http://ansinet.com/itj

ITJ

ISSN 1812-5638

INFORMATION TECHNOLOGY JOURNAL

ANSInet
Asian Network for Scientific Information
308 Lasani Town, Sargodha Road, Faisalabad - Pakistan

Information Technology Journal 10 (8): 1579-1586, 2011
ISSN 1812-5638 / DOI: 10.3923/itj.2011.1579.1586

An Algorithm Based on Diagonal Feature for Local Alignment in Large Database

[1]Decai Sun, [2,1]Xingming Sun, [3]Xiaoxia Wang and [1]Zhiqiang Ruan
[1]College of Information Science and Engineering, Hunan University, Hunan Changsha, 410082, China
[2]Jiangsu Engineering Center of Network Monitoring,
Nanjing University of Information Science and Technology, Jiangsu Nanjing, China
[3]Department of Computer Science and Technology, Hunan Institute of Technology,
Hunan, Hengyang, China

Abstract: Sequence alignment was one of the most popular operations in bioinformatics. The key issue of alignment was how to improve matching speed in a large sequences database. In this study, a full-sensitivity algorithm was proposed to solve the problem of finding all local alignments over a given length w with an error rate at most e. The proposed algorithm was implemented on a q-gram index. First, a large part of irrelevant subsequences were eliminated quickly by effective filtrating with new diagonal features. These new diagonal features were extracted from match-regions by analyzing the edit matrix of query sequence and database. Second, the unfiltered regions were verified by smith-waterman algorithm to search the true matches. The experimental results demonstrate that the proposed algorithm improves the filtration efficiency in a short filtration time and the algorithm is always faster than the well-known SWIFT on condition of low max error rate. This result is of great practical to local alignment with low error rate and short window size.

Key words: Sequence alignment, local alignment, q-gram index, q-gram filter, filter algorithm

INTRODUCTION

Searching a biological database for similarity to a given query sequence is a fundamental problem in bioinformatics. Biologists can infer the structural, functional or evolutionary relationships among them by comparing the sequences of genes and proteins (Karimi Kordestani and Omid, 2008; Haliloglu and Bostan, 2002; Khurshid and Saleem, 2000). Sequence alignment is one of the most popular operations because it can identify local alignments accurately and provide an explicit mapping between sequences. With the increment of biological sequences, the problem of sequence alignment in a biological database remains relevant today.

The first algorithms for aligning two sequences are those based on dynamic programming and the most notable algorithm is Smith-Waterman (Smith and Waterman, 1981). These algorithms have quadratic time complexity and they are full-sensitivity algorithms which guarantee that every true match in database can be found without omitting any one. To improve searching speed in large databases, heuristics algorithms are developed though they are not full-sensitivity, such as FASTA (Pearson and Lipman, 1988) and BLAST (Altschul *et al*., 1997). BLAST performs searches at an amazing speed but it can not fulfil what today's applications' need.

In contrast, filter is a two-phase alignment algorithm. In the first phase, a large part of irrelevant subsequences that do not contain any true matches are eliminated rapidly. In the second phase, the unfiltered subsequences called candidates is verified by another algorithm. A filter algorithm is full-sensitivity if its verification algorithm is full-sensitivity. QUASAR (Burkhardt *et al*., 1999) is a refined filter based on the study of Jokinen and Ukkonen (1991) which uses an index combined with suffix array and look up table. In searching process, the database is split into blocks logically and a sliding window proceeds over the query sequence. Then the number of q-grams co-occurring in the window and each block is determined and blocks whose number is less than a certain threshold are eliminated. At last, the BLAST is run on the unfiltered blocks to search the true matches. QUASAR's filtration time is shorter but the verification time is much longer for its low filtration efficiency. On the other hand Rasmussen *et al*. (2006) gets inspiration from the idea of diagonal sorting of FASTA and a local alignment algorithm based on q-gram index called SWIFT is proposed. In searching process, the database is divided into bins logically by diagonals and every W×E parallelogram whose q-grams hits isn't less than a certain threshold is reported as a candidate. At last, the candidates are verified by using BLAST or a bit-vector

Corresponding Author: Xingming Sun, Jiangsu Engineering Center of Network Monitoring,
Nanjing University of Information Science and Technology, Jiangsu Nanjing, 210044, China
Tel: 86-025-58731575

dynamic programming algorithm proposed by Myers (1999). SWIFT achieves higher filtration efficiency but its filtration time is longer. Now, the gapped-seed idea is orthogonal to the filtration method (Burkhardt and Karkkainen, 2003). It also can be applied to the proposed algorithm but it is not employed here.

