Metric-Space Search of Protein Sequence Databases

Weijia Xu[1], Rui Mao[1], Shu Wang[2], Daniel P. Miranker[1]
[1] Department of Computer Sciences
& Center of Computational Biology and Bioinformatics
[2] Department of Electrical and Computer Engineering
University of Texas at Austin

Abstract

Motivation: Growing sequence databases are instigating sequence retrieval systems that construct k-mer (hot-spot) indexes off-line to speed up the on-line query execution. For fixed k, the content of each index bucket grows along with the database, diminishing the effectiveness of the index. Thus it is important to establish effective methods beyond simply indexing BLAST hot-spots, where k equals 3 with concomitant 8,000 index buckets.

Results: We investigate an evolutionary criterion to directly retrieve an evolutionary neighborhood of k-mers. The method uses metric-space search and an overlapping k-mer representation of protein databases. We prove a lemma that enables the comparison of two k-mers using weighted-Hamming distance in-lieu of global alignment, yielding an O(k) speed-up. We evaluate the trade-offs between scalability, speed and accuracy and assess several k-nearest neighbor search algorithms. The results extend k to 6 and over 60 million buckets, achieving better scalability and maintaining comparable search accuracy as BLAST.

Availability: http://www.cs.utexas.edu/mobios/

Contact: Miranker@cs.utexas.edu

1 Introduction

The increasing workload of sequence analysis has been the driving force of several efforts to organize the k-mers of a sequence database in data structures, offline, to speed up the execution time of on-line queries (Giladi et al., 2002; Kent 2002; Fondrat and Dessen, 1995; Tan et al., 2003; Williams and Zobel, 2001; Halperin et al, 2003;). The general method to efficiently find homologous regions between sequences consists of two steps. The first step is to identify evolutionarily close k-mers (substrings of length k) between a query sequence and sequences in the database. The second step is to stitch these k-mers together to approximate a local alignment between their parent sequences and the query. Wilbur and Lipman first proposed such an algorithm. Their method finds similar sequences by identifying common k-mers between a query sequence and database sequences through hashing (Wilbur and Lipman, 1983). Similar approaches were then adopted by popular exhaustive sequence searching tools, such as FASTA and BLAST, which require a linear scan of the entire database for each search (Pearson, 1988; Altschul et.al. 1990, 1996).

A major goal of our approach is to quickly and *directly* retrieve matching k-mers within a biologically meaningful neighborhood of each k-mer in the query sequence. The use of a metric-space index to implement such retrievals was enabled by the recent development of a validated metric substitution-weight matrix for amino acids, mPAM (Xu and Miranker, 2004, Sellers 1974). This is in contrast to the generate-and-test method, pioneered by BLAST, which relies on conventional indexing or exact matching methods. In BLAST, for each k-mer in the query, all k-mers within a predetermined similarity threshold are generated. This entire set of k-mers is compared against the database for exact matches (Altschul et. al., 1990). Accuracy entails a trade-off between k-mer length (hot-spot) and similarity threshold. For protein searches BLAST performs best at its default, the length of 3 k-mers. However, such a short k-mer length results in excessive duplicate entries as the volume of data grows, which subsequently increases the workload of the algorithm performed after k-mer retrieval. Our results show that directly retrieving an evolutionarily close neighborhood around a longer k-mer not only achieves comparable accuracy but also better scalability and speed.

Metric-space indexing exploits the triangle inequality and the intrinsic clustering of a dataset to prune a search space without regard to a mapping of the data to a coordinate system (Chavez et al., 2001). We measure the distance between two k-mers using weighted Hamming distance parameterized by the mPAM substitution matrix (Xu and Miranker 2004). Hence, for each k-mer in the query sequence, a set of evolutionarily close k-mers can be retrieved with only one query. Due to the curse of dimensionality, such methods are unstable and must be verified for each application (Yianilos, 1993; Chavez, et al., 2001; Mao, et al., 2004). We exploit a tree-based metric-space index structure, the multiple vantage point tree (MVPT). We constructively show that the mPAM distance between protein k-mers induces a clustering effective for protein database retrieval. The MVPT has also been used for string proximity search where the distance between two k-mers is measured by compression distance or character edit distance (Bozkaya and Ozsoyoglu, 1999; Sahinalp, et. al., 2003).

Our method is also different from other indexed search methods, which are often tailored to one particular problem, and/or designed only for nucleotide acid retrieval. Due to alphabet size, etc., few have claimed results for general protein sequence management and retrieval. The RAMdb is mainly used to identify short subsequences (Fondrat and Dessen, 1995). The Ed-tree is designed for nucleotide sequences (Tan et al., 2003). The Cafe system uses an inverted index to support a coarse search that produces a small candidate set for detailed assessment (Williams and Zobel, 2001). Cafe is applicable to both amino acids and protein sequences where complex scoring models are not used during coarse searching. The approach of using a coarse search followed by a fine search is also used in MAP, where the coarse search is sped up by indexing a frequency vector (Kahveci, et. al., 2004). BLAT is based on simple edit distance supported by hashing to achieve O(m) scalability using O(n) memory, where m is the size of the query and n is the size of the database (Kent, 2002). SST and BLAT approaches use Hamming distance and simple edit distance, respectively. Initial success was achieved by targeting the sequence assembly problem where evolutionary criteria are unimportant. Subsequently these systems are being effectively

applied to genomic analysis problems whose data is limited to sequences from evolutionarily close organisms (Rouchka et al., 2002).

Exploitation of metric properties was first proposed by Sellers (Sellers 1974). Sellers showed that if a substitution weight matrix forms a metric distance on a set of characters, then the global alignment of sequences drawn from that character set also forms a metric. We have previously shown, using Smith-Waterman local-alignments, that the mPAM matrix is the first metric substitution weight matrix to produce biologically effective protein alignments (Xu and Miranker, 2004).

Since only global alignment and not local alignment forms a metric, our off-line index is based on the weighted Hamming distance of overlapping protein k-mers. We prove a general result which defines conditions where, by virtue of k-mer overlap, gaps (indels) can be ignored in the management of the index and weighted Hamming distance produces the same results as global alignment. This yields an O(k) improvement.

Using a curated benchmark of yeast protein sequence queries **(**Schaffer et al., 2001), we first analyze the effect of both word length and the search range of each k-mer query on the trade-off between the accuracy and speed of the system. To improve the search efficiency, we develop a heuristic *k*-nearest neighbor search and then compare its results with exact nearest neighbor and radius-limited *k*-nearest neighbor search. The results demonstrate accuracy comparable to BLAST. We further compare the search performance of our system with BLAST using a set of short query sequence against a protein sequence database downloaded from Swissprot. The results show that our current disk-based Java implementation outperforms BLAST in some cases. Lastly, we show the scalability results of querying different size databases. Thus, indexing sequence data in metric-space with relatively longer word length is a feasible approach to achieve fast and scalable search performance for large-scale data with accuracy comparable to BLAST.

