Progressive Searching of Biological Sequences

Tamer Kahveci
Department of Computer and
Information Science and Engineering
University of Florida
Gainesville, FL, 32611-6125
tamer@cise.ufl.edu

Ambuj Singh
Department of Computer Science
University of California
Santa Barbara, CA, 93106
ambuj@cs.ucsb.edu

Abstract

We consider the problem of progressive searching, and propose two k-NN (k-Nearest Neighbor) search algorithms for biological sequence databases, one for local alignment and one for global alignment. We develop a method to compute the confidence on the partial results. We also propose an early pruning strategy to reduce the total search time. Our experiments show that our techniques can achieve 75 % accuracy within the first 2.5-31 % of the iterations and 90 % accuracy within the first 12-37 % of the iterations.

1 Introduction

We investigate the problem of providing progressive access to genome sequences. We develop two models to quickly predict the proximity of database subsequences to a given query sequence. Our first model employs the Karlin-Altschul statistics [KA90] directly to database subsequences. This model computes the *unexpectedness* of an alignment by considering the distribution of letters in the entire database and query sequence. Our second model transforms database subsequences into vector space through the use of *frequency vectors* (count of each letter in a sequence) [KS01]. The frequency vectors are then clustered into MBRs (Minimum Bounding Rectangles). Then, we estimate the distance distribution of the subsequences in each MBR to the query subsequences with the help of *order statistics* [Cas88].

Using these models, we build a set of progressive k-NN (k-Nearest Neighbor) search techniques for both global (i.e., entire query sequence is aligned) and local alignment (i.e., query subsequences are aligned). Our first algorithm considers the local alignment problem. It partitions the database into overlapping blocks. Later, these blocks are ranked with the help of Karlin-Altschul statistics, and searched in this order using a highly optimized search tool, such as BLAST [AGM$^+$90]. The intermediate results are immediately reported.

Our next algorithm considers the global alignment problem. It hierarchically finds the next MBR of the index structure in ascending order of its kth-order statistics. Later, these MBRs are inspected iteratively using Needleman-Wunsch method [NW70]. We reduce the total search time of this technique by pruning the MBRs on the fly that do not contain results better than the ones found so far. We also develop a novel technique to

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

estimate the probability that the uninspected MBRs do not contain any results better than the results reported so far. This value is then reported as a *confidence estimate*.

Experimental results on DNA and protein sequences show that our methods achieve 75 % accuracy within the first 2.5-31 % of the iterations and 90 % accuracy within the first 12-37 % of the iterations. Our pruning strategy can prune up to 38 % of the database.

A recent method (named OASIS) [MPK03] uses suffix trees for accurate online searches for short query sequences. Our work differs from this paper in four ways. 1) Our index structures are very compact (i.e., 1-2 % of the database size [KS01]). 2) Our methods can handle long queries. 3) We can perform both local and global alignment. 4) We provide confidence estimates.

The rest of the paper is as follows. Section 2 presents background on sequence searching. Sections 3 and 4 discuss our progressive methods for local and global alignment respectively. Section 5 presents the experimental evaluation of our methods. Finally, we end with a brief discussion in Section 6.

2 Background on sequence searching

A sequence x can be transformed into another sequence y using three *edit operations*, namely *insert*, *delete*, and *replace*, on individual characters of the sequence x. The *edit distance* between two sequences is generally defined as the minimum number of edit operations to transform one sequence into the other. A special case of edit distance, *Gapless Edit Distance (GED)* or *Hamming Distance*, can be defined as follows.

Definition 1: Let x and y be two sequences of the same length. Let $x[i]$ be the ith character of x. We define the *Gapless Edit Distance*, *GED*(x, y), as the number of character pairs $(x[i], y[i])$ for which $x[i] \neq y[i]$. $\square$

GED is an upper bound to the edit distance, and is widely used for statistical analysis [AG96, AW94]. We will also use *GED* in our analysis in Section 4.

