Fundamenta Informaticae XXIII (2003) 1001–1018
IOS Press

Better Filtering with Gapped q-Grams

Stefan Burkhardt[*][†]
Max-Planck-Institut für Informatik
Stuhlsatzenhausweg 85, 66123 Saarbrücken, Germany
stburk@mpi-sb.mpg.de

Juha Kärkkäinen[†]
Max-Planck-Institut für Informatik
Stuhlsatzenhausweg 85, 66123 Saarbrücken, Germany
juha@mpi-sb.mpg.de

Abstract. A popular and well-studied class of filters for approximate string matching compares substrings of length q, *the q-grams*, in the pattern and the text to identify text areas that contain potential matches. A generalization of the method that uses *gapped* q-grams instead of contiguous substrings is mentioned a few times in literature but has never been analyzed in any depth. In this paper, we report the first results of a study on gapped q-grams. We show that gapped q-grams can provide orders of magnitude faster and/or more efficient filtering than contiguous q-grams. To achieve these results the arrangement of the gaps in the q-gram and a filter parameter called *threshold* have to be optimized. Both of these tasks are nontrivial combinatorial optimization problems for which we present efficient solutions. We concentrate on the k mismatches problem, i.e, approximate string matching with the Hamming distance.

1. Introduction

Given a pattern string P, a text string T, and a distance k, the *approximate string matching problem* is to find all substrings S of the text (*actual matches*) that are within a distance k of the pattern P. A

[*] Supported by the DFG 'Initiative Bioinformatik' grant BIZ 4/1-1.
[†] Partially supported by the Future and Emerging Technologies programme of the EU under contract number IST-1999-14186 (ALCOM-FT).

very common distance measure, leading to the k *differences problem*, is the *Levenshtein distance*, the minimum number of single character insertions, deletions and replacements required to transform one string into the other. A simpler variant, the k *mismatches problem*, uses the *Hamming distance*, which does not allow insertions or deletions, i.e., it is the number of non-matching characters for strings of the same length. The *indexed* versions of these problems allow the use of a precomputed index of the text, while in the *online* versions no such additional data structures are available. An extensive survey of online approximate string matching is given in [16]. Indexed approximate string matching is surveyed in [18].

Filtering is a way to speed up approximate string matching, particularly in the indexed case but also in the online case. A *filter* is an algorithm that quickly discards large parts of the text using a *filter criterion*, leaving the interesting parts, the *potential matches*, to be checked with a proper (online) approximate string matching algorithm. These two phases are the *filtration* and *verification phases*. A filter is *lossless* if it never discards an actual match; we consider only lossless filters. The ability of a filter to reduce the text area is called its *filtration efficiency*.

Many filters are based on q*-grams*, substrings of length q. The q-gram similarity (defined as a distance in [28]) of two strings is the number of q-grams shared by the strings. The q-gram filter is based on the q*-gram lemma*:

Lemma 1.1. (The q-gram lemma [10])
Let P and S be strings with (Levenshtein or Hamming) distance k. Then the q-gram similarity of P and S is at least $t = |P| - q + 1 - kq$.

As an example, the strings `ACAGCTTA` and `ACACCTTA` have Hamming and Levenshtein distance 1 and $8 - 3 + 1 - 1 \cdot 3 = 3$ common 3-grams: `ACA`, `CTT` and `TTA`.

The value t in the lemma is called the *threshold* and specifies the minimum number of q-grams that an approximate match S, a substring of T, must share with the pattern P. This condition is used as the filter criterion. The threshold given by the lemma is *optimal*, i.e., it is the highest value of t for which the filter is lossless. A filter that uses a lower value for t is also lossless but produces more potential matches, which increases the time required for verification. Filters that use q-grams are particularly suitable for indexed string matching since there are efficient data structures for indexing all text q-grams. Indexed q-gram filters are described in [9, 10, 15, 17, 19, 26].

A generalization of the q-gram filter uses *gapped* q-grams, subsets of q characters of a fixed non-contiguous *shape*. For example, the 3-grams of shape `##-#` in the string `ACAGCT` are `ACG`, `CAC` and `AGT`. Gapped q-grams have been used in [7, 20, 13]. In [7, 20], the motivation is to increase the filtration efficiency by considering multiple shapes. Pevzner and Waterman [20] use q-grams containing every $(k+1)$st character together with ordinary contiguous q-grams for the Hamming distance. Califano and Rigoutsos [7] describe a *lossy* filter that uses as many as 40 different random shapes. Their approach is effective for high k but they need a huge index (18 GB for a 100 million nucleotide DNA database). The Grampse system of Lehtinen et al. [13] uses a shape containing every hth character for some h (similar to [20]) for *exact* matching. Their motivation of using gapped q-grams is to reduce dependencies between the characters of a q-gram.

In previous work, gapped q-grams were introduced to gain the ability to use multiple different shapes or to reduce dependencies between characters. In this paper, we show that gapped q-grams have a more fundamental advantage over contiguous q-grams. When used correctly, gapped q-grams can provide much faster and/or more efficient filtering. We have observed improvements of several orders of magnitude in our experiments.

Our approach to gapped q-grams differs from previous work in two important ways. First, while regular shapes are used in [20, 13] and random shapes in [7], we use *carefully chosen arbitrary shapes*. We performed an exhaustive search of all feasible shapes and chose the best ones. As a criterion for picking the best shapes, we introduce a measure called the *minimum coverage*, which we show by experiments to correlate closely with the filtration efficiency. The best shapes typically possess little regularity, while regular shapes, including contiguous shapes, are actually among the worst shapes. A similar observation was made by Califano and Rigoutsos [7], who found out experimentally that randomly generated shapes outperform deterministically generated regular shapes. However, good shapes are rare and still far better than randomly selected ones. Califano and Rigoutsos also mention the idea of using optimized shapes but do not provide any method for choosing the shapes beyond experimental comparison.

The second deviation from previous work is that we use an *optimal* and *shape-dependent* threshold. Pevzner and Waterman [20] give equations for the threshold of their regular shapes that guarantee losslessness but are optimal only in certain special cases. Califano and Rigoutsos [7] determine the threshold based on probabilistic calculations and experiments. Their threshold does not depend on the actual shapes and may lead to missing some actual matches (lossy filter). We have not found a closed form formula for the optimal threshold of arbitrary shapes like the one provided by the q-gram Lemma. However, we present a simple definition of the optimal threshold and an efficient dynamic programming algorithm for computing it.

For the sake of simplicity, we consider only the k mismatches problem in this paper. The k differences problem has a wider range of applications, but it causes difficulties for gapped q-grams because they are affected by insertions and deletions in the gaps. A replacement in a gap will not invalidate a matching q-gram but an insertion or deletion will. An approach to tackle these difficulties by using multiple shapes is presented in [6]. Nevertheless, there are also practical applications for which the Hamming distance is a good, sometimes even a better distance measure. Examples include optical character recognition(OCR) and the detection of single nucleotide polymorphisms (SNPs) [21].

This paper is an extended and updated version of [5]. Since then other results using carefully selected gapped q-grams to improve approximate string matching have been published [14, 2]. Both publications recognize the superiority of gapped over contiguous shapes and the importance of selecting good shapes. However, they focus on the case where the threshold $t = 1$ and the filters they propose are lossy.

The rest of this paper is organized as follows. In the first part of the paper, we define shapes as a description of gapped q-grams and describe a method to compute the optimal threshold for a gapped q-gram. We then introduce a property of q-grams, the minimum coverage, that can be used to predict their filtration efficiency. In the experiments section, we analyze the prediction quality of the minimum coverage and compare gapped with contiguous q-grams using artificial and authentic DNA databases as test cases.

2. Gapped q-Grams

We start by introducing some basic terminology and notation on gapped q-grams.

Let I be a set of integers. The *size* of I, denoted by $|I|$, is the cardinality of the set. The *span* of I is $\mathrm{span}(I) = \max I - \min I + 1$. The *position* of I is $\min I$, and the *shape* of I is the set $\{i - \min I \mid i \in I\}$.

An integer set Q with position zero is called a *shape*. A shape Q with size q and span s is called a q-shape or a (q, s)-shape.

