Available online at www.sciencedirect.com

ScienceDirect

Journal of Computer and System Sciences 73 (2007) 1035–1044

JOURNAL OF COMPUTER AND SYSTEM SCIENCES

www.elsevier.com/locate/jcss

Optimal spaced seeds for faster approximate string matching

Martin Farach-Colton [a], Gad M. Landau [b], S. Cenk Sahinalp [c,*], Dekel Tsur [d]

[a] *Department of Computer Science and DIMACS, Rutgers University, USA*
[b] *Department of Computer Science, University of Haifa, Israel*
[c] *Laboratory for Computational Biology, Simon Fraser University, Canada*
[d] *Department of Computer Science, Ben-Gurion University of the Negev, Israel*

Received 21 June 2003; received in revised form 1 December 2006

Available online 14 March 2007

Abstract

Filtering is a standard technique for fast approximate string matching in practice. In filtering, a quick first step is used to rule out almost all positions of a text as possible starting positions for a pattern. Typically this step consists of finding the exact matches of small parts of the pattern. In the followup step, a slow method is used to verify or eliminate each remaining position. The running time of such a method depends largely on the quality of the filtering step, as measured by its false positives rate. The quality of such a method depends on the number of true matches that it misses, that is, on its false negative rate. A *spaced seed* is a recently introduced type of filter pattern that allows gaps (i.e. do not cares) in the small sub-pattern to be searched for. Spaced seeds promise to yield a much lower false positives rate, and thus have been extensively studied, though heretofore only heuristically or statistically. In this paper, we show how to design almost optimal spaced seeds that yield no false negatives.

Keywords: Pattern matching; Hamming distance; Seeds design

1. Introduction

Given a pattern string P of length m, a text string T of length ℓ, and an integer k, the *approximate pattern matching problem* is to find all substrings of T whose edit distance or Hamming distance to P is at most k.

The basic idea employed in many approximate pattern matching algorithms [7,14] and commonly used software tools such as BLAST [1] is *filtering* based on the use of the pigeonhole principle: Let P and S be two strings with edit distance or Hamming distance at most k. Then P and S must have identical substrings (contiguous blocks) whose sizes are at least $\lceil (m-k)/(k+1) \rceil$. This simple observation can be used to perform efficient approximate pattern matching through the following approach.

(i) *Anchor finding*: Choose some $b \leqslant \lceil (m-k)/(k+1) \rceil$. Consider each substring of P of size b, and find all of its exact occurrences in T.

* Corresponding author.
E-mail address: cenk@cs.sfu.ca (S.C. Sahinalp).

(ii) *Anchor verification*: Verify whether each initial *exact match* extends to a complete *approximate match*, through the use of (a localized) dynamic program or any other appropriate method.

When the text string T is available off-line, the anchor finding step above can be implemented very efficiently:

(i) build a compact trie of all substrings in T of size b;
(ii) search each substring in P of size b on the compact trie.

By the use of *suffix links*, the compact trie can be built in $O(\ell)$ time and the anchor finding step can be completed in $O(m)$ time, both independent of the size of b.

The running time of the anchor verification step depends on the specific method for extending an initial exact match and the value of b. As b increases, the number of *false positives* is expected to decrease, but if $b > \lceil (m-k)/(k+1) \rceil$ some actual occurrences of the pattern may be missed, yielding *false negatives*.

In the remainder of this paper we will focus on filtering processes that yield to *no false negatives*, except as noted. Under this constraint, much of the literature on pattern matching via filtering focuses on improving the specific method for verifying anchors. The fastest approximate pattern matching algorithms based on filtering have a running time of $O(\ell(1 + \frac{poly\, k \cdot polylog\, \ell}{m}))$ and thus are especially powerful for "small" values of k [2,7,14]. In general, pattern matching under edit distance can be solved in time $O(\ell k)$ [11], whereas pattern matching under Hamming distance can be solved in time $O(\ell\sqrt{k \log k})$ [2].

1.1. The performance of the filtering approach

Although the filtering approach does not always speed up pattern matching, it is usually quite efficient on high-entropy texts, such as those in which each character is drawn uniformly at random from the input alphabet (of size σ). Given a pattern P, suppose that the text string T is a concatenation of

(1) *actual matches of P*: substrings of size m whose Hamming distance to P is at most k;
(2) *high entropy text*: long stretches of characters, determined uniform i.i.d. from the input alphabet.

On this T and P we can estimate the performance of the anchor verification step and thus the filtering approach in general as follows. Let the number of actual matches be $\#occ$. Each such match (i.e. true positive) will be identified as an anchor due to the pigeonhole principle. There will be other anchors yielding false positives. It is the expected number of false positives which will determine the performance of the anchor verification step and thus the overall algorithm. The false positives, i.e. substrings from the high entropy text that will be identified as anchors, can be calculated as follows. The probability that a substring $T[i\colon i+m-1]$ is identified as an anchor, i.e. has a block of size b which exactly matches its corresponding block in P, is $\leqslant m\sigma^{-b}$. The expected number of anchors from the high entropy text is thus $\leqslant \ell m \sigma^{-b}$. This implies that the running time of the filtering approach is proportional to $\#occ + \ell m \sigma^{-b}$ as well as the time required to verify a given anchor.

The above estimate of the performance of the filtering approach is determined mostly by problem specific parameters, $\#occ$, ℓ, m or the time for verifying a given anchor, none of which can be changed. There is only one variable, b, that can be determined by the filter designer. Unfortunately, in order to avoid false negatives b must be at most $\lceil (m-k)/(k+1) \rceil$.