Similarity between sequences can also be computed based on scores. An *alignment* of sequences x and y is obtained by matching each character of x to a character in y in increasing order. All the unmatched characters in both sequences are matched with space. Each character pair is assigned a score based on their similarity. These scores are stored in a *score matrix*. Each gap incurs a (potentially large) *gap open penalty* for the first space and a (smaller) *gap extend penalty* for all the spaces. The value of an alignment is defined as the sum of the scores of all of their character pairs. *Global alignment* of sequences x and y is their highest scoring alignment. On the other hand, their *local alignment* is alignment of their subsequences with the highest score. Both global and local alignments can be determined in $O(|x| \cdot |y|)$ time and space using dynamic programming [NW70, SW81].

Several heuristics that run faster than the dynamic programming, such as BLAST [AGM$^+$90], are widely used for sequence alignment. BLAST starts by finding exactly matching seeds with the help of a hash table. Later, these seeds are extended in both directions and appended to find longer matches. (See [KLS04] for a brief discussion of other important sequence alignment tools.)

3 Progressive local alignment

There is a substantial amount of work that estimates the score of the best local alignment of two sequences using global distribution of the letters in these sequences. We first discuss Karlin Altschul statistics [KA90] which is used by BLAST to compute the unexpectedness of a result. Then we explain how we employ it for progressive local alignment.

3.1 Background on Karlin-Altschul statistics

Let $\Gamma = \{\alpha_1, \cdots, \alpha_\gamma\}$. Assume that the letters are sampled with probabilities $\{p_1, \cdots, p_\gamma\}$ in the data sequence, and $\{q_1, \cdots, q_\gamma\}$ in the query sequence. Let the score of matching letters α_i and α_j be $s(\alpha_i, \alpha_j)$ such that $\sum_{i,j} p_i p_j s(\alpha_i, \alpha_j) < 0$ and $max_{i,j}\{s(\alpha_i, \alpha_j)\} > 0$. These conditions must be satisfied for any valid scoring scheme [KA93].

Let $f(\lambda) = \sum_{i,j} p_i p_j e^{\lambda s(\alpha_i, \alpha_j)}$. BLAST uses the unique positive solution, λ^* to the equation $f(\lambda) = 1$ in the statistical computation [KA90]. Karlin and Altschul [KA90] show that the expected value for the score of the maximal alignment of two sequences can be approximated as $\frac{\ln(mn)}{\lambda^*}$, where m and n are the lengths of the two compared sequences. Finding this value requires the computation of λ^*. We propose to use a direct solution technique based on the following observations: a) $f(0) = 1$, b) $f'(0) < 0$, c) $f''(0) > 0$, where f' and f'' are the first and second derivatives of f. These three conditions imply that f is a convex function. One of the solutions to $f(\lambda) = 1$ is obviously at $\lambda = 0$, and the other root is positive. We define $g(\lambda) = f(\lambda) - 1$, and use the Newton-Raphson Method [Ham87] to find the positive root of $g(\lambda) = 0$. In our experiments, this method converged in a few iterations.

3.2 Using Karlin-Altschul statistics for progressive local alignment

Here, we develop our first progressive search technique based on Karlin-Altschul statistics. We call this technique *Karlin-Altschul Statistics-based incremental search (KAS)*. KAS partitions the database sequences into overlapping blocks (i.e., subsequences) by placing a window of length n on the database sequences. Consecutive blocks are obtained by shifting the window by Δ letters. Each such positioning of the window produces one data base block. We discuss the choice of n and Δ in Section 5. For each block, we store the frequency of all the letters.

The score of the maximal local alignment for each block s_i is estimated as $\ln(n \cdot |q|)/\lambda_i^*$, where λ_i^* is defined the same as in Section 3.1 for s_i. Since n and $|q|$ are fixed, this estimate becomes larger when λ_i^* is smaller. Given a query sequence q, KAS works in 3 steps:

Step1: (Prediction) Compute λ_i^* for q and the subsequence s_i in partition i using the Newton-Raphson method, for all i.

Step 2: (Sorting) Sort λ_i^* in ascending order.

Step 3: (Searching) Search the partitions s_i in ascending λ_i^* order using an efficient search tool like BLAST. Report intermediate results to the user for each partition.

4 Progressive global alignment

Computing statistics for global alignment is more difficult than computing statistics for local alignment. Here we present a method for estimating global alignment statistics using frequency vectors and GED distributions.