2. Algorithms and Implementation

Given a set of sequences *S* and a query sequence *w* of length *l*, the goal of a homology search is to find all sequences from *S* that are evolutionarily close to *w*. To efficiently solve this problem, we first divide *S* into a set of overlapping k-mers of length k with step size 1 and manage them in an MVP tree, *D*. During the online search, the query *w* is also divided into a set of overlapping k-mers of length k with step size 1, $\mathcal{F}=\{w_i| i=0..l\text{-}q\}$. For each k-mer w_i in $\mathcal{F}$, we select a set of k-mers R_i from database D such that all k-mers in R_i are similar to w_i. Then a heuristic chaining algorithm is used to extend and chain all fragments in $R_0 \cup R_1 \cup \ldots \cup R_{W-T}$ to obtain the result of the homology search for query W. We implement the chaining algorithm described by Gusfield with some modifications (Gusfield, 1997). Although the chaining algorithm is an important issue and a major factor in determining the accuracy of the homology search results, it is not detailed in this paper since the chaining phase only accounts for a small portion of the computational cost in our approach. In this section, we detail our algorithms and the distance function used to build and search the database.

2.1 Building index using MVP tree

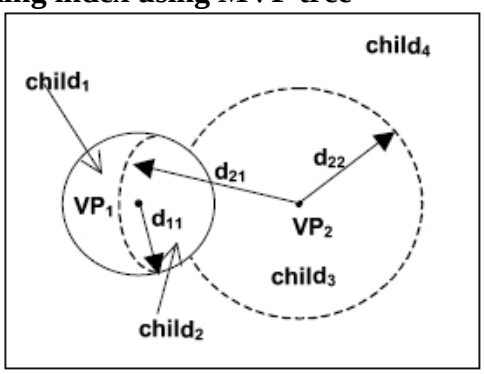

Figure 1 Vantage point tree structure

When indexing biological sequence data, we first determined that among three major classes of metric-space search algorithms, multiple vantage point methods perform the best (Chavez et al., 2001; Bozkaya and Ozsoyglu, 1999; Mao et. al., 2004). The structure of a node in multiple (two) -vantage-point tree is shown in Figure 1. Starting with the root of the tree, the distance from each point in the space is measured to two special points, the vantage points. As illustrated, bounding predicates of the form d(VP, x) > r are used to partition the data into four equal-size disjoint subsets. The process is repeated recursively. The triangle inequality, integral to metric-distance functions, is used to prune the search of the resulting tree for data retrieval operations.

Our MVP tree is implemented in Java and also serves as the disk-mapped storage manager of MoBIoS (the Molecular Biological Information System) (Miranker et al., 2004). MoBIoS is built on the Mckoi open-source relational database engine which provides a generic IO management system to map the data records to disk pages (http://www.mckoi.com/). Vantage-points were selected using the farthest-first traversal algorithm (Hochbaum and Shmoys, 1985). Parameters related to the MVP-tree structure have been evaluated in preliminary work. The number of vantage points and the termination of the recursion to build the tree were chosen to optimize retrieval times for a disk-based implementation (Xu, et. al., 2004).

2.2 Distance function

As a prerequisite to this work, and inspired by Sellers' seminal paper on sequence matching, we derived a metric version of the PAM250 substitution matrix, mPAM, and demonstrated that is has comparable accuracy to the original PAM250 (Xu and Miranker, 2004, Sellers 1974). Using mPAM, it follows from Seller's theorem that global alignment distance among k-mers forms a metric.

We extend these results to show that when the k-mers are derived as overlapping substrings from a much longer sequence, weighted Hamming distance can be substituted for global alignment distance in the management of a database of protein k-mers. Although the use of an index structure substantially reduces the total number of distance calculations, the distance calculation still dominates the computational cost of retrieval. Replacing the global alignment distance function, $O(k^2)$, with a Hamming distance function, O(k), results in a practical factor of k improvement in speed.

Definition 1 Substitution Cost Function The substitution cost function M(x,y), where x and y are symbols from an alphabet $\mathcal{A}$, returns a nonnegative real number modeling the cost of substituting sequence element x with element y. Substitution weights are usually encoded in a substitution weight matrix; If '_' denotes a gap, M(x,_) or M(_, x) returns a gap penalty g.

Definition 2 Global Alignment Distance Function Given the substitution cost function M(x,y), two sequences A: $a_1a_2\ldots a_n$ and B: $b_1b_2\ldots b_n$, where x, y, a_i and b_i are drawn from an alphabet $\mathcal{A}$, the global alignment distance function G(A, B) is defined as (Gusfield, 1997)

$$G(A,B)=S_{n+1,n+1}$$

where S is a n+1 x n+1 matrix and

$$S_{0,i}= g\times i \quad \text{for } i=0\ldots n+1$$

$$S_{j,0}= g\times j \quad \text{for } j=0\ldots n+1$$

$$S_{i,j} = \text{Min}(S_{i-1,j-1}+ M(a_i, b_i), S_{i-1,j} + M(a_i, \_), S_{i,j-1}+M(\_, b_i)) \quad \text{for } i,j \neq 0$$

Definition 3 Weighted Hamming Distance Given substitution cost function M(x,y), two sequences A: $a_1a_2\ldots a_n$ and B: $b_1b_2\ldots b_n$, where x, y, a_i and b_i are drawn from an alphabet $\mathcal{A}$, the weighted Hamming distance function H(A, B) is defined as

$$H(A,B)= \sum_{i=1}^{n} M(a_i,b_i)$$

Definition 4 Range Query Given a set of objects O and a distance function d(a,b), a range query $Q_d(q,r)$ returns a set of objects $\{o \in O \mid d(o,q) \leq r\}$.

Lemma 1 Given a substitution cost function M, a set of k-mers O, a radius r, and a gap penalty g, for all queries $q \in O$, if $r \leq 2g-1$ then $Q_H(q,r)=Q_G(q,r)$, where g is the gap penalty used in both H and G.

