International Journal of Hybrid Information Technology
Vol.9, No.1 (2016), pp. 221-232
http://dx.doi.org/10.14257/ijhit.2016.9.1.19

A Q-gram Filter for Local Alignment in Large Genomic Database

Decai Sun[1,*] and Xiaoxia Wang[2]

[1,*]*College of Information Science and Technology, Bohai University, Jinzhou 121013, China*
[2]*Teaching and Research Institute of College Computer, Bohai university, Jinzhou 121013, China.*
sdecai@163.com, wxxsdc@163.com

Abstract

Fast and exact searching for sequences similar to a query sequence in genomic databases remains a challenging task in molecular biology. In this paper, the problem of finding all e-matches in a large genomic database is considered, i.e. all local alignments over a given length w and an error rate of at most e. A new database searching algorithm called QFLA is designed to solve this problem. The proposed algorithm is a full-sensitivity algorithm which is a refined q-gram filter and implemented on a q-gram index. First, new features are extracted from match-regions by logically partitioning both query sequence and genomic database. Second, a large part of irrelevant subsequences are eliminated quickly by these new features during the searching process. Last, the unfiltered regions are verified by the well-known smith-waterman algorithm. The experimental results demonstrate that our algorithm saves time by improving filtration efficiency in a short filtration time.

Keywords: *sequence comparison, local alignment, filter algorithm, q-gram filter, q-gram index*

1. Introduction

Bioinformatics has received increasing publicity over the past few years, in large part due to its importance to the Human Genome Project. With the increment of genomic sequences, numerous large databases holding DNA and protein sequences are now readily available over the WEB, such as GenBank and PIR-PSD. Searching a genomic database for sequences similar to a given query sequence is a fundamental problem in bioinformatics [1], but as the quantity of available sequence information continues to multiply, the problem of remains very much relevant today.

The old methods for aligning two sequences are those based on dynamic programming [2]. The need for greater speed especially when searching for true matches in a large database, the heuristics algorithm has been developed, such as FASTA [3] and BLAST [4-5]. But it still cannot fulfill what today's applications' need.

Filter algorithm is a two-phase alignment algorithm [6]. In the first phase, large parts of irrelevant text which do not contains any true matches are discarded rapidly. In the second phase, the unfiltered text, called candidates, is verified by another algorithm to search all true matches. Filter algorithm is an off-line algorithm which requires an index to preprocess database. There are three different data structures are often used in the literature [7], including suffix tree, suffix array and q-gram index [8-9].

Now, there are mainly two different types in the existing filtration algorithms [10]:

1) String matching approach. The idea of the string matching approach is to search some substrings of pattern which exactly or approximately appear in text first.

ISSN: 1738-9968 IJHIT

In [11-13], the pattern is split into $k+s$ pieces, and hence at least s of the pieces must appear in any true matches. Therefore, the text that contains at least s of those pieces and requires the stated distance is verified for a complete match. In [12, 14], the pattern is split into j pieces, and hence at least one of these pieces which has at most $\lfloor k/j \rfloor$ errors with the pattern's one must appear in the true matches.

2) q-gram counting approach. The q-gram counting approach uses the q-grams of two strings for filtration. In [15], q-samples are extracted from every h characters in text, hence the text that contains a certain number of pattern's q-samples and requires the stated distance is verified. In [16-18], text is split into q-grams which are overlapped and continuous, hence the text that contains at least $|P|+1-(k+1)q$ of $|P|-q+1$ pattern's q-grams is verified.

The famous filter algorithms in literature include QUASAR, SWIFT, etc. QUASAR [17] is a refined filter based on the work of Jokinen and Ukkonen [16]. Its filtration time is short, but the verification time is very long for its low filtration efficiency. SWIFT [18] is a local alignment algorithm got inspiration from FASTA [3]. It achieves higher filtration efficiency, but its filtration time is very long. Now, the gapped-seed idea is orthogonal to the filtration method [19-21]. It also can be applied to our algorithm, but we do not employ here.

In this paper, a full-sensitivity filter called QFLA is proposed to improve the speed of finding all e-matches in a large genomic database. The rest is organized as follows. Preliminaries are introducted in Section 2. Our new filter is detailed in Section 3. Section 4 analyzes the complexity of new filter. The experimental results and performance analysis are discussed in Section 5. Lastly, the conclusions are made in Section 6.

2. Preliminaries

The first q-gram lemma of q-gram filter is proposed by Jokinen and Ukkonen [16], and the lemma is shown as follows.

Lemma 1. Let an occurrence of sequence S with at most k edit distance end at $T[j]$ in T. Then at least $|S|+1-(k+1)q$ of the $|S|-q+1$ q-grams of S occur in $T[j-|S|+1,\mathrm{L}\ , j]$.

