

A novel filtration method in biological sequence databases

Anthony J.T. Lee ^{*}, Chao-Wen Lin, Wen-Hsing Lo, Chieh-Chun Chen, Jia-Xin Chen

Department of Information Management, National Taiwan University, No. 1, Sec. 4, Roosevelt Road, Taipei 10617, Taiwan, ROC

Received 11 May 2005; received in revised form 22 August 2006

Available online 27 October 2006

Communicated by W. Pedrycz

Abstract

In this paper, we propose a new filtration method, called Transformation-based Database Filtration method (TDF), to screen out those data sequences of a DNA sequence database which cannot satisfy a given query sequence. Our proposed method consists of two phases. First, we divide each data sequence into several windows (blocks), each of which is transformed into a data feature vector using the Haar wavelet transform. The transformed data feature vectors are then stored in an index file. Second, we divide a query sequence into sliding windows, each of which is, again, transformed into a query feature vector using the Haar wavelet transform. We then search the index file to find the candidate sequences for each query feature vector and check if they match the query sequence using the sequence alignment algorithm. We transform the bound of edit distance between sequences to the bound of Manhattan distance between feature vectors. Since the Manhattan distance is much easier to compute, our proposed method can efficiently screen out impossible data sequences and guarantee no false negatives. The experimental results show that our proposed method outperforms the QUA-SAR method in terms of filtration ratio, precision, execution time and index size. The proposed method also outperforms the YM method for long query, low complexity and repetitive data.

© 2006 Elsevier B.V. All rights reserved.

Keywords: Bioinformatics; Sequence comparison; Filtration method

1. Introduction

Sequence comparison is one of the most important primitive operations in bioinformatics. By analyzing the sequences of genes and proteins, biologists can infer the structural, functional, or evolutionary relationships among them. Sequence alignment (Smith and Waterman, 1981) is one of the most popular sequence comparison methods, as it can accurately identify local alignments and provide an explicit mapping between sequences. Given two sequences S and S' , we can align them by inserting gaps in them and map each residue in S to the corresponding residue in S' . The score of such an alignment is then calculated

by consulting a scoring matrix that records the scores for each pair of residues. If the score of the alignment between S and S' is greater than a predefined threshold, we say that S and S' are similar. This algorithm yields a quadratic time complexity. However, the length of a complete human chromosome scales to millions of residues. It is impractical to search a large database with such an algorithm. Meek et al. (2003) proposed the OASIS algorithm, which is an order of magnitude faster than the Smith–Waterman's algorithm (Smith and Waterman, 1981), but such an improvement is good only for a short query sequence.

Trad et al. (2001) designed a method based on the wavelet transform to compare the similarity between protein sequences. Krishnan et al. (2004) also presented a method based on the wavelet transform to detect the motifs in a protein database. Recently, Yamada and Morishita (2005) proposed a method (hereinafter referred to as the YM method) to search for short interfering RNA (siRNA),

^{*} Corresponding author. Tel.: +886 2 33661188; fax: +886 2 33661199.

E-mail addresses: jtlee@ntu.edu.tw (A.J.T. Lee), d89003@im.ntu.edu.tw (C.-W. Lin), r91034@im.ntu.edu.tw (W.-H. Lo), r93725007@im.ntu.edu.tw (C.-C. Chen), r94725005@im.ntu.edu.tw (J.-X. Chen).

where the seed-hashing technique and pigeonhole principle are used to filter out impossible candidates. Due to its specific purpose, the YM method is designed to process short queries only. As all the mentioned methods are designed to calculate the Hamming or Euclidean distance between ungapped protein or DNA sequences, they are incapable of computing the edit distance between gapped sequences that is more frequently used in homology research.

Many methods based on the edit distance have been proposed in the last decade to speed up the process. Examples are Blast family (Altschul et al., 1990, 1997; Zhang et al., 2000), FASTA (Lipman and Pearson, 1985), FastA (Pearson, 1994), PatternHunter (Ma et al., 2002), SST (Giladi et al., 2002), and string search via transformation (Aghili et al., 2003). These filtration methods employ heuristics to improve the speed of searches, but they can have false negatives (Giladi et al., 2002). The algorithms with false negatives only find most but not all the sequences that satisfy the query. In contrast, the method without false negatives, such as QUASAR (Burkhardt et al., 1999), can find all the sequences satisfying the query. However, QUASAR is only suitable for dealing with highly similar sequences, and has a large memory requirement.

Therefore, in this paper, we propose a new filtration algorithm, called Transformation-based Database Filtration method (TDF), that can efficiently screen out those data sequences of a DNA sequence database which cannot satisfy a given query sequence. Our proposed algorithm, based on the approach of General Match (Moon et al., 2002), consists of two phases. First, we divide a data sequence into several windows, each of which is transformed into a data feature vector using the Haar wavelet transform. The transformed feature vectors will be the indices of the windows and stored in an index file. Second, we divide a query sequence into sliding windows, each of which is, again, transformed into a query feature vector using the Haar wavelet transform. For each query vector, we search the index file to find those data feature vectors whose Manhattan distances to the query feature vector are not greater than a predefined threshold. The blocks corresponding to the feature vectors thus found form the set of candidate blocks. For each candidate block, we will check whether the corresponding subsequence of the original data sequence matches the query sequence using the sequence alignment algorithm.

Our proposed method relies on the fact that if the edit distance between a query sequence and a data sequence is not greater than a user-specified threshold ε , then after transforming both sequences into feature vectors using the Haar wavelet transform, the Manhattan distance between transformed feature vectors will be less than the threshold derived from ε . Since the Manhattan distance is much easier to compute, our proposed method can efficiently screen out impossible data sequences and guarantee no false negatives.

The rest of this paper is organized as follows. The problem definition is described in Section 2. Section 3 presents

our proposed algorithm in detail. The experimental results and performance analysis are discussed in Section 4. Lastly, the conclusions are made in Section 5.

2. Problem definition

Let $S[1..n]$ be a sequence of length n . A *subsequence* $S[i..j]$ of S is a segment of consecutive entries, where $1 \leq i \leq j \leq n$. The sequence matching problem is to find the data sequences similar to a given query sequence. Two sequences S and S' are *similar* if the distance between them is not greater than a user-specified *tolerance threshold* ε . There are a wide variety of distance metrics. The *edit distance* is a frequently used metric to measure the degree of similarity between sequences in the area of bioinformatics. Given the three *edit operations* of *insert*, *delete*, and *replace*, the edit distance between two sequences S and S' , denoted by $ED(S, S')$, is defined as the minimum number of edit operations required to convert S into S' . For our purpose, we define S to be in ε -*match* with S' if $ED(S, S')$ is not greater than ε . Given N data sequences, a query sequence Q , and the tolerance ε , we aim to find all subsequences of data sequences that are in ε -*match* with Q .

2.1. Transforming a sequence into feature vectors

We use a sliding window to divide a sequence into short subsequences of length q (called q -grams), where q is the window size. For each q -gram, we can transform it into a feature vector.

Definition 1. Let S be a sequence over alphabet $\Sigma(A, C, G, T)$. A q -gram of S is a subsequence of S whose length is equal to q . We define feature vector of S with q -gram $F_q(S) = (\alpha_0, \alpha_1, \dots, \alpha_{|\Sigma|^q-1})$, where α_i is the number of occurrences of the q -gram that has an index value of i . The index value of q -gram $a_1 a_2 \dots a_q$ is denoted as $I(a_1 a_2 \dots a_q)$ and is calculated as follows:

$$i = I(a_1 a_2 \dots a_q) = \sum_{k=1}^q 4^{q-k} \cdot M(a_k)$$

$$M(a_k) = \begin{cases} 0 & \text{if } a_k = A \\ 1 & \text{if } a_k = C \\ 2 & \text{if } a_k = G \\ 3 & \text{if } a_k = T \end{cases}$$

For example, let $S = \text{AGGT}$. If $q = 1$, then $I(A) = 0$, $I(G) = 2$, and $I(T) = 3$. Thus, $F_1(S) = (1, 0, 2, 1)$. If $q = 2$, then $I(AG) = 2$, $I(GG) = 10$, and $I(GT) = 11$. Thus, $F_2(S) = (0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0)$, where α_2 is the number of occurrences of q -gram AG.

Next, we use the Haar wavelet transform (HWT) to reduce the dimensions of a feature vector.

