On Optimizing Distance-Based Similarity Search for Biological Databases

Rui Mao, Weijia Xu, Smriti Ramakrishnan, Glen Nuckolls, Daniel P. Miranker
Department of Computer Sciences
Center for Computational Biology and Bioinformatics
University of Texas at Austin
1 University Station C0500, Austin, TX 78712-0233 USA
{rmao, xwj, smriti, nuckolls, miranker}@cs.utexas.edu

Abstract

Similarity search leveraging distance-based index structures is increasingly being used for both multimedia and biological database applications. We consider distance-based indexing for three important biological data types, protein k-mers with the metric PAM model, DNA k-mers with Hamming distance and peptide fragmentation spectra with a pseudo-metric derived from cosine distance. To date, the primary driver of this research has been multimedia applications, where similarity functions are often Euclidean norms on high dimensional feature vectors. We develop results showing that the character of these biological workloads is different from multimedia workloads. In particular, they are not intrinsically very high dimensional, and deserving different optimization heuristics. Based on MVP-trees, we develop a pivot selection heuristic seeking centers and show it outperforms the most widely used corner seeking heuristic. Similarly, we develop a data partitioning approach sensitive to the actual data distribution in lieu of median splits.

1. Introduction

Distance-based indexing exploits the relative distance among data objects to support similarity search [9, 17]. The goal of the approach is the creation of general-purpose data structures that support fast scalable retrieval of complex data types. The primary requirement is that the distance function be a *metric*. That is, the distances are positive, symmetric, and satisfy the triangle inequality. Many domain-specific index structures have been developed for biological applications, such as suffix trees [15] and suffix arrays [23]. However, they are limited by their domains and are therefore not discussed in the paper. The active application of distance-based indexing began with multimedia applications (images and sound) [3, 12].

Increasingly, in the face of hyper-exponentially growing biological databases, biological applications of distance-based indexes are also being investigated [25, 33, 35]. A data structure, the index, is initialized off-line. On-line data retrieval based on proximity, either k nearest-neighbor or range search may exploit the triangle inequality, amortizing the off-line cost of creating the index and promising scalable performance. In addition, the use of tree-based data structures offers seamless integration with object-relational database management systems (e.g. DB2, Oracle, Postgres) [16].

In multimedia applications, each object is often associated with a long description vector. For example, an image might be described by frequency and/or texture histograms [3, 9, 17]. The similarity of a pair of objects is defined as the Euclidean norm of their feature vectors. A primary benefit of distance-based approaches is the casting of the similarity of a pair of high dimension descriptors to a single number.

We have determined that several established results in distance-based indexing, well founded in anticipation of multimedia applications, are not applicable to biological workloads. A primary concern is the impact of the curse of dimensionality. Informally, the curse of dimensionality states that similarity search algorithms have an exponential dependency on the dimension of the space. Analytic results concerning the indexability of very high-dimensional uniform data have shown that any indexing scheme will degrade as the number of dimensions increase, and, given the overhead of indexing schemes, at some point a simple

This research was supported by grants from the Texas Higher Education Coordinating Board and the National Science Foundation contract DBI-0241180, IIS-0325116, EIA-0121680, EF-0331453

linear scan will outperform any indexing scheme as the dimension of the problem increases. A distance function may cast a high dimensional problem to a single number, but the curse of dimensionality speaks to intrinsic properties of the data. Simple changes in representation do not overcome these intrinsic properties.

In biological applications a primary advantage of distance-based methods is that they may be used to capture similarity of data that is not naturally described by a point in a Euclidean space. For example, the similarity of biological sequences concerns evolutionary distance, often modeled by weighted-edit distance or Hamming distance. Our work in matching peptide fragmentation mass-spectra is derived from cosine distance.

We have concluded that the intrinsic dimensionality of three important biological workloads is not very high. For instance, the intrinsic dimensionality of 5-mers (overlapping fragments of length 5) from a yeast protein database under the mPAM weighted edit-distance model is roughly 2.7 or 20.5, depending on the measure. Similarly, Hamming distance on Arabidopsis' genomic sequences of length 18 (a length suitable for identifying highly conserved genomic sequences across species as well as identifying small RNA coding genes [35]) has an intrinsic dimensionality of 2.2 or 23. The intrinsic dimensionality of our third application workload, a database of analytically determined peptide mass fragmentation spectra, is 130 or 300, 2 orders of magnitude less than the number of resolvable peaks (approximately 40,000). See Section 3.

In previous work, we determined that of the three major classes of distance-based indexing methods, M-trees [10], generalized hyper-plane trees [36] and vantage-point trees [36, 39], the vantage-point tree suits our biological workloads best. In a vantage-point method, a metric space is partitioned into disjoint regions recursively to form a hierarchical structure. At each step a vantage-point, also known as a *pivot*, is selected. The distance of each point to the pivot is computed. The distance d to the pivot establishes a bipartition, with half the points inside a bounding sphere of radius d centered on the pivot and half the points outside that sphere. The method has been generalized for large disk-resident databases by considering multiple vantage-points at each level, and a higher degree of partitioning (>2 partitions). This data structure is known as a multiple vantage point tree, or MVP-tree. The result resembles a k-d tree [2]. The pivot-selection and data partition heuristics are critical to search performance [4, 39].

For pivot selection we revisit the "corner-pivot" heuristic first suggested by Yianilos. Yianilos' informal argument assumes a Euclidean space and uniform data distribution, which don't hold for biological workloads. We develop a "center-pivot" heuristic. Empirical results on synthetic uniform Euclidean data and each of the biological workloads agree with our analysis.