For any shape Q and integer i, let Q_i denote the set with shape Q and position i, i.e., $Q_i = \{i + j \mid j \in Q\}$. Let $Q_i = \{i_1, i_2, \ldots, i_q\}$, where $i = i_1 < i_2 < \cdots < i_q$, and let $S = s_1 s_2 \ldots s_m$ be a string. For $1 \leq i \leq m - \text{span}(Q) + 1$, the *$Q$-gram* at position i in S, denoted by $S[Q_i]$, is the string $s_{i_1} s_{i_2} \ldots s_{i_q}$. A Q-gram is also called a q-gram or a (q, s)-gram, where $q = |Q|$ and $s = \text{span}(Q)$.

Example 2.1. Let $Q = \{0, 1, 3, 6\}$ be a shape. Using the notation from the introduction Q is the shape `##-#--#`. Its size is 4 and its span is 7. Thus Q is a 4-shape and a $(4, 7)$-shape. The string $S =$ `ACGGATTAC` has three Q-grams: $S[Q_1] = s_1 s_2 s_4 s_7 =$ `ACGT`, $S[Q_2] =$ `CGAA` and $S[Q_3] =$ `GGTC`. This is illustrated below.

```
S = A C G G A T T A C
    A C - G - - T
      C G - A - - A
        G G - T - - C
```

We say that two strings P and S of length m *share* a Q-gram or have a *common* Q-gram at position i if $P[Q_i] = S[Q_i]$. The *Q-gram similarity* $s_Q(P, S)$ of P and S is the number of their common Q-grams, i.e., $s_Q(P, S) = |\{i \in [1, m - \text{span}(Q) + 1] \mid P[Q_i] = S[Q_i]\}|$.

Example 2.2. Let Q and S be as in Example 2.1, and let $P =$ `ACTGACTAG`. Then $s_Q(P, S) = 1$, since P and S share only the Q-gram `CGAA` at position 2:

```
P = A C G T A C T A G
    A C - ? - - T
      C G - A - - A
        G ? - ? - - ?
S = A C G G A T T A C
```

3. Threshold

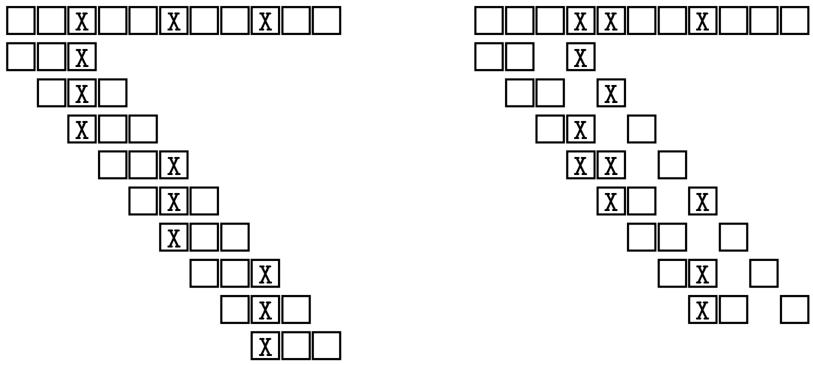
The basic idea of a Q-gram filter is to identify all text substrings whose Q-gram similarity with the pattern is at least t. A too high value for the threshold t makes the filter lossy, i.e., some actual matches might be missed. On the other hand, increasing the threshold improves the filtration efficiency, i.e., reduces the number of false positives. Thus, the optimal value of the threshold is the maximal value that does not make the filter lossy.[1]

The q-gram lemma (Lemma 1.1) gives a simple formula for the optimal value of the threshold t for contiguous q-grams (see also Lemma 3.3). A straightforward generalization for gapped Q-grams yields the formula $t = |P| - \text{span}(Q) + 1 - |Q|k$ (see Lemma 3.4 below). Pevzner and Waterman [20] give

[1] 'Lossy' threshold values may be of interest in practice but are not considered in this paper.

this formula for their regular shapes.[2] However, the formula is not optimal for gapped q-grams as shown by the following example.

Example 3.1. Let $|P| = 11$ and $k = 3$, and consider the 3-shapes `###` and `##-#`. The above formula gives thresholds 0 and -1, respectively, for the two shapes. Thus, neither shape would seem to be useful for lossless filtering in this case. However, the optimal threshold for the shape `##-#` is actually 1. By an exhaustive search of all possible arrangements of 3 mismatches one can verify that at least one `##-#`-gram is always unaffected by the mismatches. The following figure gives an example of a worst possible arrangement of 3 mismatches for both shapes.



The above example already suggests that the optimal threshold value can be defined in terms of the worst-case arrangements of mismatches. We formalize this next.

Let $P = p_1, \ldots p_m$ and $S = s_1 \ldots s_m$ be two strings of length m. Let $M(P, S)$ denote the set of positions where P and S match, i.e., $M(P, S) = \{i \in [1, m] \mid p_i = s_i\}$. Then $m - |M(P, S)|$ is the Hamming distance of P and S. Also, the Q-gram similarity can be expressed using the set $M(P, S)$:

Lemma 3.1. $s_Q(P, S) = |\{i \in [1, m - \text{span}(Q) + 1] \mid Q_i \subseteq M(P, S)\}|$.

The optimal threshold is the largest value for which the filter is still lossless. In other words, it is the lowest Q-gram similarity of any two strings of length m with Hamming distance k. By Lemma 3.1, it is enough to consider the worst possible combination of the mismatches. Hence, we get the following general Q-gram lemma.

Lemma 3.2. (The Q-gram lemma)
Let Q be a shape. For any two strings P and S of length m that are within Hamming distance k of each other, the Q-gram similarity $s_Q(P, S)$ of P and S is at least

$$t_Q(m, k) = \min_{M \subseteq [1,m], |M| = m-k} |\{i \in [1, m - \text{span}(Q) + 1] \mid Q_i \subseteq M\}|.$$

Furthermore, for every string P of length m, there exists a string S that is within Hamming distance k of P such that $s_Q(P, S)$ is exactly $t_Q(m, k)$.

[2] They give a better formula for shapes with a span close to the length of the pattern.

We have not found a closed form for the optimal threshold in the general case, but there are closed forms for special cases. In particular, the classic q-gram lemma for contiguous q-grams yields the optimal threshold as shown by the following lemma.

Lemma 3.3. Let Q be a contiguous shape, i.e., $Q = [0, q-1]$. Then
$t_Q(m,k) = \max\{0, m - \text{span}(Q) + 1 - |Q|k\} = \max\{0, m - q + 1 - kq\}$.

Proof:
The lower bound is shown by Lemma 3.4. To show the upper bound, let $M = [1,m] \setminus \{q, 2q, \ldots, kq\}$. Then Q_i is contained in M if and only if $i \in [kq+1, m-q+1]$, and the number of such Q_i's is given exactly by the formula. □

Closed forms could also be derived for other classes of regular shapes. However, regular shapes appear to be rather poor shapes.

One could also try to find good closed form *lower bounds* for the optimal threshold. Using such a lower bound as the threshold guarantees the losslessness of the filter but reduces filtration efficiency. The following lower bound we already saw in the beginning of this section.

Lemma 3.4. $t_Q(m,k) \geq \max\{0, m - \text{span}(Q) + 1 - |Q|k\}$.

Proof:
Let M be the set minimizing the value of $t_Q(m,k)$. For each $j \notin M$ there is exactly $|Q|$ integers i such that $j \in Q_i$. Therefore, at most $k|Q|$ of the sets Q_i, $i \in [1, m - \text{span}(Q) + 1]$, are not in M, and at least $m - \text{span}(Q) + 1 - k|Q|$ are in M. □

Unfortunately, this is a poor lower bound in general. Furthermore, even with a good lower bound the filtration efficiency can be reduced dramatically. A decrease of one in the threshold increases the expected number of potential matches by a factor of at least $|\Sigma|$, where Σ is the alphabet, and in many cases much more as we will see in Section 6. Example 3.1 shows that the optimal threshold may be required to achieve any filtration at all.