$T[i,\mathrm{L}\ , j]$ is a substring starting from the i th character to j th character in T. Edit distance [22] is the minimum number of insertions, deletions, and substitutions required to transform one string into another. And a q-gram is a substring length q of T or S.

Lemma 1 gives a basic feature of a match-regions where includes true matches, it guarantees that at least $t(k)=|S|+1-(k+1)q$ of the q-grams contained in S occur in this match-region. Obviously, all substrings of T with this feature are candidates.

Error rate is often used to scale the similarity between query sequence and its true matches, and it is defined as $k/|S|$. The ***matching error rate*** of filter is a limit to make sure that the error rates between query sequence and its true matches do not exceed the limit. A q-gram filter will function well when its matching error rate in $\left[0,1/q-1/|S|\right]$, otherwise, it will collapse [23].

The total searching time of filter algorithm mainly consists of filtration time and verification time. Filtration efficiency is also a key factor which can display the efficiency on eliminating irrelevant text. We redefine filtration efficiency according to [15] for better evaluating the performance of filter algorithm.

$$f_e = \frac{n-n_f}{n-n_t} \tag{1}$$

In (1), n denotes the total length of database; n_f denotes the total length of unfiltered regions; and n_t denotes the total length of true matches in database.

The aim of this paper is to accelerate the speed of finding all *e-matches* in a large genomic database. The definition of *e-matches* is presented in [24].

3. A Refined Q-gram Filter

3.1. Match-region Feature Extraction Based on Partition

In this section, some new features will be extracted from match-region and presented as lemmas. Here, let q denote the length of q-gram, $q \geq 1$; k denote the maximal edit distance, $k \geq 0$; S denote a subsequence of query sequence Q. And the whole database is regarded as a long text T logically.

Definition 1. S and T are divided into blocks with fixed length b, $b \geq q, b \geq kq+1$, and the blocks are consecutive but they do not overlap with each other. In S, there are u perfect query blocks excluding the last one if its length is shorter than b, $u = \lfloor |S|/b \rfloor$, and the residual block length is $l = |S| - ub$. In T, there are r text blocks including the last one, $r = \lceil |T|/b \rceil$. An example is shown in Figure 1.

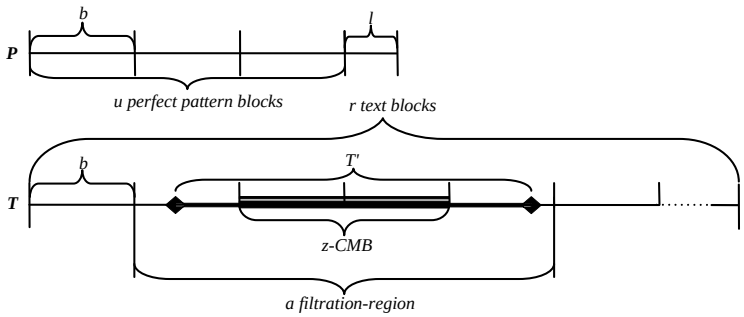

Figure 1. Partition, CMB and Filtration-Region

Lemma 2. S and T are partitioned by Definition 1. Let a subsequence T' at most k edit distance with S occur in T, i.e., $ED(T',S) \leq k$. Then T' strides at least $z = f(u,l,c)$ consecutive text blocks in T.

$$f(u,l,c) = \begin{cases} u-1, l \geq c \\ u-2, l < c \end{cases}, c = \frac{b-1}{q} - 1 \tag{2}$$

Proof. As $ED(T',S) \leq k$, hence

$$|T'| \geq |S| - k \tag{3}$$

From Definition 1, $b \geq kq+1, k \geq 0, q \geq 1$, then $k \leq (b-1)/q$.

As $|S| = ub + l$

Case 1: when $l \geq c$, then $|S| \geq ub + (b-1)/q - 1$

Replace $|S|$ by its minimal value and k by its maximal value in (3), then

$|T'| \geq ub + (b-1)/q - 1 - (b-1)/q \Rightarrow |T'| \geq ub - 1$

Hence, at least $u-1$ consecutive text blocks are strode by T' because its length is not shorter than $ub-1$.

Case 2: when $l < c$, then $|S| \geq ub$.

Replace $|S|$ by its minimal value and k by its maximal value in (3), then

$|T'| \geq ub - (b-1)/q \Rightarrow |T'| \geq (u-1)b - 1 + (b+1-(b-1)/q)$

As $q \geq 1, b > 0$, then $b+1-(b-1)/q > 0$. Hence, at least $u-2$ consecutive text blocks are strode by T' because its length is not shorter than $(u-1)b-1$.

According to Cases 1 and Cases 2, such that (2) holds.

Lemma 3. S and T are partitioned by Definition 1. Let a subsequence T' at most k edit distance with S occur in T, *i.e.*, $ED(T',S) \le k$. Then all the text blocks strode by T' excluding the last one share at least $b-kq$ common q-grams with S, and $b-kq>0$. And the last text block strode by T' shares at least $b+1-(k+1)q$ common q-grams with S.