Independent of algorithmic approach or the intrinsic dimension of the data, an important heuristic with respect to the success of distance-based indexing speaks to the data intrinsically entailing a hierarchical clustering and the ability of the initialization algorithm to reflect that hierarchy [6, 24]. To determine more effective data partitions we apply a clustering algorithm to the distances from the data points to the pivots. We set the partitioning surfaces according to those clusters. Although our heuristics can lead to unbalanced index trees, empirical results show that they outperform the canonical heuristics that traditionally favor balanced trees.

Ultimately we show that MVP-trees can be effectively optimized for biological workloads and that specific optimizations to the character of biological workloads can amount to up to a 50% performance improvement over heuristics developed primarily for multimedia applications.

The rest of the paper is organized as follows. The biological workloads are summarized in Section 2. In Section 3, we discuss intrinsic dimensionality. The MVP-tree and the bulkload heuristics are discussed in Section 4. Section 5 consists of empirical results, followed by conclusions and future work in Section 6.

2. Biological workloads

In the context of a large project aiming to build general purpose metric-space extensions to an object-relational database management system, we have developed applications in comparative genomics and proteomics [24, 25, 26, 37]. These applications have necessitated the creation of the three workloads we consider here, the amino-acid sequences of the yeast proteome, the DNA sequences of the Arabidopsis genomes, and analytically determined peptide fragmentation spectra of human and E. coli proteins.

Similarity retrieval of k-mers is an important component of many biological applications [14, 27]. The similarity between two biological sequences are often measured based on their optimal local alignment. One general approach for finding useful optimal local alignments is to divide the sequences in a sequence database and the query sequence into k-mers. The evolutionary closed k-mers were identified and then

used to determine a local alignment. Similar approaches were adopted by popular sequence searching tools, such as FASTA and BLAST [1, 31].

The similarity or distance between two k-mers is often measured by their global alignment score whose computation is often based on a substitution matrix [11].

The yeast protein database is organized as 5-mers, overlapping substrings of length 5. The similarity function is weighted-edit distance based on the recently developed metric PAM substitution matrix, mPAM [38]. In mPAM the distance between amino acids forms a metric. It follows that mPAM weighted-edit distance on sequences also forms a metric [34]. The protein sequence dataset was downloaded from Genbank in July 2003 [29]. The dataset contains FASTA formatted amino acid translations extracted from GenBank/EMBL/DDBJ records that are annotated with one or more CDS features.

The DNA sequence database contains the Arabidopsis thaliana genome organized as 18-mers. The similarity function used is simple Hamming distance. This choice of k-mer length was central to an analysis that compared the Arabidopsis and Rice genomes to identify candidate DNA regions that would be universally primerable and amplifiable for all flowering plants [37]. The data was downloaded from ftp://ftp.arabidopsis.org/home/tair/ in March 2003.

A peptide fragmentation spectrum or tandem MS spectrum is a histogram of mass over charge (m/z) ratios, or peaks. It is generated by collision-induced fragmentation of peptides that are in turn derived from the enzymatic digestion of a protein. Our mass spectra dataset consists of 653,882 predicted fragmentation spectra of peptide of E. coli and human proteins derived from the REFSEQ protein sequence database, which is available online [28]. Given a sample mass range of 4000 Da (Daltons), and resolution of representation 0.1 Da, we can visualize each spectrum as a binary vector in a space of approximately 40,000 dimensions. In practice, these spectra are stored as variable length real valued vectors.

Our distance function is a 'fuzzy' measure of the shared peaks count (SPC) [35]. It can be interpreted as a modified form of cosine distance [12, 13], and overcomes some of the drawbacks of SPC. Two peaks are marked as being equal if the absolute difference of their m/z values lies within certain tolerance. Based on cosine distance interpretation, the distance between two data points is the angle between their vector representations.

The biological workloads are summarized in Table 1 in next section. The intrinsic dimensionality listed in the table will be discussed in next section.

3. Intrinsic dimensionality

For clarity we will refer to the domain and range of the distance function separately. The distinction between these is often blurred in studies which assume data is randomly and uniformly distributed in some well-formed metric-space, such as Euclidean space or Hamming space. It is often convenient to map a biological domain to a well-formed domain. But biological data tends to be very highly structured and/or stylized such that the properties of the domain of the model may provide a very misleading characterization of the range of the data.

Consider, for example, the databases of analytically determined Maldi mass-spectra, used to interpret proteomic mass-spectroscopy experiments. Depending on the machine, a Maldi mass-spectrometer may be able to resolve 40,000 to 100,000 peaks. Thus, using the vector-space model, the domain of the distance function will have 40,000 to 100,000 dimensions. Analytically determined monoisotopic peptide fragment spectra typically average 10 to 50 peaks, independent of resolution. Further, since there is no quantitative information, peak intensity information may be disregarded, and each dimension of the vector-space is just a binary value, peak/no-peak. Thus, measured against the domain, the range of a mass-spectra database contains vast regions of empty space.

Due to the curse of dimensionality, the dimension of the data space dominates the efficiency of the search algorithms. The practical impact of the curse of dimensionality on an algorithm stems from the requirement that the algorithm form largely disjoint subsets of the data that may be pruned during search. The higher the dimension of the space, the more difficult it is to use distance to distinguish the subsets (i.e. the higher the dimension, the more data that have the same distance to a reference data point, and are thus not distinguishable for the view of the reference point).

Distance-based indexing methods only consider relative distance between data objects without interpretation to coordinate systems, but the density of the data remains an intrinsic property. The actual performance of a distance based index is dependent on the actual density and distribution of the data (the range), and not the formal definition of the legal points in the space (the domain). In many applications, the range of high dimensional data domains may have much lower intrinsic dimensionality.

Therefore, the intrinsic dimensionality of a metric space is a way to quantify a characteristic of the range of a distance function.

Two methods of measuring intrinsic dimensionality have been proposed. We will consider them both on each of three biological workloads. The first measurement is due to Chavez et al. [9].