It is possible to relax the above constraint on b by performing filtering through the use of non-contiguous blocks, namely substrings with a number of *gaps* (i.e. *do not care* symbols). To understand why searching with blocks having do not care symbols can help, consider the case when $k = 1$, that is, we allow one mismatch. When contiguous blocks are used, the maximum value of b is $\lceil (m-1)/2 \rceil$. Now consider using a block of size $b+1$ with one do not care symbol in the center position. How large can b be while guaranteeing that each substring of T with a single mismatching character with P will be found? No matter where the mismatch occurs, we are guaranteed that such a substring can be found even when $b = 2\lceil m/3 \rceil - 1$. This is a substantial improvement over ungapped search, where $b \leqslant \lceil (m-1)/2 \rceil$, reducing the time spent on false positives by a factor of $\approx \sigma^{m/6}$.

1.2. Previous work

The idea of using gapped filters (which are called *spaced seeds*) was introduced in [6], and the problem of designing the *best* possible seed was first posed in [5,13], independently. The design of spaced seeds has been extensively studied for the case when the filter is allowed to have false negatives (lossy filter), e.g. [3,4,9,13]. In this scenario, the goal is to find a seed that minimizes the false negatives rate, or in other words, a seed that maximizes the hit probability. In the above papers, the seed design problem is solved using computational methods, namely, an optimal seed is found by enumerating all possible seeds and computing the hit probability of each seed. More recent work study the design of multiple spaced seeds, and also use computational methods [12,15,16]. Unlike the case of one seed, going over all possible sets of seeds is impractical. Therefore, heuristic methods are used to find a set of seeds which may be far from optimal.

For some applications, it is desirable to construct seeds that are lossless, i.e., seeds that find *all* matches of any pattern P of length m under Hamming distance k. Alternatively, one can want to find all matches of any pattern P' of length $m' \geqslant m$ within error rate k/m. The lossless seed design problem was studied in [5], and was solved using computational methods. The problem of finding optimal lossless seeds more directly was also first posed in [5]. This *combinatorial seed design problem* has remained open both in the lossless and the lossy case.

1.3. Our contributions

In this paper we study the combinatorial seed design problem in the lossless case. We give explicit design of seeds that are (almost) optimal for high entropy texts.

Our specific results are as follows. The combinatorial seed design problem has four parameters: minimum pattern length m, the number of "solid" symbols in the seed b, the number of "do not care" symbols in the seed g, and the maximum number of allowed errors between the pattern and its match k. We denote by n the seed length, namely $n = g + b$. One can optimize any one of the parameters, given the values of the other three parameters. In this paper we focus on two variants of this optimization problem:

(1) We study the following problem: Given m, n, and g, what is the spaced seed (of length n and with g do not cares) that maximizes the number of allowed errors k? i.e. we want to find a seed which guarantees that *all* matches of a pattern of length m within Hamming distance k are found, for the largest possible value of k. Our result for this problem is *explicit construction* of seeds (for various values of m, n, and g) which we prove to be almost optimal.
(2) More interestingly, given the number of errors k and minimum pattern length m, we are interested in the seed with largest possible b such that $b + g = n \leqslant m$, which guarantees no false negatives for matches with at most k errors. Clearly this seed minimizes the time spent on false positives and thus maximizes the performance of the filtering approach for any given pattern of size m with k errors. Again, we give explicit construction of seeds that are almost optimal.

Our final result is on the design of multiple seeds: For any fixed pattern length m and number of errors k (alternatively minimum pattern length $m \leqslant m'$ and error rate k/m), we show that by the use of $s \geqslant m^{1/k}$ seeds one can guarantee to improve on the maximum size of b achievable by a single seed.

We note that the problem of maximizing b for the case when k is constant was solved in [10]. Moreover, [10] considers the problem of maximizing n for fixed k, m, and g, and solves it for the case $g = 1$. This problem is analogous to the problem of maximizing k. Our results are more general: We study the problem of maximizing k for wider range of g, and the problem of maximizing b for wider range of k.

2. Preliminaries

For the remainder of the paper, the letters A and B will be used to denote strings over the alphabet $\{0, 1\}$. For a string A and an integer l, A^l denotes the concatenation of A l times. Let $\textsc{Zeros}(A)$ be the number zeros in the string A. $A[i : j]$ denotes the substring of A that starts at the ith character of A and ends at the jth character.

For two strings A and B of equal lengths, we write $A \leqslant B$ if $A[i] \leqslant B[i]$ for all $i = 1, \dots, |A|$. Note that this differs from lexicographic ordering. We say that a string A *covers* a string B if there is a substring B' of B of length

$|A|$ such that $A \leqslant B'$. In words, A covers B if we can align A against B such that every zero in the aligned region of B, is aligned against a zero in A. We will say that such an alignment *covers* B.

The connection between the above definitions and the seed design problem is as follows: a seed of length n will be represented by a string A of length n such that $A[i] = 0$ if the ith symbol of the seed is a do not care, and $A[i] = 1$ otherwise. Given a pattern string P and a substring T' of some text T of length m, let B be a string of length m, such that $B[i] = 0$ if $P[i] \neq T'[i]$, and $B[i] = 1$ otherwise. Then, the filtering algorithm using seed A will find a match between P and T' if and only if A covers B.

We define $k(n, g, m)$ to be the maximum k such that there is a string A of length n containing g zeros that covers every string B of length m with at most k zeros. In other words, for a seed length of n with g do not cares, $k(n, g, m)$ is the maximum possible number of errors between any P and any substring T' of length m that is guaranteed to be detected by the best possible seed. Also, $b(k, m)$ is the maximum b such that there is a string A with b ones that covers every string B of length m with at most k zeros. In other words, given the maximum number of errors k, $b(k, m)$ is the maximum number of solid symbols one can have in a seed so that a match between any P and T' with k errors could be detected by the best possible seed.