The lack of a closed form for the optimal threshold does not make it impossible to compute. In the next section, we describe a dynamic programming algorithm which achieves this. We have implemented the algorithm and used it, together with a pruning technique, to compute the optimal thresholds for all shapes for $m = 50$ and $k \in \{4, 5\}$. Tables 2 and 3 in the Appendix (pages 1019–1020) give the highest threshold among (q,s)-shapes for all combinations of q and s that have shapes with positive thresholds (except $q = s = 1$).

The tables show that Example 3.1 was not an isolated instance: In many cases, especially for higher values of q, the best gapped shapes have much higher thresholds than contiguous shapes of the same or even smaller size.

However, it is not sufficient just to have gaps; the shape has to be chosen carefully. For instance, for the parameters of Example 3.1, $m = 11$, $k = 3$, $q = 3$, the shape `##-#` and its mirror image `#-##` are the only ones that have a positive threshold. As a more impressive example, for the parameter values $m = 50$, $k = 5$ and $q = 12$, there are only two shapes, `###-#--###-#--###-#` and `#-#-#---#-----#-#-#---#-----#-#-#---#`, (and their mirror images) with a positive threshold. In most cases shown in Tables 2 and 3, only a few shapes achieve the highest threshold. The distribution of threshold values for one typical case is shown in Figure 1 on page 1012.

4. Computing the Optimal Threshold

The optimal threshold could be computed by an exhaustive search of all combinations of k mismatches according to Lemma 3.2, but this is prohibitively expensive for large values of m and k. We have developed a significantly faster dynamic programming algorithm for computing the threshold.

Let us first define some notation. For a set I of integers let $I \oplus j$ denote the set $\{i + j \mid i \in I\}$, and define the operator $\ominus$ similarly. Let $[cond]$ be 1 if the condition *cond* is true and 0 otherwise.

The computation is based on the following conditional threshold.

Definition 4.1. Let Q be a shape with span s. For non-negative integers m, $k \leq m$, and a set $M \subseteq [1, s-1]$, $|M| \geq s - 1 - k$, let $t_Q(m, k, M)$ denote the optimal threshold for pattern length m and Hamming distance k under the additional condition that the matches in the last $s-1$ positions correspond to M. That is (cf. Lemma 3.2)

$$t_Q(m, k, M) = \min_{\substack{M' \subseteq [1,m],\ |M'|=m-k \\ (M' \ominus (m-s+1)) \cap [1,s-1]=M}} |\{i \in [1, m - s + 1] \mid Q_i \subseteq M'\}|.$$

The actual optimal threshold can be computed by minimizing over the appropriate conditional thresholds:

Lemma 4.1. $t_Q(m, k) = \min_{M \subseteq [1,s-1], |M| \geq s-1-k} t_Q(m, k, M)$.

The dynamic programming algorithm is based on the fact that the values $t_Q(i, *, *)$ can be computed from the values $t_Q(i - 1, *, *)$. Note that it is not possible to compute $t_Q(i, *)$ from $t_Q(i - 1, *)$; the matches in the last $s - 1$ positions need to be recorded as well. The following lemma gives the detailed recursion.

Lemma 4.2. For $s \leq i \leq m$, $0 \leq j \leq k$ and $M \subseteq [1, s - 1]$, $|M| \geq s - 1 - j$, we have

$$\begin{aligned} t_Q(s-1, j, M) &= 0 \\ t_Q(i, j, M) &= \min \begin{cases} t_Q(i-1, j - [s-1 \notin M], (M \cup \{0\} \setminus \{s-1\}) \oplus 1) \\ \qquad\qquad\qquad\qquad + [Q \subseteq (M \cup \{0\})] \\ t_Q(i-1, j - [s-1 \notin M], (M \setminus \{s-1\}) \oplus 1) \end{cases} \end{aligned}$$

Proof:
If i is less than $s = \text{span}(Q)$, $t_Q(i, *, *)$ is always zero since a string shorter than the span can not contain a matching q-gram. Let then $i \geq s$ and consider the value $t_Q(i, j, M)$. If we want to compute it from $t_Q(i - 1, j^*, M^*)$ for some j^* and M^* we need to make three important observations. First, j^* is either equal to j or to $j - 1$ depending on whether there is a character match at position i, i.e. whether M contains $s - 1$. This is due to the fact that we only increase i by one and can therefore only introduce one additional mismatch. The second observation is that the threshold $t_Q(i, j, M)$ is either equal to $t_Q(i - 1, j^*, M^*)$ (if we do not introduce an additional matching q-gram) or equal to $t_Q(i - 1, j^*, M^*) + 1$ (if we do, i.e. if $Q \subseteq (M \cup \{0\})$). Furthermore the set M^* must agree with M on $s - 2$ positions, i.e. $M^* \cap \{2, \ldots, s - 1\} = (M \cap \{1, \ldots, s - 2\}) \oplus 1$. The reason for this is that M

overlaps M^* by $s-2$ characters. This leaves us only with the two possibilities given in the lemma. The use of the conditions combined with the minimization provides us with the correct choice. □

Given the recurrence from Lemma 4.2 and Lemma 4.1 we are now able to compute the optimal threshold. We achieve this by generating the values $t_Q(i,j,M)$ for all error values j below or equal to k, for all possible sets M describing the matches in the last $s-1$ positions and for $i \le m$. Out of all values $t_Q(m,k,*)$ we select the lowest threshold.

To speed up computation, all the values $t_Q(i,*,*)$ are stored in a single array. For each possible error value j between 0 and k it contains the threshold values $t_Q(i,j,M)$ for all possible sets M containing between 0 and j errors. The size of the array is therefore $f(k,s) = \sum_{j=0}^{k}\sum_{a=0}^{j}\binom{s-1}{a} = \sum_{a=0}^{k}\binom{s-1}{a}(k-a+1)$. The binomial term $\binom{s-1}{a}$ describes the number of possible placements of a errors in a string of length $s-1$.

Updating the array from $t_Q(i-1,*,*)$ to contain the new values for $t_Q(i,*,*)$ requires evaluating the formula given in Lemma 4.2 for every entry of the array. A single evaluation can be done in constant time since only two possibilities have to be considered. Thus the array update can be computed in time $O(f(k,s))$. In order to compute $t_Q(m,*,*)$ we have to update the array $O(m)$ times. We arrive at the following result:

Theorem 4.1. The optimal threshold $t_Q(m,k)$ can be computed in time $O(mf(k,s))$ and $O(f(k,s))$ space, where $s = \text{span}(Q)$ and $f(k,s) = \sum_{a=0}^{k}\binom{s-1}{a}(k-a+1)$.

Remark 4.1. For all positive k and s, $(k+1)2^{s-1}$ is an upper bound for $f(k,s)$. For $s > 2k$, $\left(\frac{s-k}{s-2k}\right)^2\binom{s-1}{k}$ can be a tighter bound.

The above dynamic programming algorithm is much faster than an exhaustive search but is still not fast enough to process all possible shapes. As shown in this paper, there are useful shapes with large size and span. Even with the faster algorithm it would be virtually impossible to compute the threshold for all shapes up to that size and span. To reduce the search space we used a pruning technique based on the following observation.

Lemma 4.3. If $Q' \subseteq Q$, then $t_{Q'}(m,k) \ge t_Q(m,k)$.

For each combination of q and s, we generated a set of *candidate* (q,s)-shapes from the set of $(q-1,s)$-shapes with positive threshold with the guarantee that every (q,s)-shape with positive threshold is among the candidates. To achieve this we first partition the $(q-1,s)$-shapes with positive threshold into sets containing shapes which only differ from each other in the second to last position. This means that for any two shapes $Q' = \{i_1,\ldots,i_{q-1}\}$ and $Q'' = \{j_1,\ldots,j_{q-1}\}$ in the same set, i_k equals j_k for all $k \in \{1,\ldots,q-3,q-1\}$. The following shapes could be in such a set for $q=5$ and $s=6$:

$$\{0,1,2,5\},\{0,1,3,5\},\{0,1,4,5\}$$

Given such a set of shapes we then generate a new shape

$$Q = Q' \cup Q'' = \{i_1,\ldots,i_{q-2},j_{q-2},i_{q-1}\}$$

for all pairs Q', Q'' from the set. The new shape Q has size q since it contains all entries from Q' plus an extra entry from Q''. Returning to our example we would generate the shapes

$$\{0, 1, 2, 3, 5\}, \{0, 1, 2, 4, 5\}, \{0, 1, 3, 4, 5\}$$

Since we can assume that we generated all $(q-1, s)$-shapes with a positive threshold and due to Lemma 4.3, all (q, s)-shapes with positive threshold have to be supersets of $(q-1, s)$-shapes with positive threshold. Pairing shapes which differ only in the second to last entry to produce new shapes is only one method of generating all (q, s)-shapes with positive t, however it sufficiently reduces the amount of shapes generated by this algorithm to make it practical for our purposes.