To accelerate alignment in a database, an index will be constructed to preprocess database at first. Three different data structures (Navarro *et al*., 2001) are often used in the literature. Suffix array (Karkkainen *et al*., 2006) is a weak version of suffix tree. It requires much less space but poses a small penalty over the search time. Inverted index is a look up table which consists of substrings extracted from database, for example, q-gram index. SSAHA (Ning *et al*., 2001) uses a q-gram index to exact search in DNA database and only non-overlapping q-grams are considered. This reduces the index size but at the cost of a loss in sensitivity. The other index structures are illustrated by Sahin *et al*. (2007) and Fu *et al*. (2009).

The aim of this study is to accelerate local alignment and the problem to be solved is to search all e-matches and window length w in database. The problem is defined as follows.

If a subsequence *s* in database D is locally similar to query sequence Q with windows length w, there exists at least one pair subsequences (Q[i,..., i + w-1],s') with the following properties:

- Q[i,..., i + w-1] is a subsequence length of w in Q and s' is a subsequence in s
- Q[i,..., i + w-1] and s' have error rate at most e

A new filter (Local Alignment Algorithm based on DiagonalFeature, LAADF) is proposed in this study, it can efficiently screen out those subsequences where do not contain any true matches. The proposed filter is a full-sensitivity algorithm and it consists of two phases. First, each diagonal in the virtual edit matrix of query sequence and database is assigned with a counter and a sliding window proceeds over the query sequence. The q-gram hits on each diagonal is calculated and stored into corresponding counter and then the diagonal whose counter is not less than certain threshold is reported as a candidate. Second, Smith-Waterman algorithm is run on all candidates to search true matches accurately.

PRELIMINARIES

A q-gram is a subsequence with length of q which extracted from biological sequences and its position is the offset of its first character in database. In consecutive extraction, the distance between two adjacent q-grams is 1, such as Navarro and Baeza-yates (1998), an example is shown in Fig. 1a. In non-consecutive extraction, the distance is larger than 1, such as Sutinen and Tarhio (1996), an example is shown in Fig. 1b.

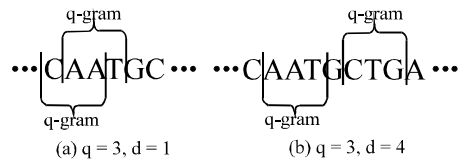

Fig. 1 (a-b): Examples of consecutive and nonconsecutive extraction

Consecutive extraction is employed here and the basic q-gram lemma based on consecutive extraction is proposed by Jokinen and Ukkonen. The basic q-gram lemma guarantees that at least t (k) = |S| + 1-(k + 1)q q-grams of S must occur in every match-region, a match-region is a subsequence where actually includes true matches. This lemma gives a basic feature of match-regions and it is used in both QUASAR and SWIFT.

Error rate is a scale to display the similarity between query sequence and its true matches and it is defined as k/|S|. The max error rate of filter is a threshold which guarantees that all subsequences whose error rates are larger than it are eliminated. A filter will work well when its max error rate in [0,1/q-1/|S|], otherwise, it will collapse (Sutinen and Szpankowski, 1998).

Filtration efficiency is a key factor which can display the ratio of irrelevant subsequences eliminated by filter (Sutinen and Tarhio, 1996).

Q-GRAM INDEX

Structure of q-gram index: Q-gram index is employed here because its searching speed is faster than that of suffix array under the circumstance of same memory space (Puglisi *et al*., 2006). Q-gram index is an inverted index whose terms are q-grams extracted from database. Q-gram index is widely used in computational biology, text retrieval etc. for its advantages of less memory consumption, easy management and simple construction. A q-gram index includes vocabulary and position lists. Vocabulary is a set of distinct q-grams whose addresses can be located quickly and each q-gram has a position list which consists of all its positions in database.

Hash table and B* tree are often used as the structure of vocabulary. Hash table is employed here for its fast locating speed. To build a q-gram index, a map function for a given alphabet Σ size of σ is constructed first, for example, a function of DNA database is shown in Eq. 1.