Definition 1: the intrinsic dimensionality of a metric space is defined as $\rho = \mu^2/2\sigma^2$, where μ and σ^2 are the mean and variance of the distribution of the pair-wise distances among the data points in the space.

Using Yianillos' asymptotic results for mean and variance [40], this measure produces the expected result, $\theta(d)$, for d-dimensional L_p spaces with random and uniform data. The hidden constants in these results make the measure more useful asymptotically than for a single data set. In fact, the constants in the definition were chosen for technical convenience.

Another way to destermine the intrinsic dimensionality is to measure how the volume of a hyper-ball, that is, the number of points contained in it, changes with respect to the radius. This measure agrees with the standard measure in d-dimensional Euclidean space with random and uniform data, since the ball with radius $c*r$ has c^d times the volume of the ball with radius r. Thus if the data is uniformly distributed, the number of data points in the hyper-ball will be proportional to r^d.

Definition 2: Let r be the radius of range queries, and n be the average number of results of range queries with radius r. The intrinsic dimension is determined as the slope coefficient of the linear regression of log(n) vs. log(r).

We avoid queries that incur a reduced volume due to boundary effects.

Except for the protein data set, we measured the dimension based on query points that were in the data domain but not necessarily in the data set. DNA queries on the *Arabidopsis thaliana* genome were drawn randomly from the rice DNA data and the mass spectra queries were from other source (Section 5).

We believe it is reasonable and useful to base the dimensionality measure in part on the intended application. Thus Definition 2 in this case is preferred since the ability to prune in distance based indexing depends on properties of the domain and the query workload, both of which are reflected in the regression based measure.

The intrinsic dimensionalities of the biological workloads estimated by the two methods are listed in Table 1. For the protein 5-mers, its ρ-value of dimension, by Definition 1, is 20.5, but is just 2.7 by regression of Definition 2. For the DNA 18-mers, its ρ-value of dimension is 23 but is just 2.2 by regression. Similarly, for mass-spectra with fussy cosine distance, its ρ-value of dimensionality is 300 while 130 by regression, much less than the dimension of their domain, a 40,000 dimensional binary vector representation.

Table 1. Summary of biological workloads

Biological workloads	Size	Distance function	Domain dimen-sionality	Intrinsic Dimen-sionality	
				Def. 1	Def. 2
Protein 5-mer	up to 200 million	Global alignment	5	20.5	2.7
DNA 18-mer	up to 30 million	Hamming distance	18	23	2.2
Mass-spectra	653,882	Cosine distance	40,000	300	130

To further reduce the dimensionality, we design a semi-metric distance function and modify the MVP tree search algorithm [35], similar to the work done in [21, 33]. The semi-metric distance between two spectra data is defined as the sum of the fuzzy cosine distance, and the absolute difference between the precursor masses of the two peptides. Precursor mass difference is defined to be zero if it is less than a given tolerance [35]. The semi-metric function looses the triangle inequality by a constant, k, i.e. $d(x,y) + d(y,z) \leq d(x,z) + k$. We modify the pruning criterion of the MVP search algorithm for semi-metric search. Given a pivot p, query q and radius r, a metric-distance search would prune all points u where $|d(u, p) - d(q,p)| > r$, while a semi-metric search prunes all points u where $|d(u, p) - d(q,p)| > r+k$ [35]. With the semi-metric, the intrinsic dimensionality of mass spectra is reduced to approximately 1, as shown in the last column of Table 1. Since the intrinsic dimension is dramatically reduced using the semi-metric, we only consider the semi-metric in the rest of this paper.

4. Multi-vantage point trees

In anticipation of large-scale disk resident databases, we consider the paginated version of MVP-trees, i.e., MVP-trees whose internal nodes and leaves are sized to fit onto a disk page. For the internal nodes this is accomplished by choosing a combination of the number of vantage-points and the number of partitions per vantage point such that the memory required to store the partitioning information and the surrogates (disk related addresses serving as pointers to children) approaches the size of a disk-page, which is usually 4

Kbytes. The number of data items stored in a leaf is inversely proportional to the size of the data objects.

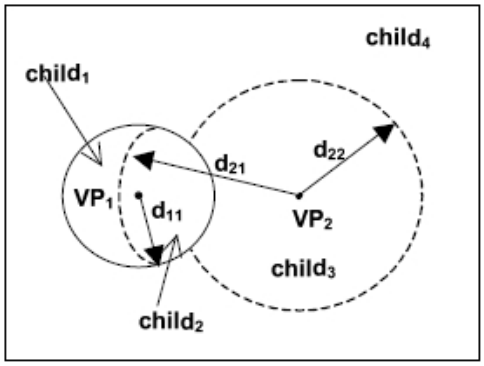


Figure 1. Partitions of a (2,2,m) MVP internal node

```
IndexNode bulkLoad(Object[] dataset, int pivotNumber) {
   if (dataset.length <= maxLeafSzie) {
      create a leaf node and return;
   }
   else {       //build an internal node
      Object [] pivot = selectPivot(dataset, pivotNumber);
      Object [] subDataset = partition(dataset, pivot);
      IndexNode [] children;
      for ( each subDataset[i] )
         children[i]=bulkLoad(subDataset[i], pivotNumber);
      Create an internal node with children, and return;
   }
}
```

Figure 2. Bulkload algorithm of MVP-Index

The topological parameterization of an MVP-index is denoted by the triple (*v, s, m*), where *v* represents the number of pivots in each node, *s* is the number of subsets into which a dataset is partitioned based on the distances to one pivot, and m is the maximum number of data points in a leaf node. For internal nodes, starting with the first pivot p_1, the data is partitioned into s disjoint intervals, ($d_{1,i}$-*min,* $d_{1,i}$-*max*), each of which defines a range predicate, i.e., for any data point x in the i^{th} subset, $d_{1,i}$-*min* <= $d(p_1,x)$ <= $d_{1,i}$-*max*. Within the node, the process repeats recursively for p_2, …, p_v. The fanout of a node is thus s^v. Figure 1 illustrates the four partitions of a (2,2,m) internal node. VP_1 and d_{11} split the data into 2 partitions. VP_2 and d_{21} and d_{22}, in turn split the 2 partitions into 4.

