loss of generality, if the domain of the data space is [0, 1]^d, we can set MAX to $\sqrt{d}$. Therefore, the key range of the i-th partition is from $i \cdot \sqrt{d}$ to $(i+1) \cdot \sqrt{d}$. As a second step, we insert each data object $o_j \in \mathcal{D}$ into a B$^+$-tree index, with its hashed key $key(o_j)$. Thus, the range query in the

high dimensional space is reduced to the existing problem of searching a B$^+$-tree with several key ranges. Since the B$^+$-tree has higher fanout, compared to high dimensional indexes [10, 8] , it is space-efficient and can incur lower I/O cost during queries.

Given a range query q with a radius r_q, for partition $\mathcal{P}_i$, we first reduce the dimensionality of q to d_i, and compute the distance $dist^{d_i}(q, p_i)$ between q and pivot $p_i \in \mathcal{P}_i$, using d_i reduced dimensions. According to triangle inequality, for any data object $o_j \in \mathcal{P}_i$, as long as it holds that $|dist^{(d_i)}(q, p_i) - dist^{(d_i)}(p_i, o_j)| > r_q$, the data object o_j can be safely pruned. In other words, we only need to search those data objects o_j in the B$^+$-tree with keys ranging from $(i \cdot MAX + dist^{(d_i)}(q, p_i) - r_q)$ to $(i \cdot MAX + dist^{(d_i)}(q, p_i) + r_q)$. After that, the retrieved candidates are further refined by calculating their real *Euclidean distance* from the query in the d-dimensional space.

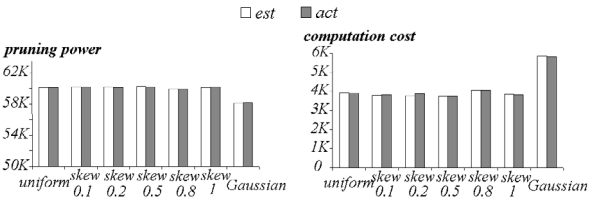
7 Experimental Evaluation

In this section, we first verify our cost model for both LDR and GDR, and then demonstrate the effectiveness and efficiency of the proposed LDR approach, with extensive experiments. Specifically, in the first set of experiments, we aim to confirm the correctness of our cost model by comparing the estimated *pruning power* and *computation cost* of range queries with the actual values in LDR and GDR. We use both real and synthetic data sets during the experiments. In particular, the real data set *mixed* is extracted from 11609 *Corel* images in 113 categories [12], which are widely used in the area of the content-based image retrieval. It contains 64-dimensional features, including 44 correlogram, 6 color moment, and 14 texture moment. For the synthetic data sets, we generate three common types, *uniform*, *Zipf* and *Gaussian*. That is, each dimension of data objects is generated by picking up a random value within [0, 1] that follows the *uniform*, *Zipf* or *Gaussian* distribution. For the *Zipf* distribution, we produce 5 data sets, *Zipf*0.1, *Zipf*0.2, *Zipf*0.5, *Zipf*0.8 and *Zipf*1, with the skewness 0.1, 0.2, 0.5, 0.8 and 1, respectively, whereas for *Gaussian* data set with the mean 0.5 and variance 0.2. In the second set of experiments, we evaluate the pruning power and I/O cost of the proposed LDR method, PRANS, compared to the state-of-art LDR technique, MMDR [14]. The tested data sets include the real data set *mixed* and the correlated synthetic data set, *CD* [14].

In the sequel, we only present the experimental result applying the *Singular Value Decomposition* (SVD) [20]. Note however, that our cost model is applicable to arbitrary dimensionality reduction techniques. Moreover, we limit the maximum reduced dimension of either LDR or GDR to 16.

