Similarity Search in Arbitrary Subspaces Under L_p-Norm

Xiang Lian and Lei Chen

Department of Computer Science and Engineering
Hong Kong University of Science and Technology
Clear Water Bay, Kowloon
Hong Kong, China
{xlian, leichen}@cse.ust.hk

***Abstract*— *Similarity search* has been widely used in many applications such as information retrieval, image data analysis, and time-series matching. Specifically, a *similarity query* retrieves all data objects in a data set that are similar to a given query object. Previous work on *similarity search* usually consider the search problem in the *full* space. In this paper, however, we propose a novel problem, *subspace similarity search*, which finds all data objects that match with a query object in the *subspace* instead of the original *full* space. In particular, the query object can specify *arbitrary* subspace with *arbitrary* number of dimensions. Since traditional approaches for *similarity search* cannot be applied to solve the proposed problem, we introduce an efficient and effective pruning technique, which assigns scores to data objects with respect to pivots and prunes candidates via scores. We propose an effective *multipivot-based* method to pre-process data objects by selecting appropriate pivots, where the *entire* procedure is guided by a formal cost model, such that the *pruning power* is maximized. Finally, scores of each data object are organized in sorted list to facilitate an efficient *subspace similarity search*. Extensive experiments have verified the correctness of our cost model and demonstrated the efficiency and effectiveness of our proposed approach for the *subspace similarity search*.**

I. Introduction

Similarity search has been used in a wide spectrum of applications such as information retrieval [18], image data analysis [15], time-series matching [2], [8], and the like. In particular, a *similarity query* retrieves all the data objects in the database that are similar to a query object. Formally, given a database $\mathcal{D}$ containing n-dimensional data objects and an n-dimensional query object q, a *similarity query* finds all data objects $o \in \mathcal{D}$, such that $dist(q, o) \leq \varepsilon$, where $dist(x, y)$ is a distance function to measure the similarity between objects x and y, and ε the user-specified similarity threshold. As a concrete example, assume that we have a very large image database, in which each image contains n features (dimensions), such as RGB histograms, texture, shape, local descriptors or *interest points*. In order to retrieve images of particular interest, we can specify an n-dimensional feature vector (possibly from a sample image) as query object (vector) and issue a *similarity query* to find all the similar images whose feature vectors are within ε distance from the query vector, where ε is a given similarity threshold.

In reality, however, since the sample image may not be always available or the number of features, n, is too large (e.g., a hundred), users have to post a query point with only a few features as well as a query radius, and then issue a range query. As in the content-based image retrieval (CBIR) [20], [14], indexes are constructed separately on different subsets of features that users are interested in. Users would issue range queries over one of indexes that contains their preferred features. It is natural that two different users can have different preferences to query features [10], and moreover they can specify different numbers of features [24], [17], [20]. These make the *similarity search* problem much more complex than the traditional one where a *full* n-dimensional match is performed between query and data objects. In this case, what is involved is a match of a k-dimensional query object to n-dimensional data objects, where $k < n$ and k may be different for different queries.

Apart from image databases, this problem is also very common in business analysis which processes data based on ad-hoc subspace in on-line analytical processing (OLAP) applications [1]. In particular, a data analyst could explicitly choose an ad-hoc subspace that he/she thinks important to search. Therefore, two data analysts may be interested in different subspaces for their own purposes, which needs the support of query processing on arbitrary subspaces. Note that, the selection of good subset of features (or subspaces) for a specific application is known as *feature selection* problem [13], which is not the focus of this paper. We are interested in the following question. Given a subset of features *on the fly*, how to conduct an efficient search over very large databases. There are two aspects of this problem. One is that query features are not known beforehand. Second, the search is based on *subspace similarity search* defined below.

Definition 1.1: (Subspace Similarity Search) Given a data set $\mathcal{D}$ containing data objects in n-dimensional space DIM and a k-dimensional query object $q^{(k)}$ in a space DIM', where $DIM' \subseteq DIM$ and $k \ll n$, let $o^{(k)}$ be a k-dimensional point (vector) obtained from the projection of data object $o \in \mathcal{D}$ on DIM'. A *subspace similarity query* retrieves all objects $o \in \mathcal{D}$, such that $dist(q^{(k)}, o^{(k)}) \leq \varepsilon$, where $dist(\cdot, \cdot)$ is a distance function and ε a user-specified similarity threshold.

According to Definition 1.1, the search subspace DIM' is specified by query object $q^{(k)}$. Since different queries may

specify different query objects, subspaces DIM' in *subspace similarity queries* may also be different. That is, subspace DIM' can be the *arbitrary* combination of k out of n dimensions in the *full* space DIM, where we assume that $k \in [k_{min}, k_{max}]$ $(1 \leq k_{min} \leq k_{max} \ll n)$. Thus, there are totally $\sum_{k=k_{min}}^{k_{max}} \binom{n}{k}$ possible subspaces for queries. In a special case where $k_{min} = k_{max}$, there are $\binom{n}{k}$ possible subspaces, which is very large for a large n.

Surprisingly, in spite of many related work on *similarity search* in the *full* dimensional space [2], [8], to the best of our knowledge, only a few previous work has studied the search efficiency problem of the *subspace similarity search*, where the *similarity search* can be performed in *arbitrary* subspaces specified by users. Yiu and Mamoulis [31] studied the *nearest neighbor* (NN) and *reverse nearest neighbor* (RNN) search in ad-hoc subspaces with help of sorted lists. In contrast, our work focuses on the *range query* in *arbitrary* subspaces.

One straightforward way to solve the *subspace similarity search* problem is to build $\sum_{k=k_{min}}^{k_{max}} \binom{n}{k}$ multidimensional indexes on data objects for *all* possible subspaces separately, and then perform the *similarity search* on one of them, whose dimensions correspond to the subspace specified by the query object. However, this approach is not feasible due to the large number of possible subspaces, which results in far more space consumption than necessary. As an example, when $n = 20$ and $k = 3$, there are more than 1,000 ($\binom{20}{3} = \frac{20 \times 19 \times 18}{3 \times 2 \times 1}$) possible subspaces, and it is unacceptable to build a thousand indexes for the *same* data set.

Another possible solution is to build a *single* n-dimensional index, such as R-tree [12], on data objects in the *full* space DIM, and perform a *similarity search* on the index in subspace DIM' by ignoring other dimensions $(DIM - DIM')$. However, when the dimensionality of data objects is high (e.g. more than 20), the n-dimensional index would encounter the "dimensionality curse" problem, having even worse query performance than a *linear scan*. Previous dimensionality reduction techniques [19], [2], [22] are useless in this case, since they are only used for the *full* space *similarity search*, rather than *subspace*.

The third possible approach is to insert every dimension of data objects into a 1-dimensional index, such as B^+-tree. Given a k-dimensional query object $q^{(k)}$ in DIM' and a similarity threshold ε, we issue k *range queries* for k dimensions of DIM', respectively, union candidate sets obtained from query results, and finally refine them by checking their real distances to $q^{(k)}$. This method incurs high space cost, since we have to store n copies of object id in the index, where n is the dimensionality of the *full* space. Furthermore, too many *false positives* (candidates that do not belong to the actual answer set) are introduced, and it is not efficient either to refine candidates. As a consequence, traditional methods for *similarity search* cannot be directly applied to the *subspace similarity search*.

In this paper, we present a novel approach, that can successfully solve the *subspace similarity search* problem for *arbitrary* subspaces. In particular, we select several data objects in $\mathcal{D}$ as *pivots*, assign 1-dimensional scores to each data object with respect to the chosen *pivots*, and finally use these scores to efficiently prune candidates. Since the *entire* search procedure including the pivot selection is based on our proposed formal cost model, which aims at maximizing the *pruning power* and reducing the *computation cost*, our method is efficient and effective to answer the *subspace similarity query*.