The bulkload algorithm is detailed in Figure 2. It is a divide and conquer algorithm. The function bulkLoad( ) takes the data set and the number of pivots as arguments, and it returns the root node of the index (sub)tree. If the data point array is small enough to fit onto a disk page, bulkLoad( ) just creates a leaf node. Otherwise, pivots are selected, the range of distances for each partition is determined and then recorded to form an internal index node. The data is divided and a recursive call to bulkLoad() is invoked for each partition.

Optimal pivot selection would require solving NP-hard clustering problems**.** Thus the quality of the index is determined by the choice of heuristic algorithm [4, 5]. Here we reconsider Yianilos' corner selection heuristic for pivots. We explicitly introduce a clustering algorithm as the basis of identifying partitions

4.1 Corner vs. center pivot selection

The Hochbaum-Schmoy's k-center clustering algorithm, also known as the farthest-first-traversal algorithm, is a fast, convenient way to identify pivots. Farthest-first-traversal gives a 2-approximation to minimizing the maximum cluster diameter [18]. Its time and space complexities are both O(n), where the number of farthest points is considered as a constant.

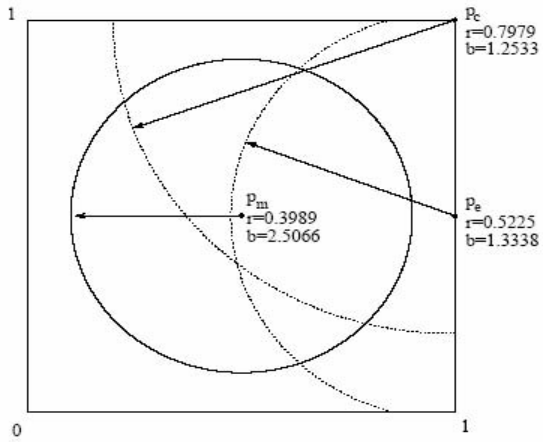


Figure 3 Yianilos' pivot heuristic. Corners minimize the partitioning surface

The algorithm is consistent with a heuristic proposed by Yianilos that *corners* form good pivots and is thus commonly used. Yianilos' arguments are best explained by including his illustration. See Figure 3.

Yianilos considers the distance restricted nearest-neighbor problem [39]. That is, nearest neighbors beyond a certain distance from the query point are of no interest. This retrieval definition is consistent with

our applications. Yianilos' algorithm assumes that the data consists of uniform points on a 2-d unit square with Euclidean distance.

The crux of Yianilos' argument is that by minimizing the surface of the partitioning sphere, (or circle in 2-D), one minimizes the probability that on any one retrieval search a neighborhood around the query point will intersect with the bounding sphere. Such intersections eliminate the opportunity to choose to search one partition to the exclusion of the other; i.e. to prune. Yianilos considers a circle, centered at point p in the square and with radius r such that the area inside and outside the circle are equal. As shown in Figure 3, he similarly considers a point on the side, a point in the corner and the center. Of the three, the circle defined by the corner pivot has the smallest boundary. This argument holds for a Euclidean distance between points in a high-dimensional unit hyper-cube. Empirical results presented in Section 5 corroborate the benefit of this heuristic for metric-spaces that fulfill all of Yianilos' assumptions.

The biological workloads we consider do not involve Euclidean metrics. Each of the metrics has symmetry and a "wrap-around" property that makes finding a corner very difficult. For these workloads, one can travel in any direction from a point and ultimately return to the same spot. Thus, with respect to the domain, every point in the space is identical and simply has no distinguishing geometric features (like a corner).

We agree with Yianilos' heuristic principle that minimizing the surface of the partitioning sphere maximizes pruning. For our biological workloads, that means minimizing the diameter of the bounding spheres. Since there is no definition of algebraic operations in metric space, a k-means algorithm cannot be applied to a metric space directly. K-center is also not applicable because its objective only considers the maximum cluster diameter. Therefore, k-median and MSD (Minimize the Sum of cluster Diameter) algorithms are our choices. K-median algorithms try to minimize the sum of distances from all data points to their cluster centers, while MSD algorithms try to minimize the sum of cluster diameters. Both of these are NP-hard problems and there are some primal-dual based constant factor approximation algorithms [7, 20]. However, these algorithms usually have high time complexity, which does not make them applicable to the case of indexes, where the datasets are large. For MVP-tree indices, we implement an MSD algorithm derived from CLARA (Clustering LARge Applications) [22]. CLARA is a simple k-median algorithm based on sampling and iteration. The time complexity of the algorithm is O(n). Interested readers are referred to [22].

Empirical results show that using centers as pivots is comparable to and in most cases outperforms the use of corners.

4.2 Data partition

It has been argued, most notably by Brin, that the effectiveness of a metric-space index depends on the algorithm's ability to capture the intrinsic hierarchical structure of the data [6]. Note that this is consistent with the heuristic goal of minimizing the diameter of the partitioning spheres [24]. Furthermore, numerous bioinformatic studies use hierarchical clustering to make important biological discoveries, suggesting that as an application domain, biological data is well suited to this approach.

We must begin by recognizing the differences between the application of hierarchical clustering to data mining and to initializing a database index. Differences include the size of the data set and the semantic quality of the results. In data mining, an important goal of clustering is to extract semantic relationships from the data with high confidence, and the dataset is relatively small and manageable. $O(n^2)$ algorithms are common. Introducing small but semantically effective improvements in the clustering even at large computational expense is desirable. When managing a database one must assume that the amount of data is arbitrarily large. Even O(nlogn) algorithms may be prohibitive. The semantic quality of the clustering is of secondary concern. The primary concern is discovering spatially distinguishable disjoint data sets that improve the pruning decisions during search.