Proof:

This lemma is proved by contradiction. From the definition of the global alignment problem, it is trivial to show that the following two properties hold:

P1. $G(x,y) \leq H(x,y)$ for any x, y, and

P2. $G(x,y) = H(x,y)$, if there is no gap in the optimal global alignment.

a) Assume there is a k-mer $m \in O$, such that $m \in Q_H(q,r)$ and $m \notin Q_G(q,r)$ when $r \leq 2g-1$

By the range query definition, $H(m, q) \leq r$. P1 implies that $G(m,q) \leq H(m, q) \leq r$. So m must be in $Q_G(q,r)$, which contradicts the assumption that $m \notin Q_G(q,r)$. Hence such m does not exist.

b) Assume there is a k-mer $p \in O$, such that $p \in Q_G(q,r)$ and $p \notin Q_H(q,r)$ when $r \leq 2g-1$

The assumption indicates that $Q_H(q,r) > r > 2g-1 \geq Q_G(q,r)$. Since $2g-1 \geq Q_G(q,r)$, there must be no space inserted into either p or q to form the optimal global alignment. Otherwise $G(p,q) \geq 2g$. From P2, it follows that $G(p,q) = H(p,q)$ and $p \in Q_H(q,r)$, which contradicts the assumption $p \notin Q_H(q,r)$. Hence, if $p \in Q_G(q,r)$, then $p \in Q_H(q,r)$ i.e. $Q_G(q,r) \subseteq Q_H(q,r)$ when $r \leq 2g-1$.

Therefore, from a) and b), $Q_H(q,r)=Q_G(q,r)$ when $r \leq 2g-1$. □

Gap penalty is usually larger than mismatch score (Gusfield, 1997). In the course of our experiments, when it became clear that there was no need to perform any queries with radius bigger than 2g-1, we moved to weighted Hamming distance and witnessed precisely a factor of k improvement in execution times.

2.3 Search algorithms

We consider standard and heuristic search methods to attain accurate, scalable results. Although there may be any number of k-mer matches in a given evolutionary distance, more conserved k-mers contribute more to determine homology. Consider, as the database grows, the number of k-mers within a fixed distance of a query k-mer grows. This by itself can prevent scalability (sublinear execution time). In addition, due to the characteristics of peptides, a single distance value can hardly define the optimal boundary between similarity and dissimilarity for all k-mers. The distribution of amino acids is non-uniform, and the distance distribution among k-mers is also non-uniform. For example, k-mers from a highly conserved protein motif occur more frequently than random, and k-mers containing highly mutable amino acids have more similar k-mers within any reasonable distance. Therefore, biasing the search toward the conserved k-mers is vital if there are many k-mers to be retrieved by a range search. Justified by similar practical aspects of other applications, Yianilos proposed radius-limited nearest neighbor search (Yianilos, 2000). The basic idea is to return the nearest neighbor provided that it is within a maximum distance. Yianilos conducted an analytic study of this approach based on the Euclidean hypercube with uniform distributions. To our knowledge, our radius-limited *k*-nearest neighbor search methods are the first to be applied to practice.

Definition 5 Radius-limited nearest neighbor (RNN) search Given a set of objects D, a distance function d(a,b), and a radius r, the RNN query returns the set of objects from D that (1) are nearest to query object q and (2) have a corresponding distance no greater than r as measured by the distance function d. If more than one object qualifies, all qualified objects will be returned.

Nearest neighbor search algorithms normally utilize a priority queue to store the nodes to be searched (Uhlmann, 1991). The priority queue orders the nodes that may contain the nearest match to the query based on certain sorting criteria. This algorithm is guaranteed to find the best match within a limiting radius. At first, it may sound like a good alternative to range search. However, using nearest neighbor search alone is inadequate to achieve the desired accuracy. Consider the case where two homologous sequences share few common k-mers. The homologous sequence in the database consists of k-mers that are very close to those in the query but not identical. The nearest neighbor search can ignore those k-mers and return the exact match randomly occurring in an unrelated sequence. In such a case,

additional k-mers need to be retrieved instead of just the nearest ones. Thus radius-limited *k*-nearest neighbor search is a better solution.

Definition 6 Radius-limited *k*-nearest neighbor (RKNN) search Given a set of objects D, a query q, a number *k*, and a distance function d(a,b), the search Q(q, *k*, r) returns up to *k* closet objects whose distance to the query object q is no greater than r: $S=\{s|\ s\in D;\ d(s,q)<r;\ d(s,q)\leq d(t,q)\ t\in D,\ t\notin S\}$.

The radius-limited *k*-nearest neighbor search Q(q,k,r) has the following property: If the number of k-mers within distance r to q is less than *k*, all of the qualified k-mers will be returned. If the number of k-mers within distance r to q is more than *k*, the *k*-nearest qualified k-mers will be returned. Intuitively, the *k* best results for k-mers with high mutability will be within a smaller radius than those for k-mers with low mutability. The worst case is that the total candidates within the given distance bound are less than *k*. The computational cost for the worst case is equivalent to the cost of a range search plus some overhead.

In addition to the one priority queue used in the NN search, the radius-limited *k*-nearest neighbor search algorithm requires a second priority queue. The priority queue *node_list* stores the internal nodes which may contain qualified k-mers and need to be searched. The additional priority queue *cached_result_list* stores the k-mers that are within the limiting radius but are not yet returned. The objects in both priority queues are sorted in the ascending order of their distance to the query k-mer. The pseudocode of the algorithm is presented in Figure 2.

```
Radius-limited-k-nearest-neighbor-search (q, k, r)
1.  Add root_node into the node_list
2.  While the node_list is not empty, get node n from the head of the node_list.
3.     if the minimum possible distance, md, between q and objects in subtree rooted at n is greater than current_target_radius
4.           move all previously found k-mers that are within current_target_radius to query q from the cached_result_ list into the
             final_result_list.
5.           If the final_result_list contains k results whose distance to q is no greater than the current_target_radius, then stop the
             search and return these results.
6.           Otherwise update the current_target_radius as md.
7.     if n is an internal node with c1…ck children
8.           For each child node ci
9.              get md as the lower distance-bound for k-mers in the subtree rooted at ci to q
10.             if md is smaller than radius r, add ci in the node_list and ordered by md.
11.    if n is a leaf node with r1…rm data objects
12.          For each data object ri , compute d as the distance between ri to q
13.             If d is smaller than radius r put ri in the cached_result_list ordered by d.
14. Return final_result_list
```

Figure 2 Pseudocode for retrieving up to k-nearest neighbors that are within distance bound r to a given query object q.

In Figure 2, lines 1 and 2 create a priority queue *node_list* for storing internal nodes to be searched and then start the search process. Lines 3-6 maintain *cached_result_list*, the priority queue of returned results. The *current_target_radius* is the expected nearest distance and is initialized as zero. If a new result is found, lines 4 and 5 check if there are already *k* results. If not, the *current_target_radius* is updated by line 6. Lines 7-10 search an internal node. Line 9 computes node *c*'s *md*, which is the lower distance-bound between query q and the k-mers stored in the subtree rooted at node *c*. If the *md* is smaller than the limiting radius, then node *c* will be put into *node_list* which is sorted based on *md*. Lines 11-13 search a leaf node. If a qualified result is found, it is put into the *cached_result_list* sorted by its distance to q.