We make the following contributions:

1) We present an efficient approach to perform the *subspace similarity search*, which selects *pivots* in the data set, assigns 1-dimensional scores to each data object with respect to *pivots*, and finally utilizes scores to efficiently prune candidates.
2) We provide a formal cost model for our proposed method, namely *multipivot-based*, in light of which *pivots* are chosen such that the *pruning power* and *computation cost* of the *subspace similarity search* can be maximized and minimized, respectively.
3) Last but not least, we demonstrate through extensive experiments that our approach can efficiently and effectively give the answer to *subspace similarity query*.

In the sequel, Section II briefly reviews the related work on *similarity search*. Finally, Section VII concludes this paper.

II. Related Work

Similarity search has been intensively studied ever since the early 1990s along with the research on the content-based image retrieval [26]. Much of the literature focuses on this topic due to its wide application to different media, whose unique characteristics may introduce new challenges to users or applications.

Previous work on *similarity search* usually consider the search problem in the *full* space, in the sense that the search space of the query object is the same as that of data objects. In general, there are two important issues involved. The first issue is the design of "good" distance functions that meet the requirement of different applications. In this paper, we focus on the search problem under L_p-norm for $1 \leq p \leq \infty$, and leave the discussion of using other interesting measures as our future work. In particular, L_p-norm has been widely used in many applications [29], for example, L_1-norm is robust against impulse noise [29], L_2-norm is often used for *similarity search* [5], [7] and *spatial queries* [12], and L_∞-norm can be applied to *atomic matching* [29].

The second issue of the *similarity search* problem is the development of efficient retrieval methods. In order to facilitate a fast *similarity search*, previous work usually construct multidimensional indexes, such as R-tree [12], for the data set, on which either *range* or *nearest neighbor query* is issued. The query performance of the index, however, degrades dramatically, that is, even worse than a *linear scan*, when the dimensionality of objects becomes high, known as the "curse of dimensionality" problem. Therefore, various dimensionality reduction techniques are proposed to reduce the dimensionality of data objects before indexing them. These techniques include

SVD [19], DFT [2], DWT [22], and so on. Furthermore, if the underlying distance function is a *metric* measure, we can utilize the property of the *triangle inequality* in the metric space to prune *false positives*. For details of the *similarity search* in high dimensional and metric spaces [30], [4], please refer to two nice surveys [3] and [6], respectively.

In contrast, *subspace similarity search* also has many important applications. As an example, in sub-image retrieval [17], [23], only a subset of local features (dimensions) are used to search for images. To the best of our knowledge, only a few previous work has studied the search problem in the *subspace*, where not all dimensions are specified by the query object. Yiu and Mamoulis [31] searched for *nearest neighbor* (NN) and *reverse nearest neighbor* (RNN) in ad-hoc subspaces. The most relevant one to ours is to retrieve NN, while our study focuses on the *subspace range query*. Specifically, they construct a sorted list for each dimension of the data set for sequential scan and compute the minimum possible distance from query point to each data (by underestimating distances along those dimensions that have not been seen yet). If a data object has been seen in all dimensions and its real distance to query point is smaller than the underestimated distances of all other objects, then this object is NN of the query point.

Finally, there are some work that either propose partial distance measures in order to closely mimic humans' recognition of similarity [10], [27] or utilize the partial distance to facilitate efficient search [21]. However, the focus of their work is still on the search in the *full* space.

III. Problem Definition

In this section, we formally define our problem of the *subspace similarity search* in detail. Specifically, as described in Definition 1.1, given a database $\mathcal{D}$ containing N data objects in the n-dimensional *full* space DIM and a query object $q^{(k)}$ in a k-dimensional subspace DIM' of DIM (i.e. $DIM' \subseteq DIM$, $k \in [k_{min}, k_{max}]$, and $k \ll n$), a *subspace similarity query* retrieves all data objects $o \in \mathcal{D}$ such that $dist(q^{(k)}, o^{(k)}) \leq \varepsilon$, where $o^{(k)}$ is a k-dimensional point obtained from the projection of data object o on subspace DIM' and $dist(\cdot, \cdot)$ is L_p-norm ($p \in [1, \infty]$) distance function. In particular, given any two n-dimensional data objects x and y, the L_p-norm distance function $L_p(x, y)$ is defined as:

$$L_p(x, y) = \sqrt[p]{\sum_{i=1}^{n} |x[i] - y[i]|^p}, \quad (1)$$

where $1 \leq p < \infty$. Furthermore, when $p = \infty$, the L_∞-norm distance function $L_\infty(x, y)$ is given by:

$$L_\infty(x, y) = max_{i=1}^{n} |x[i] - y[i]|, \quad (2)$$

where x and y are two n-dimensional data objects.

In the sequel, we formalize the problem of the *similarity search* in *arbitrary* subspaces.

***Problem** 3.1:* Assume we have a database $\mathcal{D}$ containing n-dimensional data objects in the *full* space DIM. Given any k-dimensional query object $q^{(k)}$ in the subspace $DIM' \subseteq DIM$, we want to retrieve all data objects $o \in \mathcal{D}$, such that $L_p(q^{(k)}, o^{(k)}) \leq \varepsilon$, where $L_p(\cdot, \cdot)$ is a L_p-norm distance function for $1 \leq p \leq \infty$, $o^{(k)}$ is a k-dimensional point obtained from object o projected on DIM', and $k \in [k_{min}, k_{max}]$.

Note that, we consider DIM in a numerical domain, and leave other interesting domains (e.g. categorical domain) as our future work. In a special case of Problem 3.1 where $k_{min} = k_{max}$, query objects always specify subspaces with the same number of dimensions (i.e. *fixed* value of k), however, they may query different dimensions. Table I summarizes the commonly-used symbols in this paper.

TABLE I
Meanings of Symbols Used

Symbols	Descriptions
$\mathcal{D}$	a data set of size N
$\mathcal{P}_i$	the i-th partition of the data set $\mathcal{D}$
$\lvert\mathcal{D}\rvert$ or N	the size of $\mathcal{D}$
m	the number of *pivots* or partitions
n	the dimensionality of data objects
k	the number of dimensionality in the subspace specified by the query object
piv_i	the i-th n-dimensional *pivot* in $\mathcal{D}$
o	an n-dimensional data object in $\mathcal{D}$
$q^{(k)}$	a k-dimensional query point specified by users
$x^{(k)}$	a k-dimensional point from object x projected on subspace DIM'
$\phi(x)$	the *cumulative distribution function* (CDF) following a *normal distribution*

IV. Subspace Similarity Search

In this section, we discuss details of our solutions to the *similarity search* problem in *arbitrary* subspaces. The framework for our *subspace similarity search* problem consists of three steps, *pre-processing*, *query processing* and *post-processing*. Specifically, the *pre-processing step* computes 1-dimensional scores for each data object in the data set with respect to a set of selected pivots. The second *query processing step* retrieves a candidate set $cand$ using pruning conditions on scores. Finally, the *post-processing step* refines the candidate set and outputs data objects that ε-match with the query object.

A. A Brief Review: Pruning with the Triangle Inequality

In *metric* spaces, one of the most important properties for *metric* distance functions is the *triangle inequality* defined as follows.

Definition 4.1: (Triangle Inequality) Given any three objects x, y, and z in the space, a distance function $dist$ satisfies the *triangle inequality* only if $dist(x, z) \leq dist(x, y) + dist(y, z)$ or equivalently $dist(x, z) \geq |dist(x, y) - dist(y, z)|$.

The *triangle inequality* can be used to effectively prune candidates during the *similarity search* and thus save the cost of distance computations between candidates and the query object. The basic idea is as follows. Assume we select a pivot piv in a data set $\mathcal{D}$. Now we want to issue a *similarity query* which finds objects in the data set that ε-match with query object q. Instead of directly computing the distance from query q to data object o, we can alternatively apply the *triangle inequality* to prune *false positives*. Obviously, the distance $L_2(piv, o)$

between pivot piv and any data object $o \in \mathcal{D}$ can be pre-computed, and the distance $L_2(q, piv)$ between query object q and pivot piv can be efficiently calculated upon q's arrival. Thus, as long as it holds that $|L_2(q, piv) - L_2(piv, o)| > \varepsilon$, we can safely prune object o, since by the *triangle inequality* $L_2(q, o) \geq |L_2(q, piv) - L_2(piv, o)|$, we have $L_2(q, o) > \varepsilon$, indicating that o is not the query result.