Proof. As $ED(T',S) \le k$, hence at least z consecutive text blocks are strode by T' according to Lemma 2.

Let B_i be the ith text block of z consecutive blocks and C_i be the total number of characters following B_i in T'.

As $b \ge q$, hence $C_i > q-1, 1 \le i < z$ and $C_z \ge 0$.

Let $T_i', 1 \le i < z$, be the subsequence of T' and T_i' is formed by the content of B_i and the adjacent $q-1$ characters next to B_i.

As $ED(T',S) \le k$, hence at least one subsequence S_i' satisfies $ED(T_i',S_i') \le k$.

Obviously, T_i' shares at least $b+q-1+1-(k+1)q = b-kq$ common q-grams with S according to Lemma 1.

As $b \ge kq+1$, hence $b-kq>0$ holds.

Let T_z' be a subsequence of T' and T_z' is formed by the content of B_z and the adjacent j characters next to B_z, $j \ge 0$.

As $ED(T',S) \le k$, hence at least one subsequence S_z' satisfies $ED(T_l',S_z') \le k$.

As $|T_z'| \ge b$, Hence T_z' shares at least $b+1-(k+1)q$ common q-grams with S according to Lemma 1.

Definition 2. A ***valid text block*** is a text block which shares at least $b-kq$ common q-grams with S. An ***additional text block*** is not a valid text block but it is a succeeding text block of valid text block and it shares at least $b+1-(k+1)q$ common q-grams with S. A $x+y$ ***consecutive match-blocks***, denoted $(x+y)$-CMB, is formed with x consecutive valid text blocks and y additional text block, $x \ge 0, y = 0$ or 1.

Lemma 4. S and T are partitioned by Definition 1. Let a subsequence T' at most k edit distance with S occur in T, *i.e.*, $ED(T',S) \le k$. Then at least one z-CMB (z is defined in (2)) is contained in T.

Proof. As $ED(T',S) \le k$, T' is a true match of S, hence at least one z-CMB is strode by T' in T according to Lemma 2. According to Lemma 3 and Definition 2, these consecutive text blocks are all valid text blocks except the last one, and the last one is a valid text block or an additional text block. Hence a z-CMB is formed, and Lemma 4 holds. An example is shown in Figure 1.

Lemma 4 summarizes a new match-region feature. It guarantees that a match-region contains at least one z-CMB. On the contrary, there contains true matches in large probability if a region contains a c-CMB, $c \ge z$.

3.2. New Filter

In this section, a new filter called QFLA will be presented. The new filter is designed with Lemma 4, and it includes steps of preprocessing database, input, preprocessing query, choosing filtration-regions, filtration, verification and output. The flow chart of QFLA is illustrated as Figure 2.

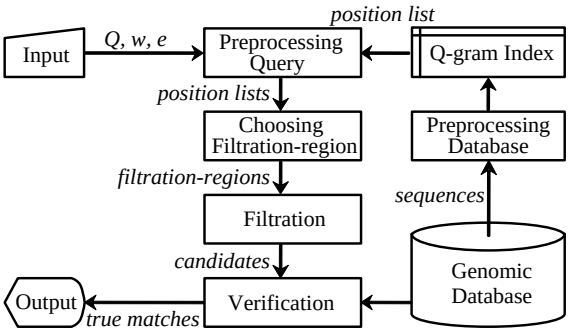

Figure 2. Flow Chart of QFLA

3.2.1. Preprocessing Database, Input and Preprocessing Query: The work of preprocessing database is to construct an index for the given database. Q-gram index [24] is employed here because its searching speed is faster than that of suffix array under the circumstance of same memory space [25]. The input include query sequence Q, window length w ($w \geq q$), and matching error rate e ($e \geq 0$). Then the maximal edit distance is computed by $k = we$. In preprocessing query, query sequence Q is split into $|Q| - q + 1$ consecutive and overlapped q-grams in order of their position, and then their position lists are located in q-gram index.

3.2.2. Choosing Filtration-Region: The key of improving filtration speed in a filter is how to choose filtration-region. It can eliminate much more irrelevant text in a very short filtration time when a good scheme is employed.

1) Partitioning Text

We consider one subsequence length w of Q first, i.e., $Q[0,\mathrm{L}\ ,w-1]$. According to Lemma 1, the length of block b can be computed by the number of consecutive match-blocks z.

$$b = \left\lfloor \frac{w+1+1/q}{z+1+1/q} \right\rfloor \tag{4}$$

When the matching error rate e is too high or the window length w is too short, it will lead to $b < q$ or $b < kq+1$, then QFLA will fail in partition. These special problems will be detailed in Section 3.3.

T can be divided into r text blocks logically by Definition 1. A array H of length r is allocated in memory to count the number of common q-grams shared between each block and $Q[0,\mathrm{L}\ ,w-1]$, for example, $H[0]$ is the counter of 0th text block.