In the next sections we will give upper and lower bounds on $k(n, g, m)$ and $b(k, m)$ for various values of parameters, effectively solving the combinatorial seed design problem. Our lower bounds on $k(n, g, m)$ and are proved by giving explicit constructions of a seeds with desired value of k. The lower bounds on $b(k, m)$ are proved using explicit constructions and probabilistic arguments. The respective upper bounds on $k(n, g, m)$ and $b(k, m)$ show that the seeds we construct are almost to optimal.

3. Spaced seeds that maximize *k*

We first present our results on how to design a seed that maximizes the number of errors k when the other parameters n, g and m are fixed. In other words, we describe a seed with length n and g do not care symbols that guarantees to find all substrings T' of size m whose Hamming distance to pattern P is as high as possible. Our results also extend to the problem of maximizing the error rate k'/m' for any pattern P' of length $m' \geqslant m$ with fixed n and g.

3.1. Maximizing *k* for constant *g*

Theorem 1. *For every fixed g, $k(n, g, m) = (2 - \frac{1}{g+1}) \cdot \frac{m}{n} \pm O(\max(1, \frac{m}{n^2}))$.*

Proof. We first show a lower bound on $k(n, g, m)$, by explicitly constructing the seed A. The main idea of the proof is that if B contains "few" zeros, then using some form of the pigeonhole principle, there will be an "isolated" zero in B. Thus, we can align A over B such that the isolated zero in B is aligned against some zero in A, and there are no more zeros in the aligned region of B.

Suppose that n is divisible by $2g + 1$. Then the seed A is the string of length n that contains zeros in positions $\frac{n}{2g+1}, 2 \cdot \frac{n}{2g+1}, \ldots, g \cdot \frac{n}{2g+1}$, and ones elsewhere.

Below we show that A covers any string of length m with $k = (2 - \frac{1}{g+1}) \cdot \frac{m}{n} - 3$ zeros or less. We will later show that this value of k is (almost) optimal.

We start with a lemma on the distribution of zeros on B which will prove to be quite useful later. Let $L = \frac{n}{2-1/(g+1)}$.

Lemma 2. *Let B be a string of length at least n with at most $(2 - \frac{1}{g+1}) \cdot \frac{|B|}{n} - 3$ zeros. Then, either there is a substring of B of length n containing no zeros, or there is a substring B' of B of length $2L$ containing exactly one zero, and the zero appears in the first L characters of B'.*

Proof. We prove the lemma using induction on $|B|$. The base of the induction is when $|B| \leqslant n + 2L$. In this case we have that B contains no zeros (since $(2 - \frac{1}{g+1}) \cdot \frac{n+2L}{n} - 3 < 1$). Now, suppose that B is a string of length greater than $n + 2L$. W.l.o.g. B contains at least two zeros, and let x_1 and x_2 be the indices of the rightmost zero and second rightmost zero in B, respectively. If $|B| - x_1 + 1 \leqslant L$, then the prefix of B of length $x_1 - 1$ contains at most $(2 - \frac{1}{g+1}) \cdot \frac{|B|}{n} - 4 \leqslant (2 - \frac{1}{g+1}) \cdot \frac{x_1 - 1}{n} - 3$ ones, and by the induction hypothesis we have that B satisfies the statement of the lemma. If $|B| - x_2 + 1 \leqslant 2L$, then the prefix of B of length $x_2 - 1$ contains at most $(2 - \frac{1}{g+1}) \cdot \frac{|B|}{n} - 5 \leqslant$

A	1010111111	A	1010111111	A	1010111111
B'	101111111111	B'	110111111111	B'	111110111111
	(a)		(b)		(c)

Fig. 1. An example for the proof of the lower bound in Theorem 1. Suppose that $n = 10$ and $g = 2$. Then, $A = 1010111111$. If B is a string of length m with at least $\frac{m}{6} - 3$ zeros, then either there is a substring of B of length 10 with no zeros, or there is a substring B' of B of length 12 that contains one zero, and the position of the zero is at most 6. If B' contains a zero at position 1 or 2, then we align $A[1]$ with $B[3]$ (a). If B' contains a zero at position 3 or 4, then we align the first zero in A with the zero of B' (b), and if B' contains zero at position 5 or 6, we align the second zero in A with the zero of B' (c).

$(2 - \frac{1}{g+1}) \cdot \frac{x_2-1}{n} - 3$ ones, and again we use the induction hypothesis. If neither of these two cases above occurs, we take B' to be the suffix of B of length $2L$. □

Now we can complete the proof for the lower bound. Let B be a string of length m with at most $(2 - \frac{1}{g+1}) \cdot \frac{m}{n} - 3$ zeros. If B contains a substring of length n with no zeros, then clearly $A \leqslant B$. Otherwise, let B' be a substring of B of length $2L$ that contains exactly one zero, and this zero appears in the first L characters of B'. There are $2L - n + 1$ ways to align A against B', and at least one of these alignments cover B' (see Fig. 1 for an example). More precisely, let j be the index such that $B'[j] = 0$, and let s be the integer for which $\frac{s}{2g+1}n < j \leqslant \frac{s+1}{2g+1}n$. Note that $s \leqslant g$ since $j \leqslant L$. For $s = 0$, the alignment of A and B' in which $A[1]$ is aligned with $B'[j+1]$ covers B'. For $s \geqslant 1$, the alignment of A' and B in which the sth zero in A is aligned against $B'[j]$ covers B'

For every n which is not divisible by $2g + 1$, let A' be the string constructed above for the length $n' = (2g + 1) \times \lceil n/(2g + 1) \rceil$, and let A be the prefix of length n of A' (if A contains less than g zeros, arbitrarily change some ones into zeros). A covers every string that is covered by A'. Therefore, the seed A described above covers any string B of length m with $k = (2 - \frac{1}{g+1}) \cdot \frac{m}{n'} - 3$ zeros or less. Thus we get the following bound for the maximum value of k that is achievable by any seed:

$$\begin{aligned} k(n, g, m) &\geqslant \left(2 - \frac{1}{g+1}\right) \cdot \frac{m}{n'} - 3 \geqslant \left(2 - \frac{1}{g+1}\right) \cdot \frac{m}{n+2g} - 3 \\ &= \left(2 - \frac{1}{g+1}\right) \cdot \frac{m}{n} - \frac{(2 - \frac{1}{2g+1})2g \cdot m}{n(n+2g)} - 3. \end{aligned}$$

