

# The IGrid Index: Reversing the Dimensionality Curse For Similarity Indexing in High Dimensional Space

Charu C. Aggarwal  
IBM T. J. Watson Research Center  
Yorktown Heights, NY 10598  
charu@watson.ibm.com

Philip S. Yu  
IBM T. J. Watson Research Center  
Yorktown Heights, NY 10598  
psyu@watson.ibm.com

## ABSTRACT

The similarity search and indexing problem is well known to be a difficult one for high dimensional applications. Most indexing structures show a rapid degradation with increasing dimensionality which leads to an access of the entire database for each query. Furthermore, recent research results show that in high dimensional space, even the concept of similarity may not be very meaningful. In this paper, we propose the *IGrid*-index; a method for similarity indexing which uses a distance function whose meaningfulness is retained with increasing dimensionality. In addition, this technique shows performance which is unique to all known index structures; the percentage of data accessed is inversely proportional to the overall data dimensionality. Thus, this technique relies on the dimensionality to be high in order to provide performance efficient similarity results. The *IGrid*-index can also support a special kind of query which we refer to as projected range queries; a query which is increasingly relevant for very high dimensional data mining applications.

## Categories and Subject Descriptors

H.2.8 [Database Management]: Database Applications

## General Terms

Dimensionality Curse, Indexing

## 1. INTRODUCTION

The similarity search problem is defined as follows: for a given target record in multi-dimensional space, find the closest record to it based on some pre-defined distance measure. This technique finds applications in numerous domains such as spatial databases, multimedia systems, data mining, and image retrieval [4, 5]. With the increasing availability of large repositories of very high dimensional data in numerous application domains, it becomes increasingly important to develop efficient query processing and similarity indexing techniques for such data.

A number of techniques such as KDB-Trees, kd-Trees, and Grid-Files are discussed in the classical database literature [26] for indexing multidimensional data. Many of these techniques were initially proposed in the context of low-dimensional spatial applications. Starting with the seminal work of Guttman on R-Trees [18], considerable work has been done on finding multi-dimensional index structures in the database arena. Variants of R-Trees such as  $R^*$ -Trees,  $R^+$ -Trees, and Hilbert R-Trees [7, 21, 27], are able to resolve various kinds of queries more efficiently, by using better packing and split methods to build the tree structure. All of these methods partition the data into ranges which are parallel to the original axis system. Subsequently, methods such as SS-Trees [28] were proposed which do not necessarily partition the data using ranges from the original set of attributes. Other prominent indexes and query processing methods may be found in [5, 20, 24, 28].

The above techniques generally work well for low dimensional problems, though they degrade rapidly with increasing dimensionality, so that each query requires the access of almost all of the data; consequently specialized techniques for high dimensional similarity indexing such as X-Trees, SR-Trees and TV-Trees [11, 22, 23] have been proposed. Because of the inherent difficulty of exact nearest neighbor search for very high dimensional data, some interesting methods have also been proposed for approximate nearest neighbors in these cases [8, 17]. Other relevant methods include the pyramid technique [9], which have been proposed for high dimensional range queries. Despite these indexing and query processing methods, it has been observed that with almost any technique, when the dimensionality is above 15 or 20, then all of the data is accessed. In fact, recent results [6] show that a simple sequential scan performs better than any of the space partitioning methods on uniformly distributed data when the dimensionality is larger than 610.<sup>1</sup> This behavior has been validated with empirical testing for significantly lower dimensionalities [15]. Consequently, a technique called the VA-File [6] was proposed which assumes that a sequential scan is inevitable and tries to build an index by compressing the time required for this sequential scan to between 12.5% – 25% of the data. This is achieved by scanning a compressed representation of the data; some additional time overhead is required for resolving the conflicts due to the information lost in compression. This method shows more promising results than the best

<sup>1</sup>In practice this threshold is well below 610, since the worst case analysis is based on very crude estimations and bounds.

space partitioning methods.

Recent research results show that in high dimensional space, even the concept of proximity may not be very meaningful [12]. These results show that for certain classes of commonly used similarity functions such as the  $L_p$ -norm, the nearest and furthest neighbor are of the same relative distance to the query point for large classes of data distributions. Several additional interesting properties of the  $L_p$ -norm may be found in [1]. The lack of relative contrast in terms of similarity is somewhat undesirable, since it is not clear whether or not the nearest neighbor is meaningful under such circumstances. For many high dimensional data mining applications, even the concept of similarity is a heuristic one, and is not well defined. There is not much consensus and literature on finding distance functions which result in the most meaningful definition of similarity. The common use of the Euclidean distance metric for indexing structures arises from their initial applicability to spatial databases (for which the  $L_2$ -norm has special meaning). For many other high dimensional application domains such as information retrieval (IR), categorical data and market basket data, the importance of designing qualitatively effective and meaningful distance functions using the aggregate statistical behavior of the data has been well understood and appreciated [13, 16, 25]. These techniques have however not been applied to the multidimensional indexing problem in the quantitative domain, where many applications have continued to use the increasingly irrelevant  $L_p$ -norm.

The lack of definiteness in measurement of similarity for high dimensional data has been explored in recent work, where the distance distributions of the data to the query point in different projections are examined in order to identify interesting query-specific features [19]. These identified features are used in order to determine the most similar objects. This technique is called *projected nearest neighbor search* and can be very valuable in understanding dimensional selectivity in the neighborhood of a query point. Since it is a difficult problem to find the best combination of dimensions in the neighborhood of a query-point, the technique is slower than a sequential scan. Thus, this system is very useful for dealing with the quality issue of nearest neighbor search and understanding the data by providing (locally) interesting combinations of dimensions which are discriminatory projections. On the other hand, the technique is not really focussed on the problem of rapid query resolution. In order to provide such capability, we need to develop a meaningful similarity function which can be expressed in closed form, and can be used in conjunction with an effective index. To this effect, we develop a class of distance functions which are somewhat similar to the  $L_p$ -norm in low dimensions, but become increasingly different with greater dimensionality. In addition, the development of a closed form distance function has implications for a wide variety of other difficult high dimensional data mining problems which rely on proximity concepts.

This paper shows how the aggregate summary behavior of the data may be used in order to design a qualitatively effective function which computes distances in a more flexible way than the straightforward summation of distances over all dimensions. We will provide a theoretical analysis of the

meaningfulness of this distance function, which shows that it does not suffer from the lack of discrimination caused by high dimensional sparsity as discussed in [12]. This theoretical analysis is supplemented with qualitative empirical evidence which suggests that this method leads to more meaningful results. A key aspect of this class of distance functions is that it turns out to be *index-friendly*; we will design a class of index structures (*IGrid* and *IGrid<sup>+</sup>*) based on these functions. These index structures show the surprising behavior of the reversal of the dimensionality curse in terms of performance; in other words the percentage of data accessed is inversely proportional to the dimensionality. This is in contrast to the behavior of almost all known indexing structures and algorithms.

This paper is organized as follows. The remainder of this section discusses the background and definition of the similarity function. In section 2, we introduce the *IGrid* index. A theoretical analysis is discussed in section 3. Section 4 discusses how inter-attribute correlations may be used in order to improve the similarity calculations. In section 5, we will discuss the application of the *IGrid*-index to a special kind of query which we refer to as projected range queries. Empirical results are presented in section 6. Section 7 discusses an overview of the results of this paper.