Definition 2. Let $v = (v_{0,0}, v_{0,1}, \dots, v_{0,2^d-1})$ be a 2^d -dimensional vector. We define the k th level Haar wavelet transform of v , $WT_k(v) = (v_{k,0}, v_{k,1}, \dots, v_{k,2^{d-k}-1})$, $1 \leq k \leq d$, as follows:

$$v_{k,i} = \begin{cases} (v_{k-1,2i} + v_{k-1,2i+1})/2 & \text{if } 0 \leq i \leq 2^{d-k} - 1 \\ v_{k,i} = (v_{k-1,2(i-2^{d-k})} - v_{k-1,2(i-2^{d-k})} + 1)/2 & \text{if } 2^{d-k} \leq i \leq 2^{d-k+1} - 1 \\ v_{k-1,i} & \text{if } 2^{d-k+1} \leq i \leq 2^d - 1 \end{cases}$$

Let $FWT_{k,2^p}(v)$ be the vector formed by the first 2^p coefficients of $WT_k(v)$. After the transformation, for each feature vector v , we can store $FWT_{k,2^p}(v)$ in an index file.

2.2. Manhattan and edit distances

In this section, we will first define the Manhattan distance. Next, given two sequences whose edit distance is not greater than a user-specified threshold ε , we can then transform each sequence into a feature vector according to Definitions 1 and 2, and find the upper bound of the Manhattan distance between both feature vectors.

Definition 3. Let $u = (u_0, u_1, \dots, u_{d-1})$ and $v = (v_0, v_1, \dots, v_{d-1})$ be two d -dimensional vectors. We define Manhattan distance between these two vectors as $MD(u, v) = \sum |u_i - v_i|$, positive distance as $PD(u, v) = \sum_{u_i > v_i} |u_i - v_i|$, and negative distance as $ND(u, v) = \sum_{u_i < v_i} |u_i - v_i|$, where $0 \leq i \leq d - 1$.

For example, let $u = \{1, 2, -1, 3, 4\}$ and $v = \{2, 0, -1, 0, 4\}$. Then, we have $PD(u, v) = |2 - 0| + |3 - 0| = 5$, $ND(u, v) = |1 - 2| = 1$, and $MD(u, v) = |1 - 2| + |2 - 0| + |-1 - (-1)| + |3 - 0| + |4 - 4| = 6$.

Lemma 1. $MD(u, v) = PD(u, v) + ND(u, v)$, where u and v are two d -dimensional vectors.

Proof. $MD(u, v) = \sum |u_i - v_i| = \sum_{u_i > v_i} |u_i - v_i| + \sum_{u_i = v_i} |u_i - v_i| + \sum_{u_i < v_i} |u_i - v_i| = PD(u, v) + 0 + ND(u, v)$. \square

Lemma 2. Let S and S' be two sequences. If $ED(S, S') = 1$, then $MD(F_q(S), F_q(S')) \leq 2q$, where q is the length of q -gram.

Proof. Since $ED(S, S') = 1$, there are three ways to convert S to S' . First, S is converted to S' by one insertion. Let $S = a_1 a_2 \dots a_{i-1} a_i a_{i+1} \dots a_n$ and $S' = a_1 a_2 \dots a_{i-1} a_i a_x a_{i+1} \dots a_n$. Since the character a_x is inserted between a_i and a_{i+1} in S' , the numbers of occurrences of the following $q - 1$ q -grams $a_{i-q+2} a_{i-q+3} \dots a_i a_{i+1}$, $a_{i-q+3} a_{i-q+4} \dots a_i a_{i+1} a_{i+2}$, \dots , $a_i a_{i+1} \dots a_{i+q-1}$ in $F_q(S')$ are smaller than the corresponding numbers in $F_q(S)$ by one. On the other hand, the numbers of occurrences of the following q q -grams $a_{i-q+2} a_{i-q+3} \dots a_i a_x$, $a_{i-q+3} a_{i-q+4} \dots a_i a_x a_{i+1}$, \dots , $a_x a_{i+1} \dots a_{i+q-1}$ in $F_q(S')$ are larger than the corresponding numbers in $F_q(S)$ by one. Therefore, $PD(F_q(S), F_q(S')) \leq q - 1$ and $ND(F_q(S), F_q(S')) \leq q$. Second, if S is converted to S' by one deletion, then $PD(F_q(S), F_q(S')) \leq q$ and $ND(F_q(S), F_q(S')) \leq q - 1$.

Third, if S is converted to S' by one replacement, then $PD(F_q(S), F_q(S')) \leq q$ and $ND(F_q(S), F_q(S')) \leq q$. However, these q -grams may not be distinct. If some of them are the same, the Manhattan distance between $F_q(S)$ and $F_q(S')$ will be smaller. Since $MD(F_q(S), F_q(S')) = PD(F_q(S), F_q(S')) + ND(F_q(S), F_q(S'))$, we can conclude that if $ED(S, S') = 1$, $MD(F_q(S), F_q(S')) \leq 2q$. \square

Corollary 1. Let S and S' be two sequences over alphabet Σ . If $ED(S, S') \leq \varepsilon$, then $MD(F_q(S), F_q(S')) \leq 2q \cdot \varepsilon$, where ε is the user-specified threshold.

Lemma 3. Given two feature vectors u and v , $MD(FWT_{d-p,2^p}(u), FWT_{d-p,2^p}(v)) \leq (2^p / |\Sigma|^q) MD(u, v)$, where p is a non-negative integer, q is a positive integer, $d = \text{ceiling}(\log_2 |\Sigma|^q)$, and $\text{ceiling}(x)$ returns the minimum integer not less than x .

Proof. Let u and v be two 2^d -dimensional vectors. According to Definition 2, for $k \geq 1$,

$$\begin{aligned} MD(FWT_{k,2^{d-k}}(u), FWT_{k,2^{d-k}}(v)) &= \sum_{i=0}^{2^{d-k}-1} |(u_{k-1,2i} + u_{k-1,2i+1})/2 - (v_{k-1,2i} + v_{k-1,2i+1})/2| \\ &= (1/2) \cdot \sum_{i=0}^{2^{d-k}-1} |(u_{k-1,2i} - v_{k-1,2i}) + (u_{k-1,2i+1} - v_{k-1,2i+1})| \\ &\leq (1/2) \cdot \sum_{i=0}^{2^{d-k+1}-1} |u_{k-1,i} - v_{k-1,i}| \\ &= (1/2) \cdot MD(FWT_{k-1,2^{d-k+1}}(u), FWT_{k-1,2^{d-k+1}}(v)) \quad (1) \end{aligned}$$

By Eq. (1), we have

$$\begin{aligned} MD(FWT_{d-p,2^p}(u), FWT_{d-p,2^p}(v)) &\leq MD(FWT_{d-p-1,2^{p+1}}(u), FWT_{d-p-1,2^{p+1}}(v))/2 \\ &\leq MD(FWT_{d-p-2,2^{p+2}}(u), FWT_{d-p-2,2^{p+2}}(v))/2^2 \\ &\vdots \\ &\leq MD(FWT_{0,2^d}(u), FWT_{0,2^d}(v))/2^{d-p} \\ &\leq MD(u, v)/2^{d-p} \\ &\leq MD(u, v) \cdot (2^p/2^d) \end{aligned}$$

Since $d = \text{ceiling}(\log_2 |\Sigma|^q)$, we have $|\Sigma|^q \leq 2^d$. Therefore, $MD(FWT_{d-p,2^p}(u), FWT_{d-p,2^p}(v)) \leq MD(u, v) \cdot (2^p/2^d) \leq MD(u, v) \cdot (2^p/|\Sigma|^q)$. \square

Theorem 1. Let S and S' be two sequences over alphabet Σ . If $ED(S, S') = \varepsilon$, then $MD(FWT_{d-p,2^p}(F_q(S)), FWT_{d-p,2^p}(F_q(S')))) \leq (2^p/|\Sigma|^q) \cdot 2q \cdot \varepsilon$, where p is a non-negative integer, q is a positive integer, and $d = \text{ceiling}(\log_2 |\Sigma|^q)$.