Upper bound

We now give an upper bound on $k(n, g, m)$, demonstrating that no seed can achieve a much higher value for k than the seed we described above. Let A be any seed of length n with g zeros. We will construct strings $B_0, \ldots, B_g$ that are not covered by A, such that at least one of these strings has at most $(2 - \frac{1}{g+1}) \cdot \frac{m}{n} + O(\max(1, \frac{m}{n^2}))$ zeros.

Let $y_1, \ldots, y_g$ be the indices of the zeros in A. Let $Y = \{y_j - y_i\colon 1 \leqslant i < j \leqslant g\}$, and let Z be the multi-set $\{\max(y_i - 1, n - y_i)\colon i = 1, \ldots, g\}$. Denote the elements of Z in a non-decreasing order by $z_1, \ldots, z_g$, and denote $z_{g+1} = n$. Define $d_i = \max(\{0, \ldots, z_{i+1}\} \setminus Y)$ for $i = 0, \ldots, g$, and $d'_i = \max(\{0, \ldots, n - 1 - z_i\} \setminus Y)$ for $i = 1, \ldots, g$.

The strings $B_0, \ldots, B_g$ are constructed as follows: Let B_0 be the prefix of length m of the string $(1^{d_0-1}0)^m$. The string B_i is the prefix of length m of $(1^{d_i-1}01^{d'_i-1}0)^m$. If either $d_i = 0$ or $d'_i = 0$ for some i, we say that B_i is undefined. See Fig. 2 for an example of this construction.

We now show that A does not cover the defined strings in $B_0, \ldots, B_g$. To see that A does not cover B_0 (if it is defined), suppose conversely that there is an alignment of A with B_0 that covers B_0. Since $d_0 < n$, the aligned region of B_0 contains a zero in some position j, and this zero must be aligned with a zero in A. Suppose that $B_0[j]$ is aligned with $A[y_i]$. We can break the string A into two parts: the characters to the left of y_i, and the characters to the right of y_i, whose lengths are $y_i - 1$ and $n - y_i$, respectively. By the definition of z_1, it follows that the size of the larger part is at least z_1. W.l.o.g. assume that the larger part is the part of the characters to the right of y_i. Since $d_0 \leqslant z_1$, the aligned region of B_0 contains another zero at position $j + d_0$. From the definitions of d_0 and Y, this position must be aligned with a one in A, contradicting the assumption that the alignment covers B_0.

Now, suppose that there is an alignment of A with B_l that covers B_l. The aligned region of B_l must contain a zero in some position j, which is aligned with $A[y_i]$ for some i. We again break the string A into two parts, and we have

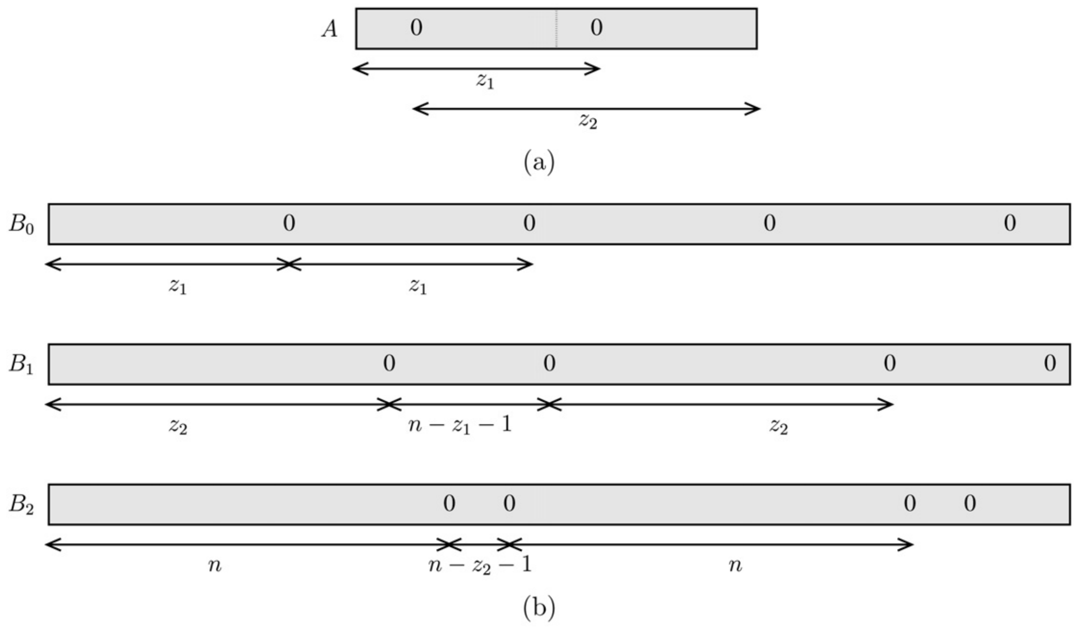

Fig. 2. An example for the proof of the upper bound in Theorem 1. (a) Shows a string A with two zeros, and the values of z_1 and z_2. (b) Shows the corresponding strings B_0, B_1, and B_2.

that either the larger part is of size at least z_{l+1}, or the smaller part is of size at most $n-1-z_l$. From the definition of d_l, d'_l, and Y, it follows that there is a zero in the aligned region of B_l that is aligned against a one in A, a contradiction.