Finding the sets of $(q-1, s)$-shapes that differ only in the second to last position can be achieved quickly in a lexicographically sorted list.

Using this simple but effective pruning method in combination with our threshold computation algorithm allowed us to find all shapes with a positive threshold for $m = 50$ and $k \in \{4, 5\}$, a total of more than 25 million shapes.

5. Measuring Filter Algorithms

The two most important quantitative properties of a filter are its speed and filtration efficiency. The first determines the filtration time, the second the verification time. For a given filter, one can either measure these properties experimentally, or compute or approximate their expected values. In this paper, we employ both approaches. A fast way to compute the expected values was needed to select the best shapes out of millions of candidates, and experimental values are used for the final evaluation. In this section, we describe and motivate our methodology.

Let us start by describing the filter algorithm. We present our implementation in some detail and mention alternative approaches in some places.

We define a *hit* as a pair (i, j) such that the q-gram at the position i in the pattern P has an occurrence at the position j in the text T. Thus, a hit (i, j) indicates a shared q-gram between the pattern and the substring of length $|P|$ starting at $j - i + 1$ in T. A substring associated with at least t hits satisfies the filter criterion and is returned as a potential match. The filtration phase consists of two steps, finding the hits and finding the potential matches.

To find the hits, a q-gram index of the text T is used. Our implementation uses a lookup table and has size $O(|T| + |\Sigma|^q)$, where Σ is the alphabet. The second term, if too large, could be removed by using other techniques such as hashing, a trie or a suffix array. The index can be constructed in time $O(|T|q)$, and all hits for a pattern P can be found in time $O(|P|q + h)$, where h is the number of hits. For contiguous q-grams, the construction time can be reduced to $O(|T|)$ and the query time to $O(|P| + h)$, but this is not generally possible for gapped q-grams. The difference in query time is not significant in practice, because it is typically dominated by h, but the construction time could make a difference. However, in practice the difference is much smaller than a factor of q due to cache effects and hidden constants.

The lookup table provides fast access to all hits for all q-grams in P. The next step is to process them in order to find the potential matches. While there are many possibilities for solving this problem, we used an array of $|T| - |P| + 1$ counters, one for each possible substring of T with length $|P|$. The array is initialized just once, at the same time as the q-gram index is constructed. Then, for each pattern, the list of h hits is scanned, the counters are incremented, and potential matches are identified in $O(h)$ time. The counter array can also be restored to the initial state in $O(h)$ time by maintaining a list of all non-zero counters.

In the verification phase, one has to examine all the substrings of T that were reported as potential matches. In the worst case this means comparing all characters in P with the corresponding substring of T, requiring time $O(|P|)$. However, this is only necessary for actual matches since they contain at most k mismatches. A random potential match is likely to have much more mismatching characters allowing the algorithm to stop earlier when it detects the $(k+1)$st mismatch. In any case, the amount of time required for the verification phase is linear in the number of potential matches.

In summary, the filtration phase runs in time $O(|P|q + h)$, while the verification time is linear in the number of potential matches. Since $|P|q$ is rather small in practice, we use h, the number of hits, as a measure of the filter speed. The number of potential matches is the obvious choice for describing the filtration efficiency. For these reasons, the number of hits and the number of potential matches are the measures used in the experiments section. An important advantage of these parameters is their independence of any implementation details.

For random texts and some real world texts, e.g. DNA databases, the value of h can be predicted with reasonable accuracy using the simple formula

$$h \approx (|P| - s + 1)\frac{1}{|\Sigma|^q}(|T| - s + 1).$$

Thus the filter speed for a given shape is primarily determined by the size q of the shape. Predicting the number of potential matches is more difficult. However, in the next section we introduce a measure called the minimum coverage that allows reasonable predictions.

A final issue we want to mention is the difference between *global* and *local* matching. Instead of searching for a pattern P in a text T, one can attempt to locate pairs of approximately matching substrings from both P and T. This problem is known as the *substring matching with k mismatches*. A vital parameter of this problem is the minimum length m required for any match. This minimum length is crucial to avoid detecting a huge number of extremely short matches.

There are two simple ways to adapt a global matching algorithm to the local matching problem. A naive solution is to separately apply the global algorithm to all substrings of length m of the pattern P. The second alternative, which we used in our experiments, is to apply the global algorithm to P only once but using $t_Q(m, k)$ as the threshold, where m is not the length of P as it normally would be, but the desired length of the substrings. This of course increases the number of potential matches as more hits are generated and the probability of finding t common q-grams grows. However, it does not strongly affect the relative performance of different shapes.

6. Minimum Coverage

As explained in the previous section, we use the number of potential matches to represent the filtration efficiency of a filter. For a given shape Q, the number of potential matches clearly depends on the threshold $t_Q(m, k)$. However, the correlation is not as strong as one might expect. We have found a large number of instances where shapes with lower thresholds produce fewer potential matches. A simple example illustrating this phenomenon is shown below.

Example 6.1. Let $m = 13$ and $k = 3$. Then both shapes `###` and `##-#` have a threshold of two. If two strings have four consecutive matching characters, they have two common 3-grams of shape `###`. In contrast, to have two common 3-grams of shape `##-#` two strings need to have at least 5 matching characters.

Motivated by the example, we define the following measure.

Definition 6.1. Let Q be a shape and t a non-negative integer. The *minimum coverage* of Q for threshold t is

$$c(Q,t) = \min_{C \subseteq \mathbb{N}, |C|=t} | \cup_{i \in C} Q_i|.$$

The minimum coverage is, in essence, the minimum number of characters that need to match between a pattern and a text substring for there to be t matching Q-grams. This gives a reasonable first order estimator for the probability of random strings having t common Q-grams. While it neglects random matches with more than c matching characters, the fact that those with exactly c are the most likely to occur results in a strong correlation between the minimum coverage and the filter efficiency. We therefore conjecture that the expected number of potential matches is roughly proportional to

$$\frac{1}{|\Sigma|^{c(Q,t)}} .$$

There are additional factors that affect this relationship. For example, some shapes have several distinct minimum covers. In our experiments, we used the number of minimum covers as a secondary criterion for choosing between shapes with the same minimum coverage. We do not analyze this further in this paper, but to support our conjecture we plotted the predicted and actual values for the number of potential matches in Figure 2 in Section 7.

An interesting observation that helps explaining and understanding the superiority of gapped q-grams is the connection between the minimum coverage and the threshold of a shape. Given a contiguous shape and a threshold t, the minimum coverage of that shape will only increase by exactly one if t is incremented by one. For gapped shapes it is often the case that an increase of t by one will raise the minimum coverage by 2 or more.

Example 6.2. For $t = 1$, shapes `###` and `##-#` have a minimum coverage of 3. If we increase t to two, the contiguous shape has a minimum coverage of 4 and the gapped shape one of 5. A second contiguous q-gram can be overlapped with the first one by adding one extra matching character, but for the gapped case the second q-gram adds one extra character at the end of the q-gram and one in the gap.

We have implemented a straightforward branch-and-bound algorithm for computing the minimum coverage. For a shape with span s using threshold t, there are potentially $\binom{(s-1)(t-1)}{t-1}$ combinations to examine. However, the bound heuristic works very well and the minimum coverage computations we performed required much less CPU time than the threshold computations.