Toward this end, given a pivot, we consider a clustering method to locate partitions rather than organizing the partitions by balancing their cardinality. When the data distribution displays structure, a histogram of the distances from the pivot to each of the data point will show peaks (dense regions) and valleys (sparse regions). Assuming query distribution is similar to the data distribution, we can decrease the probability that a neighborhood around a query will intersect with the partition boundary by placing the partition boundaries in the sparse regions, enabling successful pruning. Consequently, an index tree structure consistent with the intrinsic clustering of the data is apt to outperform an index that is inconsistent with that structure.

The use of clustering to locate partitions in lieu of balancing the cardinality of the partitions may lead to

unbalanced trees. Our algorithm ameliorates but does not solve this. Even so, we show empirically that the clustering method improves average retrieval times. Our application domain is not concerned with real-time transactions, and the concomitant criteria of consistent response times for concurrent users. These databases are intended to support large-scale data analysis. Minimizing the average search time is more important than minimizing the standard deviation of the response times.

```
ClusteringPartition(D: dataset, pi: a pivot, s: number
of partitions induced from each pivot ) {
   // each cluster is associated with a set of pivots,
      initially one cluser D is associated with all the
      pivots
   while( there exists a cluster C whose pivots set P is not
         empty) {
      for ( each pi in P) {
         compute the distances to pi;
         find s-1 split values by calling k-means( );
         compute the sizes of sub-clusters based on
            the split values;
         compute the variance of the sizes of sub-
            clusters;
      }
      find p-mv, resulting in the smallest variance;
      split C into s sub-clusters based on the split
         values from p-mv;
      remove p-mv from P;
      copy P as the pivot set for each of the s sub-
         clusters;
      remove C;
      }
   return all the clusters;
}
```

Figure 4. Algorithm of Clustering Partition

An outline of the algorithm is as follows. Pivots are selected by some methods beforehand. For each pivot, a one-dimensional k-means algorithm is run with respect to the distance from the pivot to each of the data points. In an effort to maintain a balanced tree structure, if there is more than one pivot, the variance of the size of the clusters for each pivot is computed. The pivots are sorted by increasing variance in the size of the clusters, and the MVP partitions situated on cluster boundaries. In the next step, each sub-dataset and the remaining pivots are considered similarly. The algorithm is detailed in Figure 4. One can prove that the time complexity of this algorithm is O(n), but the constants are large and one must assume the k-means clustering converges quickly.

Several research efforts relate to this approach. Chavez et al. proposed the *distance-balanced* partition algorithm [8]. In their approach the range of the distance function is partitioned into k equal size intervals, independent of the cardinality of the partitions. The time to build the tree is very fast, as the consideration of the actual data is minimal. This construction also leads to unbalanced trees, but without the benefits of considering the actual data distribution. Brin's GNAT-trees, which build upon the general hyper-plane methods of building metric-index trees, use data sampling to instill the structure of real data into the tree. Furthermore, the trees are kept balanced in height by varying the fanout of the internal nodes, e.g., internal nodes receiving larger than average partitions introduce additional partitions at the next level. In our own efforts, we developed a bidirectional bulkload algorithm of M-trees whose primary goal was to minimize the diameter of partitioning sphere through a better cluster merging step [24]. To our knowledge, this is the first effort to introduce an initialization step into MVP-trees that improves the ability of the MVP-tree to capture the intrinsic hierarchical clustering of the data.

5. Empirical results

In this section, we present data to compare the MVP-tree bulkload heuristics. The hypotheses are that for the biological workloads, center-pivot heuristic outperforms corner-pivot, and clustering partition outperforms the distance-balanced partition. We also present data to demonstrate scalability.

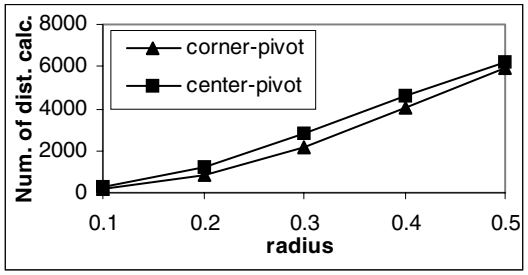
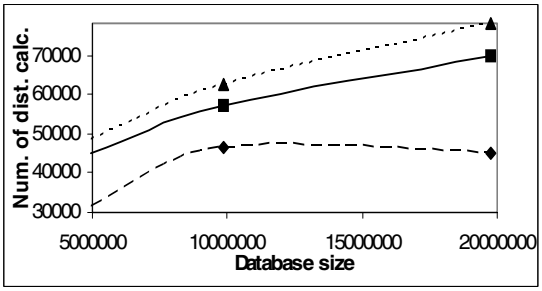


Figure 5. Pivot selection of uniform vectors

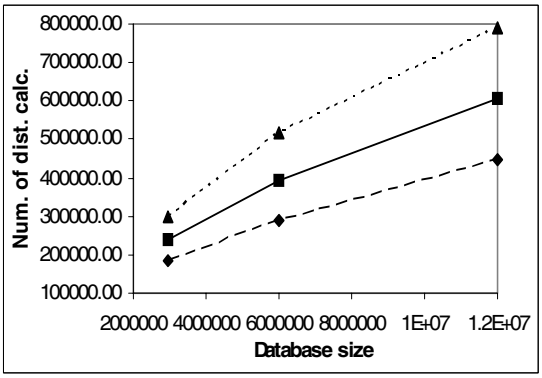
The important measures are algorithmic measures of scalability. We use system-independent measures for performance, such as number of distance calculations and number of I/O operations. The results of distance calculations and number of I/O are similar in most of the cases. Therefore, only the results of distance calculations are presented.