## 1.1 Contributions of this paper

This paper discusses an indexing technique for similarity search which is both qualitatively effective and shows performance efficient behavior. To this effect, we show that as the dimensionality increases it becomes more important to compute similarity functions in more flexible ways than is the case with the  $L_p$ -norm. The discrimination behavior of the similarity function is further enhanced by using the aggregate inter-attribute correlation behavior. We provide empirical evidence that this enhanced similarity measure is more meaningful than the euclidean metric for real data sets in high dimensional data mining applications. At the same time, this similarity function can be indexed effectively by a method called the *IGrid*-index which shows the surprisingly pleasant behavior of improved performance with increasing dimensionality. This behavior is rooted in the fact that the index and the distance measure exploit the greater statistical information available in high dimensional data in a more effective way. Thus, it solves the problem of meaningful and performance efficient similarity indexing in one unified framework. We also show how the *IGrid*-index can be utilized for effective resolution of a special kind of query called the projected range query; a query which becomes more relevant than the full dimensional query for very high dimensional applications.

## 1.2 Background and Motivation

In this section, we will illustrate some of the recent results [12] which show that in high dimensional space, the ratio of the relative distances of the different points to a given target converges to one. In order to do so, we will establish certain notations and definitions:

$d$  : Dimensionality of the data space

$N$  : Number of data points

$X_d$  : Data point from  $d$ -dimensional data distribution  $\mathcal{Q}_d$

$dist_d(X_d)$  : Distance of  $X_d$  from the origin  $(0, \dots, 0)$

$Dmin_d, Dmax_d$  : Nearest/Farthest distance of  $N$  points to origin  
 $E[X], var[X]$  : Expected value/variance of random variable  $X$   
 $Y_d \rightarrow_p c$  :  $Y_d$  converges in probability to  $c$  as  $d \Rightarrow \infty$

**THEOREM 1. Beyer et al. [12]** *If:*  
 $\lim_{d \rightarrow \infty} var \left( \frac{dist_d(X_d)}{E[dist_d(X_d)]} \right) = 0$  , *then:*  
 $\frac{Dmax_d - Dmin_d}{Dmin_d} \rightarrow_p 0$ .

**PROOF.** See [12] for proof of a more general version of this result.  $\square$

This result shows<sup>2</sup> that under certain conditions on the distance functions and data distribution, the difference between the maximum and minimum distances to a given target (the origin<sup>3</sup> in this case) does not increase as fast as the nearest distance to any point in high dimensional space. This makes a proximity query meaningless and unstable because there is poor discrimination between the nearest and furthest neighbor. Thus, in very high dimensional space a small relative perturbation of the target point in a direction away from the nearest neighbor could easily change the nearest neighbor into the furthest neighbor.

These results are valuable not just from the perspective of meaningfulness but also from the performance perspective of indexing. Most indexing methods work by using some kind of partitioning (hierarchical or flat) of the data set. This partitioning is then used in order to perform effective pruning of the data set. The key idea is that if it is already known that some neighbor  $X$  is close enough to the target, then one can prune away an entire partition by showing that the optimistic distance bound to that partition is no better than the distance to  $X$  [24]. The results in [12] show that in high dimensionality the nearest and farthest neighbor have very similar relative distances to the target. Consequently, the optimistic bounds used by most index structures are usually not sharp enough for any kind of effective pruning in partition based methods. These results also show how approximate nearest neighbor indexes [8, 17] are affected by increasing dimensionality; when the ratio of nearest to furthest distance is almost even, then the quality of an  $\epsilon$ -approximate solution which is within a pre-specified ratio  $1 + \epsilon$  of the best solution may be as bad as the furthest neighbor. Thus, the excellent  $\epsilon$ -dependent performance results of these techniques come at an increasing qualitative price with greater dimensionality and fixed  $\epsilon$ . In addition, these results expose the meaningfulness issues of a vast number of problems such as clustering which rely on the concept of proximity; some recent results [2, 3] show how these problems can be understood more effectively by using points and dimensions in a more flexible way.

<sup>2</sup>Other results [8] also show that under certain different pre-conditions, the meaningfulness behavior is not quite as bad. However, some evidence of the degradation has been demonstrated by empirical testing for real distributions [12].

<sup>3</sup>We have used the origin as the target in this case in order to simplify presentation. See [12] for a more general statement of results.

### 1.3 The Similarity Function

One of the reasons for the lack of discrimination between the nearest and furthest neighbor is the fact that for every pair of points there are dimensions with varying distances to the corresponding values in the target. The dominant components of distance functions such as the Euclidean metric are the dimensions on which the points are farthest apart; for the particular case of high dimensional data, this results in very poor measurement of similarity. This is because when the dimensionality is high, even the most similar records are likely to have a few feature values which are well separated because of noise effects and sparseness of the data; the exact degree of dissimilarity on these few noisy dimensions will determine the order of distances to the target. In general, for a given feature, we expect the values for two randomly picked records to be reasonably well separated (average separation along that range); there is no interesting statistical information in this fact. For distance functions such as the  $L_p$ -norm, the results of [12] show that the averaging effects of the different dimensions (many of which are noisy) start predominating with increasing dimensionality. A different and complementary view of similarity would be one in which a predefined proximity threshold is defined for each dimension, and the overall similarity is defined *both* by the number and quality of similarity along the dimensions on which the two records are more proximate than this threshold. Thus, the similarity function is directly affected by the number of dimensions which have this interestingly high level of proximity, and beyond a certain quality threshold, the exact degree of dissimilarity on a given dimension is not considered relevant. Since the meaningfulness problem is sensitive to the data dimensionality, the criterion for picking this proximity threshold is also dependent on the data dimensionality. Specifically, it is derived using a theoretical analysis of meaningfulness; we will revisit this issue in a later section.

The analysis indicates that this definition of similarity continues to retain its meaningfulness for higher dimensionalities in terms of the relative contrasts in distances to a given target. In addition, we will provide evidence using several high dimensional real data sets that the quality of the nearest neighbor returned by such a technique is as good or better than that provided by the  $L_p$ -norm. At the same time, it is possible to design index structures which are able to prune away a large part of the data while searching for a closest neighbor to the target.

We will first define a simple distance (similarity) function in which higher numbers imply greater similarity; later we will show how to modify this distance function in order to use proximity thresholds on the individual dimensions. Let  $X = (x_1, \dots, x_d)$  and  $Y = (y_1, \dots, y_d)$  be two sets of coordinates in  $d$ -dimensional space, so that  $(x_i, y_i) \in [l_i, u_i]$  for some set of lower and upper bounds  $l_i$  and  $u_i$ . Then, the distance (similarity) function  $IDist(X, Y)$  between  $X$  and  $Y$  is given by:

$$IDist(X, Y) = \left[ \sum_{i=1}^d \left( 1 - \frac{|x_i - y_i|}{u_i - l_i} \right)^p \right]^{1/p} \quad (1)$$

The presence of  $u_i - l_i$  in the denominator studentizes (or normalizes) the distances with respect to the different ranges

of the coordinates.

In order to incorporate the concept of proximity thresholding in the similarity function, we discretize the data into several ranges. Specifically, we assume that each dimension is divided into  $k_d$  *equi-depth*<sup>4</sup> ranges. Each of these is a contiguous range of values, such that a given range contains a fraction  $1/k_d$  of the total number of records. Specifically, we denote the  $j$ th range for dimension  $i$  by  $\mathcal{R}[i, j]$ . In order to emphasize the dependency (to be determined later) of  $k_d$  on the data dimensionality, we have used the dimensionality  $d$  in the subscript.

