International Journal on Artificial Intelligence Tools

SEQUENCE SIMILARITY SEARCH USING DISCRETE FOURIER AND WAVELET TRANSFORMATION TECHNIQUES

S. ALIREZA AGHILI
Department of Computer Science, University of California-Santa Barbara
Santa Barbara, CA 93106, USA
aghili@cs.ucsb.edu

DIVYAKANT AGRAWAL
Department of Computer Science, University of California-Santa Barbara
Santa Barbara, CA 93106, USA
agrawal@cs.ucsb.edu

AMR EL ABBADI
Department of Computer Science, University of California-Santa Barbara
Santa Barbara, CA 93106, USA
amr@cs.ucsb.edu

In this paper, we study the problem of sequence similarity search. We incorporate vector transformations and apply DFT (Discrete Fourier Transformation) and DWT (Discrete Wavelet Transformation, Haar) dimensionality reduction techniques to reduce the search space/time of sequence similarity range queries. Our empirical results on a number of Prokaryote and Eukaryote DNA contig databases demonstrate up to 50-fold filtration ratio reduction of the search space and up to 13 times faster filtration. The proposed transformation techniques may easily be integrated as a pre-processing phase on top of current similarity search heuristics/techniques such as BLAST, PatternHunter, FastA and QUASAR to efficiently prune non-relevant sequences. We study the precision of applying dimensionality reduction techniques for faster and more efficient range query searches and discuss the imposed trade-offs.

Keywords: Sequence Similarity; String Comparison; Range Query; Sequence Transformation; Biological Databases.

1. Introduction

Discovering the structure, function and evolutionary relationships among genes are the main goals of genome sequencing research. The comparative analysis of homologous sequences is a crucial part in the study of gene function, known as *genomics.* The behavior resemblance of two DNA sequences of two different organelles or species to the same external exposure may be used to infer functional or structural similarities, or mutual inclusion in the same pathway or biological mechanism. Some of the vast applications of proximity search include discovering the nature and functionality of human genome, phylogenetic analysis, drug discovery, keyword search in databases, or even user pattern analysis in the context of network security. In this study, we focus our attention on the application of sequence similarity range query search within the context of biological sequence databases. For instance, approximate sequence analysis has assisted the detection of certain strains of the *Escherichia coli(E.coli)* bacteria responsible for infant *diarrhea* and *gastroenteritis*. The researchers at the University of Chicago's Howard Hughes Medical Institute[23] discovered a protein molecule capable of transmitting a genetic trait without DNA or RNA in *yeast*, which is able to string itself together into a long fiber, much like those found in the brain in *mad cow* and human *Creutzfeldt-Jakob* diseases. In general, some of the typical applications of sequence similarity search include[6]:

- Identification of highly conserved residues/motifs which are likely to correspond to essential sites for the structure or function of the sequence.
- Phylogenetic analysis which relies on neighbor sequence search, at the protein or DNA level, to predict mutations from which it is possible to retrace evolutionary relationships among different genetic sequences. Similarly, phylogenetic trees provide the information to reconstruct the history of species and gene families.

Similarity search seeks the sequences close enough to a given query sequence either through direct alignment[19,21] or using other heuristics[4,9,14,16,20,22]. The alignment of biological sequences (*pairwise* or *multiple* alignment) is the operation of placing nucleotide or amino acid residues in columns inferring the closest common ancestral relationships. This is achieved by introducing gaps with predefined *cost*s to represent insertions or deletions into sequences. Hence, an alignment is a hypothetical model of mutations on the residue level through *edit operations* namely, *Replacement*, *Insertion* and *Deletion*. The best alignment usually refers to the one demonstrating the most likely evolutionary scenario. Let $S_1, S_2 \in \Sigma^*$ be finite ordered DNA sequences of characters (*bases*) taken from the alphabet set Σ, where $\Sigma = \{A, C, G, T\}$. Each pair of characters from Σ are assigned a replacement (substitution) *cost*. The substitution matrices[9,11,24] providing such information are built based on the structure similarity and replacement likelihood of the residues (*bases*). For instance, at the DNA level, probabilities of substitution vary according to the *nature* of the base pairs. Notably, *transitions* (substitutions between two *purines*, A and G, or two *pyrimidines*, C and T) are generally more frequent than *transversions* (substitutions between a purine and a pyrimidine). Hence, the optimal alignment of sequences S_1 and S_2 is achieved by applying the minimum number of edit operations to transform S_1 into S_2, called *Edit Distance*, or $\mathrm{ED}(S_1, S_2)$. Given two sequences of length p and q, the optimal pairwise alignment ensures the minimal transition *cost* and requires $O(pq)$-time, and $O(pq)$-space, using dynamic programming[19,21] algorithm. An example of the edit distance procedure is illustrated in the following example:

$$\begin{array}{lccccccccccccccc}
S_1 & \multicolumn{15}{l}{\text{A A C T C G A G A C C C}} \\
S_2 & \multicolumn{15}{l}{\text{A T C C G A G A G G T C C C}} \\
\cline{2-16}
 & \text{A} & \text{A} & \text{C} & \text{T} & \text{C} & \text{G} & \text{A} & \text{G} & \text{A} & - & - & - & \text{C} & \text{C} & \text{C} \\
 & & \text{R} & & \text{D} & & & & & & \text{I} & \text{I} & \text{I} & & & \\
 & \text{A} & \text{T} & \text{C} & - & \text{C} & \text{G} & \text{A} & \text{G} & \text{A} & \text{G} & \text{G} & \text{T} & \text{C} & \text{C} & \text{C}
\end{array}$$

where R, D, and I correspond to *Replacement*, *Deletion* and *Insertion* operations respectively. In the above example, a minimum of five edit operations is needed to transform S_1 to S_2. Assuming a *unit cost* for each edit operation would result in $\text{ED}(S_1, S_2) = 1\times\text{R} + 1\times\text{D} + 3\times\text{I} = 1+1+3 = 5$. Computing the optimal alignment of n sequences, each of length l requires $o(2^n l^n)$-time and $o(l^n)$-space. Unfortunately, such an algorithm is neither practical nor scalable. Following is a summary of some of the problems encountered in the sequence similarity search within the context of biological sequences:

- The quadratic *computational complexity* of the optimal sequence alignment makes it *impractical* to be applied to long sequences.
- Due to the limitations on the current knowledge of *mutations* and their corresponding probabilities, only approximate searches and heuristics[4,9,14,16,20,22] have been practically applied for comparison of sequences.
- *Scalability* is one of the most important issues that needs to be addressed. The dynamic programming algorithms are not practical for a large number of sequences, each of which might be composed of billions of residues.

In this paper, we propose the application of Discrete Fourier Transformation (DFT) and Discrete Wavelet Transformation (DWT) techniques for efficient reduction of the search space and effective filtration of the intermediate results set. These transformation techniques map each sequence into a point in a multi-dimensional coordinate system. The distance among the corresponding points is used as a similarity measure to reflect the similarity of the original sequences.

