On Using q-Gram Locations in Approximate String Matching*

Erkki Sutinen and Jorma Tarhio

Department of Computer Science
P.O. Box 26 (Teollisuuskatu 23)
FIN-00014 University of Helsinki, Finland
E-mail: {sutinen,tarhio}@cs.Helsinki.FI

Abstract. Approximate string matching with k differences is considered. Filtration of the text is a widely adopted technique to reduce the text area processed by dynamic programming. A sublinear filtration algorithm is presented. The method is based on the locations of the q-grams in the pattern. Samples of q-grams are drawn from the text at fixed periods, and only if consecutive samples appear in the pattern approximately in the same configuration, the text area is examined with dynamic programming. Practical experiments show that this approach gives better filtration efficiency than an earlier method.

1 Introduction

Background. Efficient solutions for approximate string matching are useful in many areas, such as molecular biology, text databases, and data communications. We will consider the *k differences problem*, a version of the approximate string matching problem. Given integer k and two strings, *text* $T = T[1 \ldots n]$ and *pattern* $P = P[1 \ldots m]$ over some alphabet Σ of size c, the task is to find (the end points of) all approximate occurrences of P in T. An approximate occurrence signifies substring p of T such that at most k editing operations (insertions, deletions, changes) are needed to convert p to P, in other words *edit distance* $d(P, p)$ is at most k.

There are numerous algorithms proposed for this problem. A natural solution is a modification of dynamic programming. This approach leads to $O(nk)$ algorithms [Ukk85, GaP89]. Because processing of all the text positions with dynamic programming is rather slow, many filtering techniques [Ukk92, TaU93, PeW93, ChL94, ChM94] have been proposed to reduce the text area necessary to examine using dynamic programming. Some of these approaches lead even to algorithms which are sublinear on the average.

It is typical for the k differences problem that none of the solutions is the best for every combination of problem parameters m, k, and c [JTU91]. Especially large k is troublesome for small alphabets. The key factor is $f = \frac{n-n_p}{n}$, the efficiency of the filtration phase, where n_p is the number of text positions left

* The work was supported by the Academy of Finland.

for the dynamic programming phase. Good filtration efficiency is crucial for the practical speed of an algorithm. Besides saving checking time, filtration also consumes resources, which should be taken into account in designing efficient filtration techniques.

One way to reduce n_p is to develop necessary conditions for a text area to include an approximate match of the pattern [GrL89, Ukk92, Tak94]. These conditions often deal with q-grams of the pattern, i.e. continuous substrings of length q. The idea is that whenever an approximate match occurs, it has to resemble the original pattern. This resemblance is reflected by the existence of the *same* q-grams both in the pattern and in its approximate match.

Takaoka [Tak94] presents an efficient filtration technique based on sampling. In his method every hth q-gram of the text is drawn as a sample. If a sample appears in the pattern, a neighborhood of the sample is examined using dynamic programming. Takaoka's method is a simplification of the Chang-Lawler algorithm [ChL94].

Sketch of the solution. Besides the condition for the number of common q-grams in the pattern and its approximate match, one may also utilize the fact that the preserved q-grams have to be *approximately at the same locations* both in the pattern and in its approximate match. We will present a new sublinear filtration technique based on a sampling scheme similar to Takaoka's approach and on approximate locations of the q-grams in the pattern.

An approximate location of a q-gram in the pattern is defined by dividing the pattern into blocks using sampling step $h \geq q$. Let P_0 be $(k+2)h$ characters long prefix of the pattern. We cut P_0 into $k+2$ blocks of h positions and extend each block with $k+q-1$ positions to the right. Then two consecutive blocks have an overlap of $k+q-1$ positions.

In the text, we examine every hth q-gram as a sample. Let $d_1, d_2, \ldots$ be the samples. Because $h \geq q$, the samples do not overlap. We will show that a necessary condition for an approximate match is that at least two of the $k+2$ consecutive samples $d_{j-k-1}, \ldots, d_j$ match. In other words, $d_{(j-k-2)+i} \in Q_i$ holds for at least two indices i, $1 \leq i \leq k+2$, where Q_i is the set of the q-grams of the ith block of the pattern. We will also consider a more general case, where we require that at least s of the $k+s$ consecutive samples match.

In Fig. 1 there is an example, where $m = 40, k = 2, q = 3$, and $h = 9$. Samples have been boxed. We have $d_2 \in Q_1, d_3 \in Q_2$, and $d_4 \in Q_3$ and the count of positive samples is three at $d_5 = \texttt{PQS}$ so that there is a potential approximate occurrence of the pattern.