Using the algorithm we computed the minimum coverage of all shapes with a positive threshold for $m = 50$ and $k \in \{4, 5\}$. Tables 2 and 3 in the Appendix (pages 1019–1020) give the highest minimum coverage among (q, s)-shapes for all relevant combinations of q and s. The tables show that considering the minimum coverage further improves the advantage of the best gapped shapes over the contiguous shapes. There are even cases where the contiguous shape is the q-shape with the highest threshold but some gapped q-shapes have a higher minimum coverage. One such example would be for $k = 4$ and $q = 8$. In this case, the contiguous shape has a threshold of 11 but a minimum coverage of only 18, while the best overall shape (see Table 1) achieves a minimum coverage of 29 with a threshold of only 6.[3]

As with threshold values, the shapes with the highest minimum coverage are rare. Figure 1 shows the distribution of minimum coverage in one typical case.

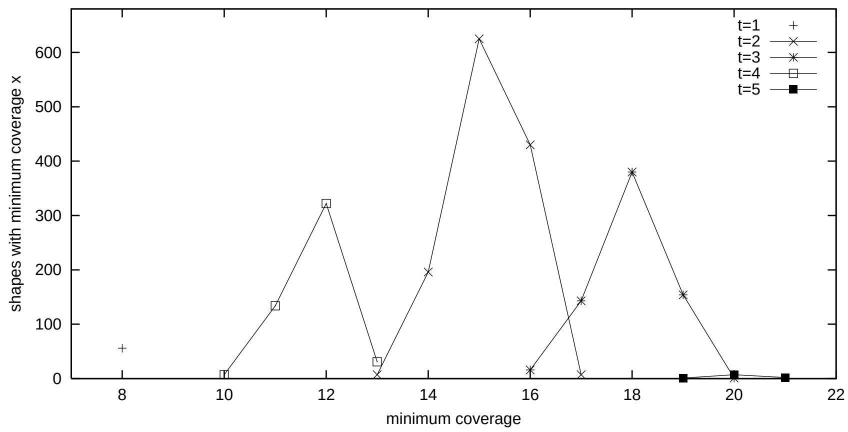


Figure 1. Distribution of threshold and minimum coverage of $(8, 17)$-grams for $m = 50$, $k = 5$. Mirror images and one shape with threshold 0 (`#######---------#`) are not included

7. Experiments

To test the q-grams in practice, we performed experiments on texts with a four-letter alphabet which is equivalent to DNA data. We used two different databases of size 50 million, one randomly generated (with even and independent distribution of characters), the other containing the first 50 million base-pairs of the GenBank Mouse EST database. The queries we used were random strings of length 500. However, the threshold used in filtering was computed for $m = 50$. As mentioned in Section 5 this will allow local matching while leading to a moderate increase in the number of potential matches. For the distance k we used the values 4 and 5. This experimental setting corresponds to high similarity local alignment problems found for example in shotgun sequencing [29] and EST clustering [12, 4]. No actual matches were found in the databases, i.e., all potential matches reported by the filter were false positives.

For randomly generated DNA as well as for actual DNA data, the expected number of hits is roughly proportional to $|\Sigma|^{-q} = 4^{-q}$. Our conjecture is that there is a similar dependence between the number of potential matches and the minimum coverage of the shape. We tested a large number of shapes using different values of k and the two databases described above. In Figure 2 the relation between the minimum coverage and the number of potential matches per query is depicted. In most cases, the

[3]The entry for $(8, 28)$-shapes in Table 2 shows a threshold 7, because there are $(8, 28)$-shapes with a threshold 7 but all of them have a minimum coverage of less than 29.

number of potential matches was averaged across 100 queries, although for some of the shapes, 1000 queries were used. It is clear that there is a strong but not stringent correlation of the form we expected.

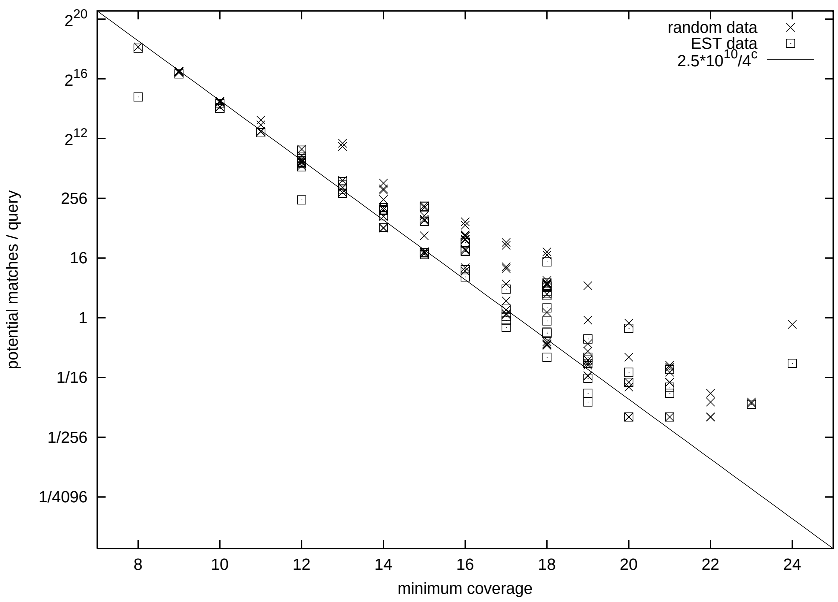


Figure 2. Correlation between expected and actual number of potential matches

For a more detailed comparison of shapes, we chose three classes of shapes for the different values of q and k:

- *Best.* This is the shape with the highest minimum coverage. To choose between multiple shapes with the same coverage we used the number of distinct covers of the minimum size, the number of distinct covers of the minimum size plus one, and the threshold as secondary keys (in this order).

- *Median.* For each span s, all (q, s)-shapes (without mirror images but including those with threshold 0) were ordered by the minimum coverage, and by the same secondary keys as for the best shape, and the median shape in this order was identified. Of these shapes (one for each s) the best one was used in the experiments. The chance that a randomly chosen shape is better than this shape is at most half. This gives some indication of the filter performance when the shape is chosen randomly without even a threshold computation as is done in [7], for example. Note that for $s = q$ there is only one shape, the contiguous q-gram. For very small q this shape is better than median shapes with a higher span and is selected as 'the' median shape.

- *Contiguous.* The classic q-grams, i.e. contiguous substrings of length q.

The chosen shapes except the contiguous ones are shown in Table 1. A missing shape means that the shape in that category had a threshold of 0. This also implies that for this value of q there is no experiment with this class of shapes. For higher values of q it is quite common that more than half of

Table 1. The shapes used in Figures 3 and 4.

$k = 4$		
q	best	median
7	##--#--------#---------##--#	###-#-----#--#--#
8	##--#------#-------#----##-#	#-#--#-#--##---#-#
9	##--#-#--------#------##--#-#	####-#-#------###
10	##--#-#-----#------#--##--#-#	###-###--#-----###
11	###-#--#------#------#-#-#--##	######-###-#-#
12	##-#-#---##---------#-##-#---##	##-##-####-###-#
13	#########-####	
14	#####-##---#####-##	
$k = 5$		
q	best	median
6	##------#--#--#-#	#-###-----#-#
7	#-##------##--#-#	##--#--#--##-#
8	##--#-#---------------##--#-#	#-#--####---#-#
9	###--#--#-#---#-##	######--#-#-#
10	###--#--#-#--###-#	##-##--#-#-###-#
11	#######-##-##	
12	###-#--###-#--###-#	

all possible shapes have a threshold of 0. In this case the median shape also has a threshold of 0 and no median data point is shown in the plots displaying our experimental results.

Figure 3 compares the chosen shapes both in theory (q vs. minimum coverage) and in practice (hits vs. potential matches). The experimental results are the averages from 1000 queries run against the random DNA database. For $k = 4$ we plotted our results for the EST database in Figure 4. In all our experiments the results between random and authentic DNA differed only slightly. The missing data points in the experimental graphs had no potential matches at all.

There are several things of interest in Figures 3 and 4. First, there is a good correlation between the predicted behavior and the actual experimental results. In most cases the correlation is very good, only for $k = 5$ and $q = 8$ as well as $k = 4$ and $q = 11$ we see a moderate difference between the two values. A possible reason for this lies in the low absolute number of potential matches (we only recorded 12 respectively 2 potential matches for all 1000 queries). Also, for $k = 4$ and $q \leq 10$ the algorithm using the best gapped shapes did not detect any potential matches. This correlates well with the expected value which was below one in all these cases.