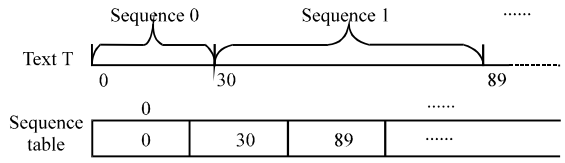

Fig. 2: An example of sequences connection and sequence table

$$I('A') = 0, I('C') = 1, I('G') = 2, I('T') = 3 \quad (1)$$

In Eq. 1, the value of I (c) is an unique integer for every character in Σ, $0 \le I(c) < \sigma$, $c \in \Sigma$. At last, each q-gram's hash value is computed by Karp-Rabin function (Karp and Rabin, 1987). It is unnecessary to compute twice for two adjacent q-grams overlap with q-1 characters. The next q-gram's hash value can be achieved in O (1) time by Karp-Rabin function.

In this study, an array with length of σ^q is allocated as the hash table. This can avoid the conflict of hash values and improve the locating speed.

Preprocessing database: Q-gram index can be constructed by only one pass over all sequences in database. Here, the database is considered as a long sequence T connected by all sequences in database logically and each sequence's start position in T is stored into an array called sequence table, as Fig. 2. Given a region in T, its corresponding sequence can be achieved quickly through sequence table.

The process of q-gram index construction is shown as follows:

Step 1: An array with length of σ^q is allocated as hash table. All sequences in database are numbered and connected in order logically. Then sequences are processed one by one in order of their number

Step 2: A sequence's content is read from database and the start position in T is stored into sequence table. Then all q-grams are extracted consecutively from the content and their positions are appended into their position lists by using their hash values

Step 3: Repeating step 2 until all sequences are processed

All q-gram's position lists are ordered lists because their positions are appended in order. At last, a q-gram index is constructed for the given database.

THE PROPOSED FILTER

Diagonal features extraction: Some definitions and lemmas are presented as follows.

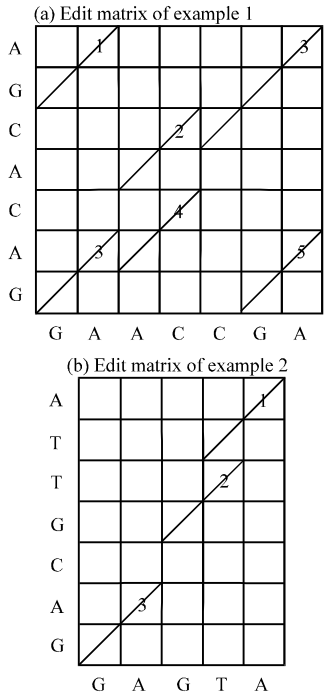

Fig. 3 (a-b): Examples of valid diagonal

Definition 1: In the edit matrix of two strings, the diagonals on their edit paths are valid diagonals.

There is only one edit distance (Levenshtein, 1966) between two strings but they may have many edit paths. In Fig. 3a, the edit distance between these two strings is 2 and they have two different edit paths in their edit matrix, such as diagonal (2, 3) and diagonal (3, 4). Therefore, diagonal 2, 3 and 4 are valid diagonals. However, diagonal 1 and 5 are not valid.

Lemma 1: Let a subsequence T' at most k edit distance with S occur in T, i.e., ED (T', S)≤k. Then there are at most k+1 consecutive valid diagonals on each edit path in their edit matrix.

Proof: The edit operations include insertion, deletion and substitution. There is only one valid diagonal when the edit distance between two strings is 0. The valid diagonals will not be changed when the operation is substitution and only one new valid diagonal will be added left or right when the operation is insertion or deletion. In a word, it will add at most k new valid diagonals with k edit operations, i.e., k+1 valid diagonals totally. So, lemma 1 holds.

Lemma 1 displays an upper bound of valid diagonals for every true match. In Fig. 3a, there are two valid diagonals on each edit path, i.e., 2≤3. In Fig. 3b, the edit distance is 2 and there are three valid diagonals totally, i.e., 3≤3.

Lemma 2: Let a subsequence T' at most k edit distance with S occur in T, i.e., ED (T', S)≤k. Then there is at least one valid diagonal whose q-gram hits is not less than c (k) = ⌈(|S| + 1)/(k + 1)⌉-q.

Proof: We suppose that there exist an edit path in the edit matrix of string T' and S, the q-gram hits on each valid diagonal of this edit path is less than c (k). By Lemma 1, there are at most k+1 valid diagonals on this edit path. So the total q-gram hits on this edit path is c_1 and $c_1 \le (\lceil (|S| + 1)/(k+1) \rceil 1\text{-}q)(k+1 \Rightarrow c_1 < |S| + 1\text{-}(k+1)q$. As ED (T', S)≤k, the total q-gram hits shared by T' and S is not less than |S| + 1-(k + 1)q by the basic q-gram lemma. This conclusion is conflictive to the assumption and then the assumption is not right. So, Lemma 2 holds.