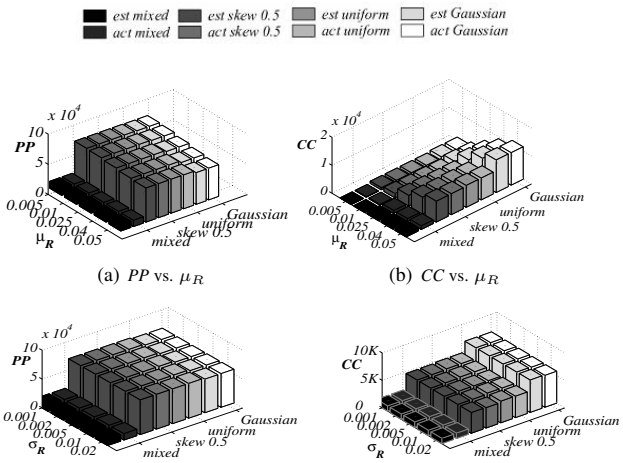
7.1 Verification of Cost Model

First, we verify the correctness of our cost model for LDR and GDR on synthetic data sets. Specifically, for each



(a) *PP* vs. data sets (b) *CC* vs. data sets

Figure 3. LDR Cost Model on Synthetic Data Sets



(a) *PP* vs. μ_R (b) *CC* vs. μ_R

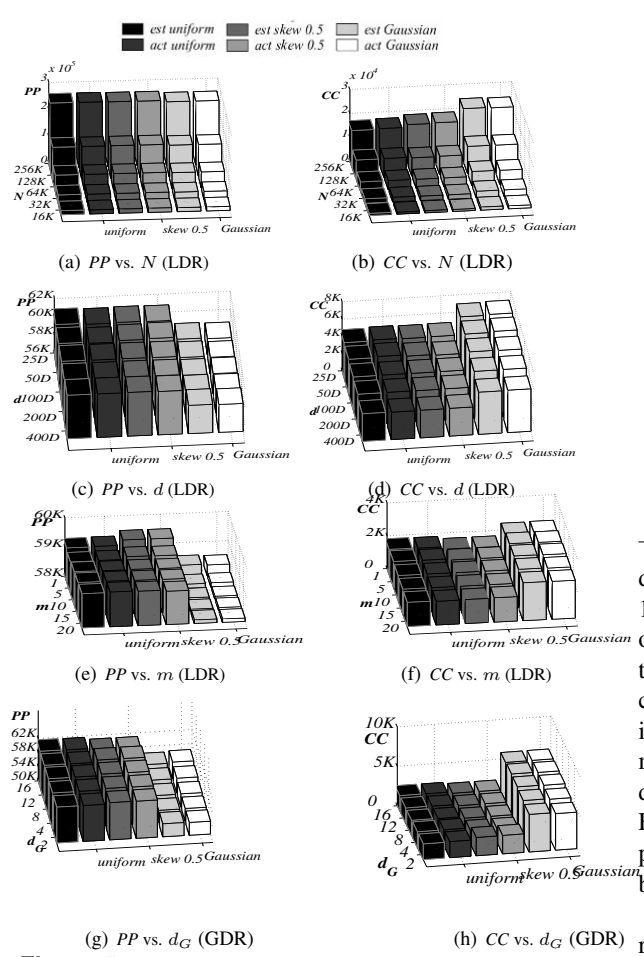
(c) *PP* vs. σ_R (d) *CC* vs. σ_R

Figure 4. Effects of the Query Radius

data set, we divide it into m partitions. Then, for data objects in each partition $\mathcal{P}_i$, we reduce their dimensionality d to either d_i locally (LDR) or d_G globally (GDR). Within each partition, we randomly choose one data object as the pivot. In order to evaluate the estimation of the model, we generate 2000 range queries, half of which are used for statistics extraction and the other half for the performance test. In particular, we randomly pick up data objects in the data set as query centers and produce radii with the mean μ_R and variance σ_R. When a range query q arrives, we consider two measures, the *pruning power* and *computation cost*, during the similarity search. Meanwhile, we also compare them with values estimated by the cost model.