Let  $X = (x_1, \dots, x_d)$  and  $Y = (y_1, \dots, y_d)$  be two records. Then the set of dimensions on which the two records are similar are those which share the same ranges. Thus, for dimension  $i$ , if both  $x_i$  and  $y_i$  belong to the same range  $\mathcal{R}[i, j]$ , then the two records are said to be in *proximity* on dimension  $i$ . The entire set of dimensions on which the two records lie in the same range is referred to as the *proximity set*. Let  $\mathcal{S}[X, Y, k_d]$  be the proximity set for two records  $X$  and  $Y$  for a given level of discretization. Furthermore, for each dimension  $i \in \mathcal{S}[X, Y, k_d]$ , let  $m_i$  and  $n_i$  be the upper and lower bounds for the corresponding range in the dimension  $i$  in which the records  $X$  and  $Y$  are in proximity to one another. Then, for a given pair of records  $X$  and  $Y$  and a level of discretization  $k_d$ , the similarity between the records is given by:

$$PIDist(X, Y, k_d) = \left[ \sum_{i \in \mathcal{S}[X, Y, k_d]} \left( 1 - \frac{|x_i - y_i|}{m_i - n_i} \right)^p \right]^{1/p} \quad (2)$$

Note that the value of the above expression will vary between 0 and  $|\mathcal{S}[X, Y, k_d]|$ , since each individual expression in the summation lies between 0 and 1.

The above use of the similarity function guarantees a non-zero similarity component only for those dimensions, in which the two records are proximate enough. The use of equi-depth partitions ensures that the probability that two records have a component in the same partitions given by  $1/k_d$ . Thus, on the average the above summation is likely to have  $d/k_d$  components. For more similar records, the number of such dimensions will be greater, and each such individual component is also likely to contribute more to the similarity value. The above function leads to the ignoring of the exact degree of dissimilarity on the distant dimensions: we will see from our empirical tests, that for the case of high dimensional data this creates a sparsity/noise reduction which outweighs the effects of information loss.

## 1.4 Use of Equi-Depth Ranges

The use of equi-depth ranges as opposed to equi-width ranges has considerable significance. In real applications, the data may be distributed in a very non-uniform way across the different attributes. As a result, simple distance functions such as the  $L_p$ -norm fail to take the aggregate behavior of the data into account while measuring similarity. The use

<sup>4</sup>In equi-depth ranges, each range contains an equal number of records. In equiwidth ranges, each range contains a similar length of values covered. The reason for picking equi-depth ranges will become clear soon.

of equi-depth partitions ensures that when a particular region is very dense, then the categorical range value is much smaller. The idea here is that the proximity of two records for a given feature value should not be treated in a uniform way across the entire range, but it should be based on how close these features are with respect to the behavior of the *entire data set*. Thus, the use of equi-depth ranges creates an implicit normalization of the data by taking into account the aggregate behavior in that range. This kind of normalization is similar in spirit to idf-normalization techniques [25] used in IR applications. Another advantage of the use of equi-depth ranges is the ability to predict and control the indexing behavior using the discretization parameter; an issue which we will revisit in later sections.

## 2. THE IGRID INDEX

The use of ranges in the similarity function provides considerable advantages in the use of an inverted index in order to perform the similarity calculations. The *IGrid*-index (Inverted Grid index) is based on the use of the inverted index on a grid representation of the data. The division of the data into ranges automatically creates a grid structure, in which each record belongs to a particular cell. We create an inverted representation of the data as follows:

- (1) For each range  $j$  for each dimension  $i$  we maintain the lower and upper bounds of the range  $\mathcal{R}[i, j]$ .
- (2) For each range for each dimension, we maintain a list of the record identifiers which lie in that range. Note that the use of  $k_d$  equi-depth partitions along each dimension ensures that the length of each inverted list will be  $N/k_d$ , where  $N$  is the total number of records.
- (3) Along with each entry in the list of record identifiers, we keep the actual coordinate for the corresponding dimension in that record.

Note that the size of the inverted representation of the data is comparable to the size of the original database. Once we have constructed this inverted representation of the data from the grid structure, the calculation of similarity is very simple. For the target record, we find the appropriate range for each dimension. Then we examine the corresponding  $d$  lists. Since there is a total of  $d \cdot k_d$  lists, the percentage of data accessed is small when the value of  $k_d$  is large. The method of calculating similarity is very similar to the method often used in information retrieval applications in using the inverted representation for calculating the similarity based on word frequencies [25].

Let  $T = (t_1 \dots t_d)$  be a given target record, and let  $m_i$  and  $n_i$  be the upper and lower bounds for the corresponding range for dimension  $i$ . We say that a record is *touched* by the target, if it lies on at least one of the  $d$  lists corresponding to the ranges in the target record. The hash table maintains a record of the similarity value of all the records which are touched by the target. An entry is added to the hash table the first time it is encountered during the examination of the data. While examining each entry on the lists with corresponding coordinate value  $x'$ , we calculate the contribution of that component (which is equal to  $\left( \frac{m_i - n_i - |t_i - x'|}{m_i - n_i} \right)^p$  as defined in Equation 2) to the similarity function and keep adding that value to the appropriate entry in the hash table. For the purpose of our results, we used the value  $p = 1$ . At

the end of the process, the hash table entry with the largest similarity value is the closest neighbor. The method can easily be generalized to find the multiple nearest neighbors.

An observation to be kept in mind is that the inverted representation always returns the IDs of the records as opposed to the records themselves. However, it is also assumed that any reasonable query would have an output which is significantly smaller than the original database (for example, for a similarity query one may ask for the most similar record out of an enormous database of records). Thus, a small amount of constant time is required in order to access the records from the database using the IDs of the returned records (using the natural index by ID), an overhead which is independent of database size and dependent only on the size of the output. If it may be assumed that the users are interested only in online queries which have responses that are small enough for interactive search and exploration, then this overhead is (asymptotically) negligible for larger and larger collections of data.

## 2.1 Performance

The performance of this technique improves with increasing  $k_d$ ; a value which we will determine by meaningfulness considerations. This is because exactly  $d$  out of the  $d \cdot k_d$  of the lists are accessed, and each list is of the same size. Thus, a fraction  $1/k_d$  of the entire inverted index is accessed. However, in order to make a fair comparison, we would also need to compare the inverted index size to that of the original database. Note that each Record Identifier occurs on exactly  $d$  lists; therefore the total number of identifiers is  $N \cdot d$ . Along with each identifier in an inverted list, we store the actual value for the corresponding feature of that record. This requires the storage of another  $N \cdot d$  values (in all). Thus, the total space requirement of the index is  $2 \cdot N \cdot d$ , whereas the original database requires  $N \cdot d$  space. This corresponds to a storage (and hence performance) overhead<sup>5</sup> of 100%. In this case, since the overhead is independent of the value of  $k_d$ , it follows that the larger the value of  $k_d$  used, the better the performance. In the next section, we will discuss how  $k_d$  is determined by meaningfulness considerations in a given dimensionality  $d$ .

