DIMACS Technical Report 2004-46
October 2004

Tight Bounds on Deterministic Seed Design

by

Martin Farach-Colton[1] Gad M. Landau[2] S. Cenk Sahinalp[3]
Dekel Tsur[4]

[1]Dept. of Computer Science and DIMACS, Rutgers University.
[2]Dept. of Computer Science, University of Haifa.
[3]School of Computing Science, Simon Fraser University.
[4]Caesarea Rothschild Institute of Computer Science, University of Haifa.

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. 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 the number of false positives it yields.

A *spaced seed* is a recently introduced type of filter that allows gaps in the pattern to be searched for. Spaced seeds promise to yield a much lower false positive rate, and thus have been extensively studied, though heretofore only heuristically.

In this paper, we show how to design spaced seeds with no false negatives optimally.

1 Introduction

Given a pattern string P of length m, a text string T of length n, and an integer k, the *approximate string matching problem* asks to find all substrings of T whose edit distance to P is at most k. A simpler variant of this problem is the *k-mismatches problem* which asks to find all substrings of T whose Hamming distance to P is at most k. The fastest algorithms for solving the approximate pattern matching problem as well as the k mismatches problem require $O(n(1 + \frac{polyk \cdot polylogn}{m}))$ time when k is small [2, 6, 11]. For large k, the approximate pattern matching problem could be solved in time $O(nk)$ through an improved dynamic programming approach [8], while the k mismatches problem could be solved in time $O(n\sqrt{k \log k})$ [2].

The basic idea employed in many approximate pattern matching algorithms [6, 11] 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 P' and S' respectively, whose size is at least $m/(k + 1)$. This simple observation can be used to perform efficient approximate pattern matching through the following approach. (i) *Anchor finding:* consider each substring of P of size b, $(b \leq (m - k)/(k + 1))$ and find all of its exact occurrences in T. (ii) *Anchor verification:* verify whether each initial exact match extends to a complete approximate match, through the use of (a localized) dynamic programming.

Among the two steps above, the anchor finding step can be implemented in a very efficient fashion, provided that T is available offline: given an upper bound on the "error rate"k/m, (i) build a compact trie of all substrings in T of size $b \leq (m-k)/(k+1)$ and (ii) search each substring in P of size b on the compact trie. Thus the bottleneck in filtering approach is the anchor verification step, whose running time is determined by the number of anchors (exact matches of the size b) found in T..

While the filtering approach need 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. 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 u.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 will be identified as an anchor due to the pigeonhole principle. The expected number of "false positives", i.e., substrings from the high entropy text which will be identified as anchors can be calculated as follows. The probability that a substring $T[i : i + m - 1]$ is identified as an anchor; i.e. has a block of size b which exactly matches its corresponding block in P is $\leq m\sigma^{-b}$ (σ is the alphabet size). The expected number of anchors from the high entropy text is thus $\leq n \cdot m \cdot \sigma^{-b}$. Thus, the running time of the filtering approach is proportional to $\#occ + nm\sigma^{-b}$ and 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, $\#occ, n, m$ or the time for verifying a given anchor, that can not be changed. There is only one variable, b, which can be determined by the algorithms designer. Unfortunately, in order to avoid false negatives (substrings of T with $\leq k$ Hamming differences with P) b must be $\leq (m-k)/(k+1)$.

It is possible to relax the above constraint on b by performing filtering through the use of non-contiguous blocks, i.e. substrings with a number of *don't care* symbols. To understand why searching with blocks having don't care symbols can help, consider the case when $k = 1$. When contigous blocks are used, the maximum value of b is $(m-1)/2$. Now consider using a block of size $b+1$ with one don't 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 $b = (2m-1)/3$. This is a substantial improvement over the ungapped search, where $b \leq (m-1)/2$, reducing the time spent on false positives by a factor of $\sim \sigma^{m/6}$.

The idea of using gapped filters (which are called *spaced seeds*) was introduced in [7] and the problem of designing the *best* possible seed was explored statistically/experimentally in [3–5, 9, 10]. Such gapped substrings of size $m/(k+1)$ may in fact guarantee to find all occurrences of P in T within Hamming distance $k' > k$. In other words, to guarantee finding all matches of P in T within a Hamming distance k', it may be possible in step (i) to search for exact matches of gapped substrings of size $m/k - 1 > m/k' - 1$ so that in step (ii) fewer number of "false-positives", i.e. initial exact matches which do not extend to a full approximate match, will be obtained. Because the running time of the algorithm is determined by the number of false positives, the net effect of the use of such a gapped substring will be an improvement in the running time.