We use the shift-add approach [BaG92] to compute the sum of matches for the $k+2$ consecutive samples. By doing this, we actually reduce the k differences problem to a variation of the k mismatches problem, where each position of the pattern has a set of accepted characters of its own. In our approach, the "transformed" pattern contains $k+2$ positions (i.e. samples) and each of them has a set of accepted q-grams. A similar transformation is applied to single characters in [TaU93].

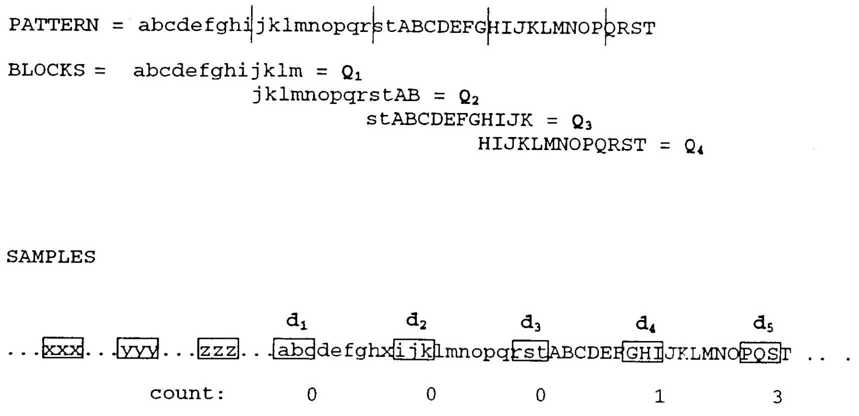

PATTERN = abcdefghi|jklmnopqr|stABCDEFG|HIJKLMNOP|QRST
BLOCKS = abcdefghijklm = Q1
jklmnopqrstAB = Q2
stABCDEFGHIJK = Q3
HIJKLMNOPQRST = Q4
SAMPLES
d1 d2 d3 d4 d5
...xxx...yyy...zzz...abcdefghxijklmnopqrstABCDEFGHIJKLMNOPQST ...
count: 0 0 0 1 3

Fig. 1. *Example of sampling.*

In earlier studies, only Holsti and Sutinen [HoS94] use the locations of the q-grams, but they consider only static texts and the details of their method are different.

Results. Let us assume that individual characters in P and T are chosen randomly. We will show that the asymptotic bound for the filtration efficiency of our method is $\Omega(1 - \frac{m+k^2}{c^q})$. The average time complexity is $O(\frac{kn}{m}\log_c m)$ for small values of k when $q = \log_c m$.

We carried out some experiments and compared our method with Takaoka's method which is among the best in practice. The filtration efficiency of our method was much better for a large range of problem parameters. For example, the number of text positions our algorithm processes with dynamic programming is less than $\frac{1}{50}$ of the corresponding number for Takaoka's method in the case of $c = 40$, $m = 40$, and $k = 8$.

Outline. The rest of this paper is organized as follows. In Section 2, we start with our sampling theorems, stating the necessary conditions for an approximate match of the pattern in a text area in terms of occurrences of q-grams. We present our algorithm in Section 3, and analyze it in Section 4. In Section 5 we review our preliminary experiments before giving concluding remarks in Section 6.

2 Sampling Based on q-Grams

Let $k \geq 0, q \geq 1$, and $s \geq 1$ be integers. In the text, every hth q-gram is examined as a sample. We call these samples *q-samples.* Distance h between the endpoints of two consecutive q-samples is the *sampling step*. Let $d_1, \ldots, d_{\lfloor n/h \rfloor}$ be the q-samples of the text. Let us assume that d_1 ends at position h.

Let us consider what the maximal value of h could be for $s = 2$. Let $p = T[j_1 \ldots j_2]$ be an approximate match of pattern P, i.e. $d(P, p) \leq k$. Since k

deletions produce the narrowest approximate match such that p is $m-k$ positions wide, p must include at least $m-k-q+1$ q-grams. We require that p includes $k+2$ non-overlapping substrings of equal size h. These conditions lead to the following bound:

$$h \leq \frac{m-k-q+1}{k+2}.$$

The basis for sampling is in Theorem 1.

Theorem 1. *Let p be a substring of T such that $d(P,p) \leq k$. If*

$$h = \lfloor \frac{m-k-q+1}{k+s} \rfloor$$

is the sampling step, $h \geq q$, then at least s of the q-samples in p occur in P.