The rest of the paper is organized as follows: Section 2, discusses the background and related work, followed by the motivation and terminology in section 3. Section 4, studies the proposed transformation techniques and their integration. Section 5 demonstrates a concise empirical performance analysis and the simulation results. Finally, section 6 concludes the work.

2. Background, Related Work

In a typical application of similarity search range query, given a protein or DNA query sequence Q and range r, it is compared with all the sequences in the database in search for sequences which are at most r edit operations far from the given query Q. However as mentioned before, because of the quadratic time involved, the dynamic programming[19,21] algorithms may not be directly or practically applied for this purpose. Several heuristics[4,7,9,14,16,20] have been proposed to speed up the homology search procedure, which are not efficient for range query over large datasets. These heuristics need to inspect the entire database while only a very small part of it might actually be of interest. The rest of this section highlights the recent research addressing this problem.

The Multi-Resolution index Structure (MRS)[13] is a technique based on *Haar wavelet* to speed-up range queries. It uses a sliding window of size $|w|$, moving over the query sequence, and for each possible location extracts the first and second *Haar wavelet* coefficients of the $|w|$-sized subsequences/windows. Hence each window is mapped to a point in $\Re^2$. Furthermore, every c trail of windows is represented with a single *Minimum Bounding Rectangle(MBR).* The trail of MBRs are subsequently processed at different resolution levels, based on different values for $|w|$. Given a range query (Q, r), the query is first divided into the maximum 2^i-sized postfix segments and each segment is searched within the respective resolution level. However, $i)$ the lower bound provided by the distance function is not tight enough for the *score* and *Edit Distance* (ED) estimations, and $ii)$ the focus of the work is on the performance of the index structure and does not analyze the filtration efficiency of using *wavelet* transformation on the incorporated biological datasets.

Chavez and Navarro[8] translate the problem of approximate string search into a range query or proximity search in a metric space. The technique is based on picking k pivots randomly and mapping each sequence to a k-dimensional vector (only keeping min and max distances) and furthermore uses the triangle inequality to prune non-relevant sequences using Suffix Trees[5] as an index structure. No empirical analysis is conducted to evaluate this approach on real or synthetic data.

SST[10] uses overlapping sliding windows of size w over the database sequences and maps them into a $\Re^{4^w}$-dimensional frequency vectors. Subsequently, SST uses k-means clustering algorithm to hierarchically cluster the database sequences. Given a query Q, it is first divided into non-overlapping windows, pruning the database windows which are farther from the given query range and finally studying the effect of window size on search time, and the error rate of input data on true positive/negative rates. Wu et al.[25] provide a concise study of DFT/DWT transformations, but only in the context of time-series databases. Similarly, Aghili et al.[1,2] incorporate Fourier, Wavelet and Singular Value Decomposition (SVD) transformation techniques to perform sequence similarity search and study the imposed trade-offs.

In this work[a], we study the effectiveness of the integration of DFT/DWT for the purpose of sequence similarity search specifically in the context of biological sequence databases.

[a] This work is an extension of the study presented in Ref. (3).

Table 1. Notations used throughout the paper.

NOTATION	DESCRIPTION
Σ	The alphabet set (e.g. Σ = A,C,G,T for DNA sequences).
T	A database of sequence files .
T_i	A file in database T consisting of a number of sequences.
$T_{i,j}$	A subsequence/block in file T_i starting at offset index j.
S	A sequence taken from the alphabet set.
$f(S)$	The frequency/numerical representation of the sequence S.
$\mathbf{ED}(S, S')$	Edit distance between the sequences S and S': The minimum number of edit operations needed to transform S to S'.
$\mathbf{FD}(f(S), f(S'))$	Frequency distance between the frequency vectors $f(S)$ and $f(S')$.
$\varpi_k(S)$	The k^{th}-level *Haar Wavelet Transformation (DWT)* of sequence S.
$X_k(S)$	The k^{th} *Discrete Fourier Transformation (DFT)* coefficient of sequence S.

3. Motivation and Terminology

This section introduces the terminologies used throughout the paper. For further clarification, a summary of the notations is provided in Table 1.

Definition 1. (Range query) Let $T = T_1, \ldots, T_N$ be a sequence file database over the alphabet Σ. Let $T_{i,j}$ denote a subsequence in file T_i starting at offset index j, for $1 \leq i \leq N$ and $0 \leq j < |T_i|$, where $|T_{i,j}| \leq |T_i| - j$. Given a query sequence pattern $Q \in \Sigma^*$ and a range r, a *range query* is the problem of finding the result set $R(Q, r)$ as the set of all the subsequences $T_{i,j}$ such that $ED(Q, T_{i,j}) < r$.

Definition 2. (Frequency vector, Alphabet vector) Let $S = s_1, \ldots, s_n$ be a sequence over the alphabet $\Sigma = \{\alpha_1, \ldots, \alpha_k\}$, where each letter $s_i = \alpha_j$ for some $1 \leq i \leq n$ and $1 \leq j \leq k$. The *frequency vector* of S, called $f(S)$ is defined as: $f(S) = [f_1, \ldots, f_k]$, where f_i corresponds to the occurrence frequency of α_i in S, and $\sum_{i=1}^{k} f_i = |S| = n$. Moreover, suppose $s_i = \alpha_j$ then the *alphabet vector* of s_i, v_{s_i}, is defined to be a $|\Sigma| \times 1$ vector where its j^{th} entry is 1 and all the other entries are filled with 0:

$$v_{s_i} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix}, where\ the\ j^{th}\ entry\ is\ 1\ and\ all\ the\ other\ entries\ i \neq j\ ,\ are\ 0.$$

Example 1. For instance, let $S = AGGTTGCAATTA$ be a sequence over alphabet $\Sigma = \{A, C, G, T\}$, then $f(S) = [4, 1, 3, 4]$ and each v_{s_i} is a 4×1 vector:

$$v_{s_1} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad v_{s_2} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \quad v_{s_3} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \quad \ldots, \quad v_{|S|} = v_{s_{12}} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

For instance, the 3^{rd} entry of v_{s_2} is 1 because $s_2 =$ 'G' which is the 3^{rd} symbol in the alphabet Σ.

One way to solve the *range query* problem is as follows: Given a query pattern Q, compare all sequences stored in the database against Q using *Edit Distance* (ED), either through direct application of dynamic programming[19,21] or other popular heuristics[4,7,9,14,16,20], and determine the answer set $R(Q, r)$. Although this approach is correct, it is not practical/scalable for two reasons. First, sequence databases may involve a large number of very large sequences (e.g., Chr_{22} is the smallest human chromosome[18] which consists of approximately 35 million base pairs) resulting in severe performance penalty. Secondly, the prohibitive computational cost of alignment or even heuristic-based sequence comparison makes it impractical, especially when $|R(Q, r)|/|T|$ is very small.