## 2.2 Avoiding Hash Table Overflow

An important issue in this technique is to avoid the hash table overflow in the process of similarity calculations. This is because the potential number of hash table entries is likely to be very large, and the hash table is always maintained in memory. In order to actually implement the system, the entire database is not indexed as one entity. Instead, the database is divided into chunks. An inverted index is built separately for each chunk, and the most similar record is found one by one for each chunk. At the end, the best match among all chunks is reported. Note that this division of database into chunks does not change the performance behavior in terms of disk accesses from the inverted index, but it eliminates the possibility of hash table overflows. The size of each chunk is determined by  $k_d$  times the total number

<sup>5</sup>We do factor in these overheads in the performance results of the empirical section. We will see that in spite of these overheads, the performance of the index is substantially better than competing methods.

of entries in the hash table. This eliminates the possibility of an overflow because exactly  $1/k_d$  of the entries in the inverted index are accessed without counting repetitions.

## 3. THEORETICAL DETERMINATION OF THE PROXIMITY THRESHOLD

Note that the similarity function is highly influenced by the number of dimensions in which the record lies in the same range as the target. In general, we expect that the nearest record will have large cardinality of the proximity set. It is insightful to look at a crude approximation of the similarity function in which only the cardinality of the proximity set is used for the purpose of measuring similarity. The discrimination behavior of this function will provide considerable insight into the case when the more refined method of using the actual coordinates are applied.

In order to analyze the discrimination behavior, let us consider two records  $X$  and  $Y$  which are picked from uniformly randomly distributed data. From the perspective of indexing and meaningfulness degradation, uniformly distributed data is the most difficult case with increasing dimensionality. Then, on any given dimension the event that the two records lie in the same range is a bernoulli random variable with success parameter  $1/k_d$ . Specifically, let us define the bernoulli random variable  $M_i$  as follows:

$$\begin{aligned} M_i = 0 & : x_i, y_i \text{ not in same bucket for dimension } i \\ M_i = 1 & : x_i, y_i \text{ in same bucket for dimension } i \end{aligned}$$

The random variable  $M_i$  has a mean of  $1/k_d$  and a variance of  $(1/k_d) \cdot (1 - 1/k_d)$ . Let  $L$  be the random variable, which is the sum of this bernoulli variable over the  $d$  dimensions. Thus, we have  $L = \sum_{i=1}^d M_i$ . Note that if we assume that  $X$  and  $Y$  are drawn from uniform distributions, then the values of  $M_i$  will also be independent and identically distributed. In such a case, when the dimensionality is high, the distribution of  $L$  approaches a normal distribution which has a mean of  $\mu(d, k_d) = d/k_d$ , and a standard deviation of  $\sigma(d, k_d) = \sqrt{(d/k_d) \cdot (1 - 1/k_d)}$ . The pre-condition of Theorem 1 when interpreted in this context would imply that the meaninglessness behavior is exhibited when  $\lim_{d \rightarrow \infty} \sigma(d, k_d)/\mu(d, k_d) = 0$ . Consequently, for the purpose of this analysis, we will analyze the behavior of the ratio  $\sigma(d, k_d)/\mu(d, k_d)$  in order to measure the effects of the discretization parameter on this result. The higher this ratio, the greater the level of discrimination among the distances to the different records. Using the above analyzed values of  $\sigma(d, k_d)$  and  $\mu(d, k_d)$ , we obtain  $\sigma(d, k_d)/\mu(d, k_d) = \sqrt{(k_d - 1)/d}$ . This value increases with the discretization parameter  $k_d$ . Thus, an increase in  $k_d$  improves the meaningfulness by ignoring the exact degree of dissimilarity on the sparse dimensions; on the other hand, picking  $k_d$  too large may result in loss of information along with noise reduction. Such a tradeoff is handled by picking the minimum value of  $k_d$  dependent on  $d$  so that the pre-condition of Theorem 1 is violated. In other words, we would like:

$$\lim_{d \rightarrow \infty} \sqrt{(k_d - 1)/d} > 0 \quad (3)$$

This implies that we should pick  $k_d$  which is at least linearly dependent on  $d$ . Consequently, for the purpose of this paper,

we will use  $k_d = \lceil \theta \cdot d \rceil$ , where  $\theta$  is some constant. Note that picking  $\theta = 1$  results in a distance function which is exactly similar to the  $L_p$ -norm for 1-dimensional data, but becomes increasingly different with increasing dimensionality. Similarly, picking  $\theta = 0.5$  and  $p = 1$  creates a distance function which is similar to the  $L_1$ -norm for 2-dimensions but it becomes different in higher dimensionalities. Assuming that  $L_p$ -norm distance functions work well for low dimensional problems, these observations provide sufficient guidance to a good choice of  $\theta$  to lie in the region of 0.5 or 1.

For this value of the discretization parameter the value of  $\sigma(d, k_d)/\mu(d, k_d)$  remains almost constant (and in fact increases slightly) with increasing dimensionality. As we shall see later in the empirical section, this also has a direct effect on the meaningfulness behavior of the similarity function which does not vary much with increasing dimensionality. Since we showed earlier that the performance of the *IGrid*-index improves with increasing  $k_d$ , a choice of  $k_d = \lceil \theta \cdot d \rceil$  results in an index structure for which the performance improves with increasing dimensionality.

#### 4. HEURISTIC IMPROVEMENTS WITH ATTRIBUTE CORRELATIONS

In high dimensional space, many of the attributes are correlated with one another. Inter-attribute correlations have often been used for designing distance functions in categorical domains where there is no natural ordering of attribute values. In such cases, the use of inter-attribute summary information provides the only possible insight into the similarity of objects by examining whether commonly co-occurring inter-attribute values are present in the two objects [13, 16]. This insight is equally relevant even for quantitative domains of data where a natural ordering of attribute values exists. The use of aggregate data behavior in order to measure similarity becomes more important for high dimensional data, where there may be considerable redundancies, dependencies, and relationships among the large number of attributes [10]. Since a lot of the proximity information may be hidden in the aggregate summary behavior of the data, the use of the linearly separable  $L_p$ -norm may be a poor representation of the similarity, when considered in light of the aggregate statistical behavior. We note here that some data domains such as text factor the correlation behavior indirectly into the distance function by using data transformation techniques such as Latent Semantic Indexing [14].

The first step is to determine pairs of attribute ranges which are strongly correlated with one another. Let us define one new pseudo-attribute corresponding to each range. Thus, there are a total of  $k_d \cdot d$  pseudo-attributes, each of which corresponds to a range from the original set of attributes. Let us denote these pseudo-attributes by  $a_1, a_2, \dots, a_{k_d \cdot d}$ . A pseudo-attribute is a 0-1 value indicating whether or not the corresponding attribute in the record contains that range. Exactly  $d$  of the pseudo-attributes take on the value of 1, whereas the others take on the value of 0. Thus, in terms of this new representation, a given record  $Z = (z_1 \dots z_d)$  can be expressed as the set  $\{i_1, \dots, i_d\}$ , where, for each  $r \in (1, d)$ ,  $a_{i_r}$  takes on the value of 1 for the record  $Z$ . For each pair of pseudo-attributes  $a_i$  and  $a_j$ , we calculate the corresponding support. The support of a pair of pseudo-attributes is equal to the percentage of records which take on the value of 1 for

both of these pseudo-attributes. Thus, this value is specific to the aggregate behavior of the entire data set which is being indexed. Let us denote this support by  $s_{ij}$ . The larger the value of this support, the greater the level of correlation between these pairs of pseudo-attributes. Thus, if we have a pair of records  $X$  and  $Y$ , such that  $X$  contains  $a_i$  and  $Y$  contains  $a_j$ , and  $s_{ij}$  is high, then this is evidential of the similarity between  $X$  and  $Y$ . Since there are a total of  $k_d \cdot d$  pseudo-attributes, a total of  $k_d^2 \cdot \binom{d}{2}$  such support values are computed. We pick a fraction  $f$  of the largest such values of the inter-attribute correlation. These are the strongly connected pairs of components.