Proof. Let p be $T[i \dots j]$. Let r be the number of q-samples in p. To estimate the lower bound for r, let us consider the situation where the leftmost q-sample of p starts as right as possible; in this case the leftmost q-sample in p starts at $(i-1)+h$. Since the rightmost possible q-sample in p starts at position $j-q+1$ and the sampling step is h (see Fig. 2), we get the inequality for the starting position of the rth q-sample in p:

$$(i-1)+rh > j-q+1-h.$$

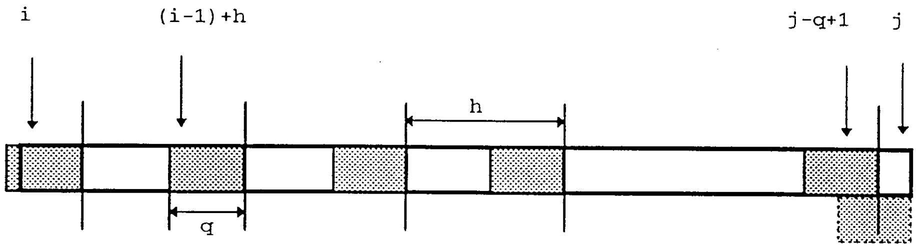


Fig. 2. *Locations of q-samples in $p = T[i \dots j]$.*

Since $|p| = j-(i-1)$, this equals

$$(r+1)h > |p| - q + 1,$$

from which we get

$$r > \frac{|p|-q+1}{h} - 1.$$

Since $h \leq \frac{m-k-q+1}{k+s}$ and $|p| \geq m-k$, we get

$$r > \frac{|p|-q+1}{m-k-q+1}(k+s) - 1 \geq k+s-1. \quad (1)$$

Let t be the number of the q-samples in p which are q-grams of P. We make an antithesis: $t \leq s-1$.

According to inequality (1), p includes $r > k+s-1$ q-samples. Now $r-t$ q-samples in p do not occur in P. Because

$$r-t > k+s-1-t \geq k+s-1-(s-1) = k$$

and because the q-samples do not overlap ($h \geq q$), p includes more than k differences with P. Thus $d(P,p) > k$ holds, which is a contradiction. □

Also Takaoka's method [Tak94] is based on Theorem 1, and a similar idea is also presented by Wu and Manber [WuM92]. In Takaoka's method, if a q-sample occurs anywhere in the pattern, the neighborhood of this sample is checked with dynamic programming. Thus Takaoka considers only the case $s = 1$. Takaoka presents a similar theorem without a proof for a *fixed* position of P.

Our approach is different, because our algorithm utilizes the locations of the q-grams in P. The algorithm examines $k+s$ consecutive q-samples together. Let p be an approximate match of P in T. Let us assume that h has been selected according to Theorem 1 so that there are at least $k+s$ consecutive q-samples in p. It turns out that then at least s of those samples must exist in the pattern, and these samples must have the same relative locations in both the pattern and the text. This requirement is stronger than the condition of Theorem 1.

We select $r = k+s$ fixed blocks from the pattern:

$$P[1\ldots h+d], P[h+1\ldots 2h+d], \ldots, P[(r-1)h+1\ldots rh+d],$$

where $d = k+q-1$. Two consecutive blocks have an overlap of $k+q-1$ positions and each block contains $h+k$ q-grams and $h+k+q-1$ characters.

The basis for the width of a block is sampling step h. In order to be able to handle q-grams, each block is extended $q-1$ positions to the right. In an approximate occurrence of P, the maximal difference of shifts of two q-grams is k positions and so each block is extended still k positions to the right.

Note that the last $m-rh-d$ q-grams of P do not necessarily occur in any block, when $rh+d < m$. This is an advantage in filtration.

Let Q_i denote the set of the q-grams of the ith block. Our approach is based on the following theorem.

Theorem 2. *Let h be as in Theorem 1. Let $p = T[i\ldots j]$ be an approximate match of P, i.e. $d(P,p) \leq k$. Then for any sequence of $k+s$ consecutive q-samples $d_{b+1}, \ldots, d_{b+k+s}$ included by p, there is integer t such that $d_{b+i+t} \in Q_i$ holds for at least s of the samples.*

Sketch of the proof. Let p include r q-samples. Theorem 1 implies that $r \geq k+s$. Let us consider an arbitrary sequence of $k+s$ consecutive q-samples in p. We know that at least s of the samples occur in P, because otherwise $d(P,p)$ would be greater than k. Let these s samples be $R = \{d_{b_1}, \ldots, d_{b_s}\}$ and let $e_1, \ldots, e_s$ be the end positions of their occurrences in P.