A solution could be mapping the problem of range query sequence similarity, $R_{ED}(Q, r)$, into a range query in a numerical/vector domain which incorporates a *Frequency Distance (FD)*, $R_{FD}(Q, r)$), to benefit from much more time/space-efficient numerical methods in the literature. One way is to use a mathematical transformation to map the *sequence domain* of sequences S_i into a *vector/frequency domain* for frequency vectors $f(S_i)$ and use an appropriate frequency distance function to estimate the edit distances of the sequence domain. If the correct mapping/transformation is applied, the *Parseval Theorem* implies that *frequency distance* is less than or equal to edit distance, or in other words $FD(f(S_i), f(S_j)) \leq ED(S_i, S_j)$ (*Distance preserving transformation*)[b]. This property is the main driving force behind using transformations. Specifically:

- The calculation of distance in the frequency domain (FD), is much more *time/space-efficient* compared to the calculation of the distance in the original sequence domain (ED).
- Range queries are much more efficiently evaluated in the frequency domain. For instance, consider a query pattern frequency vector $f(Q)$, range r and a set of frequency vectors $f(S_1), \ldots, f(S_n)$, then all the frequency vectors (and in turn sequences) $f(S)$, $where\ FD(f(Q), f(S_i)) > r$ may be pruned from the answer set without the need to investigate further (to calculate the

[b] The equality holds when all the transformed coefficients are used in the frequency distance calculations

original ED), at a very low cost. This would dramatically reduce i) the *computational cost*[13] and, ii) the required amount of *search space* $R_{FD}(f(Q), r)$ for a given range query (Q, r). However, a very important requirement is to guarantee that $R_{ED}(Q, r) \subseteq R_{FD}(f(Q), r)$ to avoid *false negatives.*

The following definitions introduce the steps used in transforming the *original domain (set of sequences)* to the *frequency domain (set of vectors)*:

Definition 3. (**Frequency Quantization**) Let $S = s_1, \ldots, s_n$ be a sequence from the alphabet Σ, *frequency quantization* of S, $S^F = [s^1, \ldots, s^n]$, is an $|\Sigma| \times |S|$ matrix of $|\Sigma| \times 1$ vectors v_{s_i}, for $1 \leq i \leq n$.

Let S be the same sequence as given in example 1, then

$$S^F = \begin{bmatrix} 1\,0\,0\,0\,0\,0\,0\,1\,1\,0\,0\,1 \\ 0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0 \\ 0\,1\,1\,0\,0\,1\,0\,0\,0\,0\,0\,0 \\ 0\,0\,0\,1\,1\,0\,0\,0\,0\,1\,1\,0 \end{bmatrix}.$$

Next, we introduce two of the famous distance preserving transformations which we deployed in our study.

Definition 4. (**Discrete Wavelet Transformation, DWT**) The k^{th}-level *Haar Wavelet Transformation (DWT)*[13] of a frequency-quantized sequence S, $\varpi_k(S)$, for $0 \leq k \leq log_2 n$, is defined as $\varpi_k(S) = [v_{k,0}, v_{k,1}, \ldots, v_{k,\frac{n}{2^k}}]$, where $v_{k,i} = [\alpha_{k,i}, \beta_{k,i}]$, for

$$\alpha_{k,i} = \begin{cases} f(c_i) & k = 0 \\ \alpha_{k-1,2i} + \alpha_{k-1,2i+1} & 0 < k \leq log_2 n, \end{cases}$$

$$\beta_{k,i} = \begin{cases} 0 & k = 0 \\ \alpha_{k-1,2i} - \alpha_{k-1,2i+1} & 0 < k \leq log_2 n, \end{cases}$$

where for $k = log_2 n$: $\alpha_{log_2 n,0} = f(S[0 : n-1])$ and $\beta_{log_2 n,0} = f(S[0 : \frac{n}{2} - 1]) - f(S[\frac{n}{2} : n-1])$ represent the first and second Haar wavelet coefficients, respectively.

For instance, for the same S as given in example 1, the 3^{rd}-level DWT of S: $\varpi_3(AGGTTGCAATTA) = \{\alpha_{3,0}, \beta_{3,0}\} = \{[4, 1, 3, 4], [-2, -1, 3, 0]\}$, represents the set of first and second wavelet coefficients.

Definition 5. (**Discrete Fourier Transformation, DFT**) The *n-point Discrete Fourier Transformation (DFT)* of a sequence $S = [S_t]$, for t=0,. . . ,n-1 is defined to be a sequence X of n complex numbers x_f of $|\Sigma| \times 1$ vectors, for $f = 0, \ldots, n-1$, and is given by

$$x_f = \tfrac{1}{\sqrt{n}} \textstyle\sum_{t=0}^{n-1} S_t e^{\frac{-j2\pi f t}{n}}, \text{ f = 0, 1, \ldots, n-1},$$

where $j = \sqrt{-1}$ is the imaginary unit. The original sequence S can be restored by the inverse transform:

$$S_t = \tfrac{1}{\sqrt{n}} \textstyle\sum_{f=0}^{n-1} x_f e^{\frac{j2\pi ft}{n}}, \text{ t = 0, 1, \ldots, n-1},$$

where x_f is a complex number and its real and imaginary parts are $|\Sigma| \times 1$ vectors.

Let $S' = ACCT$, the first and second DFT coefficients of S' are calculated as: $X_0(S') = [\frac{1}{2}, 1, 0, \frac{1}{2}]$ and $X_1(S') = \{[\frac{1}{2}, \frac{-1}{2}, 0, 0], [0, \frac{-1}{2}, 0, \frac{1}{2}]\}$, respectively.

Meanwhile, one question to be answered is *What would be the proper FD distance to deploy in the frequency domain to provide a good approximation of the edit distance of the original space?*

In the context of frequency transformations in multi-dimensional indexing, L_p-norm distance measures [15] are usually the popular choice for the frequency distance function, however the choice is application-dependent. The incorporated Frequency Distance (FD) of the feature vectors should provide an accurate estimate of Edit Distance (ED) among the sequences. For any two sequence frequency vectors X, Y: the incorporated L_1-norm is defined as the minimum number of increment, decrement or (± 1) operations needed to transform vector X into vector Y which is a lower bound on the edit distance. For instance, given $X = [0, 1, 2, 3]$ and $Y = [0, 1, 3, 2]$: $L_1(X, Y) = 1$ because X can be transformed into Y by a single ± 1 operation.

4. Transformation procedure

In this section, we provide the details of our proposed search technique. The algorithm is performed in two different stages, namely *offline* and *online*. As depicted in Figure 2, given a sequence database T, all its corresponding sequences S_i are divided into blocks and each block is mapped onto a frequency vector. The transformation techniques are applied on the extracted frequency vectors and the resulting reduced vectors are stored in an offline profile for each given sequence of the database. For instance, for a given database T, two offline files to store its DFT and DWT vectors are created, namely T_{DFT} and T_{DWT}. Given a query sequence Q (online), a similar method is used to extract its blocks and mapping them into frequency vectors. The search algorithm continues by comparing the query's vectors against the vectors of T_{DFT} or T_{DWT} and pruning the irrelevant portions of the database. The resulting similar locations of the database T to the query Q are reported accordingly. Figure 1 provides a detailed description of the proposed transformation algorithm.