Note that, L_p-norm distance functions follow the *triangle inequality*, for $1 \leq p \leq \infty$, not only in the *full* space but also in the *subspace*. This motivates us to use the *triangle inequality* to prune candidates in the *subspace* during the *subspace similarity search*, as discussed below.

B. Similarity Search in Arbitrary Subspaces

In this subsection, we propose an effective approach to solve Problem 3.1, which assumes that each query object $q^{(k)}$ can specify an *arbitrary* subspace with *arbitrary* number of dimensions, k, where $k \in [k_{min}, k_{max}]$. Due to space limit, throughout this subsection, we only consider *subspace similarity search* under L_p-norm for $1 \leq p < \infty$ in Eq. (2). The case of L_∞-norm (i.e. $p = \infty$) is similar and thus omitted.

To start with, we first consider the search problem in one *specific* subspace DIM'. Let $q^{(k)}$ be a k-dimensional query object in DIM', and $o^{(k)}$ ($piv^{(k)}$) a k-dimensional point obtained from the projection of data object $o \in \mathcal{D}$ (pivot piv) on DIM'. Since the *triangle inequality* holds under L_p-norm ($1 \leq p \leq \infty$) in DIM', according to Definition 4.1, we have:

$$L_p(q^{(k)}, o^{(k)}) \geq |L_p(q^{(k)}, piv^{(k)}) - L_p(piv^{(k)}, o^{(k)})|. \quad (3)$$

Therefore, given a *subspace similarity query* in DIM' with a query object $q^{(k)}$ and a similarity threshold ε, any data object $o \in \mathcal{D}$ can be safely pruned, if it satisfies:

$$|L_p(q^{(k)}, piv^{(k)}) - L_p(piv^{(k)}, o^{(k)})| > \varepsilon, \quad (4)$$

which can be rewritten as:

$$L_p(piv^{(k)}, o^{(k)}) < L_p(q^{(k)}, piv^{(k)}) - \varepsilon, \text{ or} \quad (5)$$

$$L_p(piv^{(k)}, o^{(k)}) > L_p(q^{(k)}, piv^{(k)}) + \varepsilon. \quad (6)$$

where $1 \leq p \leq \infty$.

Inequalities (5) and (6) are the pruning conditions of the *subspace similarity search*. In other words, if either Inequality (5) or Inequality (6) holds, point $o^{(k)}$ is guaranteed *not* to be the answer to the *subspace similarity query* in subspace DIM'. Based on this pruning heuristics, we can assign a 1-dimensional score $score(o^{(k)})$ to each data object o, which is defined as the L_p-norm distance $L_p(piv^{(k)}, o^{(k)})$ between pivot $piv^{(k)}$ and point $o^{(k)}$. Thus, any data object $o \in \mathcal{D}$ having score $score(o^{(k)})$ within the interval $[L_p(q^{(k)}, piv^{(k)}) - \varepsilon, L_p(q^{(k)}, piv^{(k)}) + \varepsilon]$ is a candidate of the query result. Note, however, that, this pruning method is only applicable to one *specific* subspace DIM' with k dimensions, but not to *arbitrary* subspace with *arbitrary* number of dimensions (within $[k_{min}, k_{max}]$).

Next, we propose an effective approach to enable handling the *similarity search* in *arbitrary* subspaces. Specifically, instead of transforming each data object o to a *single* score $score(o^{(k)})$, we use two 1-dimensional scores, $minscore(o^{(k)})$ and $maxscore(o^{(k)})$, to facilitate pruning candidates in *arbitrary* subspaces. In the sequel, we formally define the two scores of data object $o \in \mathcal{D}$ with respect to pivot piv.

Definition 4.2: (Two Scores of Data Object Under L_p-Norm, for $p \in [1, \infty)$) Given a database $\mathcal{D}$, a pivot piv, and arbitrary k-dimensional subspace DIM' ($k \in [k_{min}, k_{max}]$), for any data object $o \in \mathcal{D}$, its two scores, $minscore(o^{(k)})$ and $maxscore(o^{(k)})$, under L_p-norm ($1 \leq p < \infty$), are defined as:

$$minscore(o^{(k)}) = k \cdot min_{j=k_{min}}^{k_{max}} \{ \frac{\sqrt[p]{\sum_{i=1}^{j} (Diff[i])^p}}{j} \}, \quad (7)$$

$$maxscore(o^{(k)}) = k \cdot max_{j=k_{min}}^{k_{max}} \{ \frac{\sqrt[p]{\sum_{i=n-j+1}^{n} (Diff[i])^p}}{j} \}, \quad (8)$$

where the detailed steps to obtain $Diff[i]$ ($1 \leq i \leq n$) are described as follows.

First, we compute the (absolute) distance difference $Diff'[i]$ between pivot piv and object o along each dimension i, for all $1 \leq i \leq n$. That is,

$$Diff'[i] = |piv[i] - o[i]|, \text{ for } 1 \leq i \leq n. \quad (9)$$

Then, we sort $Diff'[1]$, $Diff'[2]$, ..., and $Diff'[n]$, and obtain a *non-decreasing* sequence $Diff[1]$, $Diff[2]$, ..., and $Diff[n]$ satisfying:

$$Diff[1] \leq Diff[2] \leq ... \leq Diff[n], \quad (10)$$

where $Diff[i] \in \{Diff'[1], Diff'[2], ..., Diff'[n]\}$, for all $1 \leq i \leq n$.

Finally, we are ready to use $Diff[i]$ to define two scores, $minscore(o^{(k)})$ and $maxscore(o^{(k)})$, for object o, as given in Eq. (7) and Eq. (8) .

Intuitively, $minscore(o^{(k)})$ and $maxscore(o^{(k)})$ are the minimum and maximum possible distances between data object o and pivot piv, respectively, taking into account *arbitrary* subspaces with dimension value $k \in [k_{min}, k_{max}]$. Therefore, we have the following lemma:

Lemma 4.1: In any subspace $DIM' \subseteq DIM$ with k dimensions ($k \in [k_{min}, k_{max}]$), we have:

$$minscore(o^{(k)}) \leq L_p(piv^{(k)}, o^{(k)}), \text{ and} \quad (11)$$

$$maxscore(o^{(k)}) \geq L_p(piv^{(k)}, o^{(k)}). \quad (12)$$

Proof. We first prove the correctness of Inequality (11). Without loss of generality, as in Eq. (7), assume that when $j = k' \in [k_{min}, k_{max}]$, operator min on RHS of Inequality (7) achieves the minimum. Therefore, we always have $\frac{\sqrt[p]{\sum_{i=1}^{k'} (Diff[i])^p}}{k'} \leq \frac{\sqrt[p]{\sum_{i=1}^{k} (Diff[i])^p}}{k}$ for *arbitrary* k values within $[k_{min}, k_{max}]$. Furthermore, since $Diff[1]$, $Diff[2]$, ..., and $Diff[k]$ are k smallest (absolute) distance differences between piv and o among all dimensions, it holds

that $\sqrt[p]{\sum_{i=1}^{k}(Diff[i])^p} \leq L_p(piv^{(k)}, o^{(k)})$. Thus, we have $minscore(o^{(k)}) = k \cdot \frac{\sqrt[p]{\sum_{i=1}^{k'}(Diff[i])^p}}{k'} \leq \sqrt[p]{\sum_{i=1}^{k}(Diff[i])^p} \leq L_p(piv^{(k)}, o^{(k)})$, which completes our proof of Inequality (11). The proof of Inequality (12) is similar. □

Based on Lemma 4.1, we illustrate pruning conditions in the following theorem.

Theorem 4.1: Assume we have a database $\mathcal{D}$ containing data objects in the n-dimensional *full* space DIM, and a *subspace similarity query* under L_p-norm ($1 \leq p < \infty$) with a k-dimensional query object $q^{(k)}$ in arbitrary subspace DIM' of DIM and a similarity threshold ε, where $k \in [k_{min}, k_{max}]$. Any data object $o \in \mathcal{D}$ can be safely pruned, if either

$$maxscore(o^{(k)}) < L_p(q^{(k)}, piv^{(k)}) - \varepsilon, \text{ or} \tag{13}$$

$$minscore(o^{(k)}) > L_p(q^{(k)}, piv^{(k)}) + \varepsilon, \tag{14}$$

holds, where $1 \leq p < \infty$.