4.1 Background on frequency vectors and MRS index structure

Let s be a sequence from an alphabet Γ, where $|\Gamma| = \gamma$. The *frequency vector*, $f(s)$, of s is defined as the γ-dimensional vector whose entries are the number of occurrences of the letters in Γ. For instance, the frequency vector of a DNA sequence has 4 dimensions since DNAs contain the letters A, C, G, and T. For example, for the DNA sequence $s = \texttt{GGTTCCTCTA}$, $f(s) = [1, 3, 2, 4]$.

The MRS index structure [KS01] maintains summary information for database sequences at resolutions of powers of two. At a given resolution w, the summary is constructed by sliding a window of length w on the database sequence. Each positioning of the window defines a frequency vector and a set of consecutive vectors defines an MBR. The size of this set determines the *capacity* of an MBR. The rest of the MBRs at this resolution are created similarly. Note that all the frequency vectors that constitute an MBR lie on the same multi-dimensional plane, called the *data plane*. This is because the entries of the frequency vectors in an MBR add up to the resolution of that MBR.

We will utilize the MRS index structure for our search techniques because of its compactness and efficiency. However, our techniques will extend to other index structures that cluster subsequences of database sequences.

4.2 GED distribution for two vectors

At the time of a database search, the query sequence is known but only the frequency vectors of database sequences are available in the index structure. Therefore, in order to plan the progressive query, we need to determine the distribution of the *GED* between two sequences using their frequency vectors.

Each frequency vector v defines an equivalence class, $\mathcal{S}_v$, of the set of sequences whose frequency vectors are equal to v. For example, if $v = [3, 0, 0, 1]$, then $S_v = \{\texttt{AAAT}, \texttt{AATA}, \texttt{ATAA}, \texttt{TAAA}\}$. Let $v = [v_1, \cdots, v_\gamma]$ be a frequency vector, where γ is the alphabet size, then the size of the equivalence class defined by v is $(\sum v_i)!/(\prod(v_i!))$.

In order to plan queries and compute statistics, we need to know the distribution of $\{GED(q, s) | s \in \mathcal{S}_v\}$. For simplicity, we will assume that s is uniformly distributed over $\mathcal{S}_v$. We do not make any assumptions on q. Using the Central Limit Theorem [Cas88], one can prove that the *GED* has a normal distribution as follows: Let the random variable Z_i represent the *GED* achieved by aligning the ith letters of both sequences for $1 \leq i \leq |s|$ (i.e., if $q[i] = s[i]$ then $Z_i = 0$. Otherwise, $Z_i = 1$). The *GED* of an alignment can be calculated as the sum of the values of all Z_is for that alignment. Since the sampling rate of all the letters is fixed and determined by the frequency vector of that sequence, the Z_is are iid (independent and identically distributed) random variables. Therefore, from the Central Limit Theorem, we conclude that the distribution of the *GED* between a query sequence and the sequences in the equivalence class of a frequency vector can be approximated using normal distribution for large $|s|$ (i.e., $|s| \geq 10$). Hence, one can compute this approximation if the mean and variance of the distribution are known. Theorems 2 and 3 develop formulas for the mean and the variance of this distribution that can be computed efficiently in $O(\gamma)$ time.

Theorem 2: (*Mean*): Let q be a sequence from alphabet $\Gamma = \{\alpha_1, \cdots, \alpha_\gamma\}$. Let $x = [x_1, \cdots, x_\gamma]$ be the frequency vector of q, and $y = [y_1, \cdots, y_\gamma]$ be a frequency vector, where $\sum_{i=1}^{\gamma} y_i = |q|$. Let $\mu_{q,y}$ be the mean of the *GED* distribution between q and the sequences in $\mathcal{S}_y$, then

$$\mu_{q,y} = |q| - \frac{\sum_{i=1}^{\gamma} x_i \cdot y_i}{|q|}. \qquad \square$$