5. Performance Analysis

We compared the application of the transformation techniques with a few other approaches. The first one so called *String* is the *q*-gram indexing method used by QUASAR[7]. We also implemented an space-efficient and a faster variation of *String*, named *Vector* and *Tuple* respectively. These implementation deploy an additional inverted table index structure and were incorporated as benchmarks to assess the performance of our proposed algorithms.

Pre-processing phase (Offline): Given a sequence database $T = \{T_1, T_2, \ldots, T_n\}$ where each $T_i \in \Sigma^*$ represents a sequence file:

(1) Let m denote the size of the shortest sequence present in any of the sequence files. Choose the window size $|w|$ to yield an optimal total number of blocks on that sequence as $j = \theta(m/log_{\Sigma}N)^{16,17}$, for database size N. This optimal j has been suggested for pattern partitioning, however the length of the pattern might not be known in advance, so we restrict the maximum size of the pattern by the size of the minimum sequence in the database to be able to benefit from the optimal partitioning.

(2) Slide the window on each of the original sequence files T_i and extract the corresponding $|w|$-sized blocks. Subsequently, partition each T_i on the starting positions of their extracted blocks, at offsets 0, $\frac{|w|}{2}$, $\frac{2|w|}{2}$, $\ldots$ into a total of $\frac{|T_i|-|w|+1}{\lfloor\frac{|w|}{2}\rfloor}$ blocks. Let $b_{i,j}$ denote the block extracted from the sequence file T_i at position j, where $1 \le i \le n$, and $0 \le j < |T_i| - |w|$.

(3) Perform *frequency quantization* (Def. 3) on each of the blocks $b_{i,j}$, extracting $b^F_{i,j}$,

(4) Use the desired DFT/DWT transformations on each of the *frequency-quantized* blocks $b^F_{i,j}$ and calculate the corresponding transformed vector $X(b^F_{i,j})$ or $\varpi(b^F_{i,j})$ coefficients in the frequency domain.

(5) Extract and store only a few coefficients (at most *three*) to represent the original subsequence/block. For the case of DFT, we keep the highest energy-concentrated-coefficients as, first, last and the second[25]. For the DWT, we keep the first and second coefficients, in which the energy of the sequence is expected to be mostly concentrated.

(6) Build an offline index structure as follows: For each of the sequence files in the database, T_i, keep a list of block vectors contained in that sequence. We keep only an index for the location of the extracted block and the corresponding fixed-size frequency vector(s), which are at most two for DWT and three for DFT. The higher precision might be achieved by choosing more coefficients which is a built-in feature in our implementation.

Query processing(Online): Given a query pattern $Q \in \Sigma^*$ and range r:

(1) Slide the $|w|$-sized window on the pattern sequence Q, partitioning it into non-overlapping segments of length $|w|$, for a total of $j' = \lfloor |Q|/|w| \rfloor$ partitions. Let Q_l denote the partition of Q starting at index l, where $1 \le l \le |Q| - |w| + 1$.

(2) For each of the extracted partitions Q_l:

- Perform *frequency quantization* (Def. 3) on Q_l block to get Q^F_l,
- Apply DFT/DWT transformations on the *frequency-quantized* blocks Q^F_l and extract the corresponding $X(Q^F_l)$ or $\varpi(Q^F_l)$ coefficient vectors,
- Search each of the coefficient block vectors, $b^F_{i,j}$, stored in the *offline* index, and prune all the subsequences/blocks for which,
  - DFT: $FD(X(Q^F_l), X(b^F_{i,j})) > \frac{r}{j'}$, or
  - DWT: $FD(\varpi(Q^F_l), \varpi(b^F_{i,j})) > \frac{r}{j'}$.

(3) Calculate ED only for the candidate result set (those not pruned) to find the subsequences S_i, where $ED(Q, S_i) < r$.

Fig. 1. Transformation procedure.

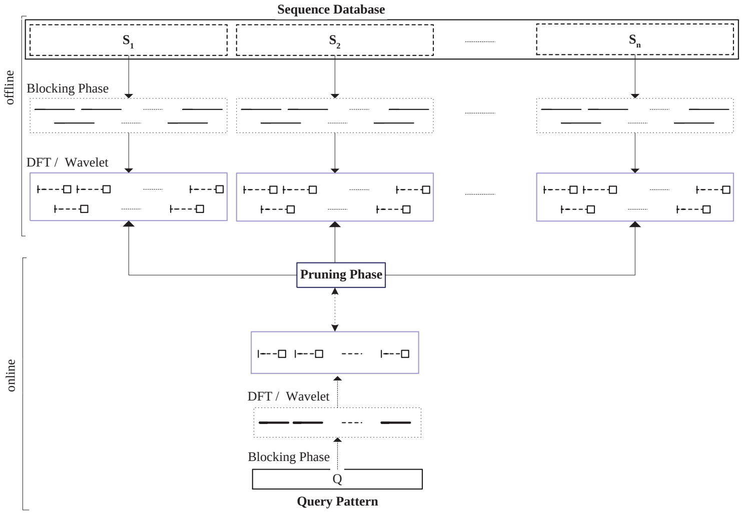

Fig. 2. The transformation procedure.

5.1. *Implementation*

String. The *String* method uses a *block addressing scheme* as follows: Each of the contig sequences of the database and the query pattern sequence are partitioned into blocks, b_i, of fixed size $|w|$ (as described in the previous section) for a total of B blocks in database. A counter C_{b_i} is associated with each block b_i of the database, respectively. For each q-gram of size q, an index structure of size $|\Sigma|^q$ is maintained (q=3). Each entry corresponds to a unique q-gram q_j followed by a list of blocks b_i (and their corresponding counters C_{b_i}) which contain the q_j q-gram. All the q-grams of the blocks of a pattern are inspected (consecutive q-grams of the blocks overlap in $q-1$ bases) and the counter C_{b_i} is incremented whenever a search for that q-gram reports *"existing"* in the b_i. After processing all the blocks and the corresponding q-grams of the pattern, each counter C_{b_i} indicates how many unique q-grams (ignoring the positional information) from Q are contained in block b_i of the database. Thereafter, all the block counters are stored in an array of size $|T|/B$ and the blocks b_i whose counters C_{b_i} contain (share) less than $max(|Q|, |b_i|)+1-(r+1)\cdot|q|$ q-grams with the pattern are pruned from the candidate set[7,12]. We employed a uniform blocking method across all different methods. Note that the *String* q-gram method is an approximate method in contrast to the dynamic programming alignment algorithm and hence potentially suffers from false positives.

Vector. The *Vector* method is very similar to the *String* method, it additionally incorporates an index structure to the *String* to save on the total amount of *space* needed to store each block. Accordingly, for each block b_i the corresponding frequency vector $f(b_i)$ is calculated (Def. 2). Each $f(b_i)$ is mapped into an integer as follows: Let $f(b) = [f_1, f_2, f_3, f_4]$ then the corresponding identifier for b_i, I_{b_i}, is defined as

$$I_{b_i} = \sum_{k=1}^{4} f_{k-1} |b_i|^{|\Sigma|-1}.$$

The same identifier translation was used to map each q-gram to an integer. This would decrease the number of entries in the index structure exponentially (for $q \gg 3$) which would be of size $q^{|\Sigma|}$. Each q-gram/block vector υ is mapped into an integer value I_υ, where $|\upsilon| \leq I_\upsilon \leq |\upsilon|^{|\Sigma|}$. However, this is not a 1-to-1 mapping, therefore only the nonzero entries are kept. The rest of *Vector* is exactly like *String*. However, *Vector* is expected to incur more false positives on average, which is due to the degeneracy of mapping and loss of the positional information during translation.