As A does not cover the strings $B_0, \dots, B_g$, we obtain that $k(n,g,m) < \min\{\text{ZEROS}(B_i)\colon i = 0, \dots, g\}$. If $z_1 \geqslant \frac{g+1}{2g+1}n$, then $d_0 \geqslant \frac{g+1}{2g+1}n - |Y| \geqslant \frac{g+1}{2g+1}n - \frac{g^2}{2}$. For large enough n, $d_0 > 0$, so B_0 is defined and

$$\text{ZEROS}(B_0) = \left\lfloor \frac{m}{d_0} \right\rfloor \leqslant \frac{m}{\frac{g+1}{2g+1}n - \frac{g^2}{2}} = \frac{2g+1}{g+1} \cdot \frac{m}{n} + O\left(\frac{m}{n^2}\right).$$

Otherwise $z_{g+1} - z_1 > \frac{g}{2g+1}n$, so there is an index $i \geqslant 1$ such that $z_{i+1} - z_i > \frac{1}{2g+1}n$. Thus, $d'_i \geqslant n - 1 - z_i - |Y| \geqslant z_{i+1} - z_i - \frac{g^2}{2} > \frac{1}{2g+1}n - \frac{g^2}{2}$, so for large n, $d'_i > 0$ (and also $d_i > 0$). Moreover, $d_i + d'_i \geqslant n - 1 + z_{i+1} - z_i - 2|Y| \geqslant n + \frac{1}{2g+1}n - g^2$, and

$$\text{ZEROS}(B_i) \leqslant 2\left\lfloor \frac{m}{d_i + d'_i} \right\rfloor + 1 \leqslant \frac{2m}{n + \frac{1}{2g+1}n - g^2} + 1 = \frac{2g+1}{g+1} \cdot \frac{m}{n} + O\left(\max\left(1, \frac{m}{n^2}\right)\right). \qquad \square$$

3.2. Maximizing k for non-constant g

In the above section we showed how to place a constant number of do not care symbols in a seed A so that it guarantees to capture all matches of a pattern P within maximum possible number of (approximately $2\frac{m}{n}$) Hamming errors. In this section we consider the case where g, the number of do not care symbols, is not constant but $g \leqslant n^{1-1/r}$ for some $r \geqslant 2$. We show how to construct an (almost) optimal seed for this case, which turns out to be capable of capturing all matches of any pattern P within $\approx r\frac{m}{n}$ Hamming errors.

Theorem 3. *For every fixed integers $r \geqslant 2$ and $l \geqslant r$, $k(n, (1+o(1))l \cdot n^{1-1/r}, m) \geqslant (r + 1 - \frac{r}{l-r+2}) \cdot \frac{m}{n} - O(\max(1, \frac{m}{n^{1+1/r}}))$.*

Proof. Recall that in the construction of Theorem 1, we were able to find an alignment of A and B such that the aligned region of B contained the character 0 at most once, and this character was aligned against one of the zeros in A. Here, we will find an alignment that contains at most r zeros in the aligned region of B, which will be aligned against zeros in A.

We first prove the theorem for the case $r = 2$. Suppose that $\sqrt{n}$ is integer, and that $\sqrt{n}$ is divisible by $3l - 2$. A is then constructed as follows.

The seed A consists of $\sqrt{n}$ blocks of size $\sqrt{n}$. The blocks numbered $\frac{2i}{3l-2}\sqrt{n}$ for $i = 1, \ldots, l-1$ contain only zeros. The other blocks contain $\sqrt{n} - 1$ ones, and a zero at the end.

We will show that A covers any string B of length m ($\geqslant n$) with

$$\textsc{Zeros}(B) \leqslant \left(3 - \frac{2}{l}\right) \cdot \frac{m}{n} - \frac{m\sqrt{n}}{\frac{n}{3-2/l}\left(\frac{n}{3-2/l} + \sqrt{n}\right)} - 5.$$

Let B such a string, and define $L = \frac{n}{3-2/l}$, We claim that either there is a substring of B of length n without zeros, or there is a substring B' of B of length $3(L + \sqrt{n})$ that contains at most two zeros, and these zeros appears in the first $2(L + \sqrt{n})$ characters of B. We omit the proof of this claim which is similar to the proof of Lemma 2. If the first case occurs then clearly $A \leqslant B$, so suppose that the second case occurs. W.l.o.g. assume that B' contains exactly two zeros, at positions j and j', with $j' < j$. Suppose that $j > \frac{2}{3l-2}n + 2\sqrt{n}$ (the case $j \leqslant \frac{2}{3l-2}n + 2\sqrt{n}$ is similar, and we omit its proof). Let $s \leqslant l - 1$ be the integer such that $\frac{2s}{3l-2}n + 2\sqrt{n} < j \leqslant \frac{2(s+1)}{3l-2}n + 2\sqrt{n}$.

Consider an alignment of A and B' in which $B'[j]$ is aligned against $A[\frac{2s}{3l-2}n]$. Note that $A[\frac{2s}{3l-2}n]$ is the last character of block number $\frac{2s}{3l-2}\sqrt{n}$ in A. If $B'[j']$ is not in the aligned region, or it is aligned against a 0 in A then we are done. Otherwise, denote by $i = j' + \frac{2s}{3l-2}n - j$ the position in A which is aligned against $B'[j']$. Let d be the distance from position i to the nearest occurrence of a zero in A to the left of i, and $d = i$ if there is no such occurrence. Since $d < \sqrt{n}$, by moving A d positions to the right, we obtain an alignment that covers B'.

The case when $\sqrt{n}$ is not integer, or when $\sqrt{n}$ is not divisible by $3l - 2$, is handled in the same way as in Theorem 1: We build a string A' of length n' as described above, where n' is the minimal integer greater than n such that $\sqrt{n'}$ is an integer divisible by $3l - 2$, and we take A to be the prefix of length n of A'.