Let us align the pattern with the text according to d_{b_1}. Let $S(i) = e_i - e_1 - (b_i - b_1) * h$ be the shift of d_{b_i} in P. Let i_{min} and i_{max} be the indices of the samples in R with which $S(i)$ get its minimum and maximum, respectively. The definitions imply $S(1) = 0$ and $S(i_{min}) \leq 0 \leq S(i_{max})$.

Clearly it is possible to select R and the corresponding occurrences in P in such a way that $S(i_{max}) - S(i_{min}) \leq k$ holds, because otherwise $d(P, p)$ would be greater than k.

Let us denote the start and end positions of block Q_j by c_j and g_j, respectively. Let d_{b_1} occur in $Q_a = P[c_a \ldots g_a]$. To complete the proof, it is sufficient to show that $c_x \leq e_1 + S(i_{min}) - q + 1$ and $e_1 + S(i_{max}) \leq g_x$ hold for $x = a - 1$, a, or $a + 1$. Then $d_{b_i} \in Q_{x+(b_i - b_1)}$ is clearly satisfied for every $d_{b_i} \in R$, which means that the value of t is $b_1 - b - x$.

There are three cases to consider.

(i) Let us assume that $g_a - e_1 \geq k$ and $e_1 - (c_a + q - 1) \geq k$. Now both $c_a \leq e_1 + S(i_{min}) - q + 1$ and $e_1 + S(i_{max}) \leq g_a$ are clearly satisfied.

(ii) Let us then consider the case $g_a - e_1 < k$. If $g_a - (e_1 + S(i_{min})) \geq k$ holds, both $c_a \leq e_1 + S(i_{min}) - q + 1$ and $e_1 + S(i_{max}) \leq g_a$ are satisfied. If $g_a - (e_1 + S(i_{min})) < k$, then $c_{a+1} \leq e_1 + S(i_{min}) - q + 1$ and $e_1 + S(i_{max}) \leq g_{a+1}$ hold.

(iii) The case $e_1 - (c_a + q - 1) < k$ is symmetric with case (ii). □

The bounds for the location of an approximate match are determined by the following theorem, when we have found enough matching q-samples.

Theorem 3. *Let us assume that s of $k+s$ consecutive q-samples $d_{b+1}, \ldots, d_{b+k+s}$ satisfy $d_{b+i} \in Q_i$ where q-sample d_{b+k+s} ends at text position j. Then an approximate occurrence of the pattern is located in text area*

$$T[j - (k + s)h - 2k - q + 2 \ldots j + m - (k + s - 1)h + k - q].$$

The width of the text area is $m + 3k + h - 1$.

Proof. Let $d_{b+t} = u$ be one of the q-samples satisfying $d_{b+t} \in Q_t$, $1 \leq t \leq k + s$. Let d_{b+t} end at position j'. We set $\Delta = j - j' = (k + s - t)h$.

Let us first study end position j_R of an approximate occurrence of P. We consider the case when j_R reaches its maximum value. This happens when q-gram u occurs at the leftmost possible position in block Q_t, i.e. the end position of u in P is $i_L = (t - 1)h + q$ (see Fig. 3). The length of the suffix of P to the right of i_L is trivially $m_R = m - i_L$.

Since we allow k differences between P and its approximate match in T, the approximate match cannot reach more than $m_R + k$ positions over j'. This means that

$$\begin{aligned} j_R &= j' + m_R + k \\ &= j - \Delta + m - i_L + k \\ &= j - (k + s - t)h + m - (t - 1)h - q + k \\ &= j + m - (k + s - 1)h + k - q. \end{aligned}$$

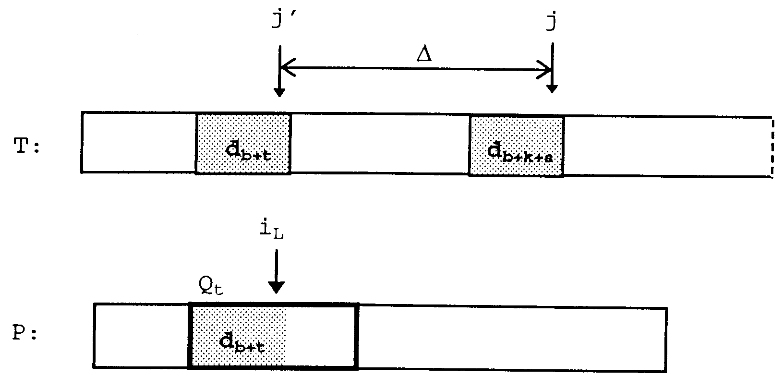


Fig. 3. *Locations of d_{b+t} in T and P.*

To find out the leftmost possible starting position j_L for the approximate match, we examine the case when j_L reaches its minimum. Now, u occurs at the rightmost position inside block Q_t, that is, it ends at position $i_R = th+q-1+k$.