Figure 3 illustrates the pruning power and computation cost of LDR with synthetic data sets, *uniform*, *Zipf0.1*, *Zipf0.2*, *Zipf0.5*, *Zipf0.8*, *Zipf1*, and *Gaussian*, comparing the estimated with actual ones, where the total data size N = 64K, the dimensionality of data objects d = 100, the number of partitions m = 10, $\mu_R = 0.025$ and $\sigma_R = 0.005$. For all data sets, our cost model can closely estimate the real values of both the pruning power and computation cost.

Figure 4 illustrates the effect of the query radius on cost model evaluation. Specifically, we compare the estimated with the actual *pruning power* and *computation cost* on data sets *mixed*, *uniform*, *skew0.5* and *Gaussian*. In particular, Figures 4(a) and 4(b) fix the radius variance σ_R of range queries to 0.005 and test the radius mean μ_R with values



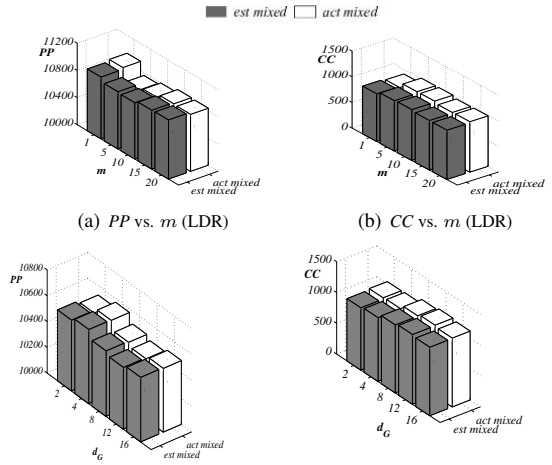
(a) *PP* vs. N (LDR) (b) *CC* vs. N (LDR)

(c) *PP* vs. d (LDR) (d) *CC* vs. d (LDR)

(e) *PP* vs. m (LDR) (f) *CC* vs. m (LDR)

(g) *PP* vs. d_G (GDR) (h) *CC* vs. d_G (GDR)

Figure 5. Cost Model Evaluation on Synthetic Data Sets



(a) *PP* vs. m (LDR) (b) *CC* vs. m (LDR)

(c) *PP* vs. d_G (GDR) (d) *CC* vs. d_G (GDR)

Figure 6. Cost Model Evaluation on the Real Data Set

0.005, 0.01, 0.025, 0.04 and 0.05, whereas Figures 4(c) and 4(d) set the radius mean to 0.025 varying the radius variance $\sigma_R = 0.001, 0.002, 0.005, 0.01, 0.02$. When the mean of the query radii increases, the number of pruned objects (i.e. pruning power) decreases and the computation cost increases, since more candidates are included. When the radius variance changes, the estimation from the cost model is always close to the real values. In summary, the estimated values based on our cost model can always mimic the actual ones closely under different μ_R. Therefore, in the subsequent experiments, we fix parameters $\mu_R = 0.025$ and $\sigma_R = 0.005$.