2) Shifting Window and Finding Filtration-regions

There is not only one subsequence of length w in Q. Actually, there are many others, e.g., $Q[1,\mathrm{L}\ ,w], Q[2,\mathrm{L}\ ,w+1],\mathrm{L}$. To align all these subsequences of Q, the technique of shifting window [17] is employed here.

First, a window of length w is placed at the start of query sequence Q, and then the q-grams in window are processed by order of their position. For example, position list of 0th q-gram $Q[0,\mathrm{L}\ ,q-1]$ is already located in q-gram index, and then we can increase the values of counters by visiting positions in 0's position list in order. Suppose that p is a position in 0th position list, then the number of block hit by p is $\lfloor p/b \rfloor$, so $H[\lfloor p/b \rfloor]{+}{+}$. Then $Q[1,\mathrm{L}\ ,q], Q[2,\mathrm{L}\ ,q+1],\mathrm{L}\ , Q[w-q,\mathrm{L}\ ,w-1]$ can be processed in the same way. Second, when we are going to process $Q[w-q+1,\mathrm{L}\ ,w]$, we find it has already exceeded the boundary of window. Then we have to shift the window forward by one character, while $Q[0,\mathrm{L}\ ,q-1]$ is shifted out of the window. Thus, we must visit 0's position list again to decrease the values of counters. Meanwhile, $Q[w-q+1,\mathrm{L}\ ,w]$ is shifted in the window, so

the counters' values are increased by visiting its position list. We make shifting for window like this until it reaches the end of Q.

It needs to visit position lists twice in total. In one position list, perhaps two or more positions hit in the same block. So, we increase or decrease the value of this counter only with one though the hits are more than one. It is easy to eliminate redundant hits because they are consecutive in the position list.

According to Lemma 4, a region contains true matches with a large probability if it contains a c-CMB, $c \geq z$. To make the choosing of filtration-region more effectively, the operation of finding z-CMB will be performed when the value of each counter is increased. For example, when visiting position list of j th q-gram $Q[j,\mathrm{L}\ , j+q-1]$, the algorithm of finding z-CMB according to the value of counter $H[i]$ will be expressed in pseudocode. The algorithm is shown as Figure 3.

Algorithm 1: Finding_z-CMB(*H, i, j, z, r*)

```
Input: Block counters H; Current counter i; q-gram position j;
Consecutive match-blocks number z; Total text blocks number r.
Output: Empty or a x-CMB.
if H[i]> = b - weq then /*It is a valid text block*/
    t ← i - 1
    while (t > i - z && t >= 0) /* Search left */
        if H[t] >= b - weq then
            t ← t - 1
        else
            break;
    a ← t + 1
    t ←i + 1
    while(t < i + z && t < r ) /* Search right */
        if H[t] >= b - weq then
            t ← t + 1
        else
            break;
    if t < i + z && t < r && H[t] >= b + 1 - (we + 1)q then
        t ← t + 1
    c ← t - 1
    if c – a + 1 >= z then
        return M^j_{a,c}
else if H[i] >= b + 1 - (we + 1)q then /*It is an additional text block*/
    SearchLeft() /*The codes same as 2-8*/
    c ← i
    if c – a + 1 >= z then
        return M^j_{a,c}
return NULL
```

Figure 3. The Algorithm of Finding *z*-CMB

Algorithm 1 illustrates the process of finding z-CMB by moving pointer towards to both left and right from current block. It is unnecessary to search out of $z-1$ blocks on both sides because the z-CMB will definitely be found by other hits once there has one in T. We let $M_{a,c}^{j}$ denote the $(c-a+1)$-CMB found by j th q-gram, a denotes the start text block number, and c denotes the end text block number. According to Lemma 2 and Lemma 4, a z-CMB is only a part of T'. We form the filtration-region by $M_{a,c}^{j}$, block $a-1$ and block $c+1$ to guarantee that all characters of T' are contained in it. An example is shown in Figure 1. The filtration-region is denoted as $F_{a-1,c+1}^{Q[j+q-w,\mathrm{L}\ ,j+q-1]}$, and $Q[j+q-w,\mathrm{L}\ , j+q-1]$ is its corresponding subsequence in Q.

To reduce filtration time in filtration phase, Filtration-regions which are overlapping or concatenating both in T and Q will be combined into one.