Let  $X = (x_1, \dots, x_d)$  and  $Y = (y_1, \dots, y_d)$  be two sets of records. Then, let  $\{x_{i1} \dots x_{id}\}$  and  $\{y_{i1} \dots y_{id}\}$  be the indices of the corresponding  $d$  pseudo-attributes which take on the value of 1. Then, the similarity between the records  $X$  and  $Y$  is equal to the number of strongly connected pairs among the pseudo-attributes in  $X$  and  $Y$ . Thus, if the inter-attribute correlation similarity is denoted by  $CIDist(\cdot, \cdot)$ , then we have:

$$CIDist(X, Y) = \text{Cardinality of } S \text{ where:} \\ S = \{(x_{ip}, y_{iq}) : a_{x_{ip}} \text{ and } a_{y_{iq}} \text{ are strongly connected}\}$$

The total similarity between two records is the sum of the proximity-threshold based similarity and correlation-based similarity. Thus, we have  $Dist(X, Y) = PIDist(X, Y) + CIDist(X, Y)$ . Each of the two records  $X$  and  $Y$  contains exactly  $d$  pseudo-attributes. Thus, there are a total of  $d \cdot (d-1)$  combinations of these pseudo-attributes, from records  $X$  and  $Y$ . Since only a fraction  $f$  of these possibilities are strongly connected on the average, it means that if the data is uniformly distributed, the contribution of  $CIDist(\cdot, \cdot)$  to the similarity function is likely to be  $f \cdot d \cdot (d-1)$ . Recall that the contribution of the  $PIDist$  component of the similarity function is of the same order of magnitude as the proximity set, which in turn is again expected to be  $d/k_d$ . Consequently, by choosing  $f = c/((d-1) \cdot k_d)$ , we obtain a similar order of magnitude for the inter-attribute correlations as well. The exact value of the constant  $c$  will determine the weightage given to the correlation component in the similarity measure, though the exact analysis of the process for finding a value of  $c$  which results in the most meaningful notion of similarity is beyond the scope of this paper. For the purpose of our experiments, we found  $c = 1$  to be appropriate.<sup>6</sup>

#### 4.1 The *IGrid*<sup>+</sup>-index

The inverted representation also allows for an efficient calculation of correlation-based similarity. For each of the  $k_d \cdot d$  pseudo-attributes, we maintain the lists of strongly connected pseudo-attributes. Note that since there are a total of  $f \cdot k_d^2 \cdot \binom{d}{2} = c \cdot d \cdot k_d/2$  strong connections, it follows that the size of the list for each of the  $d \cdot k_d$  pseudo-attributes is  $c$  on the average. We shall refer to these lists as the adjacency lists for each pseudo-attribute. Since the size of each such list is so small (typically less than 5, when  $c$  is chosen to be 1), these lists can be maintained in main memory.

<sup>6</sup>Since the expected value of  $PIDist(X, Y)$  is of the same order as the size of the proximity set, a value of  $c = 1$  balances  $PIDist(\cdot, \cdot)$  and  $CIDist(\cdot, \cdot)$  approximately.

Thus, the memory requirement for maintaining these adjacency lists is of the order of  $c \cdot d \cdot k_d/2$ . The new expanded index with these lists will be denoted as *IGrid*<sup>+</sup>. The same hash table method is used in order to perform the similarity computations in this case also except that we also need to access all the inverted lists for all the pseudo-attributes which are strongly connected to the pseudo-attributes in the target, and add one to the corresponding hash table entries for those records. Thus, for each pseudo-attribute, we access the adjacency lists of this pseudo-attribute, and find all the other pseudo-attributes which are strongly connected. Then, the inverted lists of these strongly connected pseudo-attributes are accessed. This will result in a performance overhead factor of at most  $(1 + c)$ , since for each pseudo-attribute in the target, an additional  $c$  inverted lists may need to be explored on the average. Thus, when  $c$  is chosen to be 1, the total performance overhead is only a factor of 2. Since we know that the disk access percentage is inversely proportional to dimensionality, it follows that for very high dimensional data such a constant factor would always be offset by the asymptotic scalability behavior with increasing dimensionality.

## 4.2 Edge Effects of Grid Discretization

The process of discretizing into ranges has considerable edge effects because two adjacent intervals will contain values which are very close to one another. These edge effects can be eliminated by using a second level of discretization, where each of the  $k_d$  discretized intervals are further subdivided into  $l$  equi-depth ranges. The inverted list for each of the  $l$  ranges is maintained separately. Thus, in this case there will be a total of  $k_d \cdot l$  inverted lists, each of which contain  $N/(k_d \cdot l)$  Record Identifiers. For each target record, first the most refined ranges corresponding to them are found, and then the  $\lceil (l-1)/2 \rceil$  inverted lists on either side of each of these refined ranges are explored. The resulting method ensures that for each attribute in the target, the proximity threshold explored on either side is symmetric. This eliminates the edge effects associated with discretization. The larger the value of  $l$  picked, the less the edge effects; from practical considerations we found the use of  $l = 3$  sufficient. The amount of data accessed by the technique is not affected by this further level of discretization; the number of lists explored is multiplied by a factor of  $l$ , whereas the size of each of those lists is reduced by the same factor.

## 5. PROJECTED RANGE QUERIES

The primary focus of this paper is for developing an index structure which shows effective performance and qualitative behavior for similarity search in high dimensional data. However, the applicability of this technique extends to range queries in low dimensional projections; a kind of query which becomes increasingly relevant for high dimensional data.

The traditional method of performing range queries specifies full dimensional ranges (a lower and upper bound for each dimension), and uses these in order to find the best match. As the dimensionality increases, it becomes increasingly unlikely that all ranges are relevant in a query. For example, one may wish to specify only the ranges for a small number of dimensions (say 3 or 4), and for the remaining dimensions, it is assumed that the entire range of values ought to be considered. For such queries, traditional indexes such as

the pyramid technique are no longer applicable, since they lead to the access of all the data.

The grid structure and inverted representation of the data gives an easy technique for resolving such queries as well. This is because for each projected range specified by the user, one needs to examine only those lists whose ranges have a non-zero intersection with the user-specified range. For a given dimension, the union of all the records in the corresponding equi-depth attribute grid ranges are relevant. Then the intersection of the records for the different projected dimensions specified by the user results in the relevant sets of records.

In order to consider the advantages of such a technique, let us consider the case of 1000-dimensional data, in which the level of discretization  $k_d$  has also been chosen to be 1000. Let us consider an example in which a user picks 4 of the dimensions, and for each dimension, picks a range which results in the access of a fraction  $q = 0.1$  of the inverted lists for that dimension. Thus, as a result the total fraction of the inverted index accessed will be equal to  $0.1 \cdot 4/1000$ . This is only equal to 0.04% of the data. In fact, if we assume that for most practical applications, users are only likely to pick a small number of constant dimensions for projected range queries irrespective of the data dimensionality, then the performance of the range query will also be inversely proportional to dimensionality. More specifically, it is easy to verify using the arguments similar to those above that the fraction of the index accessed is equal to  $r \cdot q/d$ , where  $r$  is the (small) number of projected dimensions specified by the user,  $q$  is the *average* fractional specificity of each range (percentage of inverted lists accessed for that dimension), and  $d$  is the total number of dimensions.