Proof. By Lemma 3, we have

$$\begin{aligned} MD(FWT_{d-p,2^p}(F_q(S)), FWT_{d-p,2^p}(F_q(S')))) &\leq (2^p/|\Sigma|^q) MD(F_q(S), F_q(S')) \\ &\leq (2^p/|\Sigma|^q) \cdot 2q \cdot \varepsilon \quad (\text{Corollary 1}) \quad \square \end{aligned}$$

Lemma 4. If $ED(Q, S) \leq \varepsilon$, then $ED(Q, S') \leq 2 \cdot \varepsilon$, where S' is extended (or shrunk) from S , $\text{len}(S') = \text{len}(Q)$, and $\text{len}(S')$ is the length of S . That is, we can add ε' alphabets to the both ends of S or delete ε' alphabets from the both ends of S , where $\varepsilon' = |\text{len}(S) - \text{len}(Q)| \leq \varepsilon$.

Proof. Since $ED(Q, S) \leq \varepsilon$, we need at most ε edit operations to align S with Q . Let S' be a sequence extended (or shrunk) from S such that $\text{len}(S') = \text{len}(Q)$. Thus, we need at most ε' more operations to align S' and Q . Therefore, $ED(Q, S') \leq \varepsilon + \varepsilon' \leq 2 \cdot \varepsilon$.

To match a query sequence Q and a data sequence S , we can divide Q into ρ disjointed windows of equal size, where $\rho > 1$. Let $p_1, p_2, \dots, p_{\rho-1}$ be the separating points. After aligning Q with S , we can divide S into ρ disjointed windows according to Q 's separating points.

Lemma 5. If $ED(Q, S) \leq \varepsilon$, there exists at least one pair of windows, U_i and V_i , such that $ED(U_i, V_i) \leq \varepsilon/\rho$, where $1 < \rho$, $1 \leq i \leq \rho$, U_i represents the i th window of Q , $Q[(i-1) \cdot w + 1..i \cdot w]$, which is one of the ρ disjointed windows divided according to Q 's separating points after aligning Q with S . Likewise, V_i is the i th window of S .

Proof. We can prove it by induction on ρ .

Basis step: For $\rho = 2$, we can divide Q into ρ equal windows, U_1 and U_2 . After aligning Q with S , S is divided into 2 disjointed windows accordingly. Assume that U_1 is in ε_1 -match with V_1 , and U_2 is in ε_2 -match with V_2 , where $\varepsilon_1 + \varepsilon_2 = \varepsilon$. Since $\varepsilon_1 + \varepsilon_2 = \varepsilon$, either ε_1 or ε_2 must be less than or equal to $\varepsilon/2$. Otherwise, $\varepsilon_1 + \varepsilon_2$ will be greater than ε . Therefore, we have that $ED(U_1, V_1) \leq \varepsilon/\rho$ or $ED(U_2, V_2) \leq \varepsilon/\rho$.

Induction hypothesis: Assume the lemma is true when Q and S are divided into ρ disjointed windows. That is, if $ED(Q, S) \leq \varepsilon$ and both of them are divided into ρ disjointed windows, then there exists at least one pair of windows, U_i and V_i , such that $ED(U_i, V_i) \leq \varepsilon/\rho$, where $1 \leq i \leq \rho$.

Induction step: Consider the case where both Q and S are divided into $\rho + 1$ disjointed windows. First, we can divide Q into 2 disjointed windows: $Q[1..w \cdot \rho]$ and $Q[w \cdot \rho + 1..w \cdot (\rho + 1)]$. After aligning Q and S , we can also divide S into two disjointed windows, S' and S'' , according to Q 's separating points. If $ED(Q[w \cdot \rho + 1..w \cdot (\rho + 1)], S'') \leq \varepsilon/(\rho + 1)$, the Lemma is proved. Otherwise, we can divided $Q[1..w \cdot \rho]$ into ρ disjointed windows. According to the induction hypothesis, there exists at least one pair of windows U_i and V_i , such that $ED(U_i, V_i) \leq ED(Q[1..w \cdot \rho], S')/\rho$, $1 \leq i \leq \rho$. Since $ED(Q[w \cdot \rho + 1..w \cdot (\rho + 1)], S'') > \varepsilon/(\rho + 1)$ in this case, $ED(Q[1..w \cdot \rho], S') < \varepsilon - \varepsilon/(\rho + 1) = \rho \cdot \varepsilon/(\rho + 1)$. Thus, $ED(U_i, V_i) \leq ED(Q[1..w \cdot \rho], S')/\rho < (\rho \cdot \varepsilon/(\rho + 1)) \cdot (1/\rho) = \varepsilon/(\rho + 1)$.

Therefore, we can prove that if $ED(Q, S) \leq \varepsilon$, there exists at least one pair of windows, U_i and V_i , such that $ED(U_i, V_i) \leq \varepsilon/\rho$, where $1 < \rho$, $1 \leq i \leq \rho$. \square

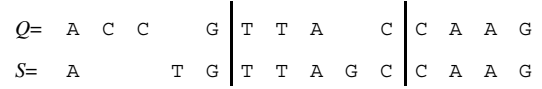


Fig. 1. Dividing Q and S into three disjointed windows.

For example, let us consider the example as shown in Fig. 1, where query sequence $Q = \text{ACCGTTACCAAG}$, data sequence $S = \text{ATGTTAGCCAAG}$, and $ED(Q, S) = 3$. We can divide Q into three disjointed windows of length 4: U_1, U_2, U_3 , and align Q with S . Then, S is divided into three disjointed windows: V_1, V_2, V_3 , according to Q 's separating points. We can find that $ED(U_2, V_2) = 1 \leq 3/3 = 1$ and $ED(U_3, V_3) = 0 \leq 3/3 = 1$. That is, there exist two pairs of windows (U_2, V_2) and (U_3, V_3), such that $ED(U_i, V_i) \leq \varepsilon/3$, where $\varepsilon = 3$, $i = 2, 3$.

3. Our proposed approach

Our proposed method, based on General Match (Moon et al., 2002), which generalized and improved on window constructing method of FRM (Faloutsos et al., 1994) and Dual Match (Moon et al., 2001), consists of two phases: processing data sequences and processing a query. Fig. 2 illustrates the flowchart of our proposed method, and the steps will be described in detail in the following sections. Note that the J -sliding and J -disjoint windows used in General Match to divide data and query sequences into several windows are different from the sliding and disjointed windows introduced in Section 2.

3.1. Processing data sequences

Here, we adopt the J -sliding window scheme (the sliding step is J) proposed in the approach of General Match (Moon et al., 2002). First, we partition a data sequences into fix-sized windows so the distance between the starting offsets of two adjacent J -sliding windows is J . Let $S_1, S_2, S_3, \dots, S_N$ be the data sequences in the database. For each data sequence S_i , $1 \leq i \leq N$, divide S_i into J -sliding windows $b_{i,j}$, where j is the starting offset in S_i , $j = 1 + (k-1) \cdot J$, $1 \leq k \leq \text{floor}((|S_i| - w)/J) + 1$ where $|S_i|$ denotes the length of S_i , w is the window size and $\text{floor}(x)$ returns the maximum integer not greater than x . That is, we divide a data sequence S_i into w -sized blocks $b_{i,j}$ starting from $S_i[(k-1) \cdot J + 1]$, $1 \leq k \leq \text{floor}((|S_i| - w)/J) + 1$. For example, let the sliding factor $J = 4$ and the window size $w = 8$. As shown in Fig. 3, the J -sliding windows starts at $S[1]$, $S[5]$, $S[9]$, etc.

For example, let $w = 8$ and $J = 4$, $S_1 = \text{AAAGCTCCTT-AGAGAGT}$ be a data sequence in the database. We can divide S_1 into J -sliding windows $b_{1,1}$, $b_{1,5}$, and $b_{1,9}$ as shown in Fig. 4.