Proof. By combining Inequality (13) with (12), it exactly yields Inequality (5), which is the pruning condition for an object o. Thus, when Inequality (13) holds, object o can be safely pruned. Similarly, by combining Inequality (14) with (11), we obtain Inequality (6), which is the second pruning condition, and completes our proof. □

From Eq. (7) and Eq. (8), one interesting observation is that, for each data object o, its two scores, $minscore(o^{(k)})$ and $maxscore(o^{(k)})$, are both *proportional* to k which is the dimensionality of the subspace specified by query object $q^{(k)}$. In other words, if query objects specify different k values, scores of the same data object would be different. Since there are totally $(k_{max} - k_{min} + 1)$ possible k values, it is not very space-efficient to store all of them for query processing. Instead, in this paper, we keep only two *average scores* for each data object o, $minavg(o)$ and $maxavg(o)$, which are defined as $minavg(o) = \frac{minscore(o^{(k)})}{k}$ and $maxavg(o) = \frac{maxscore(o^{(k)})}{k}$, respectively. In particular, from Eq. (7) and Eq. (8), $minavg(o) = min_{j=k_{min}}^{k_{max}}\{\frac{\sqrt[p]{\sum_{i=1}^{j}(Diff[i])^p}}{j}\}$, and $maxavg(o) = max_{j=k_{min}}^{k_{max}}\{\frac{\sqrt[p]{\sum_{i=n-j+1}^{n}(Diff[i])^p}}{j}\}$.

Note that, assuming we know the range $[k_{min}, k_{max}]$ of k that query objects may specify, we can pre-process each data object offline by computing two *average scores* for each data object with respect to a pivot. Given any query object $q^{(k)}$ specifying a k-dimensional subspace, the two scores, $minscore(o^{(k)})$ and $maxscore(o^{(k)})$, of any data object $o \in \mathcal{D}$ can be calculated *on the fly* multiplying $minavg(o)$ and $maxavg(o)$ by k, respectively, where the value of k is specified by query object $q^{(k)}$.

Figure 1 illustrates the detailed procedure of *subspace similarity search* under L_p-norm ($1 \leq p < \infty$) in *arbitrary* subspace with *arbitrary* number of dimensions, k, within $[k_{min}, k_{max}]$. The case of L_∞-norm is similar. Specifically, procedure SubspaceSearch_L_p includes three steps, *pre-processing*, *query processing*, and *post-processing*. In the *pre-processing step*, we compute two *average scores*, $minavg(o)$ and $maxavg(o)$, of each data object o with respect to a pivot (lines 1-2). During the *query processing* step, given any query object $q^{(k)}$ specifying a k-dimensional subspace, we obtain the candidate set $cand_1$ ($cand_2$) that contains data objects with scores violating the pruning condition in Inequality (13) (Inequality (14)). Note that, in Inequalities (13) and (14), the two scores of each data object can be computed *on the fly*, that is, $maxscore(o^{(k)}) = k \cdot maxavg(o)$ and $minscore(o^{(k)}) = k \cdot minavg(o)$. Then, we intersect $cand_1$ with $cand_2$, and obtain the final candidate set $cand$ (lines 3-5). Finally, in the *post-processing step*, each candidate o in $cand$ is refined by checking whether or not the real L_p-norm distance between $o^{(k)}$ and $q^{(k)}$ is within ε (lines 6-7).

Procedure SubspaceSearch_L_p {
Input: a data set $\mathcal{D}$, a query object $q^{(k)}$ ($k \in [k_{min}, k_{max}]$), a pivot piv, and a similarity threshold ε
Output: data objects that ε-match with $q^{(k)}$
// *pre-processing step*
(1) for each object o
(2) compute two *average scores* $minavg(o)$ and $maxavg(o)$
// *query processing step*
(3) obtain a candidate set $cand_1$ containing objects that violate Inequality (13), where $maxscore(o^{(k)}) = k \cdot maxavg(o)$
(4) obtain a candidate set $cand_2$ containing objects that violate Inequality (14), where $minscore(o^{(k)}) = k \cdot minavg(o)$
(5) $cand = cand_1 \bigcap cand_2$ // $cand$ is a candidate set
// *post-processing step*
(6) for each candidate o in $cand$
(7) if $L_p(q^{(k)}, o^{(k)}) \leq \varepsilon$, then output data object o
}

Fig. 1. Similarity Search in Arbitrary Subspaces (Under L_p-Norm, for $p \in [1, \infty)$)

V. Multipivot-Based Query Processing

Based on pruning conditions discussed in the previous section under L_p-norm ($p \in [1, \infty]$), we propose an effective *multipivot-based* approach to efficiently answer the *subspace similarity query*. Note that, the existing cost models [6] only consider the cost of the full space search rather than subspace search. Specifically, the *multipivot-based* method selects m different pivots piv_1, piv_2, ..., and piv_m from data set $\mathcal{D}$. For each data object $o \in \mathcal{D}$, we calculate two scores with respect to *every pivot* piv_i, for all $1 \leq i \leq m$. Therefore, each object has totally $2m$ scores, which takes up $O(2 \cdot m \cdot |\mathcal{D}|)$ space cost.

When a query centered at a query object $q^{(k)}$ in subspace DIM' with a radius ε arrives, we first compute the L_p-norm distance between $q^{(k)}$ and each of m pivots piv_i in subspace DIM', that is, $L_p(q^{(k)}, piv_i^{(k)})$, for $k \in [k_{min}, k_{max}]$ and $1 \leq i \leq m$. Then, we retrieve those candidates using the pruning technique discussed in Section IV-B under L_p-norm. For each of the m pivots, we can get a candidate set, and the final candidate set is obtained by intersecting all the m returned candidate sets. Note that, although we apply the existing pivot-based technique via the triangle inequality, our multipivot-based approach is novel in the sense that we use it to effectively solve the subspace search problem.

Since our pruning techniques use pivots to filter candidates, it is of great importance to select appropriate pivots in order to achieve "good" performance of the *subspace similarity query*. This motivates us to provide a formal cost model to

guide the pivot selection for our *multipivot-based* approach, in light of which either the *pruning power* is maximized or the *computation cost* minimized.

A. Cost Model

In this subsection, we model the performance of the *multipivot-based* approach, in terms of the *pruning power* PP and *computation cost* CC. Specifically, given m pivots piv_1, piv_2, ..., and piv_m, for each data object $o \in \mathcal{D}$, the *multipivot-based* approach calculates two scores $minscore_i(o^{(k)})$ and $maxscore_i(o^{(k)})$ with respect to each pivot piv_i ($1 \leq i \leq m$). Therefore, given any *subspace similarity query* with query object $q^{(k)}$ and similarity threshold $\varepsilon^{(k)}$ ($k \in [k_{min}, k_{max}]$), a data object o can be safely pruned, if there exists at least one pivot piv_i, such that:

$$maxscore_i(o^{(k)}) < L_p(q^{(k)}, piv_i^{(k)}) - \varepsilon^{(k)}, \text{ or} \tag{15}$$

$$minscore_i(o^{(k)}) > L_p(q^{(k)}, piv_i^{(k)}) + \varepsilon^{(k)}, \tag{16}$$

where $1 \leq p \leq \infty$.