For the same reason as in the case above, the approximate match in text T cannot start before

$$\begin{aligned} j_L &= j' - (i_R - 1) - k \\ &= j - \Delta - th - q + 1 - k + 1 - k \\ &= j - (k+s-t)h - th - 2k - q + 2 \\ &= j - (k+s)h - 2k - q + 2. \end{aligned}$$

The width of the text area is $m + 3k + h - 1$. □

3 Algorithm

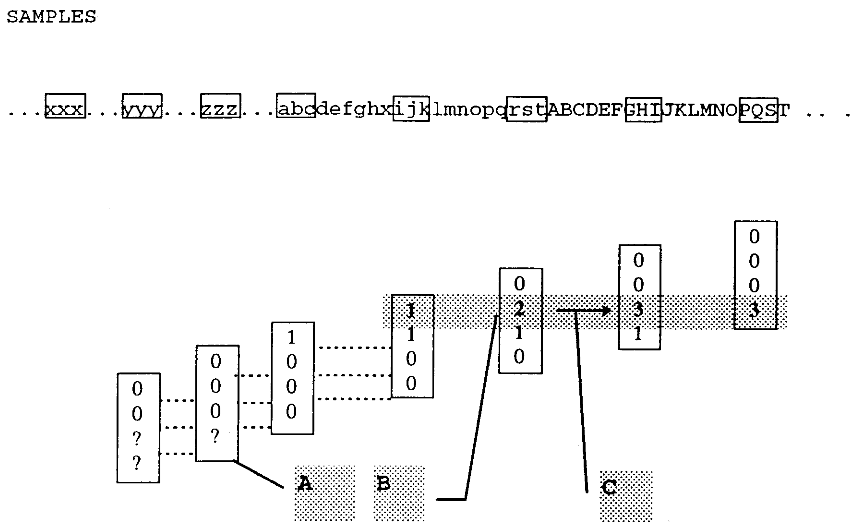
We will reduce the k differences problem to a generalized k mismatches problem, where each position of the pattern has a set of accepted characters of its own. We consider q-grams as the alphabet, q-samples as text $T' = d_1 \ldots d_{n'}$, and blocks of the original pattern as pattern $P' = Q_1 \ldots Q_{m'}$, where $n' = \lfloor n/h \rfloor$ and $m' = k + s$. A approximate match of P' with at most k mismatches ends at j, if $T'[j - m' + i] \in P[i]$, that is $d_{j-m'+i} \in Q_i$, holds for at least $m' - k$ indices i, $1 \le i \le m'$.

The transformed problem can be efficiently solved using the shift-add technique [BaG92]. We define bit matrix B as follows: $B[d,j] = 1$, if q-gram d belongs to Q_j, otherwise $B[d,j] = 0$. For each q-gram d, $B[d,*]$ gives the block profile of d.

Array $M[1 \ldots m']$ is used to compute the number of matching q-samples in an alignment of the pattern $T'[i \ldots i + m' - 1]$. An approximate match with at most k mismatches is found when $M[m'] \ge m' - k = s$. Initially, M consists of 0's. Array M is updated at each text position as follows (see also Fig. 4):

```
for $j := m'$ downto 2 do $M[j] := M[j-1]$;
for $j := 1$ to $m'$ do $M[j] := M[j] + B[d, j]$;
```

In practice, the next value of M is evaluated using bit parallel operations. Implementation details are discussed in the end of this section.



SAMPLES
. . . xxx . . yyy . . . zzz . . . abcdefghxijklmnopqrstABCDEFGHIJKLMNOPQST . . .
A
B
C

Fig. 4. *Computing the number of matching q-samples: (A) The three first elements of array M are zero, because none of the samples 'xxx', 'yyy', and 'zzz' starts an approximate match. (B) $M[2] = 2$, because samples 'ijk' and 'rst' belong to blocks Q_1 and Q_2. (C) The value of $M[2]$ is shifted to $M[3]$, corresponding to the new phase. Since sample 'GHI' belongs to block Q_3, $M[3] := M[2] + 1$.*

We are now ready to present our algorithm for approximate string matching.