However, in the case of searching k-mers with high mutability, the length of the two priority queues can increase dramatically causing the overhead of the queue operations to offset the advantages of best-first search. Therefore we propose a more aggressive heuristic search strategy, the radius-limited extended *k*-nearest neighbor search (EKNN).

Definition 7 Radius-limited extended *k*-nearest neighbor search (EKNN) Given a set of objects D and a distance function d(a,b), the EKNN query $Q_d(q, k, r)$ will return up to *k* objects from D as set *N*, which contains all of the nearest neighbors to query q and the object whose distance to q is smaller than r.

The EKNN algorithm relaxes the requirement of non-best results to be returned. Specifically, the EKNN algorithm replaces lines 4 and 5 of RKNN as presented in Figure 2 as following:

4. Move the *cached_result_ list* into the *final_result_list*.
5. If the *final_result_list* has *k* results and contains results whose distance to *q* is no greater than the *current_target_radius*, then stop the search and return these results.

In the RKNN algorithm, at any given time there is no k-mer that is closer to the query than the k-mers in the *final_result_list*. The EKNN algorithm does not have such properties. After finding the nearest neighbor, the EKNN will continue searching for the next nearest neighbors until either a total of *k* results within the limiting radius are found or all of the nodes are searched. In the case where the total number of k-mers within limiting radius is no more than *k*, both RKNN and EKNN will return the same results. In the case where the total number of k-mers within limiting radius is far more than *k*, the non-nearest results returned by EKNN can vary greatly from those of RKNN. However, in the latter case, EKNN will run much faster than RKNN. In fact, as we will show in the results section, with our test settings EKNN achieved very similar accuracy results to RKNN with much fewer distance calculations.

3. Experimental Results

The primary results demonstrate that the MVP-tree provides accurate and scalable performance for identifying homologous peptides with a metric distance function parameterized using mPAM. We will first focus on the trade-off between accuracy and speed caused by choice of fragment length and search radius. We then present the results of using nearest neighbor search algorithms, which show better scalability and trade-off between accuracy and speed than using fixed radius search.

3.1 Choice of fragment length and search radius

To study the trade-off between selectivity (the ability to reduce false positive results) and sensitivity (the ability to identify true positive results), we first determine a search radius for each fragment length that ensures biologically meaningful answers. The length of k-mer and the search radius are the two most important parameters since, independent of data structure, they determine the size of the set of k-mers returned by the search. The performance of the subsequent chaining algorithm is a function of the size of the result of the k-mer search.

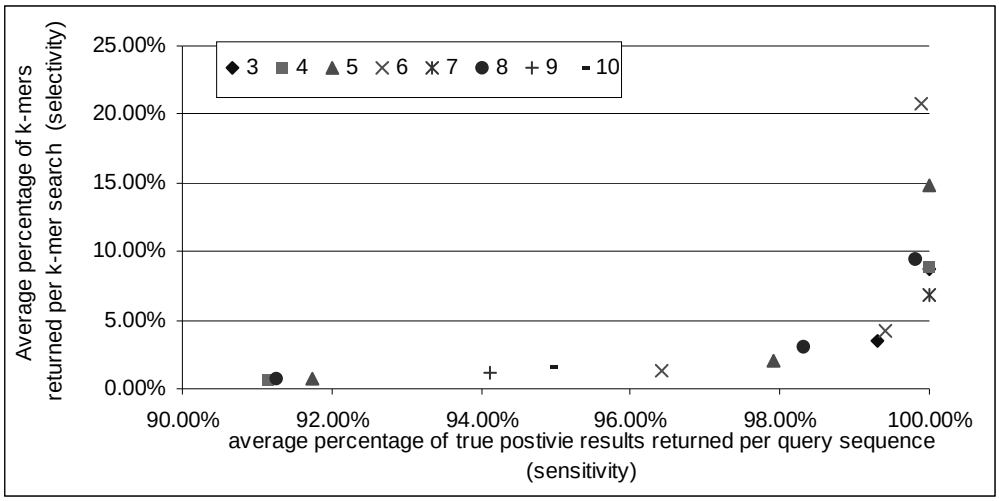

Figure 3 The trade off between selectivity and sensitivity for different search radius and fragment length combinations Range searches with various radii were run on databases of total true positive hits built from the benchmark with various k-mer lengths. The sensitivity, or average percentage of true positive hits returned per query sequence, is plotted against the selectivity, or average percentage of k-mers returned per k-mer search, for different radius and k-mer length combinations.

For this purpose, we used a curated benchmark suite furnished by NCBI (ftp.ncbi.nlm.nih.gov/pub/impala/blastest). The benchmark includes 6433 yeast protein sequences (about

2,892,155 residues) and a query set of 103 sequences whose true positive hits have been identified by human experts (Schaffer et al., 2001). The initial goal was to determine the value of the search radius that is necessary to identify all of the true positive hits for various k-mer lengths. For a sensitivity study, we only need to consider k-mers from the query sequences and their corresponding true positive hits. To reduce the computational cost of the study, only the k-mers from the true positive hits set, about 16% of the entire sequence set, were used to build databases. This reduction does not impact the qualitative study of selectivity. Every query in the benchmark was searched against those databases with various radii.

Figure 3 plots selected points whose average percentage of true positive hits returned per query sequence were higher than 90%. Table 1 details selected points whose average percentage of true positive hits returned per query sequence were higher than 95%. The average percentage of true positive hits returned per query is considered as a measure of sensitivity. The average percentage of the total k-mers in the database returned per k-mer is also computed as a reference of selectivity. Ideally, we are looking for the combination of fragment length and search radius that minimizes the size of each query results and still contains at least one k-mer from every expected answer sequence.

Fragment length	Search Radius	Average % true Positive Hits Returned per query search	Average % k-mers returned per k-mer search	TF-ratio Ratio of column 3 to column 4
3	0	99.30%	3.53%	28.13388423
4	2	100.00%	8.85%	11.30505334
5	3	97.91%	2.10%	46.53131343
5	4	100.00%	14.87%	6.726649907
6	4	96.42%	1.35%	71.42626678
6	5	99.40%	4.25%	23.36353288
7	6	98.41%	2.18%	45.19603877
8	7	98.31%	3.11%	31.65246309
8	8	99.80%	9.40%	10.6207366
9	11	99.70%	11.92%	8.36498572

Table 1 Comparison of selected results for various fragment lengths and search radii In general, using longer fragments requires a larger search radius to achieve the same true positive return rates as those obtained using shorter fragments.