Next, we convert $b_{i,j}$ into $F_q(b_{i,j})$ and apply the Haar wavelet transform to $F_q(b_{i,j})$ and calculate $WT_{d-p}(F_q(b_{i,j}))$. Then, we can extract the first 2^p coefficients of

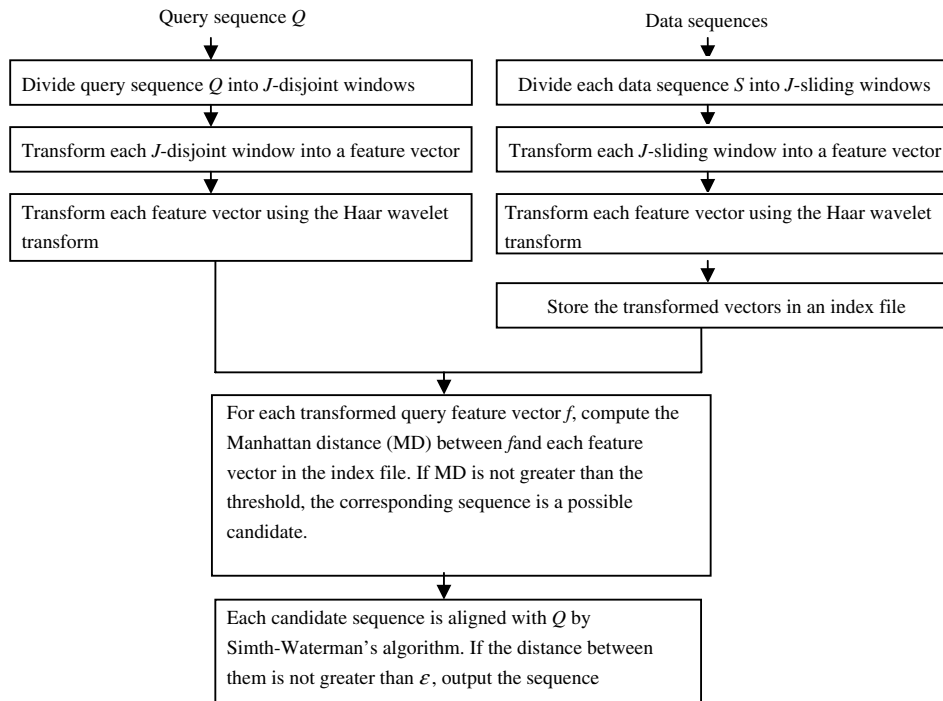


Fig. 2. The flowchart of our proposed method.

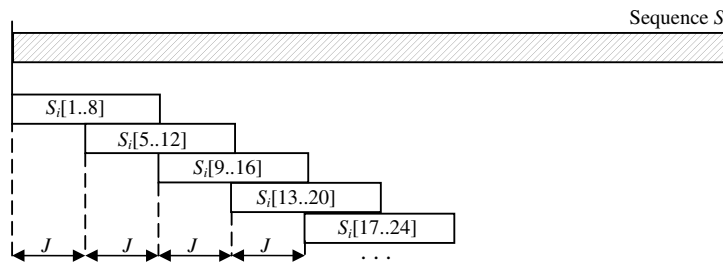


Fig. 3. An example of J -sliding windows, where $J = 4$ and $w = 8$.

$S_j =$ AAAACTCCTTAGAGAGT
 $b_{1,1} =$ AAAACTCC
 $b_{1,5} =$ CTCCTTAG
 $b_{1,9} =$ TTAGAGAG

Fig. 4. An example of a data sequence and its corresponding J -sliding windows.

$WT_{d-p}(F_q(b_{i,j}))$ and form a vector $FWT_{d-p,2^p}(F_q(b_{i,j}))$ to represent the original window $b_{i,j}$.

Let us consider the example as shown in Fig. 4. Let $q = 2$ and $p = 3$. We have $d = \text{ceiling}(\log_2|\Sigma|^q) = \text{ceiling}(\log_24^2) = 4$, $d - p = 4 - 3 = 1$, $F_2(b_{1,1}) = (3, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0)$, $F_2(b_{1,5}) = (0, 0, 1, 0, 0, 1, 0, 2, 0, 0, 0, 0, 1, 1, 0, 1)$, and $F_2(b_{1,9}) = (0, 0, 3, 0, 0, 0, 0, 0, 2, 0, 0, 0, 1, 0, 0, 1)$. After performing the Haar wavelet transform on these blocks, we have $WT_{d-p}(F_2(b_{1,1})) = WT_1(F_2(b_{1,1})) = (2, 0, 1/2, 1/2, 0, 0, 1/2, 0, 1, 0, -1/2, -1/2, 0, 0, -1/2, 0)$, $WT_1(F_2(b_{1,5})) = (0, 1/2, 1/2, 1, 0, 0, 1, 1/2, 0, 1/2, -1/2, -1, 0, 0, 0, -1/2)$, and $WT_1(F_2(b_{1,9})) = (0, 3/2, 0, 0, 0, 1, 0, 1/2, 1/2, 0, 3/2, 0, 0, 1, 0, 1/2,$

$-1/2)$. Then, for each vector, we can extract the first $2^p = 2^3$ numbers to be the feature vector: $FWT_{d-p,2^p}(F_2(b_{1,1})) = FWT_{1,2^3}(F_2(b_{1,1})) = (2, 0, 1/2, 1/2, 0, 0, 1/2, 0)$, $FWT_{1,2^3}(F_2(b_{1,5})) = (0, 1/2, 1/2, 1, 0, 0, 1, 1/2)$, $FWT_{1,2^3}(F_2(b_{1,9})) = (0, 3/2, 0, 0, 0, 1, 0, 1/2, 1/2)$. Having transformed all windows into vectors, we store these vectors into an index file.

3.2. Processing a query

Given the query sequence Q , we first divide it into J -disjoint windows Q_1, Q_2, \dots, Q_M . The J -disjoint windows of sequence Q is defined as the subsequences of length w starting from $Q[i + (j - 1) \cdot w]$, where $1 \leq i \leq J$, and $1 \leq j \leq \text{floor}((|Q| - i + 1)/w)$. An example of J -disjoint windows of sequence Q is shown in Fig. 5, where $J = 4$ and $w = 8$. Similarly, we convert each Q_m to $F_q(Q_m)$. After the conversion, we can apply the Haar wavelet transform to each $F_q(Q_m)$ and calculate the corresponding $WT_{d-p}(F_q(Q_m))$. We can use the first 2^p coefficients to form

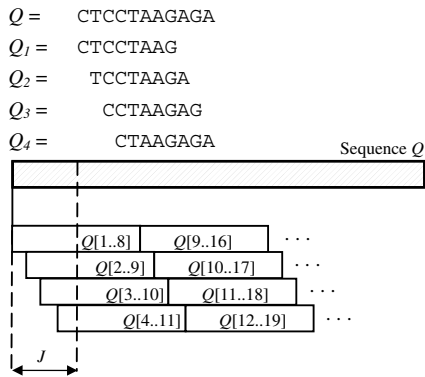


Fig. 5. An example of J -disjoint windows where $J = 4$ and $w = 8$.

a query feature vector $FWT_{d-p,2^p}(F_q(Q_m))$. Then, for each $FWT_{d-p,2^p}(F_q(Q_m))$, we search the index file and extract the candidate blocks $b_{i,j}$ so that $MD(FWT_{d-p,2^p}(F_q(Q_m)), FWT_{d-p,2^p}(F_q(b_{i,j}))) \leq (2\varepsilon/\rho) \cdot (2^p/|\Sigma|^q) \cdot 2q$. Finally, for each candidate block, we will check if the corresponding subsequence of the original data sequence is in ε -match with Q .

$Q =$ CTCCTAAGAGA
 $Q_1 =$ CTCCTAAG
 $Q_2 =$ TCCTAAGA
 $Q_3 =$ CCTAAGAG
 $Q_4 =$ CTAAGAGA

Let us consider an example as shown in Fig. 5, where $Q =$ CTCCTAAGAGA, $w = 8$ and $\varepsilon = 1$. We first divide Q into J -disjoint windows ($J = 4$). Similarly, let $q = 2$ and $p = 3$. We can apply the Haar wavelet transform to these windows, and extract the corresponding query feature vectors as follows: $FWT_{d-p,2^p}(F_2(Q_1)) = FWT_{1,2^3}(F_2(Q_1)) = (1/2, 1/2, 1/2, 1, 0, 0, 1, 0)$, $FWT_{1,2^3}(F_2(Q_2)) = (1/2, 1/2, 1/2, 1/2, 0, 1, 0)$, $FWT_{1,2^3}(F_2(Q_3)) = (1/2, 1, 1/2, 1/2, 1/2, 0, 1/2, 0)$, $FWT_{1,2^3}(F_2(Q_4)) = (1/2, 1, 0, 1/2, 1, 0, 1/2, 0)$.