All available approaches for designing spaced seeds allow *false negatives*; however it is also desirable to construct seeds that guarrantee no false negatives, i.e., that finds *all* matches of a given pattern P under Hamming distance k. Unfortunately this *combinatorial seed design* problem has remained largely open [5] since the introduction of spaced seeds in early nineties [5].

Contributions. In this paper we show how to solve the combinatorial seed design problem optimally. The combinatorial seed design problem has 4 parameters: pattern length m, seed length b, number of don't care symbols in the seed g, and error rate (or percentage error) between the pattern and its match α. Our method can optimize *any* one of the parameters, given the values of the other three parameters. For example, given m, b, g, our method can maximize the error rate α; i.e. it computes the *maximum possible error rate* α and the associated spaced seed (with length b and g don't cares) which guarantees to find *all* matches of a pattern of length m within error rate α.

More interestingly, given error rate α and pattern length m, our method can compute the largest possible b such that $b + g \leq m$, and the associated seed guarantees no false negatives. Clearly this seed minimizes the time spent on false negatives and thus maximizes the performance of the filtering approach for any given pattern of size m with error rate α.

2 Preliminaries

For the rest of the paper, the letters A, B, and C 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 k times. Let $\textsc{Ones}(A)$ be the number ones in the string A.

For two strings A and B of equal lengths, we write $A \geq B$ if $A[i] \geq B[i]$ for $i = 1, \dots, |A|$. 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 $A \geq B'$. In words, A covers B if we can align A against B such that every character 1 in the aligned region of B, is aligned against a character 1 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 P of length n will be represented by a string A of length n such that $A[i] = 1$ if the i-th letter of the seed is a gap, and $A[i] = 0$ otherwise. Given a substring T' of some text T of length m, create a string B of length m, where $B[i] = 0$ if $P[i] = T'[i]$, and $B[i] = 1$ otherwise. Then, the filtering algorithm will find a match between P and T' if and only if A covers B.

We define $\alpha(n, g, m)$ to be the maximum α such that there is a string A of length n containing g ones that covers every string B of length m with at most αm ones. Note that αm is equal to the k in the k-mismatches problem. In the next sections we will give upper and lower bounds on $\alpha(n, g, m)$ for various values of g.

3 Constant g

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

Proof. We first show a lower bound on $\alpha(n, g, m)$. The main idea of the proof is that if B contains "few" 1 characters, then using some form of the pigeonhole principle, there will be an "isolated" character 1 in B. Thus, we can align A over B such that the isolated one in B is aligned against some one in A, and there are no more ones in the aligned region of B.

Suppose that n is divisible by $2g+1$. Let A be the string of length n that contains ones in positions $\frac{n}{2g+1}, 2 \cdot \frac{n}{2g+1}, \dots, g \cdot \frac{n}{2g+1}$, and zeros elsewhere. We will show that A covers any string B of length m with at most $(2 - \frac{1}{g+1}) \cdot \frac{1}{n} \cdot m - 3$ ones. Let B such a string, and denote by $y_1, \dots, y_b$ the indices in which the character 1 appears in B.

If $y_1 > n$, then $A \geq B[1:n] = 0^n$, and we are done. In the following, we assume that $y_1 \leq n$. Let $x_1 = y_1 - \frac{g+1}{2g+1}n + 1$, $x_i = y_i - y_{i-1}$ for $i = 2, \dots, b$, and $x_{b+1} = m + 1 - y_b$. We note that the reason x_1 is defined as above rather than $x_1 = y_1$ is technical, and will be revealed in the proof of Claim 2. The average of the numbers $x_1, \dots, x_{b+1}$ is

$$\frac{\sum_{i=1}^{b+1} x_i}{b+1} = \frac{m - \frac{g+1}{2g+1}n}{b+1} > \frac{m - \frac{g+1}{2g+1}n}{\frac{2g+1}{g+1} \cdot \frac{1}{n} \cdot m - 1} \geq \frac{g+1}{2g+1}n,$$

and define $L = \frac{g+1}{2g+1}n$.

Claim 2. *There is an index $i > 1$ such that $x_i > L$ and $x_i + x_{i-1} > 2L$.*

A	0101000000	A	0101000000	A	0101000000
B'	010000000000	B'	001000000000	B'	000001000000
	(a)		(b)		(c)

Figure 1: An example of the proof of Theorem 1. Suppose that $n = 10$ and $g = 2$. Then, $A = 0101000000$. If B is a string of length m with at least $\frac{5}{3n} \cdot m - 3$ ones, then there is a substring B' of B of length 12 such that B' contains the character 1 at most once, at position at most 6. If B' does not have a one character, then every alignment of A and B' covers B'. If B' contains a one at position 1 or 2, then we align $A[1]$ with $B[3]$ (figure a). If B' contains a one at position 3 or 4, then we align the first one in A with the one of B' (figure b), and if B' contains one at position 5 or 6, we align the second one in A with the one of B' (figure c). In all of these cases, the alignment covers B'.