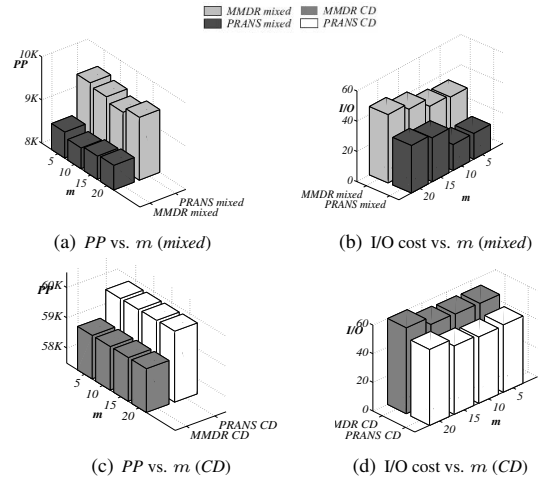
Next, we study the effectiveness of our cost model by varying different parameters N, d and m, over three synthetic data sets, *uniform*, *Zipf0.5*, and *Gaussian*. In particular, Figures 5(a) and 5(b) demonstrate the experimental result of LDR with different data size N = 16K, 32K, 64K, 128K and 256K, where d = 100 and m = 10. Figures 5(c) and 5(d) vary the dimensionality d of the data from 25 to 400, where N = 64K and m = 10. Similarly, Figures 5(e) and 5(f) present the result of LDR with the number of partition m = 1, 5, 10, 15 and 20, where N = 64K and d = 100. Furthermore, Figures 5(g) and 5(h) also study the effectiveness of our cost model using GDR method with the reduced dimension d_G = 2, 4, 8, 12, and 16, where N = 64K, d = 100, and m = 10. Note that, since we randomly select pivots in the data set, the resulting pruning power or computation cost in figures may not be the optimal ones. Therefore, curves in these figures cannot show the real trend by varying parameters. However, this does not matter, since our major concern is to verify whether our cost model can indeed estimate the real *pruning power* and *computation cost*. From the complete set of experiments, we can see that the proposed model can mimic the actual values very well, for both LDR and GDR.

Figure 6 illustrates the same set of experiments on the real data set *mixed*, where similar results to the synthetic data sets are obtained. Specifically, Figures 6(a) and 6(b) test the effect of the number m of partitions in LDR, whereas Figures 6(c) and 6(d) that of the reduced dimension d_G in GDR. In summary, our cost model can correctly estimate the actual cost on both real and synthetic data sets.

7.2 Query Performance of PRANS vs. MMDR

As a second step, we evaluate the pruning power and I/O cost by comparing two LDR approaches, MMDR and PRANS. Specifically, MMDR is a state-of-art LDR approach, which detects clusters of ellipse shapes with the help of *Mahalanobis distance*, whereas PRANS divides the data set into several partitions according to the cost model estimation, which can achieve high pruning power. Both MMDR and PRANS hash each data object to a unique key, and then index it in a B$^+$-tree using the $iDistance$ method.

We do our experiments on the real data set *mixed*, as well as the synthetic data set, *CD*. In particular, *CD* is a local correlated data set used in [14], which contains 64K 100-dimensional data objects. Figure 7 illustrates the pruning power and I/O cost on the MMDR and PRANS partitions, the number m of which is set to 5, 10, 15 and 20. Specifically, Figures 7(a) and 7(b) show the result with the real data set *mixed*, whereas Figures 7(c) and 7(d) demonstrate that



(a) *PP* vs. m (*mixed*) (b) I/O cost vs. m (*mixed*)

(c) *PP* vs. m (*CD*) (d) I/O cost vs. m (*CD*)

Figure 7. MMDR vs. PRANS on Real and Synthetic Data Sets

for the synthetic data set *CD*. For all the figures, PRANS outperforms MMDR in terms of the pruning power and I/O cost, since our PRANS is based on the cost model, such that the cost is close to optimal. In summary, by indexing partitions after the procedure PRANS, it can achieve efficient query performance, compared to MMDR.

8 Conclusions

Similarity search, for example, range queries, has a serious problem with query performance degradation in high dimensional space. Previous work to solve this problem include the *global* and *local* dimensionality reduction techniques (i.e. GDR and LDR), which reduce the dimensionality of data objects to one *fixed* lower value and various different lower ones, respectively. However, the proposed approaches are *heuristic-based*, requiring the specification of some data-dependent parameters. In particular, it is necessary to set the reduced dimension for GDR and some thresholds in LDR. Motivated by this, in this paper, we propose a formal cost model to evaluate the effectiveness and efficiency of both GDR and LDR for range queries. Furthermore, we present a novel partitioning-based LDR approach, PRANS, which is based on our cost model and can achieve good query performance in terms of the pruning power. Finally, the resulting data objects after the dimensionality reduction are indexed in a B^+-tree following the $iDistance$ framework for efficient similarity search. Extensive experiments have verified the correctness of our cost model and indicated that, compared to the existing LDR method, MMDR, PRANS can result in partitions with low query cost.