For each query feature vector $FWT_{d-p,2^p}(F_q(Q_m))$, we can calculate the Manhattan distance $MD(FWT_{d-p,2^p}(F_q(Q_m)), FWT_{d-p,2^p}(F_q(b_{i,j})))$ and screen out all blocks $b_{i,j}$ so that $MD(FWT_{d-p,2^p}(F_q(Q_m)), FWT_{d-p,2^p}(F_q(b_{i,j}))) > (2\varepsilon/\rho) \cdot (2^p/|\Sigma|^q) \cdot 2q$. Since $\varepsilon = 1$, $q = 2$, $p = 3$ and $\rho = \text{floor}(|Q|/w) = \text{floor}(11/8) = 1$, we have $(2\varepsilon/\rho) \cdot (2^p/|\Sigma|^q) \cdot 2q = (2/1) \cdot (2^3/4^2) \cdot 2 \cdot 2 = 4$. For $FWT_{1,2^3}(F_2(Q_1))$, we can calculate the following distances and find that $b_{1,1}$ and $b_{1,5}$ are candidate blocks.

$$\begin{aligned}
 MD(FWT_{1,2^3}(F_2(Q_1)), FWT_{1,2^3}(F_2(b_{1,1}))) &= 3 < 4 \\
 MD(FWT_{1,2^3}(F_2(Q_1)), FWT_{1,2^3}(F_2(b_{1,5}))) &= 1 < 4 \\
 MD(FWT_{1,2^3}(F_2(Q_1)), FWT_{1,2^3}(F_2(b_{1,9}))) &= 5 > 4
 \end{aligned}$$

Next, we can perform the above procedure for each query feature vector: $FWT_{1,2^3}(F_2(Q_2))$, $FWT_{1,2^3}(F_2(Q_3))$, and $FWT_{1,2^3}(F_2(Q_4))$. Finally, for each candidate block,

we will check if the corresponding subsequence of the original data sequence is in ε -match with Q using the sequence alignment algorithm (Smith and Waterman, 1981).

According to the algorithm of General Match, every J -sliding window of a data sequence is compared with every J -disjoint window of a query sequence. If the subsequence $S[i..j]$ of a data sequence S is in ε -match with a query sequence Q , we know by Lemma 5 that there exists at least a J -sliding window of $S[i..j]$ that is in ε/ρ -match with a subsequence of Q . By Lemma 4, we can extend (or shrink) the subsequence to a J -disjoint window so that the J -disjoint window is in $2\varepsilon/\rho$ -match with the J -sliding window.

Lemma 6. Let a data sequence S be divided into J -sliding windows of size w , and the query sequence Q into J -disjoint windows of the same size. If the subsequence $S[i..j]$ is in ε -match with Q , then at least one J -sliding window W' of $S[i..j]$ is in $2\varepsilon/\rho$ -match with a J -disjoint window W'' of Q , where $\rho = \text{floor}(|Q|/w)$.

Proof. Let $S[i..j]$ be divided into a set of J -sliding windows and Q into a set of J -disjoint windows. If the subsequence $S[i..j]$ is in ε -match with Q , by Lemma 5, we can find at least a J -sliding window W' of $S[i..j]$ that is in ε/ρ -match with a subsequence $Q[y..z]$, where $1 \leq y \leq z \leq |Q|$, and $\rho = \text{floor}(|Q|/w)$. By Lemma 4, if we extend (or shrink) $Q[y..z]$ to a J -disjoint window W'' , W'' will be in $2\varepsilon/\rho$ -match with the J -sliding window W' . \square

Theorem 2. Given a data sequence S , a query sequence Q , if $ED(Q, S[i..j]) \leq \varepsilon$, then there exists at least one pair of windows W' and W'' , so that $MD(FWT_{d-p,2^p}(F_q(W'))), FWT_{d-p,2^p}(F_q(W'')) \leq (2\varepsilon/\rho) \cdot (2^p/|\Sigma|^q) \cdot 2q$, where W' is a J -sliding window of $S[i..j]$, W'' is a J -disjoint window of Q , $\rho = \text{floor}(|Q|/w)$, w is the size of the J -sliding window, p is a non-negative integer, q is a positive integer, and $d = \text{ceiling}(\log_2 |\Sigma|^q)$.

Proof. By Lemma 6, we have $ED(W', W'') \leq 2\varepsilon/\rho$.

By Theorem 1, we have

$$\begin{aligned}
 MD(FWT_{d-p,2^p}(F_q(W')), FWT_{d-p,2^p}(F_q(W''))) & \\
 &\leq (2^p/|\Sigma|^q) \cdot 2q \cdot ED(W', W'') \\
 &\leq (2^p/|\Sigma|^q) \cdot 2q \cdot (2\varepsilon/\rho) \quad \square
 \end{aligned} \tag{2}$$

Theorem 3. Our proposed method guarantees no false negatives.

Proof. We can prove it by contradiction. Assume that there is a false negative in our proposed method, that is, there exists a data sequence S whose subsequence $S[i..j]$ is in ε -match with a query sequence Q , but for every J -sliding window X' of $S[i..j]$ and every J -disjoint window X'' of Q , $MD(FWT_{d-p,2^p}(F_q(X')), FWT_{d-p,2^p}(F_q(X''))) > (2\varepsilon/\rho) \cdot (2^p/|\Sigma|^q) \cdot 2q$. However, by Theorem 2, we know that if $ED(Q, S[i..j]) \leq \varepsilon$, then there exists at least one pair of windows W' and W'' , so that $MD(FWT_{d-p,2^p}(F_q(W'))),$

$FWT_{d-p,2^p}(F_q(W'')) \leq (2\varepsilon/\rho) \cdot (2^p/|\Sigma|^q) \cdot 2q$, where W' is a J -sliding window of $S[i..j]$, W'' is a J -disjoint window of Q , $\rho = \text{floor}(|Q|/w)$, w is the size of the J -sliding window, p is a non-negative integer, q is a positive integer, and $d = \text{ceiling}(\log_2|\Sigma|^q)$. This contradicts the assumption. Therefore, our proposed method guarantees no false negatives. \square

4. Performance analysis

In order to examine real DNA sequences that usually have multiple possible shapes and combinations, we conducted our experiments using six different real data sets all extracted from the NCBI database, version 6.2. The first data set, DS-DATA, consists of 20M base pairs (bps) of homo sapiens chromosome 19 genomic contig where the sequence is divided into subsequences of equal size. The second data set, GLO-DATA, consists of 120 beta globin gene sequences, known as homologous sequences, where the length of each sequence is different and the sequences are similar to each other. The third data set, RECOM-DATA, consists of 2.64M bps of chromosome 1 of *Deinococcus radiodurans*. RECOM-DATA contains similar sequences with reshuffled fragments. *Deinococcus radiodurans* is an extremely radiation-resistant soil bacterium that uses its multiple chromosomal copies to carry out homologous recombination system. The fourth data set, LOW-DATA, consists of 365 sequences annotated as low complexity regions from Y_chromosome. LOW-DATA is used for testing low complexity sequences (not informative). The fifth data set, REP-DATA, consists of 9878 human's alu sequences that are known to contain repetitive DNA sequences, and are used for testing repeats (similar fragments interspersed across sequences). Finally, the sixth data set, PRO-DATA, consists of 13,022 sequences including cysteine-rich, praline-rich, highly hydrophobic, and

highly charged sequences that have biased composition and appearance though they may not be similar.

We use all six data sets and compare them with the QUASAR (Burkhardt et al., 1999) and YM methods (Yamada and Morishita, 2005). Here, we modify the YM to use the edit distance as a measure to retrieve similar sequences. All methods were implemented in *Java 2 SDK* and run on an IBM Compatible PC with a Pentium IV 3.2 GHz, 1GB RAM and Windows XP. After conducting experiments with different combinations of p and q , we found that $q = 4$ and $p = 6$ are good for most of the cases listed above.

To evaluate the performance of our proposed method, we compare it with the QUASAR and YM using filtration ratio, precision, and execution time. We define filtration ratio of datasets as the number of candidate subsequences retrieved divided by that of all subsequences in the database. Precision is defined as the fraction of subsequences retrieved that are in ε -match with the given query sequence.