5.1 Corner vs. center pivot selection

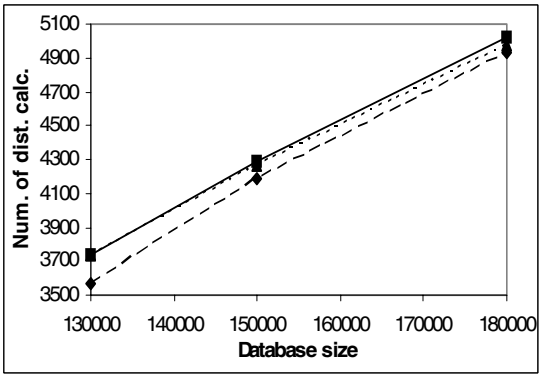
We first compare the pivot-selection heuristics of uniform Euclidean space. A number of vectors are randomly selected as queries from synthetic uniform vector datasets of size 10k in 5-d hyper-cube and hyper-ball. The average number of distance calculations of range queries with various radii is shown in Figure 5. Clearly, corner-pivot outperforms center-pivot, and Yianilos' argument is supported.



(a) Peptide 5-mer, radius = 3



(b) DNA 18-mer, radius = 3



(c) Mass-spectra, radius = 0.5

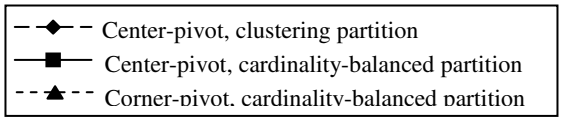
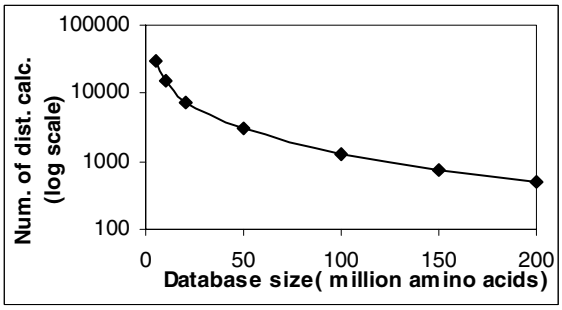
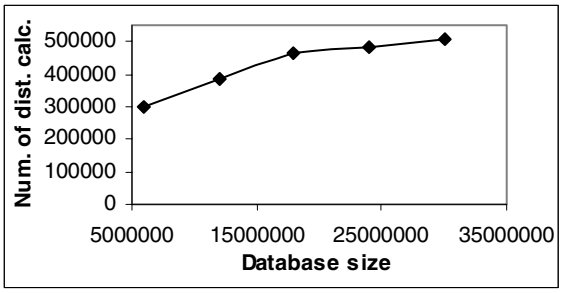


Figure 6. Pivot selection and data partition of biological workloads

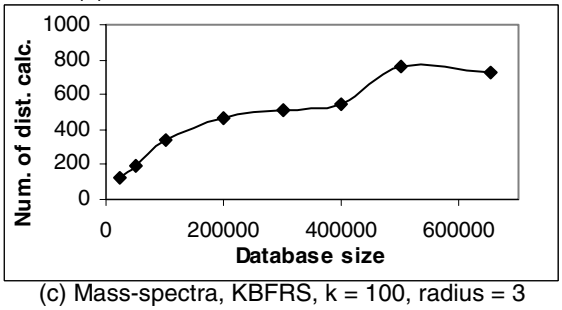
We now consider the biological workloads with respect to various combinations of heuristics and database sizes. For mass-spectra, the queries were experimental peptide fragmentation spectra drawn from the Open Proteomics Database [30]. For DNA, the query set consists of 18-mers randomly selected from rice genome. For protein, data points are randomly selected as queries. Figure 6 shows the average number of distance calculations vs. database size with various combinations of pivot selection and data partition heuristics for fixed range query radius of each workload.



(a) Peptide 5-mers, EDKNN, k = 300, d = 3



(b) DNA 18-mer, DKNN, k = 300, d = 3



(c) Mass-spectra, KBFRS, k = 100, radius = 3

Figure 7. scalability on biological workloads

Figure 6 (a) and (b) show that for peptide 5-mers and DNA 18-mers, center-pivot outperforms corner-pivot. From Figure 6 (c), we can see that center-pivot is outperformed by corner-pivot in 2 cases, but the margins are tiny. Therefore, in comparing the data partition heuristics next, we only present data with center-pivot to select pivots.

5.2 Data partition

The long dash diamond line in Figure 6 represents the data of center-pivot and clustering partition. In all three applications the clustering partition outperforms cardinality-balanced partition. For the 19M peptide database, the number of distance calculations required for the center-pivot and clustering partition heuristics, is about half of that of the traditionally used corner-pivot and cardinality-balanced methods. The superlinear speed-up for the peptide results is a real phenomenon and is explained in next sub-section, for Figure 7 (a).

5.3 Scalability

Distance restricted k-nearest neighbor (DKNN) search is of key interest in many biological applications [35, 37]. The distance restriction stipulates a maximum distance in a search result, even if that means returning fewer than k-nearest neighbors. The algorithm utilizes a priority queue to store all of the index nodes to be searched, and another priority queue to rank the search results.

In addition, we also defined two more aggressive algorithms based on the same best-first search strategy: extended distance restricted k-nearest neighbor (EDKNN) and k best first range search (KBFRS). Both aggressive algorithms traverse the index structure in the same order as DKNN. However, they may terminate earlier than normal DKNN algorithm. The KBFRS will terminate once k qualified results are found. The EDKNN will terminate after both the nearest neighbor and total k qualified results are found. If there are no more than k qualified results within the distance restriction, all three algorithms will generate same results set. If there are far more than k qualified results within the distance restriction, the aggressive algorithms will run much faster with the potential loss of accuracy. In practice, we use EDKNN in k-mer matching for homology search problem and KBFRS in a coarse filter of similar mass-spectra data retrieval.