Lemma 2 displays a new diagonal feature, that is, the edit matrix will contain at least one valid diagonal whose q-gram hits is not less than c (k) if one string contains a true match of query string.

The proposed filter: A new local alignment algorithm (LAADF) which uses a q-gram index will be detailed in this section. In the proposed algorithm, the new diagonal features described are used for filtration. LAADF includes steps of preprocessing database, input, filtration, verification and output, while the filtration phase includes preprocessing query, choosing filtration-regions and filtration. The flow chart is illustrated as Fig. 4.

- **Step 1: Preprocessing database:** The process of preprocessing database has been detailed above. The value of q differs from applications and it will be given later

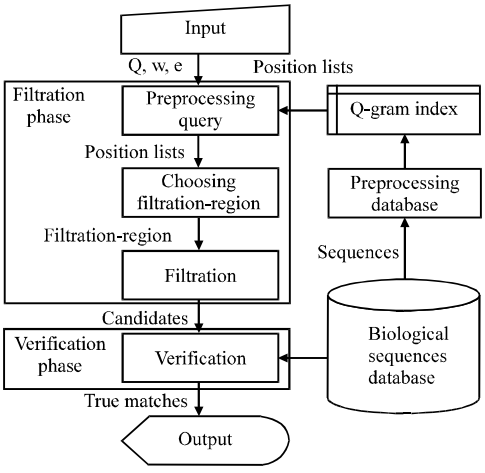

Fig. 4: Flow chart of LAADF

- **Step 2: input:** The inputs include query sequence Q (|Q| = m), window length w (w≥q) and max error rate e (e≥0). Then the max edit distance can be computed by k = we.
- **Step 3: Filtration phase:** The study of preprocessing query is to locate the position lists of all q-grams in Q. First, all q-grams are extracted consecutively from query sequence in order of their position. For example, Q [0, q-1] is the first q-gram and its position is 0. At last, all q-grams' position lists are located in vocabulary by using their hash values and these hash values are computed by Karp-Rabin function

How to choose filtration-region is the key issue of improving filtration efficiency of a filter. A good scheme of choosing filtration-region can eliminate much more irrelevant subsequences in a very short filtration time.

Here, the database is regarded as a long sequence T (|T| = n) and the first subsequence with length of w in Q is considered, i.e., Q [0,.., w-1]. Then an imaginary edit matrix of T and Q is formed in mind and an array H with length of n+m-1 is allocated in memory to count the q-gram hits on each diagonal. For example, the total q-gram hits shared by Q [0, w-1] and T on diagonal i is stored into counter H [i]

However, there are many subsequences with length of w in Q, e.g., Q [1,.., w], Q[2,.., w + 1],.... To align all these subsequence, the technique of shifting window in QUASAR is employed to process query sequence. First, a window with length of w is placed at the start of Q and then the counters are increased by visiting position list of each q-gram in the window. For example, p is a position in Q [j,..., j + q-1]'s position list and its corresponding diagonal is p-j + m-1, i.e., H [p-j + m-1]++. Q [1,..., q], Q [2,..., q + 1],..., Q [w-q,..., w-1] are processed by this way in order. Second, the window is shifted forward by one character. Then Q [0,..., q-1]'s position list is visited again to decrease the counters because this q-gram is shifted out of the window. Meanwhile, the counters are increased by visiting Q [w-q + 1,..., w]'s position list because Q [w-q + 1,..., w] is shifted in the window. The window is shifted like this until it reaches the end of Q. In total, All q-grams' position lists are visited twice.

Filtration-region is a subsequence which includes true matches in large probability. By Lemma 2, if a true match is contained in a subsequence, there must exist at least one diagonal whose counter is not less than ⌈(w+1)/(we+1)⌉-q. On the other hand, if a diagonal satisfies the threshold there will contain true matches in large probability. To find filtration-region effectively, the counters' value will be checked when the counters are increased. For example, the counter H [i] is increased

when visiting Q [j,..., j + q-1]'s position list. If H [I]≥⌈(w+1)/(we+1)⌉-q, a w×(2k+1) parallelogram formed of 2k+1 diagonals and a subsequence with length of w in Q will be considered. It is unnecessary to add more than k diagonals on both sides of diagonal i by Lemma 1. So, the leftmost diagonal of the parallelogram is i-k and the rightmost one is i+k, the nether most character in Q is j + q-w and the uppermost one is j + q-1. The covered content of this parallelogram in T is a filtration-region and it is denoted as $F_{i-k,i+k}^{Q[j+q-w,j+q-1]}$.