Algorithm A.

```
1. preprocess $P$;
2. for $i := 1$ to $m'$ do $M[i] := 0$;
3. for $j := h$ to $n$ step $h$ do
4. begin
5.     $d := T[j - q + 1 \ldots j]$;
6.     Shift_add($M, B[d, *]$);
7.     if $M[m'] \geq m' - k$ then
8.         $DP(T[j - m'h - 2k - q + 2 \ldots j + m - (m' - 1)h + k - q], P)$;
9. end
```

In Algorithm A, procedure DP searches for approximate matches of pattern P in text area $T[i_1 \ldots i_2]$. This procedure evaluates edit distance matrix

$d[0\ldots(i_2 - i_1 + 1), 0\ldots m]$ using dynamic programming, with initial values $d[i,j] = 0$ for $j = 0$ and $d[i,j] = j$ for $i = 0$:

$$d[i,j] = \min \begin{cases} d[i-1,j-1] + \delta_{T[i_1+i-1]=P[j]} \\ d[i-1,j]+1 \\ d[i,j-1]+1. \end{cases}$$

Above,

$$\delta_{a=b} = \begin{cases} 0 \text{ if a=b} \\ 1 \text{ otherwise.} \end{cases}$$

The necessary and sufficient condition for an approximate match of P ending at text position i is $d[i - i_1 + 1, m] \leq k$.

A useful heuristic. Let us assume that Algorithm A has found a potential approximate match ending at text position j. Instead of checking this potential match directly with dynamic programming, we backtrack $(m' - 1)h + \lfloor \frac{h}{2} \rfloor$ positions in the text and restart the search with new q-samples. The restarting is permitted only if $j - j_p$ is large enough, where j_p is the previous backtracking position, otherwise checking phase DP is called. This heuristic works well in practice.

Implementation details. Index Q tells for each q-gram of P the blocks containing that q-gram. The index is constructed during the preprocessing of the pattern.

Let us consider the case $s = 2$. Because the sufficient number of positive q-samples in a text area is then two, only two bits are needed for an element of M. To calculate efficiently the next value for M, we use the shift-add technique. Thus, two bits are also reserved for each block in an element of index Q.

4 Analysis

To analyze the efficiency of Algorithm A, it is essential to estimate its filtration efficiency f_A. The filtration efficiency equals to the matching probability of the mismatch problem.

In the following, we concentrate on the case $s = 2$. We assume that the texts and patterns are generated according to the i.i.d. model, i.e., characters are independently and identically distributed.

We denote by P_c the probability that P' matches, i.e.

$$P_c = Pr(\text{at least two samples match in } T'[j\ldots j+m'-1]).$$

We define:

$$P_1(i) = Pr(d \in Q_i),$$
$$P_0(i) = Pr(d \notin Q_i) = 1 - P_1(i),$$

where d is a q-gram. Since $P_1(i) = P_1(1)$ and $P_0(i) = P_0(1)$ hold for each $i, 1 \leq i \leq k+2$, we define

$$P_1 = P_1(1), P_0 = P_0(1).$$

Using these definitions, we can reformulate P_c:

$$P_c = 1 - P_0^{k+2} - (k+2)P_1 P_0^{k+1}, \qquad (2)$$

because the number of configurations with exactly one match is $k+2$. Because a block includes at most $h+k$ different q-grams, we get an upper bound for P_1:

$$P_1 \leq \frac{h+k}{c^q}.$$

By applying the formula of h we get:

$$\begin{aligned} P_1 &\leq \frac{\frac{m-k-q+1}{k+2} + k}{c^q} \\ &\leq \frac{m-k-q+1+k^2+2k}{kc^q} \\ &\leq \frac{m+k^2+k}{kc^q}. \end{aligned}$$

By setting

$$p = 1 - \frac{m+k^2+k}{kc^q}$$

and noticing that $P_0 = 1 - P_1 \geq p$, we get a lower bound for $1 - P_c$:

$$1 - P_c \geq p^{k+2} + (k+2)P_1 p^{k+1} \geq p^{k+2}.$$

Since $1 - P_c$ is the same as filtration efficiency f_A, we have obtained the following estimate for f_A:

Theorem 4. *Filtration efficiency f_A of the Algorithm A for $s = 2$ is*

$$f_A \geq \left(1 - \frac{m+k^2+k}{kc^q}\right)^{k+2}.$$

The bound of Theorem 4 is rough, and better estimates should be based on formula (2).