The results in column 5 of Table 1 show the TF-ratio, or the ratio of the average percentage of returned true positive hits to the average percentage of returned k-mers per k-mer. The combination of fragment length 6 and search radius 4 has the highest TF-ratio of 71.4. Next are fragment length 5 with radius 3 (TF-ratio 46.5) and fragment length 7 with radius 6 (TF-ratio 45.2).

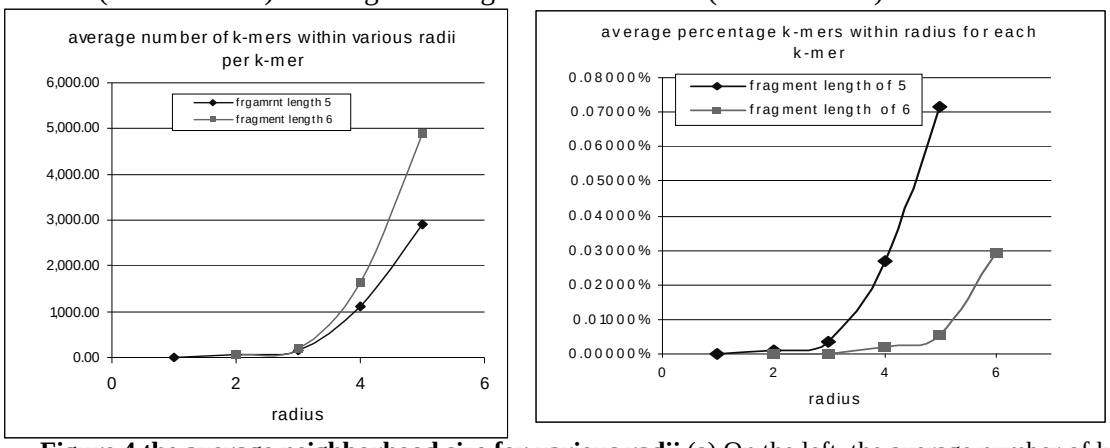

Figure 4 the average neighborhood size for various radii (a) On the left, the average number of k-mers within a given radius. (b) On the right, the average percentage of k-mers within a given radius.

For longer k-mer lengths, increasing the search radius used per k-mer range search can improve the sensitivity but also increases the total number of k-mers returned. This phenomenon is known as "the curse of dimensionality" (Bellman 1961). To illustrate this problem for k-mers of lengths 5 and 6, we exhaustively computed the numbers of fragments within various radii (the neighborhood size) for all k-mers of lengths 5 and 6 (Figure 4). For both lengths 5 and 6, the size of the neighborhood increases dramatically after radius 3. The average neighborhood size of radius 5 is about three times the average size of radius 4. Since increasing the length of k-mer from 5 to 6 also increases the entire search space from about 3M to 60M, the average number of 6-mers within the distance increases dramatically when the distance is greater than 5.

3.2 Comparison of search strategies

In this section, we evaluate the biological effectiveness of our approach using various algorithms for each k-mer search. Both the fixed radius range search and the radius-limited *k*-nearest neighbor search can achieve results with comparable accuracy to other homology search algorithms. One of the challenges in our approach is to determine if there is a way to produce biologically effective results without encroaching too far beyond the knee of the curse (which would swamp the computation time for chaining) during the k-mer search. The results in this section show that algorithms based on radius-limited *k*-nearest neighbor searches are the most preferable.

To evaluate the biological effectiveness of our approach, we use the complete yeast data set included in the benchmark as mentioned in the previous section (Schaffer et al., 2001). All 6433 yeast protein sequences are used to build an MVP-tree with 2 vantage points per node, 2 partitions per vantage points and a maximum of 150 k-mers per leaf node. The tests were conducted using Java 1.4 for Linux (SUSE 8.0; dual AMDXP 1800+ processors with 2GB memory). Each query sequence of length n was divided into a set of k-mers referred to as the query fragment set $\{f_i | i=1,\ldots n-q+1\}$. The length of the query sequence varies from 39 to 885. On average, 174 k-mer searches are needed for each query sequence. Each k-mer from the query fragment set f is sequentially searched against the database. The results are collected and used in a chaining algorithm to produce the final answers.

We use receiver-operating characteristic (ROC) scores to measure the accuracy of the search result (Gribskov and Robinson, 1996). For each query, the ROC_{50} value is computed as follows:

$$ROC_n = \frac{1}{nT} \sum_{i=1}^{n} t_i$$

where t_i is the number of true positive hits ranked ahead of the ith false positive, and T is the total number of true positives.

We first used 5-mers for both indexed sequences and query sequences. For each 5-mer from each query sequence, a set of k-mers is retrieved using fixed radius range search. The average ROC_{50} score for all queries is 0.45 and 0.53 by using radius 3 and 4 range search for each k-mer, respectively.

The results of using radius 4 range query against the 5-mer database show accuracy comparable to the results of using BLASTp (Xu and Miranker, 2004). However, as indicated in Figure 4b, radius 4 is a relatively large value for 5-mers. Due to the curse of dimensionality, each k-mer from the query sequence requires on average over 160k distance calculations for range 4 search against a database with 3 million k-mers. Due to the non-uniform distribution of the amino acids, the number of k-mers in a neighborhood of radius 4 varies greatly. For k-mers that contain more common amino acids, more matches will be found within the same radius than k-mers with less common amino acids. Hence, range 4 is not a fair measure for all k-mers. On the other hand, there are about 3 million unique peptides of length 5. As the database size grows larger, more duplicate entries will reduce the selectivity of the search and increase the workload of the successive chaining algorithm. Hence, a longer k-mer and a smaller radius are desired for better selectivity and speed. Therefore we think using a best-first search strategy on longer k-mers is the best trade off among speed, selectivity and accuracy.

To avoid excessive duplicate short strings, we build a database of 6-mers. In the previous section, we described three best-first search algorithms: RKNN, RNN, and EKNN. For those algorithms, the trade off between speed and accuracy is controlled by the value of *k* and, if applicable, the limiting radius.