The filtration-regions which are not candidates will be eliminated by using filter criterion. Here, the filter criterion selected is the basic q-gram lemma. First, the total q-gram hits of each filtration-region is computed by Eq. 2.

$$\text{AllHits}(F_{i-k,i+k}^{Q[j+q-w,j+q-1]}) = \sum_{a=i-k}^{i+k} H[a] \quad (2)$$

$F_{i-k,i+k}^{Q[j+q-w,j+q-1]}$ is discarded if it satisfies AllHits $(F_{i-k,i+k}^{Q[j+q-w,j+q-1]})$ <w+1-(we+1) q. Otherwise, it is a candidate, denoted as $V_{T[c-k-w,c+k-1]}^{Q[j+q-w,j+q-1]}$, Q [j + q-w, j + q-1] is the candidate's corresponding subsequence in Q and the corresponding subsequence in T is T [c-k-w, c + k-1], c = i-m + 1 + j + q. At last, these candidates are appended into a candidate set and candidates that are overlapping or concatenating in both T and Q are combined into one:

- **Step 4: Verification phase:** The second phase of filter algorithm is verification. Here, smith-waterman algorithm is used to verify each candidate in candidate set
- **Step 5: Output:** At last, all true matches are found and then sorted in ascending order of their error rate

Degeneration: The value of c (k) = [(w + 1)/(we + 1)-q] decreases with the increasing of e and the degeneration of LAADF happens gradually. When [(w + 1/(we + 1)-q], i.e.,

$$\frac{w-q}{w(q+1)} \le e < \frac{w-q+1}{wq}$$

It will spend a lot of time on filtration because all diagonals whose q-gram hits nonzero are checked. When

$$e \ge \frac{w-q+1}{wq}$$

the algorithm will be worst because only the basic q-gram lemma works.

ANALYSIS

The analysis for time complexity and space requirements of q-gram index are presented in this section and as well as the proposed filter. At last, the time complexity and space requirements in filtration phase of the proposed filter are compared with those of the well-known QUASAR and SWIFT.

Q-gram index: To construct a q-gram index, it needs only a single pass over the given database. The total expected time is O (n) instead of O (nq) because it takes O (1) time in both locating each q-gram's position list and hashing each q-gram by using Karp-Rabin function.

The space requirement of vocabulary is O (σ^q) because it is actually an array with length of σ^q. The space requirement of position lists is O (n) because all the q-grams in database are stored in position lists. Hence, the total space requirements are O (n + σ^q) and it is linear for σ^q<n generally.

The proposed filter: The time complexity of the proposed filter will be analyzed first. The average length of position list is n/σ^q, the total q-grams contained in query sequence is m-q + 1 and each position list is visited twice. So the total visited positions is 2 (m-q+1) n/σ^q. We suppose that the probability of a diagonal which shares at least t common q-grams is p (t) in random text, t = ⌈(|S| + 1)/(we + 1)⌉-q. Then 2k + 1 counters will be visited if a diagonal shares and only a counter will be visited if it does not shares. So the time complexity of filtration phase is O ((m-q+ 1) (p (t) we + 1) n/σ^q, it is linear for (m-q + 1) (p (t) we + 1)≤σ^q, P (t)≤1, e<1 generally. We suppose that the filtration efficiency of LAADF is f. Then the time complexity of verification is O (wn (1-f)), it is linear if f≥1-1/w. The filtration efficiency of new filter is always higher than 1-1/w in our all experiments. In conclusion, the total time complexity is O (((m-q + 1) (p (t) we +)/σ^q + w (1-f)) n) usually it is linear.

The space requirement of counters is O (n + m) and that of verification is O (w^2). Hence, the total space requirements are O (n + m + w^2), it is linear for w<<n, m<<n generally.

Comparison: The main difference among QUASAR, SWIFT and the proposed filter is filtration process. The time complexity and space requirement in filtration phase are compared in Table 1.

As stressed by Table 1, the time complexities of these three algorithms in filtration phase are all linear generally and as well as the space requirement. Moreover,

Table 1: Comparison of time complexity and space requirement in filtration phase

	Filtration phase	
Algorithms	Time complexity	Space requirement
QUASAR	$n (m-q+1)/\sigma^q + n/w$	n/w
SWIFT	$n (m-q+1)/\sigma^q + (n+m)/(2^z+1)$	$3 (n+m)/(2^z+1)$
LAADF	$n (m-q+1) (p (t)we+1)/\sigma^q$	n + m

the space requirement of SWIFT decreases with the increasing of max error rate but it is larger than that of LAADF on condition of low max error rate.