Tuple. We also implemented a third improvement named *Tuple* to tackle the *time* complexity of *String*. Each of the blocks are stored as a $|\Sigma|^q$-dimensional frequency vector (zero entries are neglected) where each entry i corresponds to the quantity of the unique q-gram q_i in that block. As for the index structure, the *Tuple* does not need any extra space to keep the q-gram index as in the *String*[7]. The transformed vectors include store information on the contained q-grams, where each q-gram count is implied by its corresponding entry. This technique provided exactly the same result as *String*, however was much more time-efficient. Hence, it is not included in our pruning graphs but is included in our timing comparison table (Table 2).

For a database of N sequences, containing B blocks, the *String* uses $(N+\frac{B}{2})(\text{int}) = O(B)(\text{int})$ space$(B \gg N)$, which is linear in space. However its computational cost is $O(|Q|B)$. Similarly, *Vector* has the same computational complexity, however is exponentially more space-efficient (only for $q \gg 3$), at the cost of more false positives on average. The amount of saved space would be approximately equal to

$$\lim_{q\to\infty} \frac{|\Sigma|^q}{q^{|\Sigma|}}.$$

Tuple is $O(|\Sigma|^q \cdot B)$-space and $O(B)$-time which is linear in the total number of blocks. We could also use a tree-based approach[10] and reduce the search time to $O(logB)$. Additionally, in all of the above methods, we also incorporated different blocking and q-gram partitioning methods, as follows:

- *Incremental* partitioning: Each of the consecutive q-grams/blocks of length t, overlapping by $t-1$ residues,
- *HalfOverlap* partitioning: Each of the consecutive q-grams/blocks of length t, overlapping by $t/2$ residues, and
- *non-overlapping* partitioning.

For both q-gram and block partitioning, the more q-gram/blocks were extracted, a higher computational cost was observed, in return for better filtration ratio, tighter FD bound and a smaller candidate set. This choice is a trade-off between *cost* versus *precision.* However, due to the space limitations and compactness of the paper, we did not include those results in this study. We implemented all the desired algorithms and transformations using $Java$, and ran our simulations on a PIII-800Mhz with 1GB of main memory.

5.2. *Considerations*

The following requirements should be fulfilled regarding any transformation technique:

- The Frequency Distance (FD) should be a fair approximation to the original Edit Distance (ED).
- If more representative coefficients are chosen in the frequency domain, then a higher precision on the real distance approximation and more filtration efficiency is expected.
- When FD has a smaller value, a *more compact space* is to be observed. This property relies upon the fact that a decrease in FD value would result in vectors being located closer to each other in the frequency space. We could also store a trail of quantized vectors as *Minimum Bounding Rectangles (MBR)* to minimize the total number of *MBR*s needed to store the frequency vectors, at the cost of less efficient filtration ratio.
- The calculation of FD should be computationally as efficient as possible and using a minimum number of coefficients should ensure reasonably effective filtration.
- The Filtration Ratio (FR) is to be maximized (incurring low or no false negatives), however the efficiency of pruning depends on: *i*) the structure of sequences, *ii*) query sequence, and *iii*) query range.

5.3. *Simulation Results*

Let $\circ$, $\diamond$ and $\star$ denote the use of 1^{st}, $(1^{st} + 2^{nd})$ and $(1^{st} + 2^{nd} + Last)$ coefficients. The energy of the coefficients are mostly concentrated[25] at the first and second coefficients of DWT, or the first and last coefficients of DFT, respectively. These observations were incorporated while deploying the appropriate coefficients for each of the transformation techniques. Figures 3-5, demonstrate the result of running *String*, *Vector* and the application of DFT/DWT transformation techniques for the choice of different coefficients on *Alu*, $Mitochondria$, and *Escherichia coli (E.coli)* contig sequence databases[18] for a random query pattern of length 16, as the query range varies from 4 to 14.

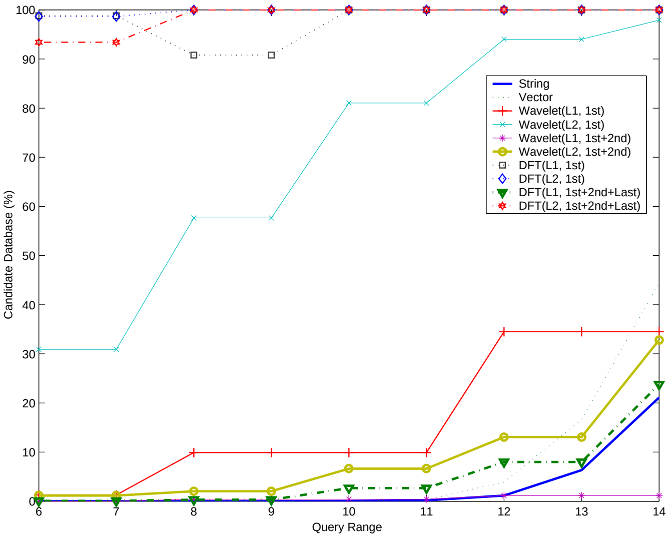

Fig. 3. The resulting candidate answer set as a function of query range on *Alu* database.

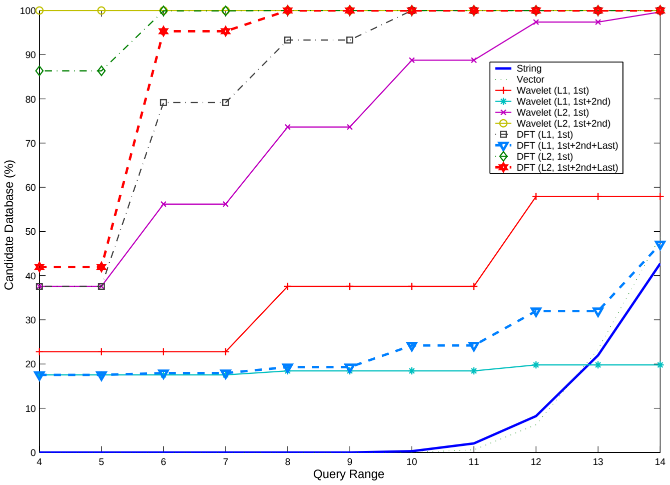

Fig. 4. The resulting candidate answer set as a function of query range on *Mitochondria* database.