We now deal with the case of $r > 2$. If $n^{1/r}$ is an integer divisible by $(r+1)(l-r+1)+1$, then we build the string A as follows: We begin by taking $A = 1^n$. We then partition A into blocks of different levels. Level i ($i = 0, \ldots, r-1$) consists of $n^{1-i/r}$ blocks of size $n^{i/r}$ each. For every $i = 0, \ldots, r-2$, and every j which is divisible by $n^{1/r}$, we change block number j in level i to consists of all zeros. Furthermore, for $j = 1, \ldots, l - r + 1$, we change block number $j \cdot \frac{l-r+2}{(r+1)(l-r+1)+1} \cdot n^{1/r}$ in level $r - 1$ to consists of all zeros. The blocks that were changed are called the *zeros blocks*.

For every string B of length m with

$$\textsc{Zeros}(B) \leqslant \left(r + 1 - \frac{r}{l-r+2}\right) \cdot \frac{m}{n} - O\left(\max\left(1, \frac{m}{n^{1+1/r}}\right)\right),$$

we have that either there is a substring of B of length n without zeros, or there is a substring B' of B of length $(r+1)/(r+1-\frac{r}{l-r+2}) + O(n^{1-1/r})$ that contains at most r zeros, and the zeros appear in the first $r/(r+1-\frac{r}{l-r+2}) + O(n^{1-1/r})$ characters of B'. Assume that the second case occurs. Suppose w.l.o.g. that B' contains exactly r zeros. We create an alignment of A and B that covers B as follows: First, we align the rightmost zero in B' with the rightmost character of the appropriate zeros block of level $r - 1$. Then, for $i = 2, \ldots, r$, we move A to the right, until the ith zero from the right in B' is either aligned against the rightmost character of some zeros block of level $r - i$, or it is outside the aligned region. By our construction, the movement of A is no more than $n^{1-(i-1)/r} - 1$ positions. Moreover, during the movements of A the following invariant is kept: For every $j \leqslant i - 1$, after the ith movement, the jth zero from the right in B' is aligned against a character of some zeros block of level j', where $r - i - 1 \leqslant j' \leqslant r - j$. In particular, at the end, all the zeros in B' are aligned against zeros in A, and therefore A covers B.

The case when $n^{1/r}$ is not an integer, or when $n^{1/r}$ is not divisible by $(r+1)(l-r+1)+1$, is handled the same as before. □

The following theorem gives an upper bound that matches the lower bound in Theorem 3, demonstrating that the seed it describes is (almost) optimal.

Theorem 4. *For every integer $r \geqslant 2$, if $g \leqslant n^{1-1/r}$ then $k(n, g, m) \leqslant r \cdot \frac{m}{n} + O(1)$.*

Proof. An *indices vector* is a vector $(i_1, \dots, i_{r-1})$ such that $1 \leqslant i_1 < \dots < i_{r-1} \leqslant n-1$. The *shift* of an indices vector $v = (i_1, \dots, i_{r-1})$ is the vector $(i_2 - i_1, i_3 - i_1, \dots, i_{r-1} - i_1, n - i_1)$, and the *$s$-shift* of v is obtained by performing s shift operations on v. Note that the r-shift of v is v. The number of distinct indices vectors is $\binom{n-1}{r-1}$.

Let A be some string of length n with g zeros. Define Y to be the set of all the indices vectors $(i_1, \dots, i_{r-1})$ such that $A[x] = A[x + i_1] = \dots = A[x + i_{r-1}] = 0$ for some integer x. As $|Y| \leqslant \binom{g}{r} < \binom{n-1}{r-1}/r$, we have that there is an indices vector $v = (i_1, \dots, i_{r-1})$ such that no s-shift of v is in Y, for every $s \geqslant 0$.

Let C be a string of length n such that $C[1] = C[i_1 + 1] = \dots = C[i_{r-1} + 1] = 0$, and the other characters of C are ones, and let B be the prefix of length m of C^m. For every substring B' of B of length n, there is an s-shift $v' = (i'_1, \dots, i'_{r-1})$ of v and an integer x, such that $B'[x] = B'[x + i'_1] = \dots = B'[x + i'_{r-1}] = 0$, and thus $A \nleqslant B'$. Therefore, A does not cover B, and it follows that

$$k(n, g, m) \leqslant \textsc{Zeros}(B) - 1 \leqslant r\lceil m/n \rceil \leqslant r \cdot \frac{m}{n} + r. \qquad \square$$

4. Maximizing *b*

We now show how to maximize b, the number of solid symbols in a seed, given the number of errors k and the pattern length m. As the number of false positives depends heavily on b, the resulting seed simply optimizes the performance of the filtering method. Our result also provides the maximum b for the problem of finding all matches of any pattern P' of size $m' \geqslant m$ within error rate $k'/m' = k/m$.

Theorem 5. *For every $k < \frac{1}{8}\log_2 m$,*

$$b(k, m) \geqslant \begin{cases} m - O(km^{1-1/(k+1)}) & \text{if } k < \log\log m, \\ m - O(m^{1-1/(k+1)}) & \text{if } k \geqslant \log\log m \end{cases}$$

and $b(k, m) \leqslant m - \Omega(m^{1-1/(k+1)})$.

Proof. We begin with showing the lower bound on $b(k, m)$. Let $s = \lfloor m^{1/(k+1)} \rfloor$.

We construct a seed A of length $n = m - 2\sum_{i=1}^{k} s^i$ by dividing it into blocks in k levels, similarly to Theorem 3: The ith level of blocks ($i = 0, \dots, k-1$) consists of blocks of size s^i each (the last $m \bmod s^i$ characters of A do not belong to a level i block). For every $i \leqslant k-1$ and every j which is divisible by s, we make block number j in level i a zeros block.