EXPERIMENTS

The proposed filter LAADF is developed with C++ and it is compared with the well-known local alignment algorithms, including QUASAR and SWIFT.

Experimental environment: The experimental database is UniGene Homo Sapiens which is downloaded from GenBank (ftp.ncbi.nih.gov/repository/UniGene/Homo-sapiens/Hs.seq.uniq.gz). The size of database is 164 MB and 123,252 unique homo gene sequences in total. The value of q in new filter is 11 for the same value in both QUASAR and SWIFT. To compare the performance of these filters, QUASAR's block size assigned is 2 w and SWIFT's bin size assigned is $k + 2^x + 1$, $x \in \mathbb{N}$, $2^x > k$.

Total 1,000 subsequences with length of 2,000 are selected from database randomly. Random edit operations are performed on these subsequences and we guarantee that the error rate between edited subsequences and the original one is below 2%. At last, these subsequences constitute the query set.

All experiments are conducted on a computer with AMD X4 630 and 4 GB main memory.

Performance: To compare the performance of filters, alignments (w = 50) for all queries in query set are conducted on the experimental database using QUASAR, SWIFT and LAADF, respectively. The average filtration time, filtration efficiency, verification time and searching time are calculated in experiments. Fig. 5-8 present the results.

Figure 5 compares the filtration time of QUASAR, SWIFT and LAADF and Fig. 6 displays their filtration efficiency. Figure 7 illustrates their verification time, while Fig. 8 compares their average searching time. Figures 5-8 show:

- LAADF is always faster than QUASAR
- LAADF is always faster than SWIFT except on condition of high max error rate

Experiments with w = 200 have also been done here and LAADF achieves the same performance as above.

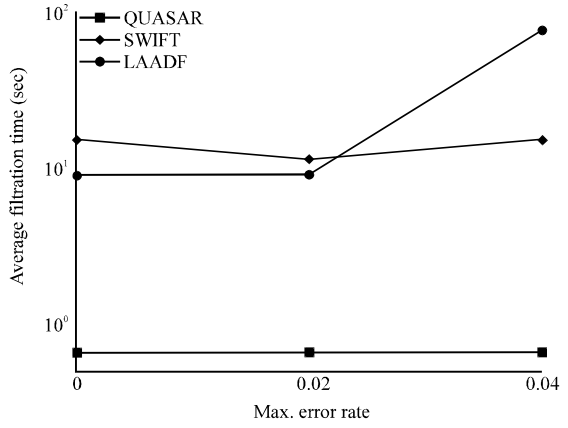

Fig. 5: Comparison of average filtration time

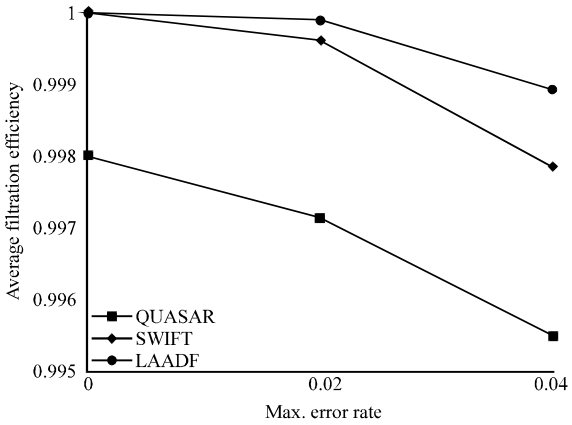

Fig. 6: Comparison of average filtration efficiency

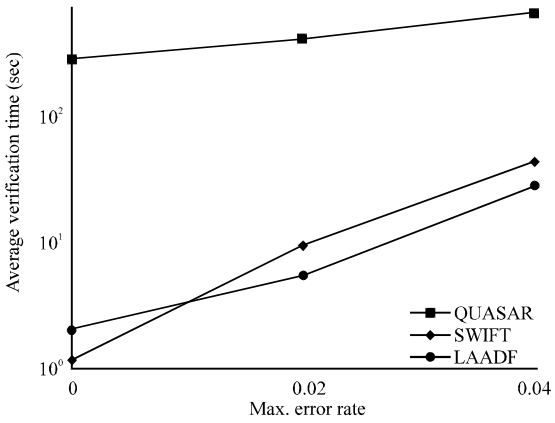

Fig. 7: Comparison of average verification time

Compare with w = 50, the results show that LAADF is better at local alignment with shorter window size.