Figure7 (a) shows the average number of distance calculations of EDKNN queries, with k equals 300 and distance restriction d equals 3, of peptide 5-mers with various database sizes.

We see in a number of cases that the number of distance calculations decreases as the database size increases. This is a real phenomena connected to the search properties of k-nearest neighbor. For peptide 5-mers, the search space is about 30 million, (20^5). Although amino acids are not uniformly distributed, the search space is gradually covered as the amount of data increases. During the nearest neighbor search, the algorithm prioritizes the nodes based on distance from the query point to the partition. The highest priority is given to nodes that may contain exact matches. As the database increases in size, the probability that a partition that may contain an exact, or near exact match actually contains matching data increases. The probability of search backtracking decreases, indicating super-linear speedups.

Figure7 (b) shows the average number of distance calculations of DKNN queries, with k equals 300 and distance restriction d equals 3, of DNA 18-mers with various database sizes. Clearly, the search scales well.

Figure 7 (c) shows average number of distance calculations of KBFRS queries, with k equals 100 and distance restriction d equals 3, of mass-spectra with various database sizes. Obviously, the number increases slowly as database size increases. Note that like the depth of a B+ tree in a relational database system, the MVP index tree has discontinuous increases in height as the database grows. Thus, the search cost increases very slowly, subject to sudden increments when the index increases in height (from 300,000 to 400,000).

6. Conclusions and future work

In this paper, we show that some traditional results on optimizing MVP-trees, instigated by multimedia applications, should not be applied to biological workloads. The reason is that biological workloads usually have low intrinsic dimensionality and the data is highly structured. The distance functions used for biological database are not Euclidean norms.

Initial construction of MVP-indexes is critical to their search performance. Because of the difference between Euclidean space vectors and biological workloads, the canonical heuristic to select corners as pivots does not work well in biological applications. As mentioned in [39], “The discrete metric is thus excluded along with many other cases”.

Further investigation of methods whose goals are to better capture the intrinsic hierarchical clustering of the data is in order. Our departure from balanced tree-structures, justified by the fact that the workloads comprise large-scale data analysis rather than on-line transaction processing, offers flexibility not normally considered in database research.

Another characteristic of biological databases, not explored in this paper, is that biological databases are usually written-once. That is, the databases grow, but

direct updates are not allowed. If records must be updated, this is usually accomplished by adding a new version. The implication is that for some databases, the frequency of queries compared to material changes in the database may justify extraordinary off-line computation time to optimize the initialization of the index structure. Consider the following scenario. Suppose BLAST searches against Genbank could be done in half the time, but only if the sequence contents of Genbank were preprocessed using a month of time on a supercomputer. It would only be weeks before there would be a net gain in CPU cycles. The productivity of scientists searching Genbank would be immediately improved. We have shown that in some cases, our heuristics and preprocesing methods for MVP-trees may double their performance. It would benefit biology if this magnitude of improvement could be gained consistently. The write-once nature of biological databases enables many different avenues. For important problems, even methods that may appear to be prohibitively expensive may be worth pursuing.

Acknowledgement

We thank Jacob Neal Sarvela for discussions concerning assessing the intrinsic dimension by regression on the range query radius and number of results.

References

[1] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman, "Basic local alignment search tool", *J. Mol. Biol*, 1990, 215: 403-410

[2] J. L. Bentley, "Multidimensional binary search trees used for associative searching", *Communications of the ACM*, September 1975, 18(9):509—517.

[3] S. Berchtold and D.A. Keim, "High-dimensional Index Structure", In *Proc. ACM SIGMOD International Conference on Management of Data*, 1998, page 501.

[4] T. Bozkaya, and M. Ozsoyoglu, "Distance-based indexing for high-dimensional metric spaces", In *Proc. ACM SIGMOD International Conference on Management of Data*, 1997, pp. 357-368.

[5] T. Bozkaya, and M. Ozsoyoglu, "Indexing Large Metric Spaces for Similarity Search Queries", *Association for Computing Machinery Transactions on Database System*, 1999, pp. 11-34.

[6] S. Brin, "Near Neighbor Search in Large Metric Spaces", In *Proc. 21st. Int. Conf. Very Large Data Bases (VLDB)*, 1995, pp. 574-584.

[7] M. Charikar and R. Panigrahy, "Clustering to minimize the sum of cluster diameters", *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, July 2001.

[8] E. Chavez and G. Navarro, "Unbalancing: The key to index high dimensional metric spaces", *Technical report*, Universidad Michoacana, 1999.

[9] E. Chavez, G. Navarro, R. Baeza-Yates, and J.L. Marroquin, "Searching in metric spaces", *ACM Computing Surveys*, 2001, Vol. 33(3), pp. 273-321.

[10] P. Ciaccia, M. Patella and P. Zezula, "M-tree: an efficient access method for similarity search in metric spaces". *Proc. 23rd Int. Conf. Very Large Databases (VLDB)*, 1997.

[11] M.O. Dayhoff, R. Schwartz, and B.C. Orcutt, *Atlas of Protein Sequence and Structure*, 1978, Vol. 5. Suppl. 3: 345-358.

[12] C. Faloutsos and D. Oard, "A survey of information retrieval and filtering methods", *Technical report*, University of Maryland, College Park, MD, 1996.

[13] A. A. Gooley and N. H. Packer, *Proteome Research: New Frontiers in Functional Genomics*, chapter of *The importance of co- and posttranslational modifications in proteome projects*. Springer-Verlag, pages 65–91, 1997.