## 6. EMPIRICAL RESULTS

In this section, we will discuss the empirical results which show that the *IGrid*-index is a meaningful and performance efficient technique for performing similarity search in very high dimensional data. In particular, since our technique changes the criterion for measuring similarity, it is very important to provide some understanding of the quality of the nearest neighbor found. This presents considerable challenges because of the inherent difficulty in measuring quality and meaningfulness directly.

In order to measure the qualitative performance, we used a technique which we refer to as the *class stripping technique*. We obtained some of the data from the UCI machine learning<sup>7</sup> repository. The data was first cleaned in order to take care of missing values, categorical attributes and non-continuous values. These data sets correspond to classification problems consisting of a set of feature variables and a special variable which is designated as the class variable. We stripped off the class variables from the data set and found the  $\gamma = 5$  nearest neighbors to each of the records in the data set using different similarity methods. In each case, we tested the number of records which matched with the class variable of the target record. If a similarity method is poor in discriminatory power, then it is likely to match unrelated random records and the class variable matching is also likely

<sup>7</sup><http://www.cs.uci.edu/~mlearn>

Table 1: Meaningfulness Behavior of the nearest neighbor

Data Set (Dimensions)	Random	Euclidean	<i>PIDist (IGrid)</i>	<i>PIDist + CIDist (IGrid<sup>+</sup>)</i>
Mechanical Analysis (8)	60	293	354	386
Musk (160)	65	255	636	671
Breast Cancer (14)	1499	2535	2619	2671
Segmentation (19)	144	688	755	802
Ionosphere (34)	926	1371	1538	1606

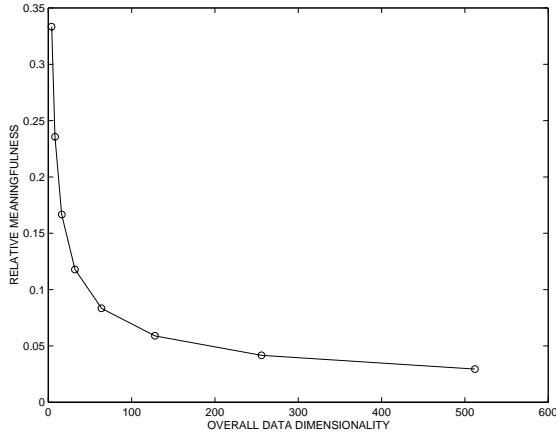


Figure 1: Meaningfulness Behavior with increasing dimensionality ( $L_2$ -norm)

to be poor. This kind of behavior would be exhibited by a random (see first column of Table 1) distance function in which the distance between two records is a uniformly distributed random number. However, when the nearest neighbor is more meaningful in the feature space, it is also likely to match the class variable closely. We concede that these results are evidential in nature, since the exact relationship between the feature variables and class variable is unknown for these data sets. However, a consistent improvement on the class variable accuracy by using the new measure for locality in the feature space does tend to be strong evidence for meaningfulness of the nearest neighbor, since the nearest neighbor was found without any information about the class variable. The above results were obtained using  $\theta = 1$  and  $c = 1$ .

As we can see from Table 1, our similarity measures significantly increase the nearest neighbor accuracy; the addition of correlation based measures improves the overall behavior even further. Since the traditional indexes are designed with respect to the Euclidean distance metric; this shows that *IGrid* index will perform qualitatively at least as well as other indices which rely on traditional distance norms.

Another interesting examination would be to test how the meaningfulness measure defined in [12] varies with increasing dimensionality. In order to do so, we generated uniform random distributions of  $N = 100$  points in increasing dimensionality, and tested both our similarity function and the euclidean distance metric for meaningfulness. The meaningfulness ratio was defined<sup>8</sup> by  $\frac{D_{max_d} - \bar{D}_{min_d}}{D_{avg_d}}$ . This is

<sup>8</sup>Note that in the euclidean distance metric, lower numbers imply greater similarity, whereas in our metric higher num-

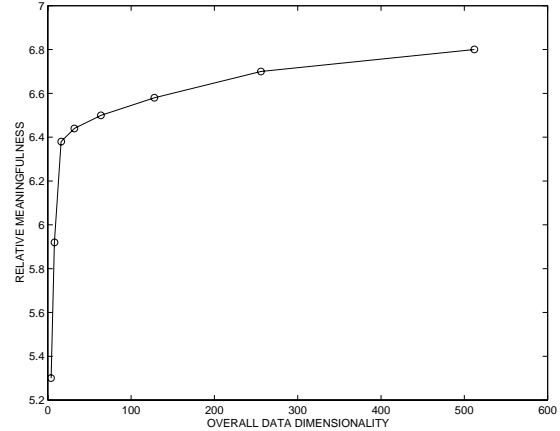


Figure 2: Meaningfulness Behavior with increasing dimensionality (PIDist)

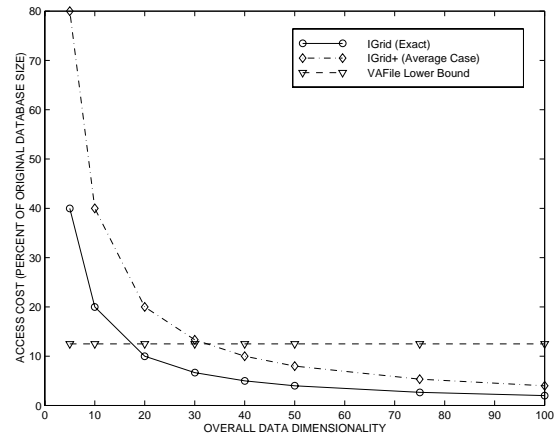


Figure 3: Performance Scalability (Increasing Dimensionality)

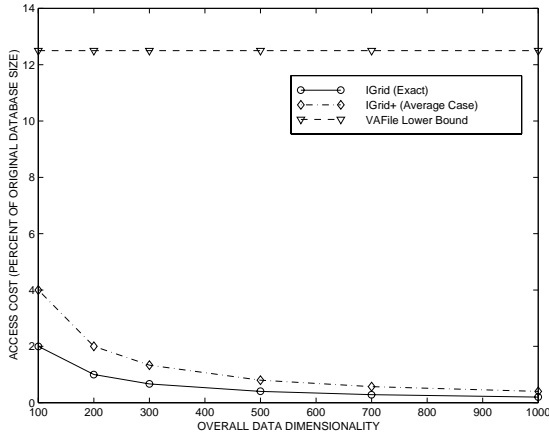


Figure 4: Performance Scalability (Increasing Dimensionality)

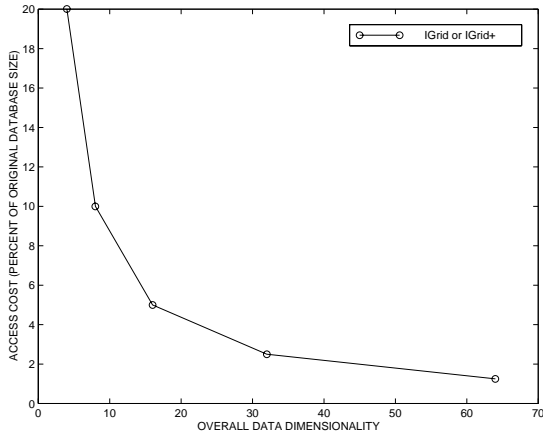


Figure 5: Projected Range Query Performance (w.r.t. Data Dimensionality)