3.2.3. Filtration, Verification and Output: The aim of filtration phase is to eliminate the filtration-regions which are not candidates by using filter criterion. Here the filter criterion is Lemma 1. First, the total hits of filtration-region $F_{x_1,x_2}^{Q[y_1,\mathrm{L}\ ,y_2]}$ is computed by (5).

$$AllHits(F_{x_1,x_2}^{Q[y_1,\mathrm{L},y_2]}) = \sum_{k=x_1}^{x_2} H[k] \tag{5}$$

The filtration-region $F_{x_1,x_2}^{Q[y_1,\mathrm{L},y_2]}$ will be discarded if it satisfies $AllHits(F_{x_1,x_2}^{Q[y_1,\mathrm{L},y_2]}) < w+1-(we+1)q$. Otherwise, the filtration-region is a candidate, and its content in T is read out by using q-gram index and database, while its corresponding subsequence in Q is read directly. For example, the content of $F_{x_1,x_2}^{Q[y_1,\mathrm{L},y_2]}$ in T is $T[x_1 b,\mathrm{L},(x_2+1)b-1]$, and its corresponding subsequence in Q is $Q[\max(y_1,0),\mathrm{L},y_2]$. At last, $F_{x_1,x_2}^{Q[y_1,\mathrm{L},y_2]}$ is appended to the candidate set.

The second phase of filter algorithm is verification. To find out all true matches accurately, the algorithm of smith-waterman [2] is used to verify each candidate in candidate set. At the end of this phase, all true matches are found out and sorted in ascending order of error rate, and then the output comes out.

3.3. Invalidation and Degeneration

Our algorithm might produce these problems in the following cases:

When the matching error rate e is too high or window length w is too short, it might lead to $b<q$ or $b<kq+1$, thus QFLA will fail in partitioning text, such as $w=60, q=11, e=0.06, z=1$. But in practical applications, e is usually very small, and the invalidation of QFLA happens rarely. QFLA will work well when $e \le (1-1/q)/w, w \ge (z+1)q-1/q$ or $e>(1-1/q)/w, w \ge z/(1-((z+1)q+1)e)$ because the length of block is valid.

It will produce the maximal block length when the value of z is 1. When $b+1-(we+1)q \le 0$, every hit will make the operation of finding z-CMB performed, which will spend a lot of time on choosing filtration-region so as to causing degeneration of QFLA, such as $w=50, q=11, e=0.04, z=1$.

4. Analysis

In this section, we will present the analysis for time and space requirements of both our index and new filter. Here, $n=|T|, m=|Q|$, σ is the vocabulary size of database.

1) Q-gram Index. To construct a q-gram index, 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 [24]. The total space requirements of q-gram index are $\Theta(\min(n,\sigma^q)+n)$ which is linear [8].

2) New Filter. We analyze the time complexity of new filter. First, there are $m-q+1$ position lists in total and each list is visited twice in filtration phase, and the average length of position list is n/σ^q. Then total number of positions read by QFLA is $2(m-q+1)n/\sigma^q$, and half of them are read to increase counters. In choosing filtration-region, weather the operation of finding z-CMB performed or not is determined by the value of current counter. We suppose that q-grams are independent and analyze them in a random text, and the probability that two sequences (length w and b, $w>b$) share at least t common q-grams is $P(t)$. a) The probability of being a valid text block is $P(t_1), t_1 = b-weq$, and the total of visited blocks is no more than $2z+1$. b) The probability of being an additional text block is $p(t_2)-p(t_1), t_2 = \max(b+1-(we+1)q,1)$, and the total of visited blocks is no more than $z+2$. c) The probability of the rest is $1-p(t_2)$, and only one block is visited. Therefore, the time complexity of preprocess query and choosing filtration-region is $O(((m-q+1)n(1+p(t_1)(2z+1)+(p(t_2)-p(t_1))(z+2)+(1-p(t_2))))/\sigma^q)$. As z is a constant in searching, and $p(t_1)\le 1, p(t_2)\le 1$, hence the complexity is $O(n(m-q)/\sigma^q)$, which will be

linear when $m-q \le \sigma^q$, and $m = \sigma^q$ in general. Last, supposing the filtration efficiency of QFLA is f, then the length of verified text is $n(1-f)$. Hence, it takes $O(wn(1-f))$ time to verify candidates. This shows that f must be kept $f \ge 1-1/w$ for the time to be linear, and the filtration efficiency of new filter is always higher than $1-1/w$ in our all experiments. In conclusion, the total time complexity is $O((m-q)n/\sigma^q + mn(1-f))$, which is usually linear. Second, we analyze the space requirements in searching. It takes $O(n/b)$ space to store array H and takes $O(w^2)$ space to verify one candidate using smith-waterman algorithm. Thus, the total space requirements is $O(n/s + w^2)$, which will be linear when $w^2 \le n$, and $w = \sqrt{n}$ in general.

5. Experimental Results

To compare the performance of relate filters, the algorithms of QUASAR [17], SWIFT [18] and QFLA are developed with C++.

5.1. 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 total size of database is 164MB, and it has 123,252 unique homo gene sequences in all. 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 $2w$ and SWIFT's bin size assigned is $k + 2^x + 1, x \in ¥, 2^x > k$.