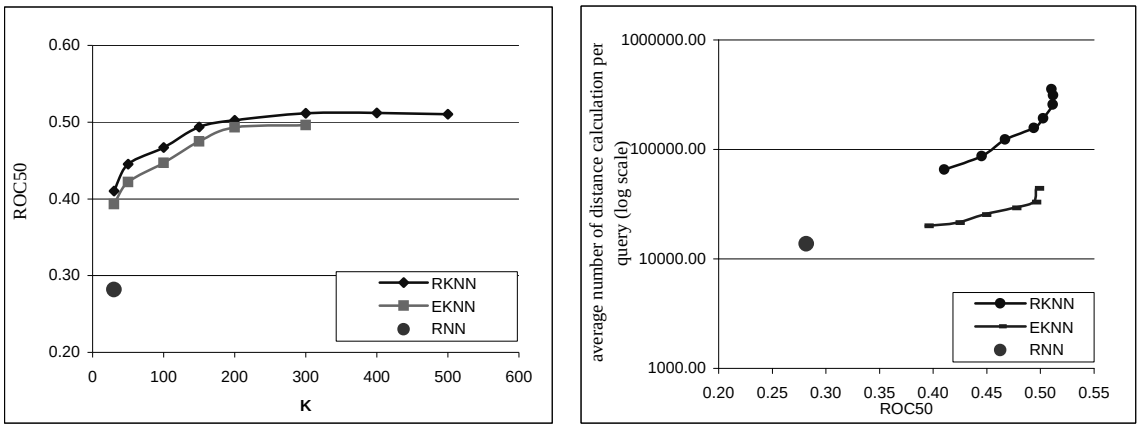

Figure 5 (a) on the left, the average ROC_{50} score per query sequence with various algorithms and *k* values; (b) on the right, the average number of distance calculations needed for various average ROC_{50} scores per query sequence.

Figure 5(a) illustrates the average ROC_{50} values for RKNN and EKNN algorithms with various *k* values and the RNN algorithm of which the *k* value is fixed by definition. Figure 5(b) plots the average number of distance calculations (as a measure of CPU time during the database search) needed per query to achieve various ROC_{50} levels. For both EKNN and RKNN algorithms, a large *k* value results in better accuracy. The improvement of accuracy gets smaller once the *k* value reaches 300, and the average ROC_{50} score seems capped at 0.51. Although the RNN search has the least computational cost, the accuracy of the search results is much lower than other methods. The EKNN search significantly increases accuracy compared with the RNN search and achieves similar accuracy as the RKNN algorithm. However, the RKNN search is about one order of magnitude slower than EKNN. Therefore, we believe the EKNN is a good approximation for RKNN search that yields much better performance.

3.3 Comparison of search speed

Although our approach is implemented in Java and integrated into a full database system, the EKNN algorithm still performs reasonable fast. In this section, we present some timing results of our approach using the EKNN algorithm with various parameters against another benchmark. The yeast benchmark used in the accuracy study is a small dataset of about 3 million amino acids. The lengths of the query sequences vary from 39 to 885 amino acids with an average number of 174. For further speed tests, we adopted another benchmark with a larger data set and short query sequences. The short query set minimizes the effect caused by the difference of chaining algorithms. This benchmark was used for the OASIS analysis and consists of the SWISS-PROT database and a hundred queries that were randomly selected from the ProClass motif database (Meek et.al., 2003). The SWISS-PROT data set was downloaded in Oct. 2004 and contains 163496 peptides with about 52M amino acids. The queries range in length from 6 to 56 amino acids with an average length of 16.

Since the accuracy of the EKNN algorithm is affected by the value of *k* and the limiting distance r, we conducted the speed comparison with several different configurations. The results were compared to the BLASTp program running on the same machine, a Linux computer with a 2.6GHZ P4 CPU and 4G of memory.

Figure 6 shows the average search time per query over the 100 queries. The BLAST search was conducted with an e-value of 20,000. The EKNN algorithm is parameterized with various *k* values and limiting radii. With a limiting radius of 5, the algorithm outperformed BLAST when *k* is no bigger than 500. The algorithm also outperformed standard BLAST with limiting distance of 6 even for *k* value of

1000. Note that the *k* is the number of k-mers returned per k-mer search and is not the number of final results per query sequence.

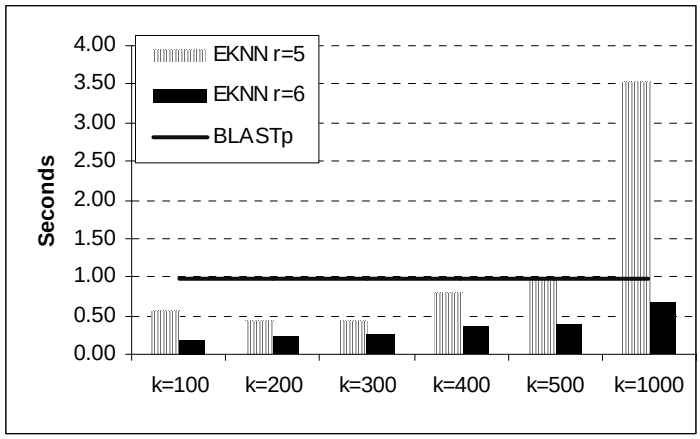

Figure 6 Comparison of average running time per query between using Blast for homology search and using indexed search with various k values. The Blast time, shown by a solid line, is the average wall clock time per query using an e-value of 20000. The average indexed search time includes both k-mer searching time and subsequent chaining time per query. The striped bar and solid bars show the results of using limiting radius 5 and 6, respectively.

3.4 Scalability of best first search

This section demonstrates the scalability of our approach. The test data sets are sequentially extracted from a large protein sequence data file (ftp://ftp.ncbi.nih.gov/genbank/genpept.fsa.z, downloaded from NCBI in July 2003) so that their size varies. The same short query set used for speed comparison was searched against those newly generated databases.

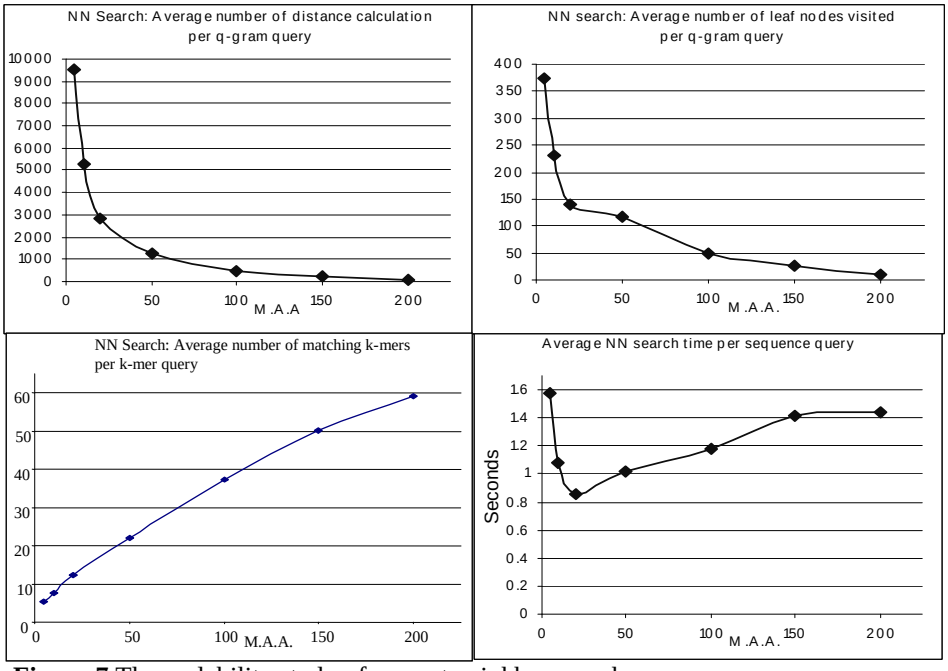

Figure 7 The scalability study of nearest neighbor search.