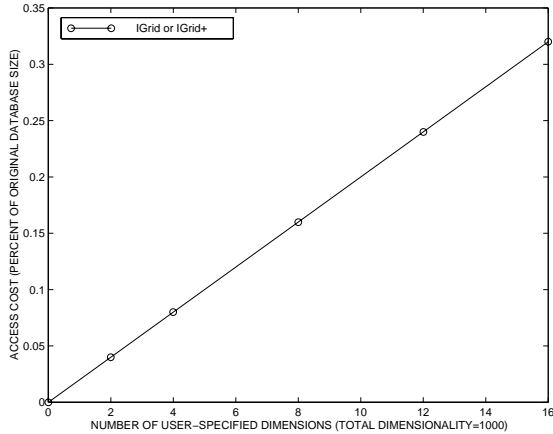


Figure 6: Projected Range Query Performance (w.r.t. User-specified dimensionality)

similar to that introduced in Section 1.2, except that we use the average distance value in the denominator for the sake of robustness. This ratio converges to 0 with increasing dimensionality for large classes of data distributions [12]. An example of such a metric is the Euclidean distance metric for which the meaningfulness ratio is illustrated in Figure 1. However, the *PIDist* function retains its meaningfulness (see Figure 2) with increasing dimensionality because it intentionally ignores the distant and noisy dimensions to the target while measuring similarity. As predicted by our theoretical analysis, the meaningfulness actually improves slightly with increasing dimensionality. This is also guaranteed by our choice of discretization parameter which results in the violation of Beyer's pre-condition for meaningfulness. As evident from the results on the real data, this actually improves the quality of similarity by measuring the number and quality of similarity of the limited number of dimensions on which the two records are very similar, rather than letting the sparse and distant dimensions dominate the similarity function. When the dimensionality is high, this turns out to be a more robust measure of similarity than traditional distance norms.

## 6.1 Performance of Similarity Queries

In the previous subsection we discussed the qualitative behavior of the nearest neighbor in terms of meaningfulness. In this section, we will discuss the performance of the nearest neighbor method in terms of the percentage of data accessed. An important observation is regarding the space overhead involved in the inverted representation of the data. The inverted representation of the data requires the storing of the ID values on each list, as well as the exact coordinate for the corresponding dimension in that ID value. Since each record occurs on exactly  $d$  of the  $d \cdot k_d$  lists, it follows that the total space required to store the inverted representation is  $N \cdot d$ ; exactly the same as the original database representation. However, the storage of both the coordinate value and Record Identifier for each entry in the inverted representation requires an additional 100% overhead. This overhead is significant from the performance point of view, since greater storage in data representation translates to greater access cost. Consequently, in all subsequent performance charts, the performance access results have been presented in terms of the size of the original data. Thus, if the entire inverted index is accessed then the corresponding performance coordinate on the chart would be 200%. One of the interesting observations about the *IGrid*-index is that the exact access fraction of the index can be predicted analytically (as  $2/[\theta \cdot d]$  of the original database size) irrespective of the nature of the underlying data distribution; in the case of the *IGrid*<sup>+</sup>, the average case behavior is within a factor  $1 + c$  of the performance of the *IGrid*-index. The exact curve for the *IGrid*-index and the average-case curve for the *IGrid*<sup>+</sup>-index are presented in Figures 3 and 4 for different data dimensionalities for the case when  $\theta = 1$  and  $c = 1$ .

Since we present the results for very high dimensional data which is beyond the range of traditional indexing techniques such as the X-Tree, we compare the results of our method with those of the VA-File for similarity queries. The VA-File bers imply greater similarity. However, since our intention is only to check the relative behavior of meaningfulness, we can use the same function for both.

**Table 2: Accuracy For Different Values of  $\theta$**

$\theta$	Accuracy (1-nearest neighbor)
1	160
0.5	162
0.25	157
Euclidean	102

[6] works on the assumption that for very high dimensionalities, space partitioning methods are outperformed by simple sequential scan methods. Consequently, the VA-File compresses the data to about 12.5 – 25% of the original size, and then uses the sequential scan on the compressed data in conjunction with some overhead required for the conflict resolution arising from the information lost in compression. The relative time for conflict resolution increases with dimensionality for the method discussed in [6]; therefore this technique is not free of the dimensionality curse. In our experiments, we compare the *IGrid*-index against the *VA-File lower bound*, which ignores the time required for conflict resolution, and compares the least possible percentage of data accessed by the VA-File versus the *IGrid*-index. (Thus, in reality, our results are likely to outperform the VA-File by an even greater margin.) As in the case of the qualitative comparison with the Euclidean distance metric, we evaluate both the indices *IGrid* and *IGrid*<sup>+</sup>. The index *IGrid*<sup>+</sup> requires an additional amount of disk I/O, since it also accesses the inverted lists corresponding to the strongly correlated pseudo-attributes. However, when the dimensionality is high enough, the *IGrid*-index always requires much smaller amount of I/O than the VA-File because of its inversion with dimensionality. In fact, for all dimensionalities 30 or above, both the *IGrid* and *IGrid*<sup>+</sup> indices outperform the VA-File. An important point to be noted is that because of the effect of increasing dimensionality on the data representation and discretization, all of the performance results are completely independent of the nature of the underlying distribution and vary as  $1/d$ . Details are illustrated in Figures 3 and 4.

## 6.2 Parametric Stability

Our analysis in earlier sections indicates that the discretization parameter  $k_d = \lceil \theta \cdot d \rceil$  should be linearly increasing with dimensionality. Note that a choice of  $\theta$  equal to 0.5 or 1 creates a distance function which is quite similar to the  $L_p$ -norm for 1-dimensional and 2-dimensional problems, but is increasingly different for high dimensional problems. These low dimensional cases (in which the  $L_p$ -norm works well) provided us the reference point for picking  $\theta = 1$  in all experiments of this paper. It is useful to see how stable the nature of the similarity function was to small changes in the value of this parameter. We would like to have a similarity function which does not change too dramatically with small changes in  $\theta$ . The accuracy values of the *IGrid*-index for the musk data set for different values of  $\theta$  are illustrated in Table 2. The accuracy values do not change too dramatically for values of  $\theta$  between 0.25 and 1, though the accuracy value of the Euclidean function is significantly different. On examining the actual points which were returned as the nearest neighbors, we found that the set of nearest neighbors returned for these different values of  $\theta$  were highly overlapping (80 – 90%), whereas the set of neighbors returned by

the Euclidean function was very different.

It is important to understand that an index which is built for a particular value of  $\theta = \theta_0$  can automatically resolve<sup>9</sup> queries for all values of  $\theta = \theta_0/\Delta$  for any value of  $\Delta > 1$ . This is achieved by accessing the  $\lceil \Delta \rceil$  closest ranges to the current range for each dimension while performing the similarity calculations. This means that it is desirable to construct the index for larger values of  $\theta$ , which automatically provides the user the capability to interactively change the similarity function in order to examine different kinds of solutions. Such interactive ability may provide the key to high dimensional similarity searches which are often heuristically defined.

## 6.3 Performance of Projected Range Queries

We also study the behavior of the technique with respect to projected range queries. For the purpose of projected range queries, we picked  $n = 4$  dimensions at random, and specified a bounding rectangle whose side was drawn from an exponential distribution with average side of 10% of the total. The results for uniformly distributed data are illustrated in Figure 5. The use of equi-depth ranges ensures that the curve can be generalized to arbitrary distributions when the bounding rectangle is drawn in such a way that the *depth* (fraction of points enclosed in that range) of each side of the bounding rectangle is drawn from the same exponential distribution. As we can see, the percentage of data accessed rapidly decreases with dimensionality and is in fact inversely proportional to it. Other index structures cannot handle such partial queries at all in high dimensional space, since the entire range is relevant for the other dimensions. This results in all the data being accessed in space partitioning methods. The improvement in behavior of the projected range query with dimensionality is particularly pleasing in light of the increased relevance of such queries in very high dimensional data.