Stability: The searching time of a given algorithm varies from queries and the variation displays its stability. Here,

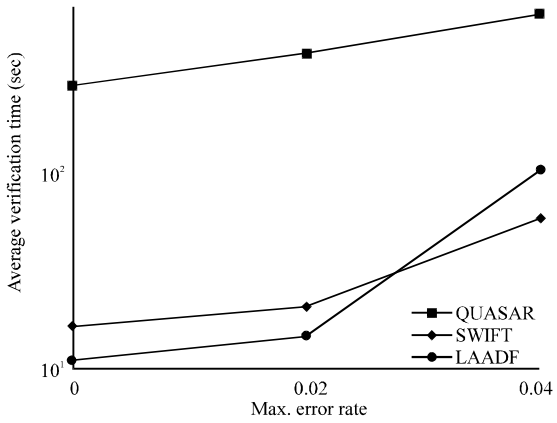

Fig. 8: Comparison of average searching time

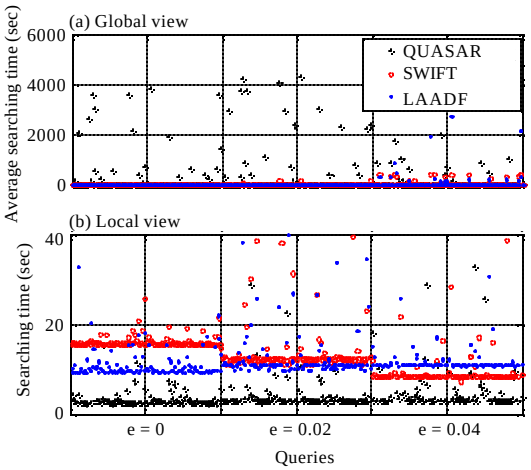

Fig. 9 (a-b): Comparison of stability

120 queries is searched on the experimental database (w = 50). The comparison of algorithms' searching time is shown as Fig. 9.

Figure 9a shows a global view of stability for three algorithms while Fig. 9b gives a local view. QUASAR is unstable because it changes sharply for its lower filtration efficiency. Both LAADF and SWIFT are relatively stable on condition of low max error rate. However, they will become unstable when the max error rate is high. LAADF is relatively stable on condition of low max error rate.

DISCUSSION

The first notable filter algorithm in 1990's is QUASAR. The technique of shifting window and the basic q-gram lemma are also employed in QUASAR. However, QUASAR's searching speed is not higher than that of our proposed filter for its lower filtration efficiency and longer verification time. By filtrating with the new extracted diagonal features, LAADF can eliminate more irreverent subsequences in filtration phase, so the verification time is decreased greatly though its filtration time is a little longer.

The well-known filter algorithm SWIFT is the closest precursor to the work presented in this study. The basic q-gram lemma and a diagonal feature are also used in SWIFT. But SWIFT's filtration time is very long though it achieves high filtration efficiency. Compared with SWIFT, LAADF achieves even higher filtration efficiency because more rigorous filter criterion is fused with new extracted diagonal features and the basic q-gram lemma. However, LAADF's filtration time is longer than that of SWIFT on condition of high max error rate for the degeneration of it. In a word, LAADF improves the searching speed on condition of low max error rate.

All the analysis and experimental results demonstrate that the proposed filter is a fast local alignment algorithm on condition of lower max error rate and the new extracted diagonal features works well in the proposed filter. The present of new filter is of great practical to local alignment with low error rate and short window size.

CONCLUSION

This study has presented an efficient filter algorithm to perform local alignment in a biological database. New filter enhances filtration efficiency by using new extracted diagonal features. Comparing with SWIFT, the proposed filter saves searching time on condition of low max error rate. However, the searching speed is a little slower than that of SWIFT when the max error rate is high. Actually it performs very well for local alignment with low error rate and short window size in bioinformatics. We are going to improve it and continue to research on extracting new match-region features. Now, the gapped-seed idea is orthogonal to the filtration algorithm and we will apply these new match-region features to gapped q-gram filtration algorithm.