Figure 7 shows the scalability of the RNN search. During nearest neighbor search, the algorithm always first looks for the node that may contain the nearest neighbor, starting with the expectation of exact matches. However, when the data size is relatively small, the chance that the node does not contain the nearest neighbor is greater. As the database size grows, such chances diminish. Hence, the search

backtracks less when the database size is larger. The results show that although the average number of matching k-mers increases linearly as the database size increase, both the average number of distance calculations and the average number of leaf nodes visited decreases dramatically.

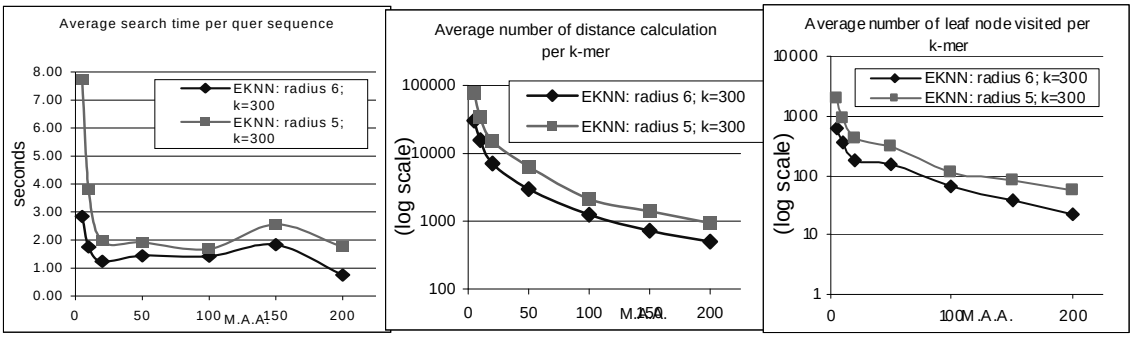

Figure 8 Scalability study of the EKNN algorithm with k =300, and a limiting radius of 5 and 6.

We expect the EKNN algorithm to have similar behavior when database size increases. Figure 8 shows the average search time per query sequence, the average number of distance calculations and the number of leaves visited per k-mer search using the EKNN algorithm with a k of 300 and a limiting-radius of 5 or 6. The results show the same decreasing behavior as the RNN searches.

4. Discussion

The homology search problem is an important practical problem that is often solved by a general strategy consisting of several heuristic algorithms. The quality of the results depends on many factors, such as the scoring model, the alignment algorithms, the similar fragment retrieval algorithm and the chaining algorithm. In this paper, we focus on the algorithms used to efficiently retrieve similar fragments.

Our homology search algorithm modifies a general framework for computing local-alignments by matching k-mers that was first proposed and analyzed contemporaneously with the development of BLAST (Wu et al., 1990). The primary change is that, by virtue of the metric-space index, we are able to look up, on-line, all matching k-mers in the database for each query fragment. In other words, we replace the generate-and-test method with a more powerful index that directly finds matching fragments based on evolutionary criteria.

Important to the effectiveness of the index structure is to maximize the number of indexable buckets by increasing the length of the k-mers. To maintain query sensitivity, as k increases, so must the evolutionary distance. As detailed by Myers' the useful neighborhood grows polynomial of k (Myers, 1994). Thus, there is a computational need for generate-and-test methods to keep k small. The optimization of trade-off for BLAST resulted in hot-spots determined by 3-mers. The number of unique 3-mer peptides is just 8000. For the yeast proteome of roughly three-million amino acids this results in an average occupancy of nearly 400 k-mers per bucket. Thus, indexing methods for protein sequence management are more often based on identifying commonly conserved motifs and/or similarity models that are effective only between more closely related organisms. Using a metric-space indexing method we have shown that there is a *sweet-spot* in the trade-offs at k=6. There are over 60 million unique 6-mer peptides enabling an effective protein index based on evolutionary criteria.

We have demonstrated that the use of radius-limited k-nearest neighbor search algorithms provides a resolution to this trade-off. Furthermore, the database is queried on a per k-mer basis. So the longer the homology region is, the less chance that a "good" answer will be missed. Our most aggressive heuristic algorithm (EKNN) enables speed and scalability without appreciable loss in accuracy. Although we confirm empirically that the curse of dimensionality affects biological sequence data, radius-limited k-nearest neighbor search may be a way to achieve good accuracy without encroaching fatally into the unstable region.

The number of nearest neighbors to be retrieved is an important factor in determining the quality of the final results. Our choice is based on the empirical results. How to determine the number of nearest neighbors to be retrieved, especially when the database is growing? This question has been studied but in

a different form. Key elements of BLAST include filtering-out low complexity regions and computing a probabilistic significance score (e-value) for each output sequence output (Altschul and Gish, 1996; Promponas et al., 2000; Wootton and Federhen 1993). These elements taken together suggest that the ultimate algorithm for scalable sequence retrieval will comprise a depth-first search strategy where the scope of the search is parameterized by the anticipated significance of the next matching fragments. We leave this as an open issue for further pursuing.

Our implementation is not a standalone application or index structure, but rather a part of an extension to object-relational databases intended as general-purpose support of biological data. The approach follows the approach of spatial databases that extend object-relational database to support geographic data (Shekhar and Chawla, 2002). In that regard, we have shown that starting from the underpinnings presented in this paper, solutions to many bioinformatics problems can be solved concisely and quickly using simple SQL queries (Miranker, et. al., 2004). Our concentration has been on managing large-scale biological data across biological disciplines and enabling databases management systems to serve biologists as well as they serve businesses. In that regard this paper serves as a demonstration of the feasibility of integrating protein sequences as a fundamental data type of database systems and the use of general-purpose SQL query engines to integrate biological sequence analysis with other data analysis at a fine-grain level (i.e. directly, not by hiding the details of using BLAST as an external utility). Despite the overhead of the generality of our implementation, disk I/O and Java, on large-scale problems we can outperform BLASTp, a highly optimized main-memory system developed in C. Therefore, we are very pleased with the raw speed of our approach. If speed rather than a database programming model is the preferred goal, then these algorithms may be implemented in C.

Acknowledgement

We acknowledge Willard Saint Willard, students in the laboratory who have participated in MoBIoS project. This research was supported in part by grants from DBI-0241180 and IIS-0325116.