We select 1,000 subsequences at length 2000 from genomic database randomly, and then make random edition to them, including insertions, deletions and substitutions. We guarantee that the error rates between new sequence and the original one are below 2%. Last, these new sequences construct the query set.

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

5.2. Parameter z

The value of z is the number of CMB in QFLA. The searching speed would be different when using different z for the same query. To find out the optimal value of z, we conduct alignments ($w=50$ and $w=200$) on the experimental database. Figure 4 illustrate the experimental results.

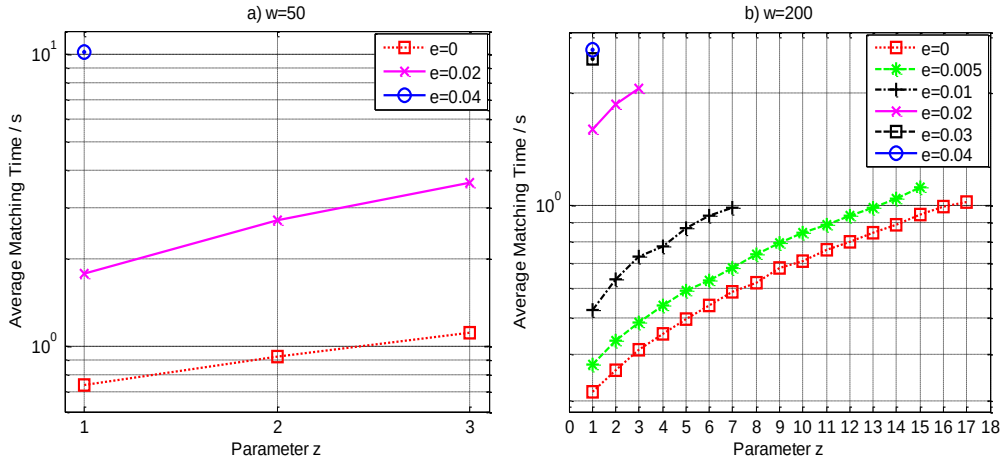

Figure 4. Parameter *z*

Figure 4 shows that QFLA performs the best in both cases of $w=50$ and $w=200$ by using $z=1$. There is no other value can be chosen for z except 1 in QFLA when $e=0.04$. When $z=1$, the length of block will be at its maximum, and the threshold of valid text block and that of additional text block are both increased. This makes the filter criterion of each block more rigorous, but makes the number of CMB minimal.

5.3. Performance

5.3.1. Matching Error Rate Effect on Filters: The Performance of each filter varies from different matching error rate. We use query set to conduct alignments ($w=50$) on the experimental database using QUASAR, SWIFT and QFLA respectively. The average filtration time, filtration efficiency, verification time and searching time are calculated in experiments. Figure 5-8 present the experimental results.

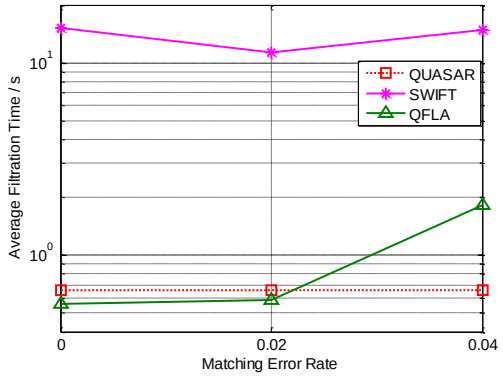

Figure 5. Comparison of Average Filtration Time

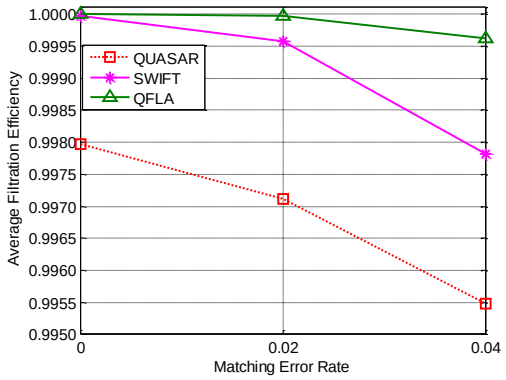

Figure 6. Comparison of Average Filtration Efficiency

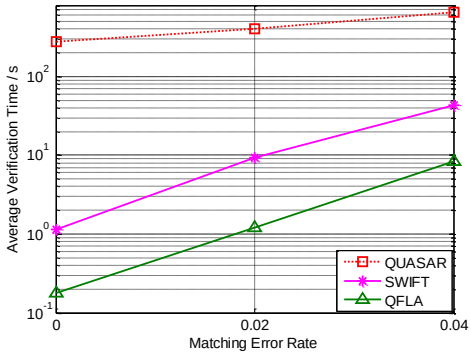

Figure 7. Comparison of Average Verification Time

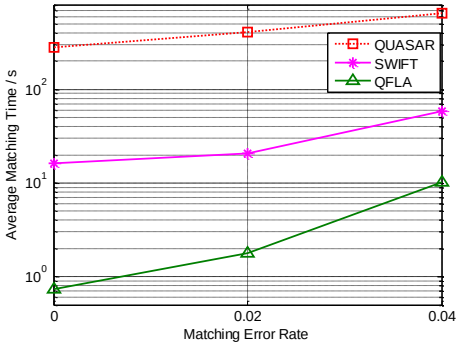

Figure 8. Comparison of Average Searching Time

Figure 5 compares the filtration time of QUASAR, SWIFT and QFLA, and Figure 6 displays their filtration efficiency, while Figure 7 illustrates their verification time. All these three figures show:

1) QUASAR's filtration time is the shortest because its method of choosing filtration-region is simple. However, its filtration efficiency is the lowest, which makes its verification time the longest.

2) SWIFT's filtration time is the longest because its method of choosing filtration-region is complex and it devours a large amount of memory in searching. But its filtration efficiency is the second highest, which makes its verification time the second shortest.

3) QFLA achieves the highest filtration efficiency costing the second shortest filtration time, and its verification time is the shortest because it can eliminate much more irrelevant text by effective finding of z-CMB.

At last, Figure 8 compares the average searching time of QUASAR, SWIFT and QFLA. QFLA is almost one order of magnitude faster than SWIFT, while SWIFT is over one order of magnitude faster than QUASAR. The searching process of QFLA is accelerated by improving filtration efficiency in a short filtration time. In a word, the searching speed of QFLA is faster than that of QUASAR and SWIFT, especially when the matching error rate is low.

We also do experiments with $w = 200$, and the results show that QFLA achieves the same performance as above.

5.3.2. Stability: As above, the query set is processed as a batch and the results are averaged. It is well known that the searching speed of an algorithm varies from queries, and the variation displays the stability of the algorithm. Here, we use 120 queries to conduct alignments ($w = 50$) on the experimental database. The comparison of every query's searching time using three algorithms respectively is shown as Figure 9.

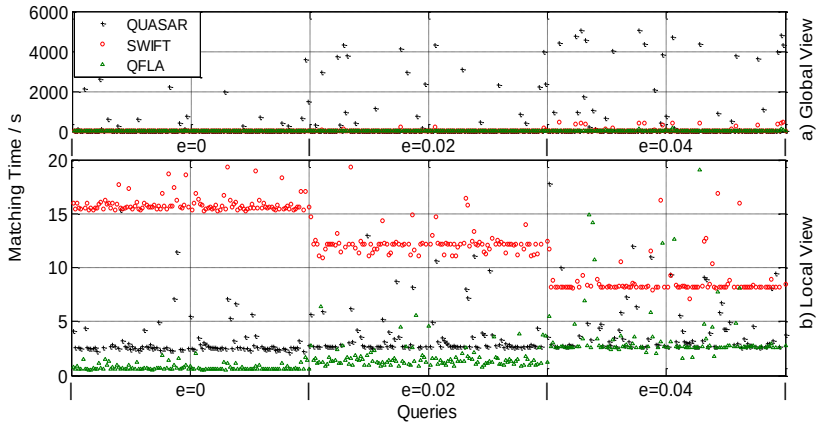

Figure 9. Comparison of Stability

Figure 9 a) shows a global view of stability for three algorithms. QUASAR is unstable because it changes sharply though most queries' searching time is short. Comparing with QUASAR, QFLA and SWIFT are stable algorithms. Figure 9 b) gives a local view, it illustrates that QFLA is more stable than SWIFT, especially when matching error rate is low.

5.4. Discussion

Here, the difference among QFLA, QUASAR and SWIFT is discussed. First, QFLA is faster than QUASAR and SWIFT in our all experiments. 1) QFLA's searching speed is higer than that of QUASAR though their filtratio time is close. This is because of QFLA's shorter verification time (higher filtration efficiency). 2) QFLA is faster than SWIFT because QFLA achieves shorter verification time (higher filtration efficiency) and even shorter filtration time. Second, QFLA is more stable than QUASAR and SWIFT especially under low matching error rate. Last, QFLA defeats QUASAR and SWIFT under low matching error rate, but it will be invalid when the matching error rate is too high or the window length is too short (see 3.3).

6. Conclusion

This paper is mainly aiming to solve the problem of finding all e *-matches with window length* w in large genomic database. We described the theoretical framework and a workable solution for an efficient q-gram filter called QFLA. Our algorithm enhances filtration efficiency in a short filtration time and accelerates searching speed by decreasing verification time. This result is of great practical importance to numerous applications, such as whole-genome alignment and approximate text retrial. However, it will be invalid when the matching error rate is too high or the window length is too short. 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.

Acknowledgements

This work is supported by the Doctoral Scientific Research Foundation of Liaoning province (20141138), Natural Science Foundation of Liaoning Province of China (2013010251-401), Foundation of Liaoning Province Education Department of China (L2013422), NSFC (61232016, 61173142, 61173141, 61173136 and 61202462).