Theorem 3: (*Variance*): Let q be a sequence from alphabet $\Gamma = \{\alpha_1, \cdots, \alpha_\gamma\}$. Let $x = [x_1, \cdots, x_\gamma]$ be the frequency vector of q, and $y = [y_1, \cdots, y_\gamma]$ be a frequency vector, where $\sum_{i=1}^{\gamma} y_i = |q|$. Let $\sigma_{q,y}^2$ be the variance of the *GED* distribution between q and the sequences in $\mathcal{S}_y$, then

$$\sigma_{q,y}^2 \approx \sum_{i=1}^{\gamma} \left(\frac{x_i \cdot y_i}{|q|} \cdot \left(1 + \frac{(x_i - 1)(y_i - 1)}{|q| - 1} - \frac{x_i \cdot y_i}{|q|}\right)\right). \qquad \square$$

4.3 GED distribution of a query to an MBR

So far, we considered the GED distribution between the frequency vector of a query sequence and that of a database sequence. Now, we consider the GED distribution between a query frequency vector and a set of database frequency vectors, specifically those contained in an MBR of an index structure. This computation poses two problems. First, it requires knowing the *GED* distributions of all frequency vectors contained in the MBR. However, an MBR does not store the individual frequency vectors it contains. It only maintains their span. Second, even if all the frequency vectors within an MBR are known, computing the GED distribution of all of them separately is time consuming.

We resolve these two problems by choosing a representative frequency vector for each MBR and assuming that an MBR contains a number of iid random variables based on the representative frequency vector. The representative frequency vector of an MBR has to be the vector which is closest to the rest of the vectors in the MBR. Furthermore, the sum of the entries of this vector must be equal to w, where w is the resolution of the MBR. This vector is found in two steps as follows: First, the centroid of the MBR is calculated as the average of the lower and higher coordinates of the MBR. Later, this centroid is projected onto the plane which contains frequency vectors for resolution w.

Once the representative frequency vector of an MBR has been chosen, we construct a random variable, X, having the GED distribution between the given query vector and the representative vector of that MBR as explained in Section 4.2. If the MBR contains c frequency vectors at resolution w, then the *GED* distribution of the query to the MBR is approximated by assuming $\lceil c/w \rceil$ random variables independent and identical to X. This is justified since the MBRs of the MRS index structure contains $\approx c/w$ independent subsequences.

The *GED* distribution between a frequency vector and a set of iid frequency vectors can be calculated using order statistics. The kth-order statistics of a set of random variables $\mathcal{X} = \{X_1, \cdots, X_n\}$ is defined as the random variable $Y_k \in \mathcal{X}$ that has the kth smallest value. Let X be a random variable having the same distribution as X_is ($1 \leq i \leq n$), then the cumulative distribution function of Y_k can be calculated as

$$P(Y_k \leq d) = \sum_{j=k}^{n} C_j^n \cdot P(X \leq d)^j \cdot (1 - P(X \leq d))^{n-j},$$

where C_j^n is n choose j. The probability mass function for Y_k is computed as

$$k \cdot C_k^n \cdot P(X \leq d)^{k-1} \cdot (1 - P(X \leq d))^{n-k} \cdot P(X = d).$$

Let M_d be the number of X_is ($1 \leq i \leq n$ and $0 \leq d$) with value less than or equal to d, then

$$P(M_d \leq m) = 1 - P(Y_{m+1} \leq d).$$

We use this distribution function to predict the number of sequences that are within a specified GED to a given query sequence in an MBR.

4.4 Using frequency statistics for progressive global alignment

Our second progressive search technique uses statistical information about the database subsequences. We call this technique the *Frequency Statistics-based incremental search (FS)* technique.

FS builds an MRS index structure on the database. Each MBR of this index defines a partition (i.e., subsequence) of $w + c - 1$ letters, where w and c are the resolution and the capacity of that MBR. Similar to KAS, FS works in three steps:

Step 1: (Prediction) In this step, the frequency vectors of all possible w-letter subsequences of the query are determined and their representative q_w is computed as their mean. For each MBR, the mean of the kth order statistics is then computed using q_w. This value is then normalized by w. This process is repeated for all resolutions w in the index, and the results are accumulated to determine the cumulative order statistics for each partition.

Step 2: (Sorting) The partitions are sorted in descending order of cumulative order statistics.

Step 3: (Searching) The partitions are aligned to the query sequence based on the ordering obtained in previous step, using an appropriate method, such as Needleman-Wunsch dynamic programming [NW70] The partial results are reported to the user immediately along with the confidence value.