From the probabilistic point of view, for a given subspace DIM' with k dimensions, the pruning power $PP_{multipivot-based}(k)$ can be obtained by summing up the probability with which each data object can be pruned by either Inequality (15) or (16). Specifically, we have:

$$\begin{aligned} & PP_{multipivot-based}(k) \\ = & \sum_{j=1}^{|\mathcal{D}|-m} Pr\{(\bigcup_{i=1}^{m} \text{data object } o \text{ can be pruned by } piv_i) | o \in \mathcal{D}\} \\ = & (|\mathcal{D}| - m) \cdot Pr\{\bigcup_{i=1}^{m} ((maxscore_i(o^{(k)}) < L_p(q^{(k)}, piv_i^{(k)}) - \varepsilon^{(k)}) \\ & \bigcup (minscore_i(o^{(k)}) > L_p(q^{(k)}, piv_i^{(k)}) + \varepsilon^{(k)}))\} \end{aligned} \tag{17}$$

Let A_i be the event that $maxscore_i(o^{(k)}) < L_p(q^{(k)}, piv_i^{(k)}) - \varepsilon^{(k)}$ and B_i the event that $minscore_i(o^{(k)}) > L_p(q^{(k)}, piv_i^{(k)}) + \varepsilon^{(k)}$. Eq. (17) can be simplified as:

$$\begin{aligned} & PP_{multipivot-based}(k) \\ = & (|\mathcal{D}| - m) \cdot Pr\{\bigcup_{i=1}^{m} (A_i \bigcup B_i)\} \\ = & (|\mathcal{D}| - m) \cdot (1 - Pr\{\overline{\bigcup_{i=1}^{m} (A_i \bigcup B_i)}\}) \\ = & (|\mathcal{D}| - m) \cdot (1 - Pr\{\bigcap_{i=1}^{m} (\overline{A_i} \bigcap \overline{B_i})\}). \end{aligned} \tag{18}$$

Now we denote $(\overline{A_i} \bigcap \overline{B_i})$ as event C_i, which indicates that object o *cannot* be pruned, using either $minscore_i(o^{(k)})$ or $maxscore_i(o^{(k)})$, by a pivot piv_i. Since pivots piv_i for $1 \leq i \leq m$ are *randomly* selected in the data set, events C_i and C_j are independent for $i \neq j$. Thus, we have $Pr\{\bigcap_{i=1}^{m} C_i\} = \prod_{i=1}^{m} Pr\{C_i\}$. Next, we illustrate the key step to derive $Pr\{C_i\}$. In particular, it holds that:

$$Pr\{C_i\} = Pr\{\overline{A_i}\} - Pr\{\overline{A_i} \bigcap B_i\}, \tag{19}$$

which can be shown using Venn Diagram. Furthermore, since event B_i can always infer event $\overline{A_i}$ (note that, $minscore^{(k)}$ is always not greater than $maxscore^{(k)}$), we have:

$$Pr\{\overline{A_i} \bigcap B_i\} = Pr\{B_i\}. \tag{20}$$

Therefore, we can rewrite Eq. (18) as follows:

$$\begin{aligned} & PP_{multipivot-based}(k) \\ = & (|\mathcal{D}| - m) \cdot (1 - \prod_{i=1}^{m} Pr\{\overline{A_i} \bigcap \overline{B_i}\}) \\ = & (|\mathcal{D}| - m) \cdot (1 - \prod_{i=1}^{m} (Pr\{\overline{A_i}\} + Pr\{\overline{B_i}\} - 1)). \end{aligned} \tag{21}$$

Let $P(k)$ be the probability that query objects specify a subspace DIM' having k dimensions, where $k \in [k_{min}, k_{max}]$. The *expected pruning power* $E(PP)$ of *multipivot-based* approach is given by:

$$E(PP) = \sum_{i=k_{min}}^{k_{max}} P(k) \cdot PP_{multipivot-based}(k). \tag{22}$$

As a second step, for a subspace DIM' with k dimensions ($k \in [k_{min}, k_{max}]$), the *computation cost* $CC_{multipivot-based}(k)$ is defined by:

$$CC_{multipivot-based}(k) = k \cdot (m + |\mathcal{D}| - PP_{multipivot-based}(k)) \tag{23}$$

with which the *expected computation cost* $E(CC)$ given a workload of queries is denoted as:

$$E(CC) = \sum_{i=k_{min}}^{k_{max}} P(k) \cdot CC_{multipivot-based}(k). \tag{24}$$

Finally, we address the remaining issue of obtaining $Pr\{\overline{A_i}\}$ and $Pr\{\overline{B_i}\}$ for $1 \leq i \leq m$ in Eq. (21). In particular, we first consider $Pr\{\overline{A_i}\}$, where $\overline{A_i}$ is the event that $maxscore_i(o^{(k)}) \geq L_p(q^{(k)}, piv_i^{(k)}) - \varepsilon^{(k)}$ holds.

Without loss of generality, we assume that query object q can locate anywhere in the data space with any query radius $\varepsilon^{(k)}$, which is independent of objects o (piv_i) in the data set. Therefore, score $maxscore_i(o^{(k)})$ and distance $L_p(q^{(k)}, piv_i^{(k)})$ are distance-independent, in the sense that the score with respect to pivot piv_i and object o cannot infer any information about distance $L_p(q^{(k)}, piv_i^{(k)})$ from pivot to query object (since query point is independent of pivot). Furthermore, the query radius $\varepsilon^{(k)}$ is also independent of both score and distance. Therefore, we can apply the *Central Limit Theorem* (CLT) [28] to compute $Pr\{\overline{A_i}\}$. Specifically, assume that $maxscore_i(o^{(k)})$ is a random number generated from a random variable $Y_{max,i}^{(k)}$ with mean $\mu_{max,i}$ and variance $(\sigma_{max,i}^{(k)})^2$. Let mean and variance of $L_p(q^{(k)}, piv_i^{(k)})$ from random variable $Y_{piv,i}^{(k)}$ be $\mu_{piv,i}^{(k)}$ and $(\sigma_{piv,i}^{(k)})^2$, respectively. Similarly, let mean and variance of $\varepsilon^{(k)}$ from random variable $Y_R^{(k)}$ be $\mu_R^{(k)}$ and $(\sigma_R^{(k)})^2$, respectively.

According to CLT, for a number z (e.g. $z = L_p(q^{(k)}, piv_i^{(k)}) - maxscore_i(o^{(k)}) - \varepsilon^{(k)}$) generated from $Z = Y_{piv,i}^{(k)} - Y_{max,i}^{(k)} - Y_R^{(k)}$, we have $\frac{z - (\mu_{piv,i}^{(k)} - \mu_{max,i}^{(k)} - \mu_R^{(k)})}{\sqrt{((\sigma_{piv,i}^{(k)})^2 + \sigma_{max,i}^{(k)})^2 + (\sigma_R^{(k)})^2}}$ following

a *normal distribution* with *cumulative distribution function* (CDF) $\phi(x)$. That is,

$$\begin{aligned} Pr\{\overline{A_i}\} &= Pr\{maxscore_i(o) \geq L_p(q^{(k)}, piv_i^{(k)}) - \varepsilon^{(k)}\} \\ &= Pr\{\frac{z - (\mu_{piv,i}^{(k)} - \mu_{max,i}^{(k)} - \mu_R^{(k)})}{\sqrt{((\sigma_{piv,i}^{(k)})^2 + (\sigma_{max,i}^{(k)})^2 + (\sigma_R^{(k)})^2}} \\ &\leq \frac{-(\mu_{piv,i}^{(k)} - \mu_{max,i}^{(k)} - \mu_R^{(k)})}{\sqrt{((\sigma_{piv,i}^{(k)})^2 + (\sigma_{max,i}^{(k)})^2 + (\sigma_R^{(k)})^2}}\} \\ &= \phi(\frac{-(\mu_{piv,i}^{(k)} - \mu_{max,i}^{(k)} - \mu_R^{(k)})}{\sqrt{((\sigma_{piv,i}^{(k)})^2 + (\sigma_{max,i}^{(k)})^2 + (\sigma_R^{(k)})^2}}) \end{aligned} \quad (25)$$

Note that, although CLT assumes a summation of large number of random variables, there are some work [11], [16] indicating that 3 variables can achieve a good approximation, which will later be confirmed by our cost model verification in the experiments. Similarly, we can obtain the probability $Pr\{\overline{B_i}\}$ of event $\overline{B_i}$ ($1 \leq i \leq m$). Without loss of generality, let $minscore_i(o^{(k)})$ be a random number generated from a random variable $Y_{min,i}^{(k)}$ with mean $\mu_{min,i}^{(k)}$ and variance $(\sigma_{min,i}^{(k)})^2$. According to CLT [28], we have:

$$\begin{aligned} Pr\{\overline{B_i}\} &= Pr\{minscore_i(o) \leq L_p(q^{(k)}, piv_i^{(k)}) + \varepsilon^{(k)}\} \\ &= \phi(\frac{-(\mu_{min,j}^{(k)} - \mu_{piv,j}^{(k)} - \mu_R^{(k)})}{\sqrt{(\sigma_{min,j}^{(k)})^2 + (\sigma_{piv,j}^{(k)})^2 + (\sigma_R^{(k)})^2}}) \end{aligned} \quad (26)$$

B. Data Pre-Processing and Query Processing

Recall that, in the *multipivot-based* method, each data object can be pruned by any of m chosen pivots in the data set. We summarize the goal of pivot selection as follows.