Since the YM does not use windows, we will only test the window size effect between TDF and QUASAR. Figs. 6 and 7 illustrate the filtration ratio and precision versus window size between TDF and QUASAR where the query length and tolerance threshold are set to 1024 and 10, respectively. The number of windows decreases when the window size increases. However, the filtration ratio of the TDF is affected not much by the window size as shown in Fig. 6. On the other hand, the number of candidates retrieved grows when the window size increases in the QUASAR, especially for LOW-DATA and REP-DATA. Since both methods guarantee that all data sequences satisfying the query will be retrieved, this means when calculating the precision with the increasing window size, the numerator (the number of data sequences satisfying the query) remains the same while the denominator (the number of retrieved candidates) gets larger. Thus, the precision drops for both methods when the window size increases as shown in Fig. 7. The TDF can filter out more data

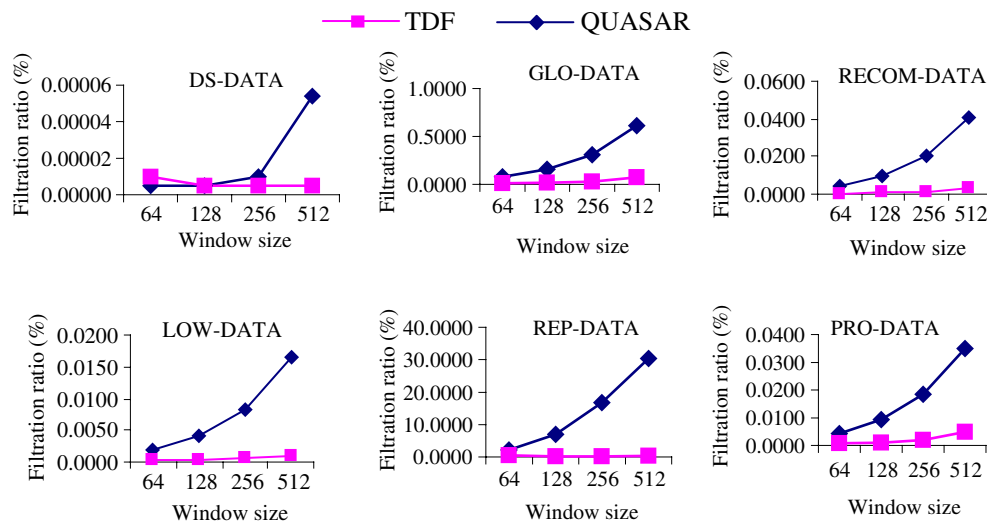


Fig. 6. Filtration ratio versus window size.

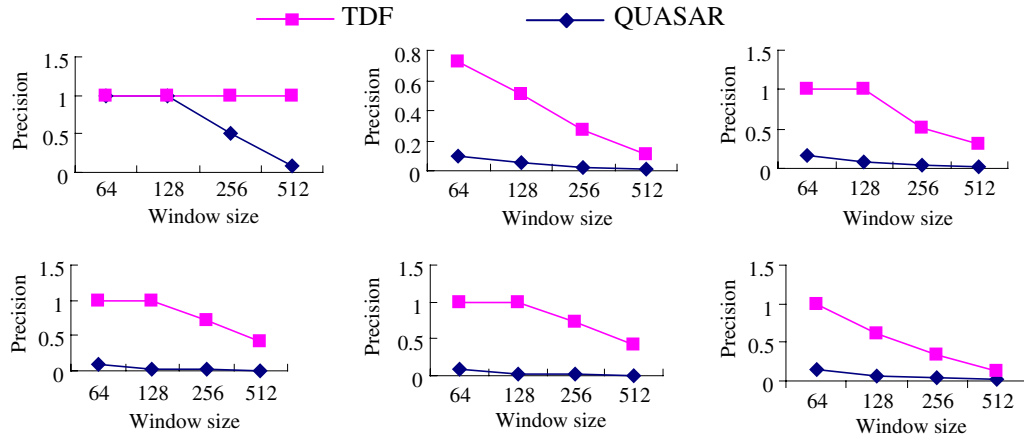


Fig. 7. Precision versus window size.

sequences and has higher precision than the QUASAR for most cases.

After comparing the window size effect between TDF and QUASAR, we illustrate the filtration ratio, precision, and execution time versus tolerance and query length for the TDF, QUASAR, and YM, where the window size for the TDF and QUASAR is set to 256. Figs. 8–10 illustrate the filtration ratio, precision and execution time versus tolerance for the TDF, QUASAR and YM, where the query length is set to 1024. As the tolerance is getting larger, the number of candidates retrieved grows at a faster rate than that of similar subsequences for the TDF and QUASAR. Consequently, the filtration ratio increases and the precision decreases for the both methods as shown in Figs. 8 and 9.

For the YM, all the distances between query subsequences and seed patterns (Yamada and Morishita, 2005) are either 0 or 1. The same set of candidates is retrieved for all the cases of tolerance thresholds. Since the number of data sequences satisfying the query increases when the

tolerance threshold increases, the precision of the YM increases when the tolerance increases as shown in Fig. 9. However, in the cases of LOW-DATA and REP-DATA, the YM may retrieve a lot of candidates which can not satisfy the query. For these two data sets the precision of the TDF is better than that of the other approaches as shown in Fig. 9.

The execution time increases as tolerance increases for all three approaches and the TDF has the best performance for most cases as shown in Fig. 10. For the LOW-DATA and REP-DATA data sets, the TDF runs 15–200 times faster than the QUASAR and YM. Since the QUASAR and YM use exactly-matched patterns to filter out data sequences, the number of data sequences left is much greater than that of the other data sets.

Figs. 11–13 illustrate the filtration ratio, precision and execution time versus query length for the TDF, QUASAR and YM, where the tolerance threshold is set to 10. For the TDF, since the window size is set to 256, there will be only one disjoint window when the query length is

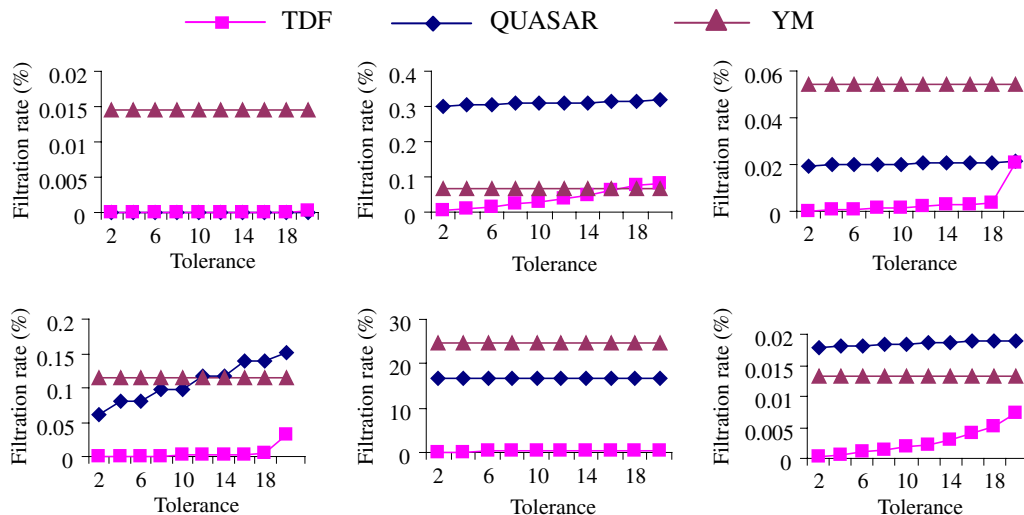


Fig. 8. Filtration ratio versus tolerance.