Confidence estimate exhibits *how confident the user should be with the current partial results*. Confidence estimates are computed based on the results discovered so far and the *GED* distributions of the uninspected

MBRs as follows. Let d_k be the GED to the kth closest sequence reported so far. $P(Y_1 \geq d_k)$ for an MBR represents the probability that that MBR does not contain any match whose distance is less than d_k. This value is computed for each uninspected MBR using the formulas given in Section 4.3. Later, these results are multiplied to find the probability that the closest subsequence in the remaining MBRs is not closer than the kth most similar subsequence found so far.

FS ranks database subsequences according to their similarity to the given query sequence in the frequency domain. These subsequences are then examined in this order, thus potentially finding better results earlier. However, FS still has to search the entire database in order to ensure the absence of false dismissals. The total search time of FS can be reduced by pruning the lower ranking subsequences that do not contain any result better than the kth-NN found so far. We propose to prune low quality subsequences in two steps:

Step 1: Let S_k be the score of the kth best match found so far. An upper bound, L, on the length of the kth best match is estimated. Define w to be the minimum resolution available in the MRS index structure, for which $w \geq L$.

Step 2: An upper bound, S_{upper}, to the score of the best alignment between the query and database subsequences contained in the next uninspected MBR at resolution w is computed. If $S_{\text{upper}} < S_k$, then this MBR is pruned. Otherwise, it is inspected for alignments. An efficient computation of S_{upper} is developed in our earlier paper [KLS04].

5 Experimental results

We used two classes of sequence datasets in our experiments: **DNA dataset** (`ftp://ftp.ncbi.nih.gov`) contains chromosome 18 (*chr-18*, acc. `NT_000864`), from *homo sapiens* database and the genetic code of *Escherichia coli* (*E.Coli* acc. `U00096`). each containing more than 4 M base pairs. **Protein dataset** (`ftp://ftp.expasy.ch`) contains all the proteins in the SWISSPROT database (available by January 2004). This dataset contains approximately 68 M residues.

We downloaded the source code of BLAST, and implemented the MRS index structure for window sizes $256 \leq w \leq 2048$, and box capacity $c = 1000$. We used BLAST for the local alignment of DNA sequences and the standard dynamic programming algorithm for the global alignment of protein sequences. For KAS, we used $n = 3047$, $\Delta = 1000$. These parameters produce the same blocks as FS.

We extracted query sets from chr-18 dataset for $|q| = \{500, 1000, 2000, 4000\}$ each containing 100 queries. Later, we generated six new query sets from each of these query sets by modifying these queries with 5, 10, 20, 30, 40, and 50 % mutation probability. We generated two query sets from the protein dataset for $|q| = \{256, 512\}$. Later, we created three more query sets by modifying these queries with 5, 10 and 20 % mutation rates. We used BLAST's default scoring scheme for DNAs in our experiments. For protein dataset, we used BLOSUM62 score matrix.

We calculate the *accuracy* of the partial results for k-NN queries as the sum of the scores of the k best matches found so far divided by the sum of the scores of the actual k best matches.

Local alignment results: First, we inspect the performance of KAS and FS for local alignment of varying proximity of query sequences to database. We use the query sets from chr-18 in this experiment set. We generate a larger dataset, namely *chr-18/E.Coli* by appending two dissimilar datasets chr-18 and E.Coli, and perform queries on this dataset. The purpose of this experiment is to see whether our techniques can distinguish the distant regions in E.Coli from the homologous regions of chr-18. Figure 1 shows the average score found by FS, FS-prune, and KAS for 1-NN queries at different iterations for 5% mutation rate of $|q| = 4000$ query set. Since chr-18 and E.Coli datasets have approximately same number of base pairs, they have similar number of MBRs. As evident from this figure, all of our techniques find the optimal results before half of the database is inspected.