Furthermore, the performance of the median shapes illustrates that, while a randomly chosen shape is likely to be better than a contiguous one, much better results can be achieved by a careful choice of the shape. In many cases there are huge differences between the best shape and the median shape. For very high values of q it is quite unlikely that a randomly picked shape is one of those with a positive threshold.

Finally we would like to point out that given a fixed value of q the best gapped shapes can reduce the amount of potential matches by several orders of magnitude. It is also worth noting that due to the higher values of q that are possible with gapped shapes, the number of hits for a query can be reduced by

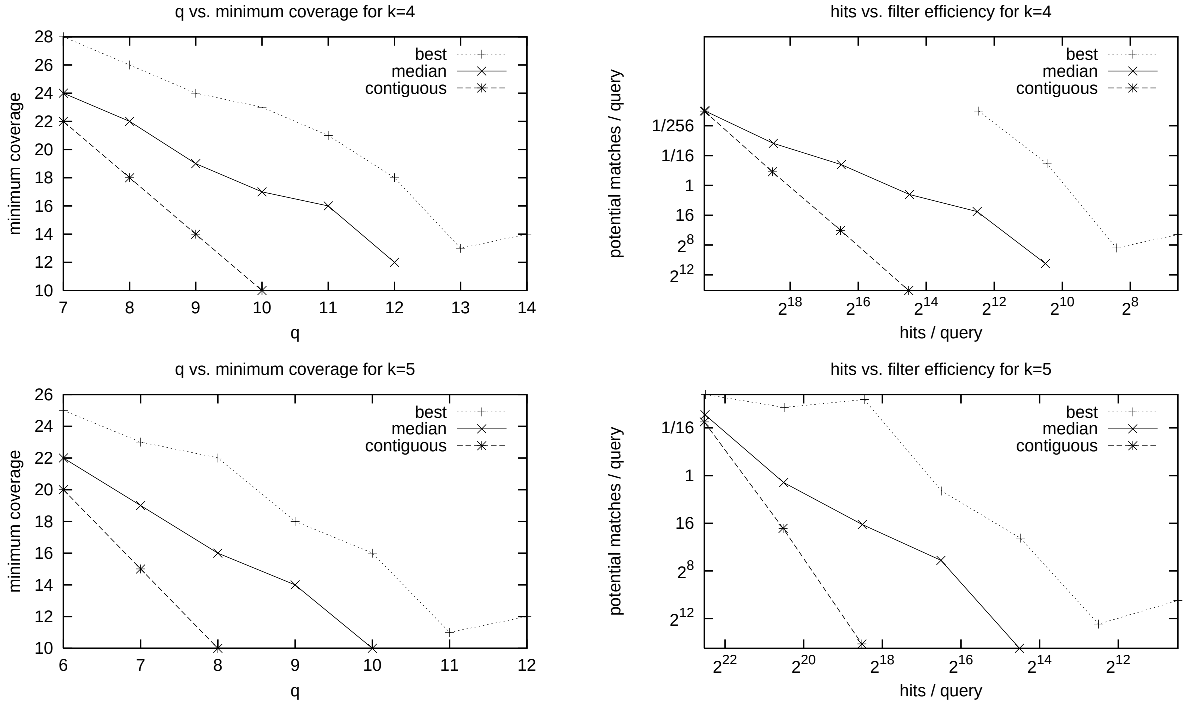


Figure 3. Comparison of three classes of shapes, random DNA

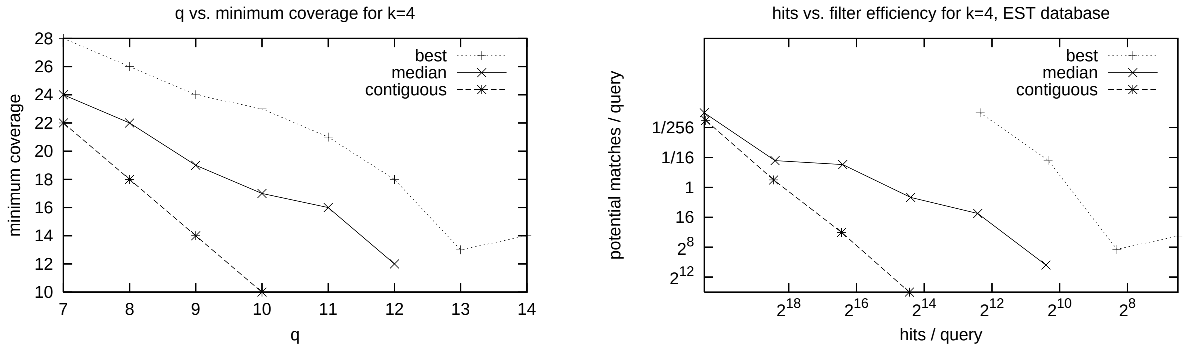


Figure 4. Comparison of three classes of shapes, EST DNA

a factor of roughly 2^8 for both $k = 4$ and $k = 5$.

8. Concluding Remarks

We have shown that suitably chosen gapped q-grams can significantly improve the performance of the basic q-gram filter for the k mismatches problem. While interesting in itself, it also opens the door to the possibilities of gapped q-grams in numerous other algorithms and applications where contiguous q-grams and related methods have been found to be useful. We have taken some first steps in studying these possibilities [6, 11], but a lot more remains to be done.

There are several variants of the basic q-gram filter, including sampling (use only a subset of all q-grams) [8, 15, 17, 19, 24, 25, 26, 27], approximate q-grams (allow errors in the q-grams) [8, 15, 19, 25], and multiple shapes [7, 20]. A generic technique for computing thresholds applicable to these and other variants of q-gram filters is described in [11]. This technique is applicable to the Levenshtein distance as well as the Hamming distance, and has already been used for computing the thresholds for an extension of gapped q-gram filters to the k differences problem [6].

Unfortunately, for many variants the threshold computation becomes much slower and/or the number of possible filters explodes. Thus, an exhaustive search of all possibilities is often impractical, and more direct methods for finding good filters, perhaps based on combinatorial properties of shapes, are needed. This remains an important open problem.

An additional avenue of study is the use of gapped q-grams in *lossy filters*. Several popular approximate string matching algorithms use lossy methods to increase their speed [1, 7]. Results from a set of experiments we conducted to investigate this approach show definite promise [3, pages 95–116] as do those in [14, 2]. However, the realm of lossy filters in general remains largely an uncharted territory.

It appears likely that other applications based on substring matching could benefit from using gapped q-grams, perhaps even in areas beyond string matching. One example is DNA sequencing by hybridization (SBH) where the problem is, in essence, to reconstruct a string given its q-grams. Preparata et al. [22, 23] have shown that SBH can be significantly improved by using gapped probes (q-grams) instead of contiguous ones.

Acknowledgements

We thank Kurt Mehlhorn for useful remarks.

References

[1] Altschul, S. F., Gish, W., Miller, W., Myers, E. W., Lipman, D. J.: Basic Local Alignment Search Tool, *Journal of Molecular Biology*, **215**, 1990, 403–410.

[2] Buhler, J., Keich, U., Sun, Y.: Designing Seeds for Similarity Search in Genomic DNA, *Proceedings of the 7th Annual International Conference on Computational Molecular Biology (RECOMB-03)*, April 2003, To appear.

[3] Burkhardt, S.: *Filter Algorithms for Approximate String Matching*, Ph.D. Thesis, Universität des Saarlandes, Germany, 2002.

[4] Burkhardt, S., Crauser, A., Ferragina, P., Lenhof, H.-P., Rivals, E., Vingron, M.: q-Gram Based Database Searching Using a Suffix Array, *Proceedings of the 3rd Annual International Conference on Computational Molecular Biology (RECOMB-99)*, ACM Press, 1999.

[5] Burkhardt, S., Kärkkäinen, J.: Better Filtering with Gapped q-Grams, *Proceedings of the 12th Annual Symposium on Combinatorial Pattern Matching*, number 2089 in LNCS, Springer, 2001.