Acknowledgement

Funding for this work was provided by the Hong Kong RGC grants DAG05/06.EG03 and National Grand Fundamental Research 973 Program of China under Grant No.2006CB303000.

References

[1] R. Agrawal et al. Efficient similarity search in sequence databases. *FODO*, 1993.
[2] S. Berchtold et al. The pyramid-technique: Towards breaking the curse of dimensionality. *SIGMOD*, 1998.
[3] S. Berchtold et al. The X-tree: An index structure for high-dimensional data. *VLDB*, 1996.
[4] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. When is "nearest neighbor" meaningful? *Lecture Notes in Computer Science*, 1999.
[5] Y. Cai and R. Ng. Indexing spatio-temporal trajectoires with Chebyshev polynomials. *SIGMOD*, 2004.
[6] K. Chakrabarti and S. Mehrotra. The hybrid tree: An index structure for high dimensional feature spaces. *ICDE*, 1999.
[7] K. Chakrabarti and S. Mehrotra. Local dimensionality reduction: A new approach to indexing high dimensional spaces. *VLDB*, 2000.
[8] P. Ciaccia et al. M-tree: An efficient access method for similarity search in metric spaces. *VLDB*, 1997.
[9] C. Faloutsos et al. Fast subsequence matching in time-series databases. *SIGMOD*, 1994.
[10] A. Guttman. R-trees: a dynamic index structure for spatial searching. *SIGMOD*, 1984.
[11] X. He et al. Tensor Subspace analysis. *NIPS*, 2006.
[12] J. Huang et al. Image indexing using color correlograms. *CVPR*, 1997.
[13] H. V. Jagadish et al. iDistance: An adaptive B^+-tree based indexing method for nearest neighbor search. *TODS*, 30(2) 2005.
[14] H. Jin et al. An adaptive and efficient dimensionality reduction algorithm for high-dimensional indexing. *ICDE*, 2003.
[15] I. Kamel and C. Faloutsos. Hilbert R-tree: An improved R-tree using Fractals. *VLDB*, 1994.
[16] K. V. Ravi Kanth et al. Dimensionality reduction for similarity searching in dynamic databases. *SIGMOD*, 1998.
[17] N. Katayama and S. Satoh. The SR-tree: an index structure for high-dimensional nearest neighbor queries. *SIGMOD*, 1997.
[18] E. Keogh et al. Locally adaptive dimensionality reduction for indexing large time series databases. *SIGMOD*, 2001.
[19] R. Kohavi and D. Sommerfield. Feature subset selection using the wrapper method: Overfitting and dynamic search space topology. *SIGKDD*, 1995.
[20] F. Korn et al. Efficiently supporting ad hoc queries in large datasets of time sequences. *SIGMOD*, 1997.
[21] King-Ip Lin et al. The TV-tree: An index structure for high-dimensional data. *VLDBJ*, 1994.
[22] R. T. Ng and J. Han. Efficient and effective clustering methods for spatial data mining. *VLDB*, 1994.
[23] I. Popivanov and R. J. Miller. Similarity search over time series data using wavelets. *ICDE*, 2001.
[24] S. Rasetic et al. A trajectory splitting model for efficient spatio-temporal indexing. *VLDB*, 2005.
[25] T. Seidl and H. Kriegel. Optimal multi-step k-nearest neighbor search. *SIGMOD*, 1998.
[26] R. Weber et al. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. *VLDB*, 1998.
[27] E. W. Weisstein. Central Limit Theorem. *http://mathworld.wolfram.com/CentralLimitTheorem.html*.
[28] B-K Yi and C. Faloutsos. Fast time sequence indexing for arbitrary L_p norms. *VLDB*, 2000.
[29] C. Yu et al. Progressive KNN search using B^+-trees. 2001.