Table 1 summarizes the number of iterations performed to achieve accuracies of 0.75 and 0.9 for various

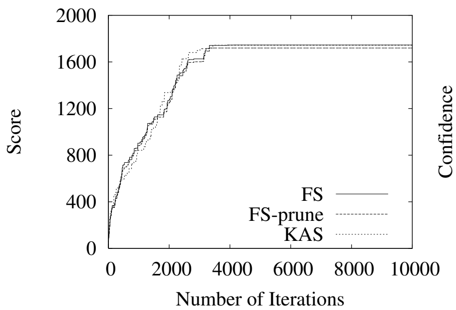

Figure 1: The score found at different iterations for query set generated from chr-18 (5 % mutation rate) on chr-18/E.Coli dataset.

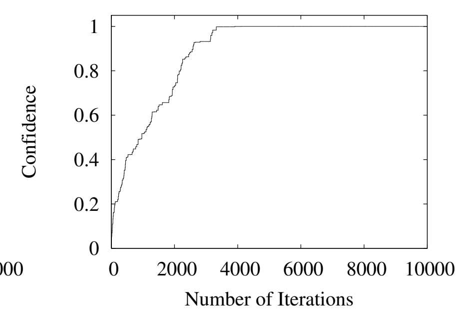

Figure 2: The confidence of FS for 1-NN queries for the experiment in Figure 1.

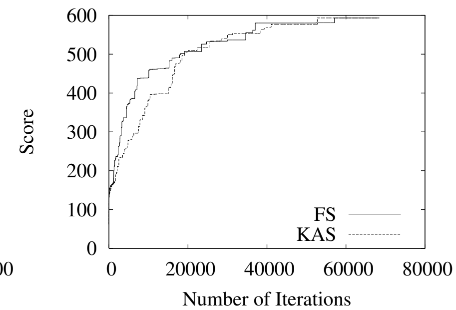

Figure 3: The score found on protein dataset for $|q| = 256$, and 20 % mutation rate.

Table 1: The number of iterations to achieve 0.75 and 0.9 accuracy for various mutation rates. The number of iterations for non-progressive techniques is 8863. (i.e., the number of database blocks.) The results for accuracy = 0.9 are shown in parenthesis.

Mutation Ratio	5	10	20	30	40	50
FS	2095 (2588)	2123 (2719)	1866 (2739)	1771 (2537)	1903 (2758)	1766 (2917)
FS- prune	2096 (2560)	2140 (2719)	1866 (2739)	1771 (2537)	1903 (2758)	1766 (2917)
KAS	1846 (2401)	1613 (2347)	1759 (2431)	1362 (2236)	1583 (2834)	1857 (2811)

mutation rates of the query set. For all mutation rates, all the techniques can obtain 0.9 accuracy after 25-31 % of the MBRs are inspected, and 0.75 accuracy after 15-24 % of the MBRs are inspected. Although there is no clear winner among these three techniques, KAS finds high scoring results slightly faster on the average. FS and FS-prune overlap for mutation rates greater than 10 %.

Evaluation of pruning rate: Unlike FS-prune, both FS and KAS search the entire database to ensure that there is no remaining match better than the reported ones. FS-prune reduces total search time by pruning low scoring regions of the database. In the experiments in Table 1, we achieved 38.9 %, 17.3 %, and 12.1 % pruning for mutation rates 5 %, 10 %, and 20 % respectively. For larger mutation rates, the pruning went down to zero.

For 1, 2, 4, and 8-NN, FS-prune eliminated 38.9 %, 38.9 %, 37.5, and 17.5 % of the database respectively. The amount of database pruned reduces slowly as the number of NN increases. However, even when $k = 8$, FS-prune is 21 % faster than both FS and KAS.

Evaluation of confidence estimates Figure 2 shows how the confidence of FS changes over iterations for the experiment in Figure 1. FS achieves 90% confidence after performing only 30% of the iterations. Another important point is that the confidence of FS corresponds closely to its accuracy in Figure 1. That is, FS can accurately report the user the quality of the partial results with respect to the final results before all the final alignments are found. This experimentally substantiates the theoretical developments of Sections 4.2 and 4.3. This is important, because the user can estimate the score of the best matches of the final alignment at intermediate steps by inspecting the partial results and the confidence estimates.

Global alignment results: In this experiment, we evaluate the quality and performance of FS, FS-prune, and KAS for global alignment of protein sequences. We use the standard dynamic programming method with BLOSUM62 score matrix in this experiment.