[6] Burkhardt, S., Kärkkäinen, J.: One-Gapped q-Gram Filters for Levenshtein Distance, *Proceedings of the 13th Annual Symposium on Combinatorial Pattern Matching*, number 2373 in LNCS, Springer, 2002.

[7] Califano, A., Rigoutsos, I.: FLASH: A Fast Look-Up Algorithm for String Homology, *Proceedings of the First International Conference on Intelligent Systems for Molecular Biology*, AAAI Press, 1993.

[8] Chang, W. I., Marr, T. G.: Approximate String Matching and Local Similarity, *Proceedings of the 5th Annual Symposium on Combinatorial Pattern Matching*, number 807 in LNCS, Springer, 1994.

[9] Holsti, N., Sutinen, E.: Approximate String Matching Using q-Gram Places, *Proceedings of the 7th Finnish Symposium on Computer Science*, University of Joensuu, 1994.

[10] Jokinen, P., Ukkonen, E.: Two Algorithms for Approximate String Matching in Static Texts, *Proceedings of the 16th Symposium on Mathematical Foundations of Computer Science*, number 520 in LNCS, Springer, 1991.

[11] Kärkkäinen, J.: Computing the Threshold for q-Gram Filters, *Proceedings of the 8th Scandinavian Workshop on Algorithm Theory (SWAT)*, number 2368 in LNCS, Springer, 2002.

[12] Krause, A., Vingron, M.: A Set-Theoretic Approach to Database Searching and Clustering, *Bioinformatics*, **14**, 1998, 430–438.

[13] Lehtinen, O., Sutinen, E., Tarhio, J.: Experiments on Block Indexing, *Proceedings of the 3rd South American Workshop on String Processing (WSP'96)*, Carleton University Press, 1996.

[14] Ma, B., Tromp, J., Li, M.: PatternHunter: Faster and More Sensitive Homology Search, *Bioinformatics*, **18**, 2002, 440–445.

[15] Myers, E. W.: A Sublinear Algorithm for Approximate Keyword Searching, *Algorithmica*, **12**(4/5), 1994, 345–374.

[16] Navarro, G.: A Guided Tour to Approximate String Matching, *ACM Computing Surveys*, **33**(1), 2001, 31–88.

[17] Navarro, G., Baeza-Yates, R.: A Practical q-Gram Index for Text Retrieval Allowing Errors, *CLEI Electronic Journal*, **1**(2), 1998, `http://www.clei.cl`.

[18] Navarro, G., Baeza-Yates, R., Sutinen, E., Tarhio, J.: Indexing Methods for Approximate String Matching, *IEEE Data Engineering Bulletin*, **24**(4), 2001, 19–27, Special issue on Managing Text Natively and in DBMSs.

[19] Navarro, G., Sutinen, E., Tanninen, J., Tarhio, J.: Indexing Text with Approximate q-Grams, *Proceedings of the 11th Annual Symposium on Combinatorial Pattern Matching*, number 1848 in LNCS, Springer, 2000.

[20] Pevzner, P. A., Waterman, M. S.: Multiple Filtration and Approximate Pattern Matching, *Algorithmica*, **13**(1/2), 1995, 135–154.

[21] Picoult-Newberg, L., Ideker, T. E., Pohl, M. G., Taylor, S. L., Donaldson, M. A., Nickerson, D. A., Boyce-Jacino, M.: Mining SNPs From EST Databases, *Genome Research*, **9**(2), 1999, 167–174.

[22] Preparata, F. P., Frieze, A. M., Upfal, E.: On the Power of Universal Bases in Sequencing by Hybridization, *Proceedings of the 3rd Annual International Conference on Computational Molecular Biology (RECOMB-99)*, ACM Press, 1999.

[23] Preparata, F. P., Upfal, E.: Sequencing-by-Hybridization at the Information-Theory Bound: An Optimal Algorithm, *Proceedings of the 4th Annual International Conference on Computational Molecular Biology (RECOMB-00)*, ACM Press, 2000.

[24] Shi, F.: Fast Approximate String Matching with q-Blocks Sequences, *Proceedings of the 3rd South American Workshop on String Processing*, Carleton University Press, 1996.

[25] Sutinen, E., Tarhio, J.: On Using q-Gram Locations in Approximate String Matching, *Proceedings of the 3rd Annual European Symposium on Algorithms*, number 979 in LNCS, Springer, 1995.

[26] Sutinen, E., Tarhio, J.: Filtration with q-Samples in Approximate String Matching, *Proceedings of the 7th Annual Symposium on Combinatorial Pattern Matching*, number 1075 in LNCS, Springer, 1996.

[27] Takaoka, T.: Approximate Pattern Matching with Samples, *Proceedings of the 5th International Symposium on Algorithms and Computation*, number 834 in LNCS, Springer, 1994.

[28] Ukkonen, E.: Approximate string matching with q-grams and maximal matches, *Theoretical Computer Science*, **92**(1), 1992, 191–212.

[29] Weber, J., Myers, E.: Human Whole Genome Shotgun Sequencing, *Genome Research*, **7**, 1997, 401–409.

A. Threshold and Minimum Coverage Tables

The following pages contain the highest threshold and minimum coverage of (q, s)-shapes for $m = 50$ and $k \in \{4, 5\}$ and all relevant combinations of q and s. Given a value for q (x-axis) and the span s (y-axis), the table contains two numbers. The first is the highest threshold of any (q, s)-shape for the respective values of m and k, the second is the highest minimum coverage. For entries marked with ^{a}, the shapes with the highest threshold and the highest minimum coverage are different.

Table 2. The best thresholds/minimum coverages for $m = 50$ and $k = 4$.