We need to show that A covers every string B of length m with k zeros. Let B be such string, and let $y_1, \dots, y_k$ be the indices in which the character 0 appears in B. Let $y_{k+1} = m + 1$. If $y_{i+1} - y_i \leqslant 2s^i$ for all $i \leqslant k$ then we have that $y_1 \geqslant m + 1 - \sum_{i=1}^{k} 2s^i = n + 1$. Therefore $B[1:n] = 1^n$, and A covers B. Otherwise, let j be the maximum index such that $y_{j+1} - y_j > 2s^j$. Note that from the maximality of j, $y_{j+1} = m + 1 - \sum_{i=j+1}^{k}(y_{i+1} - y_i) \geqslant m + 1 - \sum_{i=j+1}^{k} 2s^i$, so $y_{j+1} - n > 2s^j$.

We align A over B such that $A[n]$ is aligned with $B[\max(n, y_j)]$. Then, for $i = j, j-1, \dots, 1$, we move A to the right until $B[y_i]$ is against the rightmost character of some level $i-1$ zeros block in A. The total movement of A is at most $\sum_{i=1}^{j} s^j \leqslant 2s^j < y_{j+1} - \max(n, y_j)$, and therefore at the end of this process the alignment covers B.

We therefore have that $b(k, m) \geqslant n - g$, where g is the number of zeros in B. The lower bound for $k < \log\log m$ follows from the fact that $g = O(km^{1-1/(k+1)})$.

Now, suppose that $k \geqslant \log\log m$. We randomly construct a string A of length $n = m - \lfloor m^{1-1/(k+1)} \rfloor$: Each character of A is 0 with probability $p = 100/m^{1/(k+1)}$, and 1 otherwise, where all the choices are independent. By Markov's inequality, with probability at least 1/2, the number of zeros in A is at most $2pn = O(m^{1-1/(k+1)})$. We will show that with probability at least 2/3, A covers every string of length m with k zeros. Therefore, there is a string of length n with $O(m^{1-1/(k+1)})$ zeros that covers every string of length m with k zeros, and the lower bound follows.

Let B be some string of length m with k zeros at positions $y_1, \dots, y_k$. There is a set $X \subseteq \{0, \dots, m-n\}$ of size at least $(m-n)/k^2$ such that for every $x, x' \in X$, there are no indices i and j such that $y_i + x = y_j + x'$. (Such a set X can be built by starting with $X = \phi$ and $X' = \{0, \dots, m-n\}$, and then moving elements from X' to X. Each element added to X rules out at most $k(k-1)$ elements in X', and therefore $|X| \geqslant (m-n+1)/(k(k-1)+1) > (m-n)/k^2$.)

For every $x \in X$, the probability that the alignment that aligns $A[1]$ with $B[1+x]$ does not cover B is at most $1-p^k$ (the probability is less than $1-p^k$ if some of the zeros of B are outside the aligned region). From the definition of X, the events above are independent, so the probability that A does not cover B is at most $(1-p^k)^{|X|}$. Using the union bound we obtain that the probability that there is a string B that is not covered by A is at most

$$\binom{m}{k}\left(1-p^k\right)^{(m-n)/k^2} \leqslant m^k e^{-p^k m^{1-1/(k+1)}/k^2} < \frac{1}{3}.$$

Upper bound

Denote $M = \lceil \frac{1}{2} m^{1-1/(k+1)} \rceil$. Let A be a string with b ones and g zeros, and suppose that $b \geqslant m - M + 1$. We will show that there is a string B of length m with k zeros that is not covered by A. Clearly, $g \leqslant m - b \leqslant M - 1$.

Define $I_i = \{(i-1)M + 1, \ldots, iM\}$ for $i = 1, \ldots, 2m^{1/(k+1)} - 1$. Let Y be the set of all k-tuples $(j, i_1, \ldots, i_{k-1})$ such that (1) $j \leqslant 2m^{1/(k+1)} - 1$ and $M + 1 \leqslant i_1 < \cdots < i_{k-1} \leqslant jM$, and (2) there is an index $x \in I_j$ such that $A[x] = A[x - i_1] = \cdots = A[x - i_{k-1}] = 0$. We have that

$$|Y| \leqslant \binom{g}{k} < \binom{M}{k} < \frac{\binom{m-2M}{k}}{M}.$$

Since the number of k-tuples that satisfy (1) above is at least $\binom{m-2M}{k}/M$, we have that there is a k-tuple $(j, i_1, \ldots, i_{k-1})$ that satisfies (1) but does not satisfy (2). We now construct a string B of length m that contains zeros in positions $jM, jM - i_1, \ldots, jM - i_{k-1}$ and ones elsewhere. If A covers B, then consider some alignment of A and B that covers B, and suppose that $A[1]$ is aligned with $B[y]$. Since the length of A is at least $b \geqslant m - M + 1$, we have that $y \leqslant M$. Therefore, $B[jM]$ is aligned with $A[x]$ for some $x \in I_j$. It follows that $(j, i_1, \ldots, i_{k-1}) \in Y$, a contradiction. Therefore, A does not cover B, and the upper bound follows. □

Theorem 6. *For every $k \geqslant \frac{1}{8}\log_2 m$, $b(k,m) = \Omega(\frac{m}{k}\log\frac{m}{k})$ and $b(k,m) = O(\frac{m}{k}\log m)$.*

Proof. The lower bound is trivial if $k = \Omega(m)$, so assume that $k < \frac{m}{10}$. By Theorem 5, there is a string A of length $n = \frac{1}{8} \cdot \frac{m}{k} \log \frac{m}{k}$ that contains $b = \Theta(n)$ ones and covers every string of length $8n$ with at most $\log n$ zeros. If B is a string of length m with at most k zeros, then by the pigeon-hole principle, there is a substring B' of B of length $8n$ that has at most $k \cdot \frac{8n}{m} = \log \frac{m}{k} \leqslant \log n$ zeros, and therefore A covers B'. Thus, A covers B.