References

[1] K.-M. Chao and L. Zhang, Springer Publishers, Berlin, **(2008)**.
[2] T. F. Smith, M. S. Waterman, Journal of Molecular Biology, vol. 147, no, 1, **(1981)**.
[3] R. P. William and J. L. David, Proc. Natl. Acad. Sci. USA, vol. 85, no. 8, **(1988)**.
[4] S. F. Altschul, W. Gish, W. Miller, E. W. Myers and D. J. Lipman, Journal of Molecular Biology, vol. 215, no. 3, **(1990)**.
[5] S. F. Altschul, T. L. Madden, A. S. Alejandro, J. Zhang, Z. Zhang, W. Miller and D. J. Lipman, Nucleic Acids Research, vol. 25, no. 17, **(1997)**.
[6] A. J. T. Lee, C.-W. Lin, W.-H. Lo, C.-C. Chen and J.-X. Chen, Pattern Recognition Letters, vol. 28, no. 4, **(2007)**.
[7] G. Navarro, R. Baeza-Yates, E. Sutinen and J. Tarhio, IEEE Data Engineering Bulletin, vol. 24, no. 4, **(2001)**.
[8] G. Navarro and R. Baeza-yates, CLEI Electronic Journal, vol. 1, no. 2, **(1998)**.
[9] Z. Ning, J. C. Anthony and C. M. James, Genome Research, vol. 11, no. 10, **(2001)**.
[10] C. W. Lu, C. L. Lu and R. C. T. Lee, Theoretical Computer Science, vol. 481, no. 0, **(2013)**.
[11] S. Wu and U. Manber, Communications of the ACM, vol. 35, no. 10, **(1992)**.
[12] G. Navarro and R. Baeza-Yates, Journal of Discrete Algorithms, vol. 1, no. 1, **(2000)**.
[13] Y.-I. Chang, J.-R. Chen and M.-T. Hsu, Applied Intelligence, vol. 33, no. 1, **(2010)**.
[14] R. Baeza-Yates and G. Navarro, Algorithmica, vol. 23, no. 2, **(1999)**.
[15] E. Sutinen and J. Tarhio, "Filtration with q-samples in approximate string matching", Proceedings of the 7th Annual Symposium on Combinatorial Pattern Matching, **(1996)** June 10-12, Laguna Beach, CA, USA.
[16] P. Jokinen and E. Ukkonen, "Two algorithms for approximate string matching in static texts", Proceedings of the 16th International Symposium Proceedings on Mathematical Foundations of Computer Science, **(1991)** September 9-13, Kazimierz Dolny, Poland.
[17] S. Burkhardt, A. Crauser, P. Ferragina, H.-P. Lenhof, E. Rivals and M. Vingron, "Q-gram based database searching using a suffix array", Proceedings of the 3rd Annual International Conference on Computational Molecular Biology, **(1999)** April 11-14, Lyon, France.
[18] K. R. Rasmussen, J. Stoye and E. W. Myers, Journal of Computational Biology, vol. 13, no. 2, **(2006)**.
[19] S. Burkhardt and J. Karkkainen, Fundamenta Informaticae, vol. 56, no. 1-2, **(2003)**.
[20] L. Egidi and G. Manzini, Journal of Computer and System Sciences, vol. 79, no. 7, **(2013)**.
[21] S. Ilie, BMC Research Notes, vol. 5, no. 123, **(2012)**.
[22] V. I. Levenshtein, Soviet Physics Doklady, vol. 10, no. 8, **(1966)**.
[23] E. Sutinen and W. Szpankowski, "On the collapse of the q-gram filtration", Proceedings of the International Conference on FUN with Algorithms, **(1998)** June 18-20, Isola d'Elba, Italy.
[24] D. Sun, X. Sun, X. Wang and Z. Ruan, Information Technology Journal, vol. 10, no. 8, **(2011)**.
[25] S. J. Puglisi1, W. F. Smyth and A. Turpin, "Inverted files versus suffix arrays for locating patterns in primary memory", Proceedings of the 13th Symposium on String Processing and Information Retrieval, **(2006)** October 11-13, Berlin, Germany.

Authors

Decai Sun, He received his BE at Daqing Petroleum Institute, China, in 2002, ME at Hunan University, China, in 2009, PhD in computer science and technology from Hunan University, China, in 2012. He works as a lecturer at College of Information Science and Technology in Bohai university. His research interests include Computational Biology, Information Retrieval, and Approximate String Matching.

Xiaoxia Wang, She received his BE at Daqing Petroleum Institute, China, in 2002, ME at University of South China, China, in 2009. He works as a lecturer at Teaching and Research Institute of College Computer in Bohai University. His research interests include Computational Biology, Information Retrieval, and Intrusion Detection.