We also tested how the performance varied with increasing number of projected dimensions picked by the user. The average range specificity for each side was again chosen to be 10% and the total number of dimensions  $d$  was 1000. As, we can see from Figure 6, even for a relatively large number of dimensions specified from the range query, the specificity of retrieval continued to be very high.

## 7. WHERE IS THE PARADOX?

In this paper, we discussed the *IGrid*-index, a method whose performance improves with increasing dimensionality of the data. These results are in contradiction with all the performance results for other indexes in high dimensional space. The key here is to understand that most indexing structures and algorithms have been developed in the past based on particular distance norms such as the  $L_p$ -norm, which have natural physical interpretations in low dimensional space, but are poor representations of similarity in high dimensional space because of the noise effects of sparsity. For example, even though the euclidean distance metric has a natural physical interpretation for spatial databases in 2- or 3-dimensions, the results of [12] show its meaninglessness in

<sup>9</sup>This is only true for *IGrid* but not for *IGrid*<sup>+</sup>.

high dimensional space because of poor contrast in the distances to the different points. This is *also* the reason that the high dimensional index structures and algorithms cannot prune away large subsets of points easily while searching for the nearest neighbor; there is no discrimination to begin with. In high dimensional data mining applications, the notion of similarity itself is heuristical to begin with; therefore it makes sense to choose measures which lead to greater contrast between the different points. The understanding of meaningfulness of the nearest neighbor is critical in developing distance measures which use only a small fraction of the least noisy information available in high dimensional data in order to measure similarity. The use of such a strategy in order to measure similarity has a surprisingly pleasant side effect; in high dimensional space one has greater flexibility in picking the information which provides better statistical evidence of similarity rather than noise; therefore by picking a higher quality threshold for defining the proximity set, one is able to continue to obtain meaningful nearest neighbors, while improving the indexing performance.

## 8. REFERENCES

- [1] C. C. Aggarwal, A. Hinneburg, D. A. Keim. On The Surprising Behavior of Distance Metrics in High Dimensional Space. *IBM Research Report, RC 21739*, 2000.
- [2] C. C. Aggarwal et al. Fast Algorithms for Projected Clustering. *ACM SIGMOD Conference Proceedings*, pages 61–72, 1999.
- [3] C. C. Aggarwal, P. S. Yu. Finding Generalized Projected Clusters in High Dimensional Spaces. *ACM SIGMOD Conference Proceedings*, pages 70–81, 2000.
- [4] C. C. Aggarwal, J. L. Wolf, P. S. Yu. A New Method For Similarity Indexing of Market Basket Data. *ACM SIGMOD Conference Proceedings*, pages 407–418, 1999.
- [5] S. Arya. Nearest Neighbor Searching and Applications. *Ph. D. Thesis, University of Maryland, College Park, MD*, 1995.
- [6] R. Weber, H.-J. Sheck, S. Blott. A Quantitative Analysis and Performance Study for Similarity Search Methods in High Dimensional Spaces. *VLDB Conference Proceedings*, pages 194–205, 1998.
- [7] N. Beckman, H.-P. Kriegel, R. Schneider, B. Seeger. The R\*-Tree: An Efficient and Robust Method for Points and Rectangles. *ACM SIGMOD Conference Proceedings*, pages 322–331, 1990.
- [8] K. P. Bennett, U. Fayyad, D. Geiger. Density-Based Indexing for Approximate Nearest Neighbor Queries. *ACM SIGKDD Conference Proceedings*, pages 233–243, 1999.
- [9] S. Berchtold, C. Böhm, H.-P. Kriegel. The Pyramid Technique: Towards Breaking the Curse of Dimensionality. *ACM SIGMOD Conference Proceedings*, pages 142–153, 1998.
- [10] B.-U. Pagel, F. Korn, C. Faloutsos. Deflating the Dimensionality Curse Using Multiple Fractal Dimensions. *ICDE Conference Proceedings*, pages 589–598, 2000.
- [11] S. Berchtold, D. Keim, H.-P. Kriegel. The X-Tree: An Index Structure for High Dimensional Data. *VLDB Conference Proceedings*, pages 28–39, 1996.
- [12] K. Beyer et al. When is Nearest Neighbors Meaningful? *ICDT Conference Proceedings*, pages 217–235, 1999.
- [13] G. Das, H. Mannila, P. Ronkainen. Similarity of Attributes by External Probes. *KDD Conference Proceedings*, pages 16–22, 1998.
- [14] S. Deerwester et al. Indexing by Latent Semantic Analysis. *Journal of the American Society for Information Science*, 41(6): pages 391–407, 1990.
- [15] U. Shaft, J. Goldstein, K. Beyer. Nearest Neighbor Query Performance for Unstable Distributions. *Technical Report TR 1388, University of Wisconsin at Madison*, 1998.
- [16] V. Ganti, J. Gehrke, R. Ramakrishnan. CACTUS-Clustering Categorical Data Using Summaries. *ACM SIGKDD Conference Proceedings*, pages 73–83, 1999.
- [17] A. Gionis, P. Indyk, R. Motwani. Similarity Search in High Dimensions via Hashing. *VLDB Conference Proceedings*, pages 518–529, 1999.
- [18] A. Guttman. R-Trees: A Dynamic Index Structure for Spatial Searching. *ACM SIGMOD Conference Proceedings*, pages 47–57, 1984.
- [19] A. Hinneburg, C. C. Aggarwal, D. A. Keim. What is the Nearest Neighbor in High Dimensional Spaces? *VLDB Conference Proceedings*, 2000.
- [20] R. Jain, D. A. White. Similarity Indexing: Algorithms and Performance. *SPIE Storage and Retrieval for Image and Video Databases IV*, 2670: pages 62–75, 1996.
- [21] I. Kamel, C. Faloutsos. Hilbert R-Tree: An improved R-Tree Using Fractals. *VLDB Conference Proceedings*, pages 500–509, 1994.
- [22] N. Katayama, S. Satoh. The SR-Tree: An Index Structure for High Dimensional Nearest Neighbor Queries. *ACM SIGMOD Conference Proceedings*, pages 369–380, 1997.
- [23] K.-I. Lin, H. V. Jagadish, C. Faloutsos. The TV-tree: An Index Structure for High Dimensional Data. *VLDB Journal*, 3(4): pages 517–542, 1994.
- [24] N. Roussopoulos, S. Kelley, F. Vincent. Nearest Neighbor Queries. *ACM SIGMOD Conference Proceedings*, pages 71–79, 1995.
- [25] G. Salton, M. J. McGill. Introduction to Modern Information Retrieval. *Mc Graw Hill*, New York.
- [26] H. Samet. Design and Analysis of Spatial Data Structures. *Addison Wesley*, 1989.
- [27] T. Sellis, N. Roussopoulos, C. Faloutsos. The R+ Tree: A Dynamic Index for Multidimensional Objects. *VLDB Conference Proceedings*, pages 507–518, 1987.
- [28] D. A. White, R. Jain. Similarity Indexing with the SS-Tree. *ICDE Conference Proceedings*, pages 516–523, 1996.