Upper bound

Suppose that A is a string of length n with $b \geqslant \frac{m}{k}\log m$ ones. Let $z_1, \ldots, z_b$ be the positions of the ones in A. Construct a collection of sets $S_1, \ldots, S_m$, where $S_x = \{y \in \{0, \ldots, m-n\}$: $\exists j$ s.t. $z_j + y = x\}$. For every $y = 0, \ldots, m-n$, there are b sets among $S_1, \ldots, S_m$ that contain y. Therefore, there is a set cover of $\{0, \ldots, m-n\}$ using $l \leqslant \log(m-n+1)/\log(1/(1-b/m))$ sets from $S_1, \ldots, S_m$ [8], namely, there are indices $x_1, \ldots, x_l$ such that $\bigcup_{i=1}^{l} S_{x_i} = \{0, \ldots, m-n\}$. Now, let B be a string of length m that contains zeros in positions $x_1, \ldots, x_l$ and ones elsewhere. If we align A and B, where $A[1]$ is aligned with $B[1+y]$ for some $0 \leqslant y \leqslant m-n$, then there is an index x_i such that $y \in S_{x_i}$, that is, there is an index j such that $z_j + y = x_i$. This implies that the character $A[z_j] = 1$ is aligned against $B[x_i] = 0$. Therefore, A does not cover B. The number of zeros in B is at most

$$\frac{\log(m-n+1)}{\log\frac{1}{1-b/m}} \leqslant \frac{m}{b} \cdot \log m \leqslant k,$$

and the upper bound follows. □

4.1. Multiple seeds

To model multiple seeds, we define that a set of strings $\{A_1, \ldots, A_s\}$ covers a string B if at least one string A_i from the set covers B. Let $b(k,m,s)$ be the maximum b such that there is a set of strings $\{A_1, \ldots, A_s\}$ that covers every string B of length m with at most k zeros, and each string A_i contains at least b ones. The following theorem shows that using multiple seeds can give better results than one seed, namely, $b(k,m,m^{1/k})$ is slightly larger than $b(k,m)$ (see Theorem 5).

Theorem 7. *For every $k < \log\log m$, $b(k, m, m^{1/k}) \geqslant m - O(km^{1-1/k})$.*

Proof. We take $s = \lfloor m^{1/k} \rfloor$, and build a string A of length $n = m - \sum_{l=0}^{k-1} s^l$, that has $k-1$ levels of blocks as in the proof of Theorem 5. Then, we build strings $A_1, \ldots, A_{s-1}$, where A_i is obtained by taking the string A and adding zeros at positions $js - i$ for $j \geqslant s$. It is easy to verify that $\{A_1, \ldots, A_{s-1}\}$ covers every string of length m with at most k zeros. □

References

[1] S. Altschul, W. Gisch, W. Miller, E. Myers, D. Lipman, Basic local alignment search tool, J. Mol. Biol. 215 (3) (1990) 403–410.
[2] A. Amir, M. Lewenstein, E. Porat, Faster algorithms for string matching with k mismatches, J. Algorithms 50 (2) (2004) 257–275.
[3] B. Brejová, D.G. Brown, T. Vinar, Vector seeds: An extension to spaced seeds, J. Comput. System Sci. 70 (3) (2005) 364–380.
[4] J. Buhler, U. Keich, Y. Sun, Designing seeds for similarity search in genomic DNA, J. Comput. System Sci. 70 (3) (2005) 342–363.
[5] S. Burkhardt, J. Kärkkäinen, Better filtering with gapped q-grams, Fundam. Inform. 56 (1–2) (2003) 51–70.
[6] A. Califano, I. Rigoutsos, FLASH: A fast look-up algorithm for string homology, in: Proc. 1st International Conference on Intelligent Systems for Molecular Biology (ISMB), 1993, pp. 56-64.
[7] R. Cole, R. Hariharan, Approximate string matching: A simpler faster algorithm, SIAM J. Comput. 31 (6) (2002) 1761–1782.
[8] M. Karpinski, A. Zelikovsky, Approximating dense cases of covering problems, Electronic Colloquium on Computational Complexity (ECCC) 4 (4) (1997).
[9] U. Keich, M. Li, B. Ma, J. Tromp, On spaced seeds for similarity search, Discrete Appl. Math. 138 (3) (2004) 253–263.
[10] G. Kucherov, L. Noé, M.A. Roytberg, Multiseed lossless filtration, IEEE/ACM Trans. Comput. Biol. Bioinform. 2 (1) (2005) 51–61.
[11] G.M. Landau, U. Vishkin, Fast parallel and serial approximate string matching, J. Algorithms 10 (2) (1989) 157–169.
[12] M. Li, B. Ma, D. Kisman, J. Tromp, Patternhunter II: Highly sensitive and fast homology search, J. Bioinform. Comput. Biol. 2 (3) (2004) 417–440.
[13] B. Ma, J. Tromp, M. Li, Patternhunter: Faster and more sensitive homology search, Bioinformatics 18 (3) (2002) 440–445.
[14] S.C. Sahinalp, U. Vishkin, Efficient approximate and dynamic matching of patterns using a labeling paradigm, in: Proc. 37th Symposium on Foundation of Computer Science (FOGS), 1996, pp. 320–328.
[15] Y. Sun, J. Buhler, Designing multiple simultaneous seeds for DNA similarity search, J. Comput. Biol. 12 (6) (2005) 847–861.
[16] J. Xu, D.G. Brown, M. Li, B. Ma, Optimizing multiple spaced seeds for homology search, in: Proc. 14th Annual Symposium on Combinatorial Pattern Matching (CPM), 2004, pp. 47–58.