Let B denote the total number of blocks in the database. In Figures 3-5, the vertical axis shows the fraction of the database that is left for further investigation (those not pruned), that is $\frac{|R_{FD}(f(Q),r)|}{B}\%$. The horizontal axis represents the corresponding query range. In Figure 3, as expected, *Vector* gives more false positives, and $DFT_{L_1,\star}$ demonstrates the best Filtration Ratio (FR), for $(23.37)^{-1} \leq FR_{Alu} \leq (0.07)^{-1}$ incurring no false negatives for the inspected query range. On the other hand, $DWT_{L_1,\diamond}$ results in false negatives compared with

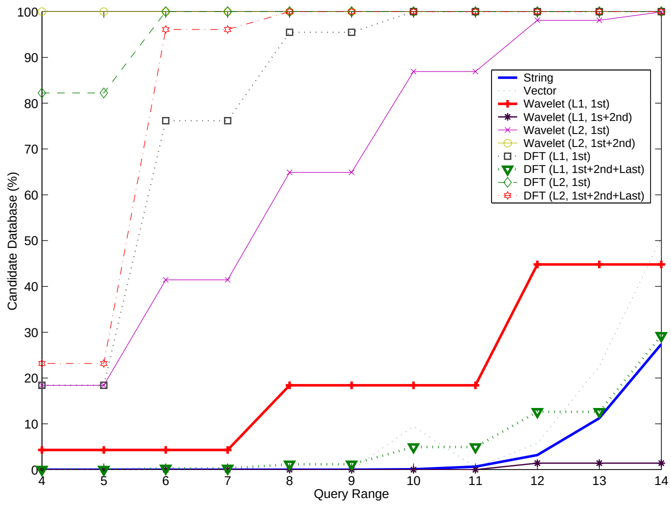

Fig. 5. The resulting candidate answer set as a function of query range on *Escherichia coli(E.coli)* database.

Table 2. The timing comparison (in *seconds*) of running 11 different range queries on the described techniques, for three *contig* sequence datasets, for a random query of length 16.

SPECIES	***String***	***Vector***	***Tuple***	$DWT_{L_1,\diamond}$	$DWT_{L_2,\diamond}$	$DFT_{L_1,\star}$	$DFT_{L_2,\star}$
Alu	**4.67**	5.09	5.5	4.62	4.5	**5.21**	6.92
Mitocondria	**1921.7**	3209.9	176.74	152.3	152.6	**175**	236.2
E. coli	**534.7**	929	259.59	221.3	224.9	**289**	369.9

String, which indicates that it gives poorer performance. We will investigate this behavior later in the section. A similar behavior is observed in Figure 4, $DFT_{L_1,\star}$ gives the best filtration with no false negatives, for $(47.13)^{-1} \leq FR_{Mito} \leq (17.54)^{-1}$, but $DWT_{L_1,\diamond}$ incurs false negatives as before. Figure 5, depicts the best expectations, no false negatives of any kind on any of the transformations. Again $DFT_{L_1,\star}$ gives the best estimates with $(29.21)^{-1} \leq FR_{E.Coli} \leq (0.02)^{-1}$. In all the figures, using more coefficients (on L_1 or L_2) leads to more efficient vector transformation and more effective filtration.

Table 2 shows the timing comparison, resulting from running 11 different range queries (4-14) on three real datasets. Compared with *String* and *Vector* q-gram methods, our proposed transformation techniques were always faster (including the offline index construction overhead) and the performance improves to 11-13 faster running time while very closely approximating the *String* q-gram method. Figure 6 shows the distribution of *True Positive Rate (TPR)* of $DFT_{L_1,\star}$ compared with *String* on the same range queries investigated earlier. For instance, joining the results of Figures 3-6 and Table 2 for the case of *Mitochondria* dataset, $DFT_{L_1,\star}$ effectively prunes up to $100 - (17.54)^{-1} \simeq 99\%$ of the database. It takes $\frac{1921.7}{175} \simeq 9\%$

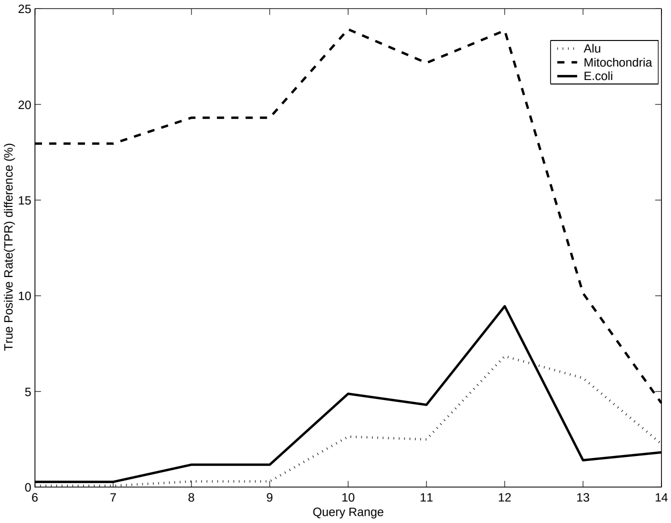

Fig. 6. *True Positive Rate(TPR)* of $DFT_{L_1,\star}$ compared with *String q*-gram method on *Alu*, *Mitochondria* and *E.coli* datasets.

of the total time needed for the *String q*-gram method and incurs no false negatives. However, as we mentioned before, the transformation application is unfortunately very much data dependent. Therefore, we ran the experiments on a much wider range of environments and inspected the performance improvements.

Figures 7-9, demonstrate the results of producing 100 random query patterns Q of length 8 and 32, and performing the range queries for $1 \leq r \leq |Q|$. For all the datasets and on all the experiments, it can be observed that the $DFT_{L_1,\star}$ transformation does not produce any false negatives up to the range query $r \leq |Q| - \varepsilon$, for $\varepsilon \leq 3$, however due to the blocking method used in the *String* method, more false positives are expected[7]. Inspecting the bottom section of Figure 7 on the range 28-32, it seems that $DFT_{L_1,\star}$ is finding less results than *String* by causing a larger space reduction. This artifact may appear as if $DFT_{L_1,\star}$ is suffering from false negatives. However, the *String* is an approximation method itself which incurs false positives, hence the difference set may not actually pertain to false negatives for $DFT_{L_1,\star}$. The filtration being less than that of q-graming method, does not necessarily imply false negatives.

For this purpose, we investigated every single candidate block produced by *String*, $DFT_{L_1,\star}$ and $DWT_{L_1,\diamond}$ on a random portion of *Alu* database for some random query patterns of length 16 and performed a range query of 14 which had "seemingly" caused false negatives, on all the datasets and inspected the corresponding precision and recall, as depicted in 10-13. In the inspected configurations, *String*, $DFT_{L_1,\star}$ and $DWT_{L_1,\diamond}$ filtrations, reduced the database size to 7.75%, 14.08% and 1.41%, respectively. $DWT_{L_1,\diamond}$ produced false positives, in addition to a couple of