References

Altschul, S.F. & Gish, W. (1996) Local alignment statistics. *Methods Enzymo.*,266, 460-480

Altschul, S.F. Gish, W., Miller, W., Myers, E. W. & Lipman, D. J. (1990) Basic local alignment search tool. *J. Mol. Biol.*, 215, 403-410

Bellman, R. (1961), Adaptive Control Processes: A Guided Tour, Princeton University Press. New Jersey, U.S.

Bozkaya, T. & Ozsoyoglu, M. (1999) Indexing Large Metric Spaces for Similarity Search Queries. *ACM Transactions on Database System*, 24(3):361-404

Chavez, E., Navarro, G., Baeza-Yates, R. & Marroquin, J.L. (2001) Searching in metric spaces. *ACM Computing Surveys*, 33(3):273-321

Halperin, E., Buhler, J., Karp, R., Krauthgamer, R. and Westover, B. (2003) Detecting Protein Sequences Via Metric Embeddings *In Proceedings of the 11th International Conference on Intelligent Systems for Molecular Biology* 122-199, Brisbane, Australia, Jun. 29-Jul. 03, 2003

Fondrat, C. and Dessen, P. (1995) A rapid access motif database (RAMdb) with a search algorithm for the retrieval patterns in nucleic acids or protein data banks. *Computer Applications in the Biosciences*, 11(3):273-279

Giladi, E., Walker, G. M., Wang, J.Z. & Volkmuth, W. (2002) SST: an algorithm for finding near-exact sequence matches in time proportional to the logarithm of the database size. *Bioinformatics*. 18(6):873-879

Gribskov, M. & Robinson, N. L. (1996) Use of receiver operating characteristic (ROC) analysis to evaluate sequence matching. *Computers and Chemistry*. 20(1):25-33

Gusfield, D. (1997) Algorithms on Strings, Trees and Sequences Computer Science and Computational Biology. 449-454 Press Syndicate of the University of Cambridge, USA

Hochbaum, D. S. & Shmoys, D. B. (1985) A best possible heuristic for the k-center problem. *Mathematics of Operational Research*, 10(2):180-184

Kahveci, T., Ljosa, V., & Singh, A.K. (2004) Speeding up whole-genome alignment by indexing frequency vectors, *Bioinformatics* 20(13) 2122-2134

Kent, W. J. (2002) BLAT-The BLAST like alignment tool. *Genome Res.* 12:656-664

Mao, R., Lei, V.I., Ramakrishnan, S., Xu, W., Miranker, D.P. (2004) On Metric-Space Indexing and Real Workloads, *Technical Report*, Department of Computer Sciences, University of Texas at Austin, 2004

Meek, C., Patel, J.M., and Kasetty, S. (2003) OASIS:An Online Accurate Technique for Local-alignment Searches on Biological Sequences *in Proceedings of the 29th VLDB Conference*, 910-921, Berlin, Germany, Sep. 9-12, 2003
Miranker, D.P., Briggs, W.J., Mao, R., Ni, S., and Xu, W. (2004) Biosequence Use Cases in MoBIoS SQL, *Data Engineering Bulletin*, 27 (3): 3-11
Myers, E.W. (1994) A sublinear algorithm for approximate keyword searching. *Algorithmica.* 12(4/5):345-374
Pearson, W. R. & Lipman, D. J. (1988) Improved tools for biological sequence comparison. *Proc. Natl Acad. Sci.* USA, 85:2444-2448
Promponas, V. J., Enright, A. J., Tsoka, S., Kreil, D. P., Leroy, C., Hamodrakas, S. J., Sander, C. and Ouzounis, C. A. (2000) CAST: an iterative algorithm for the complexity analysis of sequence tracts *Bioinformatics* 16(10):915-922
Rouchka, E. C. Gish, W. & States D. J. (2002) Comparison of whole genome assemblies of the human genome. *Nucleic Acids Res.*, 30(22):5004-5014
Sahinalp, S.C., Tasan, M., Macker, J., Ozsoyoglu, Z.M., (2003) Distance Based Indexing for String Proximity Search, *in proceedings of IEEE Data Engineering Conference*, 125-137 Banglore, India, March 03-05, 2003
Schaffer, A. A. Aravin, L. Madden, T. L., Shavirin, S., Spouge, J. L., Wolf, Y. I., Koonin, E. V. & Altschul, S.F. (2001) Improving the accuracy of PSI-BLAST protein database searches with composition-based statistics and other refinements. *Nucleic Acids Res.*, 29(14):2994-3005
Sellers, P.H. (1974) On the theory and computation of evolutionary distances. *J. Appl. Math.* (SIAM). 26:787-793
Shekhar, S. & Chawla, S. (2002) Spatial Databases: A Tour, Prentice Hall, USA
Tan, Z., Cao, X., Ooi, B.C. & Tung, A.K.H. (2003) The ed-tree: an index for large DNA sequence databases *In Proc. 15th International Conference on Scientific and Statistical Database Management* 151-160 Jul. 9-11, 2003 Cambridge, MA
Uhlmann, J. (1991) Satisfying general proximity/similarity queries with metric trees *Information Processing Letters* 40:175-179
Wilbur, W.J. and Lipman D.J. (1983) Rapid similarity searches of nucleic acid and protein data banks. *Proceedings of the National academy of Science*, 80:726-730
Williams, H.E. and Zobel, J. (2002) Indexing and Retrieval for Genomic Databases *IEEE Transactions on Knowledge and Data Engineering*, 14(1): 63-78,
Wootton. J.C. and Federhen S. (1993) Statistics of local complexity in amino acid sequences and sequence databases *Comput. Chem.* 17:149-163.
Wu, S. Manber, U. Myers, G. and Miller, W. (1990) An O(NP) Sequence Comparison Algorithm. *Information Processing Letters* 35(6): 317-323
Xu, W. & Miranker, D.P. (2004) A metric model for amino acid substitution *Bioinformatics* 20(8):1214-21
Xu, W., Miranker, D.P., Mao, R., & Wang, S. (2003) Indexing Protein Sequences in Metric Space, *Technical Report TR-04-06* University of Texas at Austin, Department of Computer Sciences. Dec, 2003
Yianilos, P. (1993) Data structures and algorithms for nearest neighbor search in general metric spaces. *In Proc. 4th ACM-SIAM Symposium on Discrete Algorithms*, 311-321, Austin, TX. Jan. 25-27, 1993
Yianilos, P. (2000) Locally Lifting the Curse of Dimensionality for Nearest Neighbor Search, *In Proc. ACM-SIAM Symposium on Discrete Algorithms*, Jan. 9-11, 2000, San Francisco, California