(*Pivot Selection*) *Given a data set $\mathcal{D}$, we want to obtain m pivots from $\mathcal{D}$ such that either the expected pruning power $E(PP)$ (in Eq. (22)) is maximized or the expected computation cost $E(CC)$ (in Eq. (24)) is reduced during subspace queries.*

In the sequel, we focus on maximizing $E(PP)$. Note, however, that this assumption does not conflict with our goal of minimizing $E(CC)$. Maximizing the *pruning power* can reduce the computation cost, as indicated in Eq. (23).

The basic idea of our *multipivot-based* approach is as follows. Initially, we randomly obtain m pivots piv_i ($1 \leq i \leq m$), and evaluate the *expected pruning power* $E(PP)$ based on the cost model (Eq. (22)). Then, we iteratively invoke a procedure, which takes $\mathcal{D}$, piv_i, and $E(PP)$ as input, until the stopping condition discussed later is satisfied. Each time we randomly select a data object piv that is not one of the current pivots and replace one random pivot piv_i with the new one piv. Next, we re-evaluate the *expected pruning power* $E(PP')$ using Eq. (22). If new pivot list results in greater pruning power, that is, $E(PP') > E(PP)$, we update and return new pivot list and *expected pruning power*; otherwise, do nothing. Note that, here we apply the same stopping condition as that in CLARANS [25], that is, either the number of trials that procedure fails to swap a pivot exceeds a threshold or the number of successful swaps exceeds a threshold. In order to avoid obtaining the *local optimum* result, we pre-process data for several passes by randomly selecting different initial pivots and use the best set of pivots that is expected to have high *pruning power*.

One interesting issue is how to choose the number of pivots, m. In our work, since data pre-processing is performed offline, we can choose different values of m within the range $[1, N/2]$ (since when $m > N/2$, $E(PP)$ is always smaller than that when $m = N/2$), and pick up the best one that maximizes the *expected pruning power* $E(PP)$ given in Eq. (22). Heuristically, we can choose m value with respect to *intrinsic dimensionality* of the data set [9].

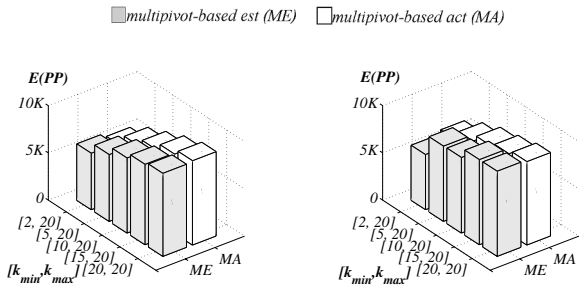
Up to now, we have pre-processed data by selecting pivots in the data set and computing scores for each data object with respect to pivots. Then, we organize the score vector of each object (containing $2m$ scores) in a list, $list$, sequentially sorted on one of scores with respect to a pivot. When a *subspace similarity query* is issued, we access the sorted list in either ascending or descending order, prune data objects using $2m$ scores (in Inequalities (15) and (16)), and output candidate pairs. Moreover, query processing can abort early during the scanning of list, when the remaining data objects with the sorting score satisfy the pruning condition.

VI. Experimental Evaluation

In this section, we demonstrate through extensive experiments the efficiency and effectiveness of our proposed *multipivot-based* approach to process the *subspace similarity search* under L_p-norm, where $1 \leq p \leq \infty$. Specifically, we conduct our experiments over two real data sets, *layout* and *sstock*. The first data set, *layout*, contains 68K 32-dimensional feature vectors representing the density of colors from *sub-images* of *Corel Image* data collections (available at: [*http://kdd.ics.uci.edu/*]). Since the dimensionality of this data set is small, in order to test the scalability of our proposed approach with respect to the dimensionality, we randomly concatenate any two 32-dimensional vectors, and obtain a 64-dimensional vector, resulting in a data set of size 34K. The second data set, *sstock*, is obtained from 193 company stocks' daily closing price from late 1993 to early 1996, including 50K stock price series of length 128. For both data sets, we normalize each dimension of the vector to interval $[0, 1]$.

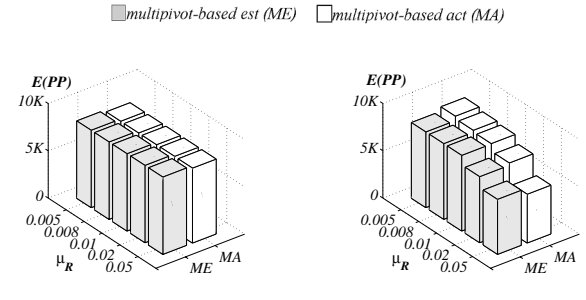
In our experiments, we select 2000 random points from each data set as query objects, and synthetically produce the query workload by randomly picking up k ($\in [k_{min}, k_{max}]$) out of totally n dimensions as subspace in which the *similarity search* is performed. Moreover, query radii are generated from a distribution with mean μ_R and variance σ_R^2.

In the sequel, Section VI-A first verifies the correctness of our proposed cost model which would be used for the *multipivot-based* approach in answering the *subspace similarity query*. Section VI-B illustrates the experimental result of the query performance with our proposed approach, in terms of the *wall clock time*, compared with the B$^+$-tree method (mentioned in Section I) and the *linear scan*. We run the



(a) *layout* (b) *sstock*

Fig. 2. Cost Model Verification (PP vs. $[k_{min}, k_{max}]$)



(a) *layout* (b) *sstock*

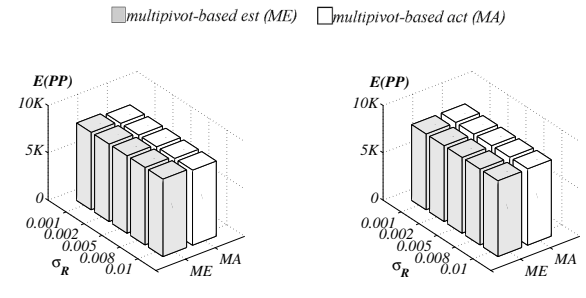
Fig. 3. Cost Model Verification (PP vs. μ_R)

experiments on Pentium 4 3.4GHz PC with 1G memory. All experimental results are averaged over 50 runs.

A. Cost Model Verification

Before we show the efficiency of our proposed *multipivot-based* approach, we first verify the correctness of our cost model, which is the basis of this approach. In particular, due to space limit, we only report the results of the cost model verification over *layout* and *sstock*, under L_2-norm and with respect to three parameters $[k_{min}, k_{max}]$, μ_R, and σ_R. In fact, similar results have been obtained by varying other parameters (e.g. m, n, N, and L_p-norms).

Specifically, we compare the *expected pruning power* $E(PP)$ of queries, which is estimated from the cost model, with the actual one, by varying parameters $[k_{min}, k_{max}]$, μ_R, and σ_R. We use half of the query objects (i.e. 1000 query points) to extract statistics for the cost model estimation, and the other half to test the real query performance. Note that, the estimated $E(PP)$ in this set of experiments is only used for verifying the correctness of cost model, and it is thus not the final pruning power after applying the *multipivot-based* approach.