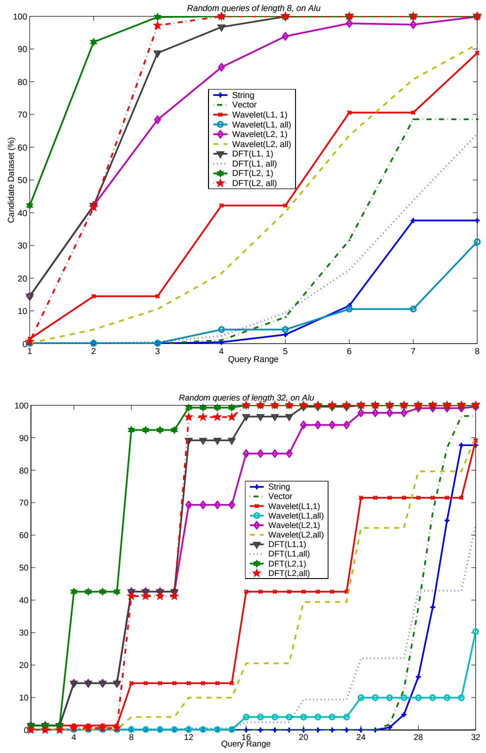

Fig. 7. Random query patterns of various length, and range queries on *Alu*.

false negatives! However, $DFT_{L_1,\star}$ caught all the actual k-distant blocks of the database. Thereafter we inspected the recalls. The *String* method depicted a better performance by producing less false positives compared with $DFT_{L_1,\star}$. However, $DFT_{L_1,\star}$ did not miss any actual k-distant block while $DWT_{L_1,\diamond}$ generated false negatives and missed some of the correct results. Both $DFT_{L_1,\star}$ and *String* resulted in a precision of 1, not missing any correct results, in contrast to $DWT_{L_1,\diamond}$. The calculation of the distance in the *String* method was based on the difference in the number of shared q-grams, however, we used the frequency vector difference for DFT and DWT. For this reason, none of the sets of false positives produced by DFT and *String* were a subset of one or another. The intersection of the results produced by them consisted of all the k-distant blocks in addition to a few false positives.

Inspecting the experimental evaluations, provided in this section, depicts the efficiency and effectiveness of $DFT_{L_1,\star}$ in reducing the intermediate result set. The $DFT_{L_1,\star}$ transformation may effectively be applied as a pre-processing phase to prune irrelevant sequences for query ranges up to half the length of the query sequence pattern ($r \leq |Q|/2$).

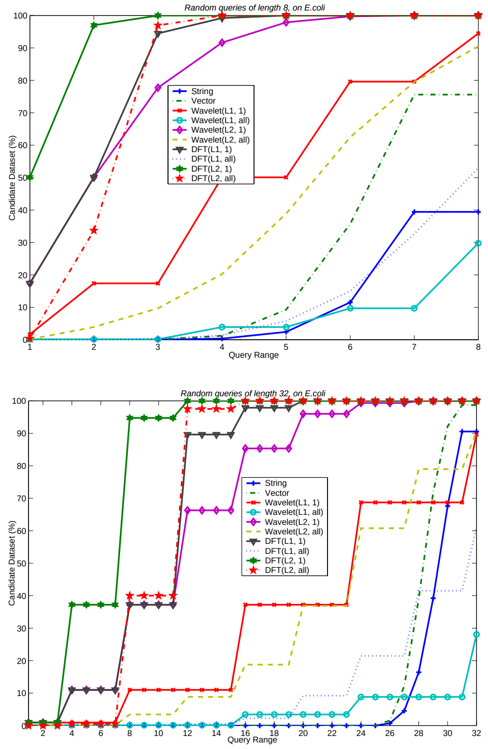

Fig. 8. Random query patterns of various length, and range queries on *Escherichia coli(E.coli)*.

6. Conclusion

In this paper, we studied the application of *Discrete Fourier Transformation (DFT)* and *Haar Discrete Wavelet Transformation (DWT)* transformations on biological sequences and evaluated the specific problem of range query. Transformation methods may be applied to prune most of the non-desired sequences and reduce the real search problem to only a fraction of the database. Such transformations may be incorporated as a pre-processing phase for any of the known heuristic approaches such as BLAST[4], PatternHunter[14], QUASAR[7], FastA[20], and even the sequence alignment[19,21]. Our results show that applying the transformation technique results in a high accuracy and faster database pruning, when the behavior of the studied transformations is taken into account before applying the appropriate range query. The filtration ratio is very much data dependent and no generalization on the min/max filtration ratio or true positive rates can be suggested. However, the empirical results show promising performance behavior, especially on $DFT_{L_1,\star}$, incurring no false negatives, high filtration ratio, while being considerably faster than the q-gram based methods.

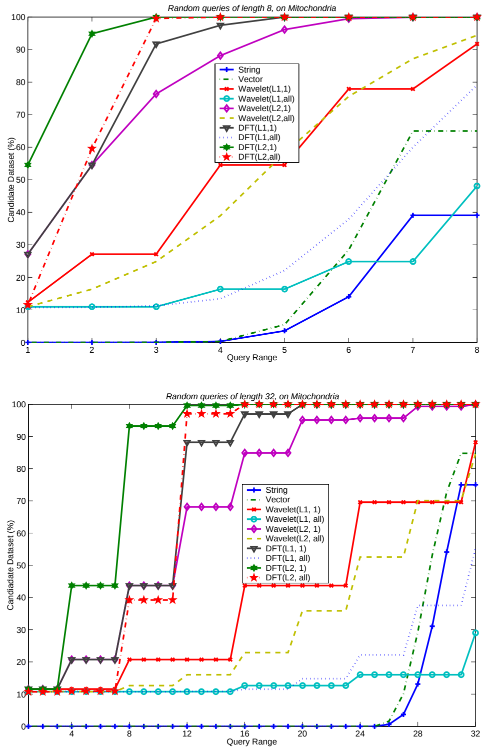

Fig. 9. Random query patterns of various length, and range queries on *Mitochondria*.

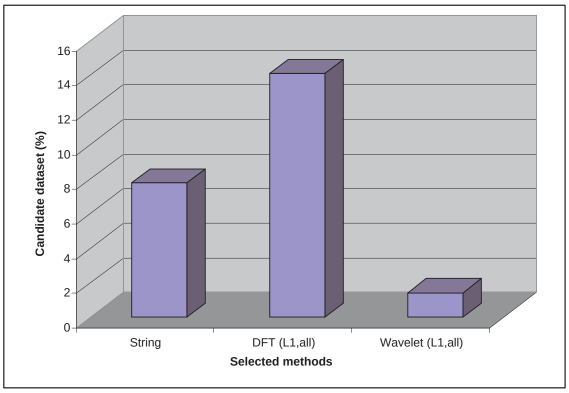

Fig. 10. Resulting candidate percentages of the *Alu* database for *String*, $DFT_{L_1,\star}$, and $DWT_{L_1,\diamond}$.

Acknowledgments

This research was supported by the NSF grants under CNF-04-23336, IIS02-23022, IIS02-09112, and EIA00-80134.