Figure 3 plots the average score found during various iterations for the query set $|q| = 256$ with 20 % modification. We do not plot the results for other query lengths and mutation rates since they have similar behavior. The experiments show that FS is slightly better than KAS. In this experiment, FS achieved 75 % accuracy after only 15 % of the data is processed. On the other hand KAS processed 24 % of the dataset to achieve the same quality. However, KAS catches up with FS as accuracy increases to 90 %. In this case FS searches 36 % of the data while KAS searches 37 %.

6 Discussion

In this paper, we considered progressive subsequence searching for biological sequences. We defined a new statistical model, based on frequency vectors, for the analysis of the distance distribution between a query and a set of sequences based on their frequency vectors. We presented formulas to compute this distribution in $O(\gamma)$ time, where γ is the alphabet size.

We proposed two novel progressive subsequence search techniques called *KAS* and *FS*. KAS splits the database into overlapping blocks and computes the average frequency of all the letters in each block. It employs Karlin-Altschul statistics to predict the maximal local alignment score for each block, and ranks the blocks based on this prediction. FS organizes the database as a set of MBRs with the help of the MRS index structure. The MBRs of the index structure are reordered based on the order statistics of their representative frequency vectors at various resolutions. These MBRs are then searched iteratively using any traditional sequence search tool. The partial results are reported at each iteration along with their confidence estimates. We also reduced the total run time of FS by pruning the unpromising MBRs. This technique is called *FS-prune*.

Our experimental results showed that the proposed techniques achieve high accuracies quickly. In our experiments, our techniques achieved 75 % accuracy within the first 2.5-31 % of the iterations and 90 % accuracy within the first 12-37 % of the iterations. In our experiments, KAS performed slightly better for the local alignment queries and FS performed better for the global alignment queries. Our pruning strategy eliminated up to 38 % of the database in these experiments. The confidence value computed by FS corresponded closely to its accuracy.

References

[AG96] S.F. Altschul and W. Gish. Local alignment statistics. *Methods in Enzymology*, pages 460–480, 1996.

[AGM+90] S. Altschul, W. Gish, W. Miller, E. W. Meyers, and D. J. Lipman. Basic Local Alignment Search Tool. *Journal of Molecular Biology*, 215(3):403–410, 1990.

[AW94] R. Arratia and M.S. Waterman. A phase transition for the score in matching random sequences allowing deletions. *The Annals of Applied Probability*, 4(1):200–225, 1994.

[Cas88] E. Castillo. *Extreme Value Theory in Engineering (Statistical Modeling and Decision Science Series)*. Academic Press, New York, 1988.

[Ham87] R. Hamming. *Numerical Methods for Scientists and Engineers*. Dover Pubns, 2 edition, April 1987.

[KA90] S. Karlin and S.F. Altschul. Methods for Assessing the Statistical Significance of Molecular Sequence Features by Using General Scoring Schemes. *Proceedings of the National Academy of Sciences of the United States of America (PNAS)*, 87(6):2264–2268, 1990.

[KA93] S. Karlin and S.F. Altschul. Applications and Statistics for Multiple High-Scoring Segments in Molecular Sequences. *Proceedings of the National Academy of Sciences of the United States of America (PNAS)*, 90(12):5873–5877, 1993.

[KLS04] T. Kahveci, V. Ljosa, and A.K. Singh. Speeding up whole-genome alignment by indexing frequency vectors. *Bioinformatics*, 2004. to appear.

[KS01] T. Kahveci and A. Singh. An Efficient Index Structure for String Databases. In *International Conference on Very Large Databases (VLDB)*, pages 351–360, Roma, Italy, September 2001.

[MPK03] C. Meek, J. M. Patel, and S. Kasetty. OASIS: An Online and Accurate Technique for Local-alignment Searches on Biological Sequences. In *International Conference on Very Large Databases (VLDB)*, 2003.

[NW70] S. B. Needleman and C. D. Wunsch. A General Method Applicable to the Search for Similarities in the Amino Acid Sequence of Two Proteins. *Journal of Molecular Biology*, 48:443–53, 1970.

[SW81] T.F. Smith and M.S. Waterman. Identification of Common Molecular Subsequences. *Journal of Molecular Biology*, March 1981.