Proof. Among $x_1, \ldots, x_{b+1}$, there is an x_j which is greater than or equal to the average of these numbers, namely, $x_j > L$. We arbitrarily pick such an element x_j. If $j = 1$, then $y_1 = x_1 + L - 1 > 2L - 1 > n$, contradicting the assumption that $y_1 \leq n$. Thus, $j > 1$. Now, if $x_j + x_{j-1} > 2L$, we are done. Otherwise, we mark the elements x_j and x_{j-1}. The average of the unmarked elements from $x_1, \ldots, x_{b+1}$ is greater than L (since $x_j + x_{j-1} \leq 2L$). Therefore, there is an unmarked element $x_{j'}$ with $x_{j'} > L$. If $x_{j'} + x_{j'-1} > 2L$, we are done. In the case where $x_{j'} + x_{j'-1} \leq 2L$, the element $x_{j'-1}$ must be unmarked, otherwise, $x_{j'-1} > L$, and therefore, $x_{j'} + x_{j'-1} > 2L$, a contradiction. We mark $x_{j'}$ and $x_{j'-1}$ and continue with this process.

The process above cannot mark all elements, because when there are only two unmarked elements x_{l-1} and x_l, we have that $x_l + x_{l-1} > L$ (since the average of the unmarked elements is always greater than L). Therefore, at some step, we will find an element x_i that satisfies the requirements of the claim. □

By Claim 2, the string $B' = B[y_i - 2L : y_i - 1]$ has at most one occurrence of the character one, which may appear at position at most L. There are $2L - n + 1 = \frac{1}{2g+1}n + 1$ ways to align A against B', and at least one of these alignments cover B' (see Figure 1 for an example). More precisely, If $x_i > n$, then the suffix of length n of B' contains only zeros, and therefore A covers B. Otherwise, let s ($1 \leq s \leq g$) be the integer for which $x_i \in \{n - \frac{s}{2g+1}n + 1, \ldots, n - \frac{s-1}{2g+1}n\}$. Since $x_{i-1} > \frac{2g+2}{2g+1}n - x_i \geq \frac{s}{2g+1}n$, it follows that the alignment of A and B in which $B[y_{i-1}]$ is aligned with $A[\frac{s}{2g+1}n]$, the aligned region of B is a substring of B', and thus this alignment covers B.

For every n which is not divisible by $2g+1$, let A' be the string constructed above for the length $n' = (2g+1)\lceil n/(2g+1) \rceil$, and let A be the prefix of length n of A' (if A contains less than g ones, arbitrarily change some zeros into ones). A covers every string that is covered by A'. Therefore, A covers any string B of length m with at most $\left(2 - \frac{1}{g+1}\right) \cdot \frac{1}{n'} \cdot m - 2$ ones.

Thus,

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

Upper bound We now give an upper bound on $\alpha(n,g,m)$. Let A be some string of length n with g ones. 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 $((2-\frac{1}{g+1})\cdot\frac{1}{n}+O(\frac{1}{\min(m,n^2)}))\cdot m$ ones.

Let $y_1,\ldots,y_g$ be the indices of the ones in A. Let $Y=\{y_j-y_i : 1\leq i<j\leq g\}$, and let Z be the multi-set $\{\max(y_i-1,n-y_i) : i=1,\ldots,g\}$. 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 $(0^{d_0-1}1)^m$. The string B_i is the prefix of length m of $(0^{d_i-1}10^{d'_i-1}1)^m$. If either $d_i=0$ or $d'_i=0$ for some i, we say that B_i is undefined.

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 alignment region in B_0 contains one in some position j, and this one must be aligned with a one 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 $d_0\leq z_1$, the aligned region of B_0 contains another one at position $j+d_0$. From the definitions of d_0 and Y, this position must be aligned with a zero 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 alignment region in B_l must contain one is 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 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 one in the aligned region of B_l that is aligned against a zero in A, a contradiction.