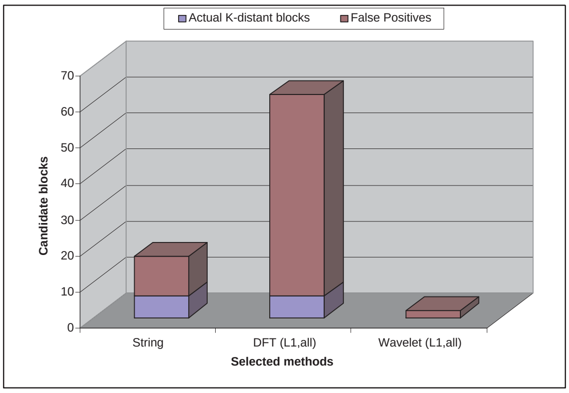

Fig. 11. Resulting candidate block distribution of *String*, $DFT_{L_1,\star}$, and $DWT_{L_1,\diamond}$.

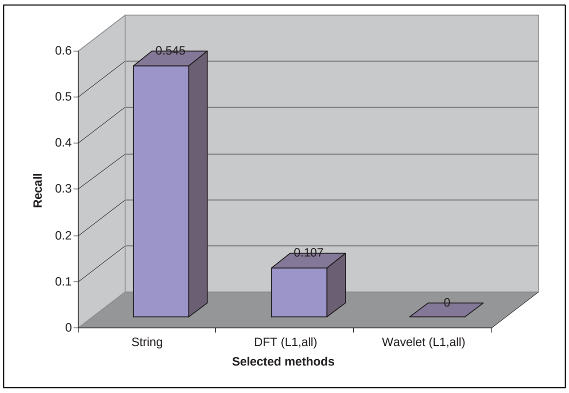

Fig. 12. Resulting recalls for *String*, $DFT_{L_1,\star}$, and $DWT_{L_1,\diamond}$.

References

1. S.A. Aghili, D. Agrawal and A. El Abbadi, *Efficient Filtration of Sequence Similarity Search Through Singular Value Decomposition*, (BIBE 2004), pp. 403–410.
2. S.A. Aghili, D. Agrawal and A. El Abbadi, *BFT: Bit Filtration Technique for Approximate String Join in Biological Databases*, (SPIRE 2003), pp. 326–340.
3. S.A. Aghili, D. Agrawal and A. El Abbadi, *Filtration of String Proximity Search via Transformation*, (BIBE 2003), pp. 149–157.
4. S. Altschul, W. Gish, W. Miller, E. Myers and D. J. Lipman, *Basic Local Alignment Search tool*, (Journal of Molecular Biology 1990(215)), pp. 403–410.
5. A. Apostolico, *The myriad virtues of subword trees*, (Combinatorial Algorithms on Words, NATO ISI Series, Springer-Verlag 1985), pp. 85–96.
6. A.D. Baxevanis and B.F. Francis Ouellette, *Bioinformatics: A Practical Guide to the Analysis of Genes and Proteins*, (Wiley Interscience 2001).
7. Stefan Burkhardt, Andreas Crauser, Paolo Ferragina, Hans-Peter Lenhof, Éric Rivals and Martin Vingron, *q-gram Based Database Searching Using a Suffix Array*

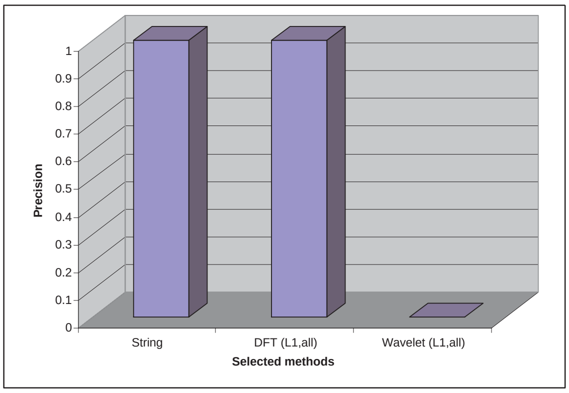

Fig. 13. Resulting precisions for *String*, $DFT_{L1,\star}$, and $DWT_{L1,\diamond}$.

(QUASAR), (RECOMB 1999), pp. 77–83.

8. E. Chavez and G. Navarro, *A Metric Index for Approximate String Matching*, (LATIN 2002), pp. 181–195.
9. M.O. Dayhoff, R.M. Schwartz and B.C. Orcutt, *A model of evolutionary change in proteins*, (Atlas of Protein Sequence Structure. National Biomedical Research Foundation, Washington, DC 1978(5)), pp. 345–352.
10. E. Giladi, M. G. Walker, J.Z. Wang and W. Volkmuth, *SST: an algorithm for finding near-exact sequence matches in time proportional to the logarithm of the database size*, (Bioinformatics 2002(18)(6)), pp. 873-877.
11. S. Henikoff and J.G. Henikoff, *Amino acid substitution matrices from protein blocks*, (Proceedings National Academy of Science 1992(89)), pp. 10915–10519.
12. P. Jokinen and E. Ukkonen, *Two Algorithms for Approximate String Matching in Static Texts*, (Proceedings of MFCS 1991(16)), pp. 240–248.
13. T. Kahveci and A.K. Singh, *Efficient Index Structures for String Databases*, (VLDB 2001), pp. 351–360.
14. B. Ma, J. Tromp and M. Li, *PatternHunter: faster and more sensitive homology search*, (Bioinformatics 2002(18)(3)), pp. 440–445.
15. Carl D. Meyer, *Matrix Analysis and Applied Linear Algebra*, (SIAM, Philadelphia, 2000).
16. G. Navarro and R.A. Baeza-Yates, *A hybrid indexing method for approximate string matching*, (Journal of Discrete Algorithms 2000(1)(1)), pp. 205–239.
17. G. Navarro, R.A. Baeza-Yates, E. Sutinen and J. Tarhio, *Indexing Methods for Approximate String Matching*, (IEEE Data Engineering Bulletin 2001(24)(4)), pp. 19–27.
18. NCBI, *National Center for Biotechnology Information(NCBI) website*, (http://www.ncbi.nih.gov/).
19. S.B. Needleman and C.D. Wunsch, *General method applicable to the search for similarities in the amino acid sequence of two proteins*, (Journal of Molecular Biology 1970(48)), pp. 443–453.
20. W.R. Pearson, *Using the 5 program to search protein and DNA sequence databases*, (Methods in Molecular Biology 1994(25)), pp. 365–389.
21. R. Smith and M.S. Waterman, *Identification of common molecular subsequences*, (Journal of Molecular Biology 1981(148)), pp. 195–197.
22. J. D. Thompson, D.G. Higgins and T.J. Gibson, *CLUSTAL W: improving the sensi-*

tivity of progressive multiple sequence alignment through sequence weighting, position specific gap penalties and weight matrix choice, (Nucleic Acids Research 1994(22)), pp. 4673–4680.

23. University of Chicago's Howard Hughes Medical Institute, *New type of DNA-free inheritance in yeast is spread by a madcow mechanism*, (http://www.uchospitals.edu/news/1997/19970530-prion-fibers.html).
24. D. Wheeler, *Weight Matrices for Sequence Similarity Scoring*, (http://www.techfak.uni-bielefeld.de/bcd/Curric/PrwAli/nodeD.html).
25. Y. Wu, D. Agrawal and A. El Abbadi, *A Comparison of DFT and DWT based Similarity Search in Time-Series Databases*, (CIKM 2000), pp. 488–495.