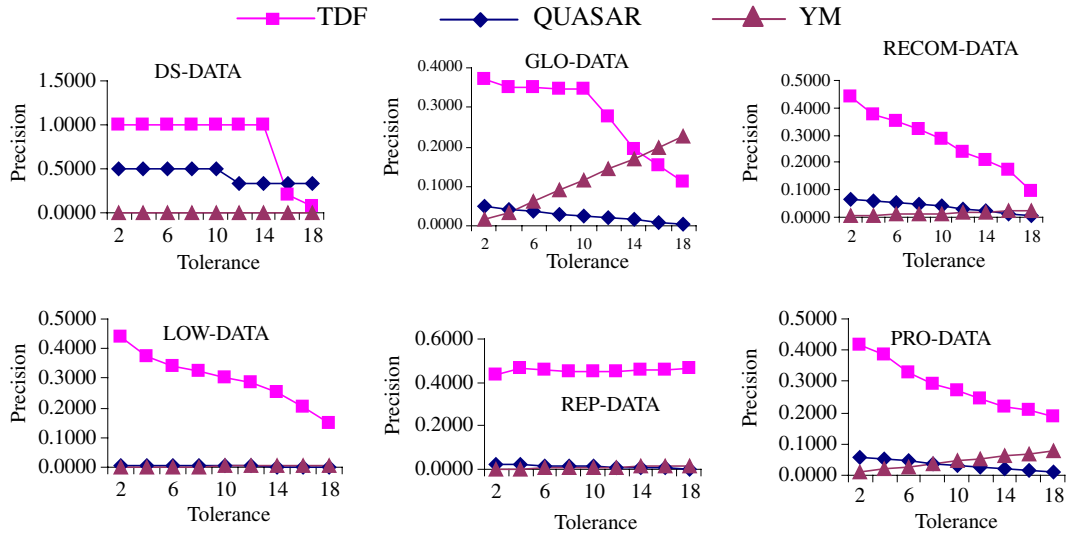


Fig. 9. Precision versus tolerance.

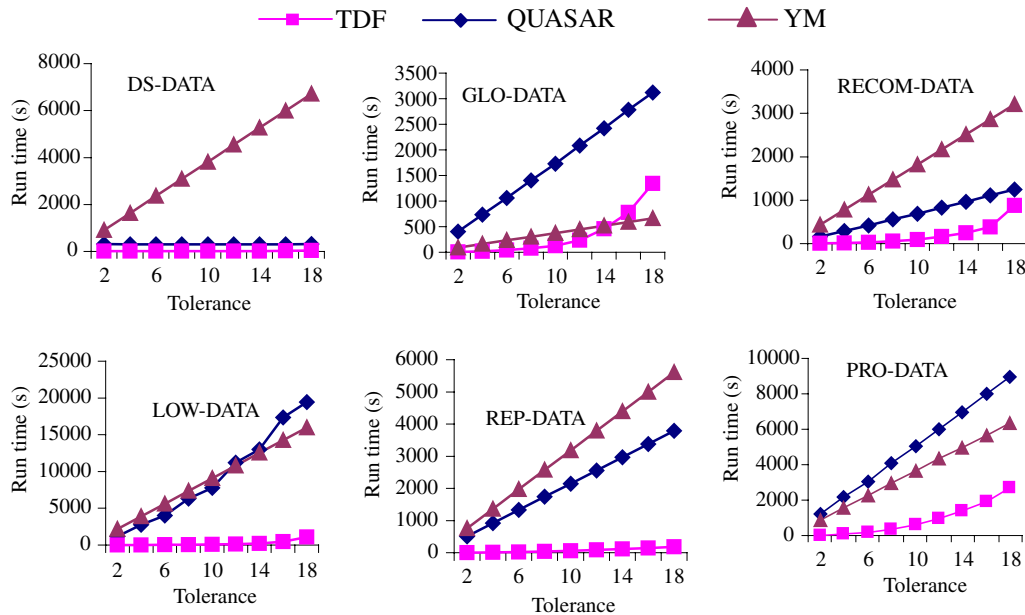


Fig. 10. Execution time versus tolerance.

256. The threshold of the Manhattan distance is large when there is only one window and gets much smaller when there are more windows (as shown in Eq. (2)). As shown in Fig. 11, the filtration ratio is high when there is only one window (i.e. the window size is equal to 256) but much lower when the number of window is greater than one. When the query length is set to the size of the J -sliding window ($=256$), the filtration ratio of the TDF increases sharply in the cases of DS-DATA, RECOM-DATA, LOW-DATA, and PRO-DATA. This means that for the TDF, the query length should be at least twice the size of a J -sliding window to have a better filtration ratio.

The QUASAR uses a q -mer to slide over the window and count the number of exact matches, and hence it is less

affected by the number of windows. The TDF has a better filtration ratio than the QUASAR when the query length is not less than 512 since it can screen out much more data sequences in all data sets. Generally speaking, the QUASAR retrieves about 10–80 times more candidates than the TDF. With a better filtration ratio and no false negative candidates, the TDF has better precision and performance than the QUASAR in most cases as shown in Figs. 12 and 13.

The YM is specially designed to process short queries, its filtration ratio increases as the length of a query increases. When the query length is not less than 512, the YM has a better filtration ratio than the TDF. However, when the query length is greater than 512, the TDF is better than the YM as shown in Fig. 11. In the case of

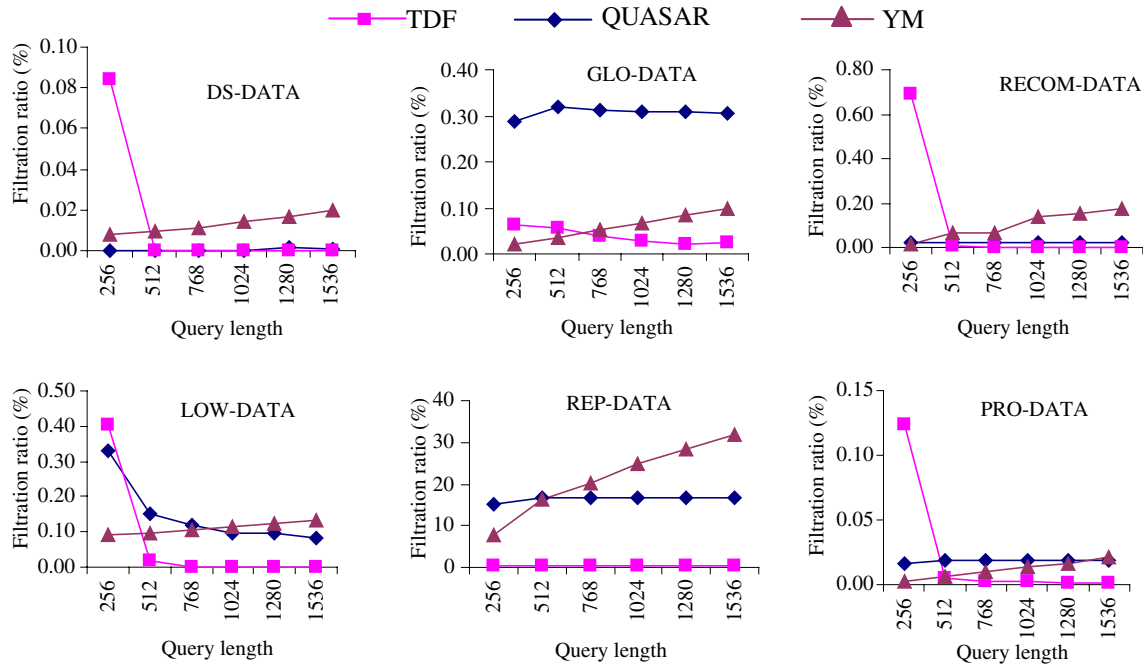


Fig. 11. Filtration ratio versus query length.

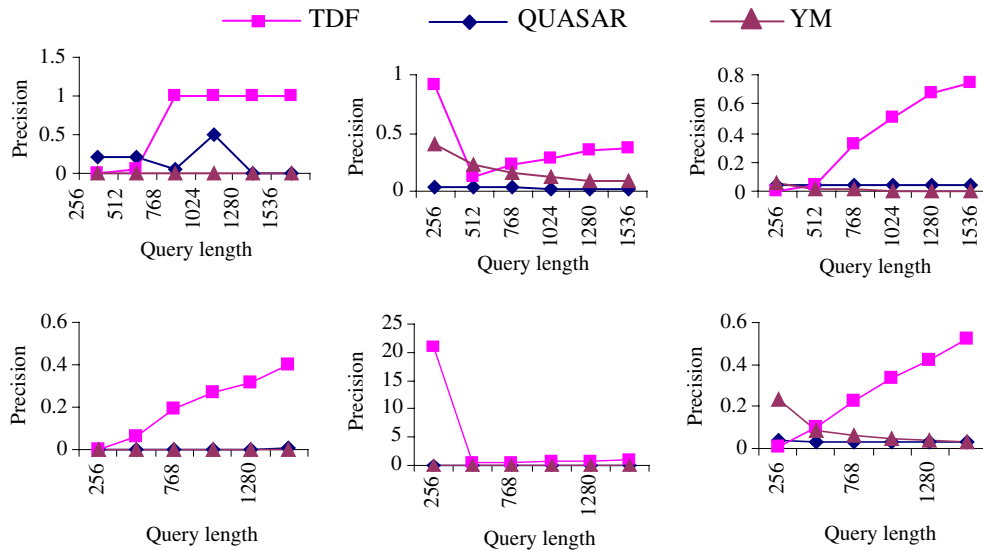


Fig. 12. Precision versus query length.