[14] L. Gravano, P.G. Panagiotis Ipeirotis, H.V. Jagadish, N. Koudas, S. Muthukrishnan, L. Pouri, and D. Srivastava, "Using q-grams in a DBMS for Approximate String Processing", *IEEE Data Engineering Bulletin*, 2001, 24(4): 28-34.

[15] D. Gusfield, Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology, Cambridge Univ Press, Jan 15, 1997.

[16] J.M. Hellerstein, J.F. Naughton, and A. Pfeffer, "Generalized Search Trees for Database Systems", *Proc. 21st Int'l Conf. on Very Large Data Bases*, Zürich, September 1995, pp. 562-573.

[17] G. R. Hjaltason and H. Samet, "Index-driven similarity search in metric spaces", *ACM Transactions on Database Systems (TODS)* , December 2003, Volume 28 Issue 4.

[18] D.S. Hochbaum, and D.B. Shmoys, "A best possible heuristic for the k-center problem", *Mathematics of Operational Research*, 1985, Vol. 10(2), pp.180-184.

[19] Q. Iqbal, and J.K. Aggarwal, "Perceptual Grouping for Image Retrieval and Classification", *3rd IEEE Computer Society Workshop on Perceptual Organization in Computer Vision*, Vancouver Canada, July 8, 2001, pp. 19.1-19.4.

[20] K. Jain and V.V. Vazirani, "Primal-dual approximation algorithms for metric facility location and k-median problems", In *Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science*, October 1999.

[21] E. Karakoc, Z. M. Ozsoyoglu, S. C. Sahinalp, M. Tasan, and X. Zhang, "Novel Approaches to Biomolecular Sequence Indexing". In the *Bulletin of the IEEE Technical Committee on Data Engineering,* September 2004, Vol. 27 No. 3, pp. 40-47.

[22] L. Kaufman and J.R. Peter, *Finding groups in data: An introduction to cluster analysis*, John Wiley & Sons, 1990

[23] U. Manber and G. Myers. "Suffix arrays: A new method for on-line string searchs", In 1st ACM-SIAM

Symposium Discrete Algorithms, San Francisco, 1990, pages 319--327.

[24] R. Mao, W. Xu, N. Singh, and D.P. Miranker, "An Assessment of a Metric Space Database Index to Support Sequence Homology", In the *proceeding of the 3rd IEEE Symposium on Bioinformatics and Bioengineering*, 2003, Washington D.C, March 10-12.

[25] D. P. Miranker, W. J. Briggs, R. Mao, S. Ni, and W. Xu, "Biosequence Use Cases in MoBIoS SQL". In the *Bulletin of the IEEE Technical Committee on Data Engineering*, September 2004,Vol. 27 No. 3, pp. 3-11.

[26] D.P. Miranker, W. Xu, and R. Mao, "Architecture and Application of MoBIoS, a Metric-Space DBMS to Support Biological Discovery", *15th International Conference on Scientific and Statistical Database Management. (SSDBM03)*, 2003, pp. 241-244.

[27] E.W. Myers, "A sublinear algorithm for approximate keyword searching", *Algorithmica*, 1994, 12(4/5): pp. 345-374.

[28] NCBI Mass Spectra data website: ftp://ftp.ncbi.nih.gov/blast/db/FASTA/swissprot.gz

[29] NCBI protein data website: ftp://ftp.ncbi.nih.gov/genbank/genpept.fsa.z

[30] Open proteomics database.http://bioinformatics.icmb.utexas.edu/OPD/.

[31] W. R. Pearson, and D. J. Lipman, "Improved tools for biological sequence comparison", *Proc. Natl Acad. Sci./ USA*, 1988, 85:2444-2448.

[32] J.T. Prince, M.W. Carlson, R. Wang, P. Lu, and E. M. Marcotte, "The need for a public proteomics repository", *Nature*, 2004, 22(4):471–472.

[33] S. C. Sahinalp, M. Tasan, J. Macker, and Z. M. Ozsoyoglu, "Distance-Based Indexing for String Proximity Search", In *IEEE Data Engineering Conference*, 2003.

[34] P.H Sellers, "On the theory and computation of evolutionary distances', *J. Appl. Math. (SIAM)*, 1974, 26: 787-793.

[35] R. R. Smriti, R. Mao, A. A. Nakorchevskiy, J. T. Prince, W. S. Willard, W. Xu, Edward M. Marcotte, and Daniel P. Miranker, "A fast coarse filtering method for protein identification by mass spectrometry", The University of Texas at Austin, Department of Computer Sciences, *Technical Report TR-05-06*. March 9, 2005.

[36] J.K. Uhlmann, "Satisfying General Proximity/Similarity Queries with Metric Trees", *Information Processing Letter*, November 25, 1991, Vol. 40(4), pp.175-179.

[37] W. Xu, W.J. Briggs, J. Padolina, W. Liu, C.R. Linder, and D.P. Miranker, "Using MoBIoS' Scalable Genome Joins to Find Conserved Primer Pair Candidates Between Two Genomes", in *proceedings of 12th International Conference on Intelligent system for Molecular Biology*, Galsgow, UK, July31-Aug05, 2004.

[38] W. Xu, and D.P. Miranker, "A metric model for amino acid substitution", Bioinformatics, 2004, 20(8):1214-21.

[39] P. Yianilos, "Data structures and algorithms for nearest neighbor search in general metric spaces", In *Proc. 4th ACM-SIAM. Symposium on Discrete Algorithms (SODA'93)*, 1993, pp. 311-321.

[40] P. Yianilos, "Excluded middle vantage point forests for nearest neighbor search", In *DIMACS Implementation Challenge, ALENEX'99*, Baltimore, MD, 1999.

[41] W. Zhang and B. T. Chait, "Profound - an expert system for protein identification using mass pectrometric peptide mapping information", *Anal. Chem*., 2000, 72(11):2482–2489.