Next we estimate the time complexity of our algorithm. Let us consider separately the four major phases of the algorithm:

1. Preprocessing of the pattern creates index Q, implemented as a hash table of size $m - q + 1$. The natural mapping of q-gram d to integer $v(d)$ of base c needs $O(q)$ time using Horner's rule. Hashing $v(d)$ and handling possible collisions can be made in constant time on the average. Since each subsequent q-gram can be processed in $O(1)$ time using its predecessor (cf. the Rabin-Karp algorithm [KaR87]), the total time for hashing all the q-grams of the pattern is $O(m)$. Storing the locations of a q-gram of the pattern involves evaluating $2k/h \leq 2k/q$ different blocks, and so the preprocessing time of the whole pattern is $O(mk/q)$.

2. Applying the shift-add approach needs $O(k/w)$ time for a q-sample, where w is the word size. A shift-add operation works in constant time for small values of k.
3. The processing of q-samples consists of two parts. Evaluating all the $\lceil n/h \rceil$ q-samples needs $O(nq/h)$ time, assuming that efficient hashing is applied. Since block profiles of q-samples are shift-added to M, the whole phase consumes $O(\frac{nqk^2}{mw})$ time, where we have used the approximation $h = \Omega(m/k)$, which holds for $k + q \leq \frac{1}{2}m + 1$ and $s = O(k)$. For small k, the time complexity is $O(nqk/m)$.
4. Dynamic programming is applied only to the filtered q-sample locations. Dynamic programming for a text of length n_0 requires $O(n_0 k)$ time. Based on Theorem 3 and Theorem 4, we conclude that the time complexity of this phase is $O(\frac{n}{h}(1-f_A)(m+h)k) = O(\frac{n}{h}\frac{m+k^2+k}{c^q}(m+h)k)$, where we have used the approximation

$$(1 - \frac{m+k^2+k}{kc^q})^{k+2} \leq 1 - (k+2)\frac{m+k^2+k}{kc^q}.$$

The time for the processing of the q-samples dominates over the other phases; therefore, we can collect our results to the following theorem:

Theorem 5. *Let w be the word size in bits. The average time complexity of Algorithm A for $s = 2$ is $O(\frac{k^2qn}{mw})$ in a general case and $O(\frac{n}{m}qk)$ for $k \leq w/2$.*

When $q = \log_c m$, the time complexity is $O(\frac{k^2n}{mw}\log_c m)$ or $O(\frac{kn}{m}\log_c m)$. The latter is the same as the average time complexity of the Chang-Lawler algorithm and Takaoka's algorithm.

5 Experimental Results

We have compared the filtration efficiency of our algorithm for $s = 2$ with that of Takaoka's algorithm.

The texts and patterns in the first test were generated according to the i.i.d. model, i.e., the characters are independently and identically distributed. Table 1 shows the results for alphabet size $c = 40$. Alg. A is the basic version of our method and Alg A' is augmented with the backtracking heuristic. The text is 500,000 characters long, pattern length m is 40, and error level k varies from 0 to 12, i.e., from 0% to 30% (where the relative error is defined as k/m). We have counted the number of columns (i.e. the total width of the area) processed in the dynamic programming phase to evaluate the filtration efficiency of the algorithms.

As the results show, all the three algorithms lose their filtration power at error levels over 30%. This is because a higher error level k means a lower sampling step h and, therefore, also smaller q. In particular, our algorithms reach $q = 1$ at error level 30% corresponding to $k = 12$ in Table 1 (Takaoka's algorithm reaches

Table 1. *Processed columns for* $c = 40$, $m = 40$, *and* $n = 500,000$.

k	Takaoka	Alg. A	Alg. A'
0	440	58	58
2	1,496	65	54
4	2,448	65	56
6	4,056	71	65
8	5,996	83	69
9	191,737	21,356	440
10	261,605	29,729	1,362
11	251,332	59,746	5,052
12	272,392	500,000	500,000

$q = 1$ at $k = 13$). A sharp increase in the number of columns is characteristic for these algorithms, when h and q approach to one.

The optimal error level for these algorithms seems to be at about 0–20%. For these error levels, Takaoka's method evaluates under 1.2% of the columns; our algorithms do not examine more than 0.02% of the text columns. For the error levels 22.5–27.5%, Algorithm A' still preserves its efficiency while evaluating at most 1% of the columns, while Takaoka's method evaluates 38–50% of the columns.

Our preliminary implementations show that the number of evaluated columns is reflected also in the execution time of the algorithms. The difference between Takaoka's algorithm and Algorithm A' is very clear at the error levels of 22.5–27.5%: our algorithm is about four times faster.