	q												
s	2	3	4	5	6	7	8	9	10	11	12	13	14
2	41/42												
3	40/41	36/38											
4	39/40	35/38	31/34										
5	38/39	34/38	30/34	26/30									
6	37/38	33/38	29/34	25/30	21/26								
7	36/37	32/38	28/34	24/30	20/26	16/22							
8	35/36	31/38	27/34	23/30	19/26	15/22	11/18						
9	34/35	30/38	26/34	22/30	18/26	14/22	10/18	6/14					
10	33/34	29/38	25/34	21/30	17/26	13/22	9/18	5/14	1/10				
11	32/33	28/36	24/34	20/30	17/27	14/24	10/20	7/17	4/14	0/0			
12	31/32	27/35	23/34	20/31	17/28	13/24	10/21	8/19	5/16	2/13	0/0		
13	30/31	26/34	22/34	19/31	16/28	13/25	10/22	8/20	6/18	3/15	1/12	0/0	
14	29/30	25/33	21/34	18/31	15/28	12/25	10/23	8/21	5/18	4/17	2/15	1/13	0/0
15	28/29	24/32	20/34	17/31	14/28	12/26	9/23	7/21	5/19	3/17	2/16	1/13	0/0
16	27/28	23/31	19/34	16/31	14/29	11/26	9/24	7/22	5/20	3/18	2/16	1/13	0/0
17	26/27	22/30	18/33	16/31[a]	13/29	11/27	9/25	7/23	5/21	4/20	2/17	1/13	0/0
18	25/26	21/28	17/32	15/31	12/29	10/27	8/25	6/23	5/22	3/20	2/18	1/13	0/0
19	24/25	20/27	17/30[a]	14/30	12/29	9/27	7/25	6/24	4/22	3/20	2/18	1/13	1/14
20	23/24	19/26	16/29[a]	13/30	11/29[a]	9/28	7/26	5/24	4/23	3/21	2/18	1/13	0/0
21	22/23	18/25	15/29	12/29	10/28	8/27	6/26	5/24	3/20	2/18	2/17	1/13	0/0
22	21/22	17/24	15/28[a]	12/29[a]	9/29	7/26	6/24	4/22	3/21	2/18	1/12	1/13	0/0
23	20/21	17/24	14/28	12/28[a]	9/28	7/26[a]	5/25	4/22	2/17	2/17	1/12	0/0	0/0
24	19/20	17/24	15/26	12/29[a]	10/27[a]	7/27	5/25	4/20[a]	2/17	1/11	1/12	0/0	0/0
25	20/21	17/24	15/28[a]	12/29[a]	10/29[a]	7/26	5/25	4/23	3/20	2/17	1/12	0/0	0/0
26	21/22	17/24	16/29[a]	12/30	11/31[a]	8/28	6/27	4/24	3/22	2/18	1/12	1/13	0/0
27	20/21	16/23	16/29[a]	12/30	12/30[a]	8/27	7/28[a]	5/26	4/24	3/21	2/18	1/13	0/0
28	19/20	15/21	15/29	11/30	11/30[a]	8/29	7/29[a]	6/25[a]	4/25	3/22	2/19	1/13	1/14
29	18/19	14/20	14/28	10/28	10/30	8/27[a]	7/27[a]	5/27	4/25	3/22	2/19	1/13	1/14
30	17/18	13/19	13/26	9/25	9/30	7/27	6/27	4/25	4/24	3/23	2/19	1/13	1/14
31	16/17	12/18	12/25	9/24	9/28[a]	6/27	6/27[a]	4/25	4/24	2/19	2/19	1/13	1/14
32	15/16	11/17	11/23	9/24[a]	7/25	5/25	5/25	3/22	3/22	2/18	2/18	1/13	1/14
33	14/15	12/17[a]	10/20	8/24	8/24[a]	5/25	4/24	3/22	2/18	2/18	1/12	1/13	0/0
34	13/14	12/18	9/19	8/24	8/24[a]	5/24	4/23	4/21[a]	2/18	2/17	1/12	1/13	0/0
35	12/13	12/18	8/18	8/24	8/25[a]	4/22	4/23	4/21	2/17	2/17	2/16	1/13	1/14
36	11/12	11/17	7/16	7/22	7/24	4/22	4/21	4/21	2/17	2/17	2/16	1/13	1/14
37	10/11	10/15	8/16[a]	6/19	6/21	4/22	4/21	3/18	2/16	2/17	2/16	0/0	0/0
38	9/10	9/14	8/16[a]	5/15	5/20	4/22	4/19[a]	3/18	2/17	2/14	1/12	0/0	0/0
39	8/9	8/13	8/16[a]	5/14[a]	4/18	4/21	4/19[a]	2/16	2/17	2/14	1/12	0/0	0/0
40	7/8	7/12	7/16	5/14	3/15	3/18	3/19	2/15	2/12	1/11	1/12	0/0	0/0
41	6/7	6/10	6/15	6/14[a]	4/14[a]	2/13	2/14	2/14	1/10	1/11	1/12	1/13	0/0
42	5/6	5/9	5/13	5/14	4/14[a]	2/12	2/13	2/14	1/10	0/0	0/0	0/0	0/0
43	4/5	4/8	4/10	4/14	4/14	3/12[a]	2/9	1/9	1/10	0/0	0/0	0/0	0/0
44	3/4	3/6	3/9	3/12	3/14	3/12[a]	2/10	1/9	1/10	0/0	0/0	0/0	0/0
45	2/3	2/5	2/7	2/9	2/11	2/12	2/11	1/9	0/0	0/0	0/0	0/0	0/0
46	1/2	1/3	1/4	1/5	1/6	1/7	1/8	1/9	1/10	0/0	0/0	0/0	0/0

Table 3. The best thresholds/minimum coverages for $m = 50$ and $k = 5$.

	q										
s	2	3	4	5	6	7	8	9	10	11	12
2	39/40										
3	38/39	33/35									
4	37/38	32/35	27/30								
5	36/37	31/35	26/30	21/25							
6	35/36	30/35	25/30	20/25	15/20						
7	34/35	29/35	24/30	19/25	14/20	9/15					
8	33/34	28/35	23/30	18/25	13/20	8/15	3/10				
9	32/33	27/35	22/30	18/26	14/22	9/17	5/13	0/0			
10	31/32	26/34	21/30	18/27	13/22	10/19	6/15	3/12	0/0		
11	30/31	25/33	20/30	16/26	13/23	10/20	7/17	4/14	2/12	0/0	
12	29/30	24/32	19/30	16/27	12/23	9/20	7/18	4/15	2/13	0/0	0/0
13	28/29	23/31	19/30[a]	15/27	12/24	9/21	6/18	4/16	2/14	1/11	0/0
14	27/28	22/30	17/30	14/27	11/24	8/21	6/19	4/17	2/14	1/11	0/0
15	26/27	21/28	17/29[a]	13/27	10/24	8/22	5/19	3/17	2/15	1/11	0/0
16	25/26	20/27	16/29[a]	13/27	10/24	7/22	5/20	3/17	2/15	1/11	0/0
17	24/25	19/26	15/28	12/27	9/25	7/23	5/21	3/18	2/15	1/11	0/0
18	23/24	18/25	14/27	11/27	8/25	6/23	4/20	3/18	2/16	1/11	0/0
19	22/23	17/24	14/26[a]	11/26[a]	8/25	6/22	4/20	2/15	1/10	1/11	1/12
20	21/22	16/23	13/25[a]	10/24	7/24	5/22	3/18	2/15	1/10	1/11	0/0
21	20/21	15/21	12/25	9/23	7/22	5/21	3/18	2/15	1/10	0/0	0/0
22	19/20	15/21	12/23[a]	9/23	6/21	4/20	2/14	1/9	1/10	0/0	0/0
23	18/19	15/21	12/23[a]	9/23	6/21	4/19	2/14	1/9	0/0	0/0	0/0
24	18/19	15/21	13/24	9/23	7/20[a]	4/19	2/14	1/9	0/0	0/0	0/0
25	19/20	15/21	13/25[a]	9/23[a]	7/22	4/20	3/16	1/9	1/10	0/0	0/0
26	20/21	15/21	14/26[a]	9/24	8/24	5/23	3/19	2/15	1/10	0/0	0/0
27	19/20	14/20	14/26[a]	9/24	9/24[a]	6/23[a]	4/19	2/15	1/10	1/11	0/0
28	18/19	13/19	13/26	8/24	8/24	5/22	4/20	3/17	2/15	1/11	0/0
29	17/18	12/18	12/25	8/23	8/24[a]	5/22	5/22	2/15	2/15	1/11	0/0
30	16/17	11/17	11/23	7/21	6/21	4/20	4/20	2/15	2/15	1/11	0/0
31	15/16	10/15	10/20	7/21	7/20[a]	4/18	4/18[a]	2/14	2/14	1/11	0/0
32	14/15	10/15	9/19	7/19[a]	5/19	3/17	3/16	1/9	1/10	0/0	0/0
33	13/14	10/15[a]	8/18	6/19	6/19[a]	3/17	2/14	1/9	1/10	1/11	0/0
34	12/13	11/17	7/16	6/19	6/20[a]	3/17	3/14	2/12	1/10	1/11	0/0
35	11/12	11/17	6/15	6/19	6/20[a]	3/16	3/14	2/14	1/10	1/11	0/0
36	10/11	10/15	6/13[a]	5/15	5/20	2/13	2/13	2/14	1/10	0/0	0/0
37	9/10	9/14	7/15[a]	4/14	4/18	3/13[a]	2/13	2/13	1/10	1/11	1/12
38	8/9	8/13	7/15[a]	4/12[a]	3/15	2/13	2/12	1/9	1/10	1/11	0/0
39	7/8	7/12	7/15[a]	4/12	3/12	2/12	2/12	1/9	1/10	1/11	0/0
40	6/7	6/10	6/15	4/13	2/10	2/11	1/8	1/9	1/10	0/0	0/0
41	5/6	5/9	5/13	5/13[a]	3/10[a]	1/7	1/8	1/9	0/0	0/0	0/0
42	4/5	4/8	4/10	4/13	3/10	1/7	1/8	1/9	0/0	0/0	0/0
43	3/4	3/6	3/9	3/12	3/12	2/8	1/8	0/0	0/0	0/0	0/0
44	2/3	2/5	2/7	2/9	2/10	2/9	1/8	0/0	0/0	0/0	0/0
45	1/2	1/3	1/4	1/5	1/6	1/7	1/8	0/0	0/0	0/0	0/0