ACKNOWLEDGMENTS

This study is partially supported by National Basic Research Program of China (973 Program) under Grant No. 2009CB326202 (2009.4-2011.8, Hunan University) and 2010CB334706 (2010.4-2013.3, Hunan University). Key Program of National Natural Science Foundation of China under Grant No. 60736016 (2008.1-2011.12, Hunan University). National Natural Science Foundation of China under Grant No. 60973128 (2010.1-2012.12, Hunan University), 60973113 (2010.1-2012.12, Hunan University),

61073191 (2011.01-2013.12, Hunan University) and 61070196 (2011.01-2013.12, Hunan University). PAPD (2011.04-2016.3, Jiangsu Province). Scientific Research Fund of Hunan Provincial Education Department of China under Grant No. 10C0589 (2010.9-2012.11, Hunan Institute of Technology).

REFERENCES

Altschul, S.F., T.L. Madden, A.A. Schaffer, J. Zhang, Z. Zhang, W. Miller and D.J. Lipman, 1997. Gapped BLAST and PSI-BLAST: A new generation of protein database search programs. Nucl. Acids Res., 25: 3389-3402.

Burkhardt, S., A. Crauser, P. Ferragina, H.P. Lenhof, E. Rivals and M. Vingron, 1999. Q-gram based database searching using a suffix array. Proceedings of the 3rd Annual International Conference on Computational Molecular Biology (RECOMB'99), Lyon, France, pp: 77-83.

Burkhardt, S. and J. Karkkainen, 2003. Better filtering with gapped q-grams. Fund. Inform., 56: 51-70.

Fu, Y., X. Wang and S. Li, 2009. Distributed index based on geographic hashing table for mobile Ad hoc networks. Inform. Technol. J., 8: 1197-1204.

Haliloglu, K. and H. Bostan, 2002. Nucleotide sequence analysis for assessment of variability of potato leafroll virus and phylogenetic comparisons. J. Biological Sci., 2: 582-586.

Jokinen, P. and E. Ukkonen, 1991. Two algorithms for approximate string matching in static texts. Math. Found. Comput. Sci., 520: 240-248.

Karimi Kordestani, G. and K. Omid, 2008. Sequence analysis and phylogeny of vird4 protein of type iv secretion system in gram-negative bacteria. Biotechnology, 7: 523-529.

Karkkainen, J., P. Sanders and S. Burkhardt, 2006. Linear work suffix array construction. J. ACM., 53: 918-936.

Karp, R.M. and M.O. Rabin, 1987. Efficient randomized pattern-matching algorithms. IBM. J. Res. Dev., 31: 249-260.

Khurshid, R. and A. Saleem, 2000. Amino acid sequence homology and modeling of cysteine protease cathepsin S. Pak. J. Biol. Sci., 3: 639-641.

Levenshtein, V., 1966. Binary codes capable of correcting deletions, insertions and reversals. Soviet Physics Doklady, 10: 707-707.

Myers, G., 1999. A fast bit-vector algorithm for approximate string matching based on dynamic programming. J. ACM., 46: 1-13.

Navarro, G. and R. Baeza-yates, 1998. A practical q-gram index for text retrieval allowing errors. CLEI Electron. J., 1: 1-16.

Navarro, G., R. Baeza-Yates, E. Sutineny and J. Tarhio, 2001. Indexing methods for approximate string matching. IEEE Data Eng. Bull., 24: 19-27.

Ning, Z.M., A.J. Cox and J.C. Mullikin, 2001. SSAHA: A fast search method for large DNA databases. Genome Res., 11: 1725-1729.

Pearson, W.R. and D.J. Lipman, 1988. Improved tools for biological sequence comparison. Proc. National Acad. Sci. USA., 85: 2444-2448.

Puglisi, S.J., W.F. Smyth and A. Turpin, 2006. Inverted files versus suffix arrays for locating patterns in primary memory. Lect. Notes Comput., 4209: 122-133.

Rasmussen, K.R., J. Stoye and E.W. Myers, 2006. Efficient q-gram filters for finding all epsilon-matches over a given length. J. Comput. Biol., 13: 296-308.

Sahin, Y.G., S.M. Basarici and T. Ercan, 2007. Face matrix: A quick search and indexing method for suspect recognition in police departments. Inform. Technol. J., 6: 607-612.

Smith, T.F. and M.S. Waterman, 1981. Identification of common molecular subsequences. J. Mol. Biol., 147: 195-197.

Sutinen, E. and J. Tarhio, 1996. Filtration with q-samples in approximate string matching. Combinatorial Pattern Matching, 1075: 50-63.

Sutinen, E. and W. Szpankowski, 1998. On the collapse of the q-gram filtration. Proceedings of International Conference on FUN with Algorithms (FUN'98), Elba, Italy, pp: 178-193.