(a) *layout* (b) *sstock*

Fig. 4. Cost Model Verification (PP vs. σ_R)

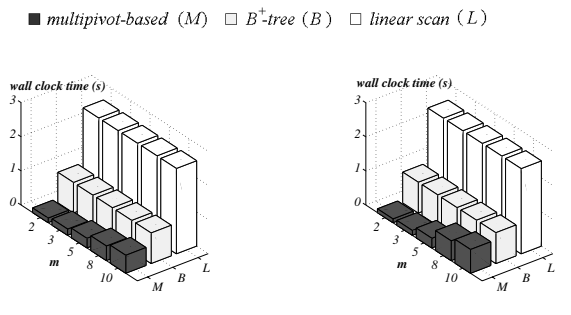
Figure 2 illustrates the estimated and actual $E(PP)$ of the *multipivot-based* approach, over *layout* and *sstock* data sets, by setting the range $[k_{min}, k_{max}]$ of k to $[2, 20], [5, 20], [10, 20], [15, 20], [20, 20]$, where $m = 5$, $N = 10K$, $\mu_R = 0.01$, $\sigma_R = 0.005$, and $n = 64$. As illustrated in Figure 2, when the range length $(k_{max} - k_{min})$ is large (e.g. 20-2=18), the pruning power becomes low. This is due to the relaxation of two scores in Eq. (7) and Eq. (8). With different $(k_{max} - k_{min})$, for both data sets, the estimated $E(PP)$ using our approach can closely mimic actual values, which confirms the correctness of our cost model.

Next, we study the effect of the query radius on the cost model estimation. Specifically, Figure 3 illustrates the estimated and actual *pruning power* over data sets *layout* and *sstock*, with different radius mean $\mu_R = 0.005, 0.008, 0.01, 0.02, 0.05$, where $[k_{min}, k_{max}] = [10, 20]$, $N = 10K$, $\sigma_R = 0.0005$, and $n = 48$ for *layout* ($n = 64$ for *sstock*). Furthermore, Figure 4 varies the variance σ_R of query radii from 0.005 to 0.01. For all figures, the estimated $E(PP)$ is very close to the actual value, which confirms the correctness of our cost model under different query radii.

By verifying cost model under a variety of parameter settings, we infer that, our cost model can well estimate the real *pruning power* of *subspace similarity queries*. Therefore, given a set of selected pivots, we can accurately evaluate the query performance of the *multipivot-based* method using the cost model, in light of which we select pivots to maximize the pruning power, as shown in next subsection.

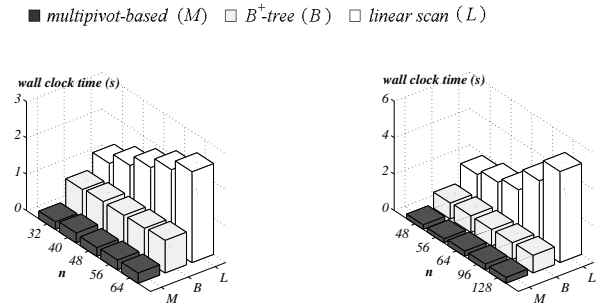
B. Performance of Query Processing

In this set of experiments, we study the query performance of *subspace similarity queries* using our method over both real data sets, *layout* and *sstock*, in terms of the *wall clock time*. Specifically, we incorporate the I/O cost into the *wall clock time* by penalizing $10ms$ for each page access, where the page size is set to 10K. As mentioned in Section I, in order to answer *subspace similarity queries*, it is infeasible to build indexes over all possible subspaces, and moreover directly indexing the *full* space would result in query performance even worse than a *linear scan* (due to the dimensionality curse). In this set of experiments, we compare our approach with the B$^+$-tree method. In particular, this method constructs one B$^+$-tree for each dimension of the n-dimensional *full* space. For a *subspace similarity query* specifying a subspace with k dimensions and a search radius ε, we issue k *range queries* with radius ε, over k B$^+$-trees, corresponding to k specified dimensions in the subspace, respectively. Then, we combine all candidates returned from *range queries* and obtain the final result. Furthermore, we also give the query performance of the *linear scan* in terms of the *wall clock time*. Due to space limit, in the sequel, we only present the experimental result under



(a) *layout* (b) *sstock*

Fig. 5. Query Efficiency vs. m



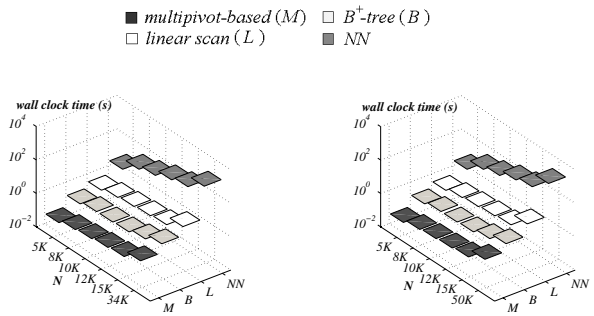
(a) *layout* (b) *sstock*

Fig. 6. Query Efficiency vs. n

L_2-norm. Similar experimental results have been obtained under other L_p-norms ($p \neq 2$) and with other parameters (e.g. $[k_{min}, k_{max}]$, μ_R, and σ_R).

Figure 5 studies the effect of the number of *pivots*, m, on the performance of *subspace similarity search*, comparing three methods, *multipivot-based*, B$^+$-tree, and the *linear scan*, over both real data sets *layout* and *sstock*. In particular, we vary parameter m from 2 to 10, where $[k_{min}, k_{max}] = [10, 20]$, $N = 10K$, $n = 64$, $\mu_R = 0.01$ and $\sigma_R = 0.005$. As indicated by figures, the *wall clock time* of the *multipivot-based* method is much smaller than that of B$^+$-tree and *linear scan*, which confirms the efficiency and effectiveness of our proposed approach. When m increases, the *wall clock time* of the *multipivot-based* method becomes higher, which is a bit counter-intuitive, however, reasonable. Although large m can provide high pruning power, more spaces are needed to store scores for data objects which thus incurs more page accesses.

Next, Figure 6 illustrates the query performance of the three approaches over two real data sets, by varying the dimensionality n of the *full* space, where $[k_{min}, k_{max}] = [10, 20]$, $N = 10K$, $m = 5$, $\mu_R = 0.01$ and $\sigma_R = 0.005$. Specifically, according to the total available dimensions of each data set, we set $n = 32, 40, 48, 56, 64$ for data set *layout*, and $n = 48, 56, 64, 96, 128$ for *sstock*. Due to the increased dimensionality, the *linear scan* has to access more disk pages, which leads to the increasing *wall time clock* with respect to n. From the experimental results, our proposed *multipivot-based* approach outperforms both B$^+$-tree and *linear scan* methods in terms of the *wall clock time*.



(a) *layout* (b) *sstock*

Fig. 7. Scalability Test (*wall clock time* vs. N)

C. Scalability Test

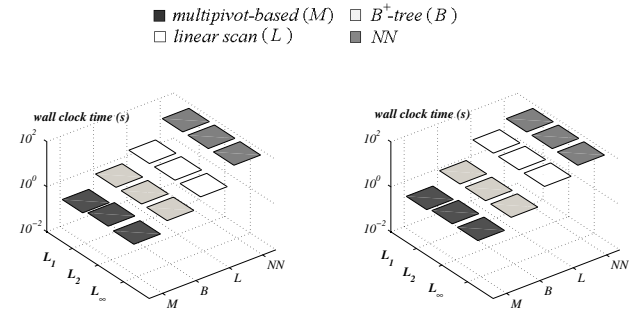
Figure 7 evaluates the scalability of our proposed approach to answer the *subspace similarity query*, by varying the data size N. Apart from B$^+$-tree and *linear scan*, we also compare our approach with a range query method, modified from NN [31], over subspaces. Specifically, we maintain n sorted lists for n dimensions of data objects, respectively, for sequential scan, and every time we encounter an object, we will calculate its minimum possible distance to query point by underestimating other unseen dimensions (update minimum distance of other objects as well). If any object has the minimum distance higher than query radius, it can be pruned; otherwise, it is the answer in case all dimensions have been seen. Specifically, we test the *wall clock time* over two real data sets, *layout* and *sstock*, by varying N from $5K$ to $34K$ in *layout* data set and from $5K$ to $50K$ in *sstock*, where $[k_{min}, k_{max}] = [10, 20]$, $m = 5$, $n = 64$ $\mu_R = 0.01$ and $\sigma_R = 0.005$. Since NN method needs to update minimum distances of data objects, it incurs high computation cost and thus has the highest *wall clock time*. Moreover, similar to previous results, the *multipivot-based* method outperforms both B$^+$-tree and *linear scan*.