REP-DATA, since a lot of repetitive subsequences are hashed to the same bucket, the filtration ratio of the YM decreases quickly.

Table 1 lists the index sizes used in the three approaches. The index size of the QUASAR is about 1–3 times more than that of the TDF while the index size of the YM is about 50–100% more.

In summary, the TDF outperforms the QUASAR in terms of execution time, filtration ratio, precision, and index size. It also outperforms the YM in case of long query, low complexity, and repetitive data.

5. Concluding remarks and future work

In this paper, we propose a filtration method called Transformation-based Database Filtration method (TDF) that can efficiently screen out those data sequences of a DNA sequence database which cannot satisfy a given query sequence. Our proposed method consists of two phases. First, we divide a data sequence into several windows (blocks), each of which is transformed into a data feature vector using the Haar wavelet transform. The transformed data feature vectors are then stored in an index file.

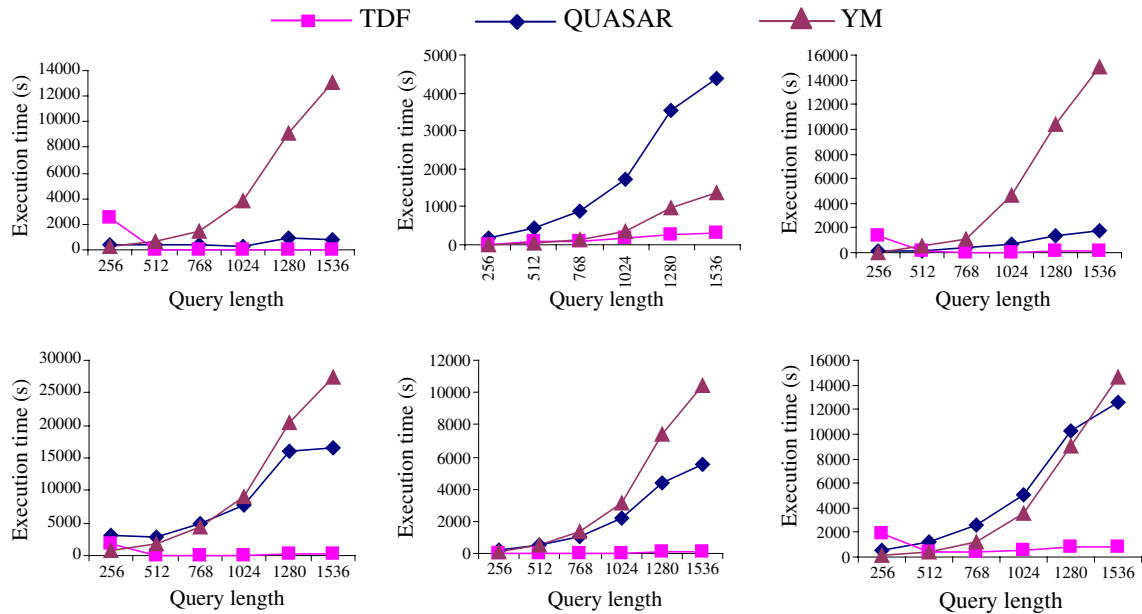


Fig. 13. Execution time versus query length.

Table 1
Index size

Size (KB)	DS-DATA	GLO-DATA	RECOM-DATA	LOW-DATA	REP-DATA	PRO-DATA
QUASAR	169,998	2801	19,607	47,864	44	173,510
YM	78,400	1620	10,000	23,900	30	80,000
TDF	37,091	912	5678	13,432	20	44,970

Second, we divide a query sequence into several sliding windows, each of which, again, transformed into a query feature vector using the Haar wavelet transform. We then search the index file to find the data feature vectors whose Manhattan distances to the query feature vector are less than a predefined threshold. The blocks corresponding to the feature vectors thus found form the set of candidate blocks. For each candidate block, we check if the corresponding subsequence of the original data sequence is matched with the query sequence using the sequence alignment algorithm. Our proposed method guarantees no false negatives. The experimental results show that our proposed method is superior to the QUASAR in terms of filtration ratio, precision, execution time and index size. Our proposed method also outperforms the YM for long query, low complexity and repetitive data.

Issues for further study include how to design or apply an index structure to speed up the query processing and how to group the query feature vectors into minimum bounding rectangles in order to facilitate range queries against the index structure.

Acknowledgements

The authors are grateful to the anonymous referees for their helpful comments and suggestions. They would also

like to thank Chia-Ming Hsu for constructing the databases and doing parts of the experiments.

References

- Aghili, A., Agrawal, D., Abbadi, A.E., 2003. Filtration of string proximity search via transformation. In: Proc. Third IEEE Internat. Symposium on Bioinformatics and Bioengineering, pp. 149–157.
- Altschul, S.F., Gish, W., Miller, W., Myers, E., Lipman, D.J., 1990. Basic local alignment search tool. *J. Mol. Biol.* 215, 403–410.
- Altschul, S.F., Madden, T.L., Schaffer, A.A., Zhang, J., Zhang, Z., Miller, W., Lipman, D.J., 1997. Gapped BLAST and PSI-BLAST: A new generation of protein database search programs. *Nucl. Acids Res.* 25, 3389–3402.
- Burkhardt, S., Crauser, A., Ferragina, P., Lenhof, H.P., Rivals, E., Vingron, M., 1999. Q-gram based database searching using a suffix array (quasar). In: Proc. Third Internat. Conf. on Computational Molecular Biology, pp. 77–83.
- Faloutsos, C., Ranganathan, M., Manolopoulos, Y., 1994. Fast subsequence matching in time-series databases. In: Proc. ACM SIGMOD Internat. Conf. on Management of Data, pp. 419–429.
- Giladi, E., Walker, M.G., Wang, J.Z., Volkmut, W., 2002. SST: An algorithm for finding near-exact sequence matches in time proportional to the logarithm of the database size. *Bioinformatics* 18, 873–879.
- Krishnan, A., Li, K.B., Issac, P., 2004. Rapid detection of conserved regions in protein sequences using wavelets. *Silico Biol.* 4, 133–148.
- Lipman, D.J., Pearson, W.R., 1985. Rapid and sensitive protein similarity searches. *Science* 227, 1435–1441.

- Ma, B., Tromp, J., Li, M., 2002. Pattern Hunter: Faster and more sensitive homology search. *Bioinformatics* 18, 440–445.
- Meek, C., Patel, J., Kasetty, S., 2003. OASIS: An online and accurate technique for local-alignment searches on biological sequences. In: *Proc. 29th VLDB Conf.*
- Moon, Y.S., Whang, K.Y., Loh, W.K., 2001. Duality-based subsequence matching in time-series databases. In: *Proc. 17th IEEE Internat. Conf. on Data Engineering*, pp. 263–272.
- Moon, Y.S., Whang, K.Y., Han, W.S., 2002. General match: A subsequence matching method in time-series databases based on generalized windows. In: *Proc. CM SIGMOD Internat. Conf. on Management of Data*, pp. 382–393.
- National Center for Biotechnology Information (NCBI). Available from: <http://www.ncbi.nlm.nih.gov/>, 2004.
- Pearson, W.R., 1994. Using the fasta program to search protein and DNA sequence databases. *Methods Mol. Biol.* 25, 365–389.
- Smith, T.F., Waterman, M.S., 1981. Identification of common molecular subsequences. *J. Mol. Biol.* 147, 195–197.
- Trad, C.H.D., Fang, Q., Cosic, I., 2001. An overview of protein sequence comparisons using wavelets. In: *Proc. IEEE-EMBS*, pp. 115–119.
- Yamada, T., Morishita, S., 2005. Accelerated off-target search algorithm for siRNA. *Bioinformatics* 21, 1316–1324.
- Zhang, Z., Schwartz, S., Wagner, L., Miller, W., 2000. A greedy algorithm for aligning DNA sequences. *J. Comput. Biol.* 7, 203–214.