Table 2. *Processed columns for an English text,* $n = 492,459$.

m	k	Takaoka	Alg. A'
4	1	441,591	193,623
8	1	71,650	850
8	2	440,606	367,015
8	3	492,366	488,726
16	1	153,708	716
16	2	120,713	1,455
16	3	174,011	7,751
16	4	333,872	491,958
16	5	492,438	492,427

Tests with other alphabets demonstrate similar behavior. Results from another test with an English text are shown in Table 2.

6 Concluding Remarks

We have presented a sublinear filtration algorithm for approximate string matching with k differences. Our experiments show that the new approach gives considerably better filtration efficiency than Takaoka's algorithm. It is possible to apply the method also to static texts [SuT95].

The number of positive samples is a parameter of our method. Besides the experiments reported for $s = 2$, we simulated the behavior of our algorithm for values $s > 2$. The efficiency of our algorithm grows clearly when s increases. On the other hand, an increase in s decreases h and q, and the shift-add operation gets slower. The relationships between these parameters determine the limits of the applicability of our method.

One might expect that our algorithm for $s = 1$ would be the same as Takaoka's algorithm, but that is not the case. The speed and the filtration efficiency of these algorithms are almost the same, but their operations are different.

The idea of s positive samples can easily be incorporated with Takaoka's method. The filtration efficiency of the resulting algorithm lies between that of Takaoka's algorithm and our algorithm. However, this variation cannot outperform our approach, because the order of the q-grams is not taken into account.

Limited backtracking with a phase shift clearly improves the efficiency of our method. It would be possible even to make several subsequent phase shifts after a potential match. Every additional shift might improve the filtration, but on the other hand, consumes more time. Finding an optimal value for the number of shifts is left for further study. The idea of phases may also be applied in parallel processing: h processors may be used in such a way that the ith processor starts at text position i.

References

[BaG92] R. Baeza-Yates and G. Gonnet: A new approach to text searching. *Communications of ACM* **35**, 10 (1992), 74–82.

[ChL94] W. Chang and E. Lawler: Sublinear approximate string matching and biological applications. *Algorithmica* **12** (1994), 327–344.

[ChM94] W. Chang and T. Marr: approximate string matching and local similarity. *Combinatorial Pattern Matching, Proceedings of 5th Annual Symposium* (ed. M. Crochemore and D. Gusfield), *Lecture Notes in Computer Science* **807**, Springer-Verlag, Berlin, 1994, 259–273.

[GaP89] Z. Galil and K. Park: An improved algorithm for approximate string matching. *Proceedings of 16th International Colloquium on Automata, Languages and Programming* (ed. M. Chytil et al.), *Lecture Notes in Computer Science* **372**, Springer-Verlag, Berlin, 1989, 394–404.

[HoS94] N. Holsti and E. Sutinen: Approximate string matching using q-gram places. *Proc. Seventh Finnish Symposium on Computer Science* (ed. M. Penttonen), University of Joensuu, 1994, 23–32.

[GrL89] R. Grossi and F. Luccio: Simple and efficient string matching with k mismatches. *Information Processing Letters* **33** (1989), 113–120.

[JTU91] P. Jokinen, J. Tarhio, and E. Ukkonen: A comparison of approximate string matching algorithms. Report A-1991-7, Department of Computer Science, University of Helsinki, 1991.

[KaR87] R. Karp and M. Rabin: Efficient randomized pattern-matching algorithms. *IBM Journal of Research and Development* **31** (1987), 249–260.

[PeW93] P. Pevzner and M. Waterman: A fast filtration algorithm for substring matching problem. *Combinatorial Pattern Matching, Proceedings of 4th Annual Symposium* (ed. A. Apostolico et al.), *Lecture Notes in Computer Science* **684**, Springer-Verlag, Berlin, 1993, 197–214.

[SuT95] E. Sutinen and J. Tarhio: Information retrieval based on q-gram locations. In preparation.

[Tak94] T. Takaoka: Approximate pattern matching with samples. *Proceedings of ISAAC '94, Lecture Notes in Computer Science* **834**, Springer-Verlag, Berlin, 1994, 234–242.

[TaU93] J. Tarhio and E. Ukkonen: Approximate Boyer-Moore string matching. *SIAM Journal on Computing* **22**, 2 (1993), 243–260.

[Ukk92] E. Ukkonen: Approximate string matching with q-grams and maximal matches. *Theoretical Computer Science* **92**, 1 (1992), 191–211.

[Ukk85] E. Ukkonen: Finding approximate patterns in strings. *Journal of Algorithms* **6** (1985), 132–137.

[WuM92] S. Wu and U. Manber: Fast text searching allowing errors. *Communications of ACM* **35**, 10 (1992), 83–91.

This article was processed using the LaTeX macro package with LLNCS style