Finally, we did experiments with the same settings under L_p-norms ($p \neq 2$), whose results are similar. Due to space limit, however, we only demonstrate one set of experiments in Figure 8, under L_1-, L_2-, and L_∞-norms, where $[k_{min}, k_{max}] = [10, 20]$, $N = 10K$, $m = 5$, $n = 64$ $\mu_R = 0.01$ and $\sigma_R = 0.005$. The experimental results show that our multipivot-based method can outperform the other three under L_p-norm ($1 \leq p \leq \infty$), due to the effective pruning via scores.

In summary, we have demonstrated through extensive experiments the efficiency and effectiveness of our *multipivot-based* method for *subspace similarity queries*.

VII. Conclusions

Similarity search plays an important role in many applications such as information retrieval, image data analysis, and time-series matching. Previous work usually consider the search problem in the *full* space. In contrast, we propose a novel problem, *subspace similarity search*, that is, given an *arbitrary* subspace with *arbitrary* dimensions, retrieve data



(a) *layout* (b) *sstock*

Fig. 8. Query Efficiency vs. L_p-norms

objects that are similar to a query object in the subspace which is specified by query objects *on demand*. This problem has many practical usages such as in multimedia databases. However, traditional methods to solve the *similarity search* problem in the *full* space cannot be applied in this case. Motivated by this, in this paper, we propose an efficient and effective approach to answer *subspace similarity queries* under L_p-norm for $1 \leq p \leq \infty$. In particular, we assign 1-dimensional scores to each object, which are computed with respect to pivots and used for pruning *false positives* during the *subspace similarity search*. The *entire* procedure is formalized with a cost model, in light of which data are pre-processed such that the *pruning power* during query processing is maximized. Extensive experiments have verified the correctness of our cost model under various parameters, and demonstrated the efficiency and effectiveness of our method under L_p-norm. As our future work, it would be interesting to adapt the *multipivot-based* method to other variations of subspace search (e.g. *nearest neighbor*). Moreover, from our experiments, when $(k_{max} - k_{min})$ is small, high pruning power can be achieved. Thus, another interesting direction is to pre-process data using several small intervals in $[k_{min}, k_{max}]$ separately, and answer subspace queries (with k) with respect to the interval containing k. The resulting solution can handle the case where the full space has very high dimensionality, which may require a cost model to make a trade-off between space and computation cost.

ACKNOWLEDGMENT

Funding for this work was provided by Hong Kong RGC Grant No. 611907 and National Grand Fundamental Research 973 Program of China under Grant No. 2006CB303000.

REFERENCES

[1] Data warehousing and OLAP: a research-oriented bibliography. *http://www.ondelette.com/OLAP/dwbib.html*.

[2] R. Agrawal, C. Faloutsos, and A. N. Swami. Efficient similarity search in sequence databases. In *Proc. 4th Int. Conf. of Foundations of Data Organization and Algorithms*, pages 69–84, 1993.

[3] C. Böhm, S. Berchtold, and D. A. Keim. Searching in high-dimensional spaces: Index structures for improving the performance of multimedia databases. *ACM Comput. Surv.*, 33(3):322–373, 2001.

[4] T. Bozkaya and M. Ozsoyoglu. Distance-based indexing for high-dimensional metric spaces. In *SIGMOD*, pages 357–368, 1997.

[5] T. Bozkaya and M. Ozsoyoglu. Indexing large metric spaces for similarity search queries. *ACM Trans. Database Sys.*, 24(3):361–404, 1999.

[6] E. Chaźez, G. Navarro, R. Baeza-Yates, and J. L. Marroquń. Searching in metric spaces. *ACM Comput. Surv.*, 33(3):273–321, 2001.

[7] L. Chen, M. T. Özsu, and V. Oria. Robust and fast similarity search for moving object trajectories. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 491–502, 2005.

[8] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos. Fast subsequence matching in time-series databases. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 419–429, 1994.

[9] R. F. S. Filho, A. J. M. Traina, C. Traina Jr., and C. Faloutsos. Similarity search without tears: The OMNI family of all-purpose access methods. In *Proc. 17th Int. Conf. on Data Engineering*, pages 623–630, 2001.

[10] K-S. Goh, B. T. Li, and Ed. Chang. Dyndex: a dynamic and non-metric space indexer. In *Proc. 10th ACM Int. Conf. on Multimedia*, pages 466–475, 2002.

[11] C. M. Grinstead and J. L. Snell. Introduction to probability. pages 333–337. AMS, 1997.

[12] A. Guttman. R-trees: a dynamic index structure for spatial searching. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 47–57, 1984.

[13] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *J. Mach. Learn. Res.*, 3:1157–1182, 2003.

[14] X. He. Incremental semi-supervised subspace learning for image retrieval. In *ACMMM*, 2005.

[15] X. He, D. Cai, and P. Niyogi. Tensor subspace analysis. In *NIPS*, 2005.

[16] G. Jovanovic-Dolecek. Demo program for central limit theorem. In *Circuits and Systems*, 1997.

[17] Y. Ke, R. Sukthankar, and L. Huston. An efficient parts-based near-duplicate and sub-image retrieval system. In *ACMMM*, pages 869–876, 2004.

[18] R. Kohavi and D. Sommerfield. Feature subset selection using the wrapper model: Overfitting and dynamic search space topology. In *The First International Conference on Knowledge Discovery and Data Mining*, pages 192–197, 1995.

[19] F. Korn, H. Jagadish, and C. Faloutsos. Efficiently supporting ad hoc queries in large datasets of time sequences. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 289–300, 1997.

[20] M. Koskela, J. Laaksonen, and E. Oja. Use of image subset features in image retrieval with self-organizing maps. In *CIVR*, pages 508–516, 2004.

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

[22] K.P.Chan and A.W-C Fu. Efficient time series matching by wavelets. In *Proc. 15th Int. Conf. on Data Engineering*, pages 126–133, 1999.

[23] H. Lejsek, F. H. Ásmundsson, B T. Jónsson, and L. Amsaleg. Scalability of local image descriptors: A comparative study. 2004.

[24] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, 2004.

[25] R. T. Ng and J. Han. Efficient and effective clustering methods for spatial data mining. In Jorgeesh Bocca, Matthias Jarke, and Carlo Zaniolo, editors, *20th International Conference on Very Large Data Bases, September 12–15, 1994, Santiago, Chile proceedings*, pages 144–155, Los Altos, CA 94022, USA, 1994. Morgan Kaufmann Publishers.

[26] W. Niblack, R. Barber, W. Equitz, M. Flickner, E. Glasman, D. Petkovic, P. Yanker, C. Faloutsos, and G. Taubin. The QBIC project: querying images by content using color, texture and shape. In *Proc. 5th Int. Symp. on Storage and Retrieval for Image and Video Databases*, pages 173–185, 1993.

[27] A. K. H. Tung, R. Zhang, N. Koudas, and B. C. Ooi. Similarity search: A matching based approach. In *VLDB*, pages 631–642, 2006.

[28] E. W. Weisstein. Central Limit Theorem. *http://mathworld.wolfram.com/CentralLimitTheorem.html*.

[29] B-K Yi and C. Faloutsos. Fast time sequence indexing for arbitrary L_p norms. In *Proc. 26th Int. Conf. on Very Large Data Bases*, pages 385–394, 2000.

[30] P. N. Yianilos. Data structures and algorithms for nearest neighbor search in general metric spaces. In *SODA*, pages 311–321, 1993.

[31] M. L. Yiu and N. Mamoulis. Reverse nearest neighbors search in ad-hoc subspaces. In *ICDE*, page 76, 2006.