As A doesn't cover the strings $B_0,\ldots,B_g$, we obtain that $\alpha(n,g,m)<\frac{1}{m}\min\{\textsc{Ones}(B_i) : i=0,\ldots,g\}$. If $z_1\geq\frac{g+1}{2g+1}n$, then $d_0\geq\frac{g+1}{2g+1}n-|Y|\geq\frac{g+1}{2g+1}n-\frac{g^2}{2}$. For large enough n, $d_0>0$, so B_0 is defined, and

$$\textsc{Ones}(B_0)\leq\frac{m}{d_0}\leq\frac{1}{\frac{g+1}{2g+1}n-\frac{g^2}{2}}\cdot m=\left(\frac{2g+1}{g+1}\cdot\frac{1}{n}+O(\tfrac{1}{n^2})\right)\cdot m.$$

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

Moreover, $d_i + d_i' \geq n - 1 + z_{i+1} - z_i - 2|Y| \geq n + \frac{1}{2g+1}n - g^2$, and

$$\textsc{Ones}(B_i) \leq 2\left\lceil \frac{m}{d_i + d_i'} \right\rceil \leq \left(\frac{2}{n + \frac{1}{2g+1}n - g^2} + \frac{2}{m} \right) \cdot m = \left(\frac{2g+1}{g+1} \cdot \frac{1}{n} + O(\tfrac{1}{\min(m,n^2)}) \right) \cdot m.$$

□

4 Non-constant g

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

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 1 at most once, and this character was aligned against one of the ones in A. Here, we will find an alignment that contains at most r ones in the aligned region of B, which will be aligned against ones 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 $3l - 2$. Let A be a string that consists of $\sqrt{n}$ blocks of size $\sqrt{n}$. Blocks number $\frac{2i}{3l-2}\sqrt{n}$ for $i = 1, \ldots, l-1$ contain only ones. The other blocks contain $\sqrt{n} - 1$ zeros, and a one at the end.

We will show that A covers any string B of length m with

$$\textsc{Ones}(B) \leq (3 - \frac{2}{l}) \cdot \frac{1}{n} \cdot m - \frac{\sqrt{n}}{\frac{l}{3l-2}n(\frac{l}{3l-2}n + \sqrt{n})} \cdot m - 4.$$

Let B such a string, and denote by $y_1, \ldots, y_b$ the indices of the ones in B.

If $y_2 \geq n + \sqrt{n}$, then the string $B' = B[1 : n + \sqrt{n} - 1]$ contains at most one occurrence of the character 1, at position y_1. If $y_1 > n$ we are done. Otherwise, consider the alignment of A against B in which $A[1]$ is aligned against $B[1]$. In the aligned region of B, the 1 in position y_1 of B is aligned against a zero in A (i.e., $A[i] = 0$). Let d be the distance from position y_1 to the nearest occurrence of 1 in A to the left of y_1, and $d = y_1$ if there is no such occurrence. From the construction of A, $d < \sqrt{n}$. Therefore, if we move A d positions to the right along B, the 1 at position y_1 will either be aligned with a 1 of B (if $d \neq y_1$), or will be outside the aligned region of B (if $d = y_1$). Moreover, the aligned region of B will contain no more ones (as $y_2 \geq n + \sqrt{n}$). Therefore, this alignment covers B. For the rest of the proof we assume that $y_2 < n + \sqrt{n}$.

Let $x_1 = y_1 - \frac{2l}{3l-2}n + 1$, $x_i = y_i - y_{i-1}$ for $i = 2, \ldots, b$, and $x_{b+1} = m + 1 - y_b$. We claim that there is an index $i > 2$ such that $x_i > \frac{l}{3l-2}n + \sqrt{n}$ and $x_i + x_{i-1} + x_{i-2} > \frac{3l}{3l-2}n + 3\sqrt{n}$. The proof of this claim is similar to the proof of Claim 2.

If $x_i > n$, then A covers B. Otherwise, suppose that $x_i \in \{n - s\frac{2}{3l-2}n + \sqrt{n}, \ldots, n - (s-1)\frac{2}{3l-2}n + \sqrt{n} - 1\}$ for $s \in \{2, \ldots, l-1\}$. Note that $x_{i-1} + x_{i-2} > s\frac{2}{3l-2}n$. Consider an alignment of A and B in which $B[y_{i-1}]$ is aligned against $A[\frac{2}{3l-2}n]$ ($A[\frac{2}{3l-2}n]$ is the last character of block number $\frac{2}{3l-2}\sqrt{n}$ in A). The alignment region of B contains at most two ones: at y_{i-1} and at y_{i-2}. The 1 in position y_{i-1} is aligned against a 1 in A, but the 1 in position y_{i-2} can be aligned against a 0. Let $j = \frac{2}{3l-2}n - y_{i-1} + y_{i-2}$ be the position in A which is aligned against $B[y_{i-2}]$. Let d be the distance from position j to the nearest occurrence of 1 in A to the left of j ($d = 0$ if $A[j] = 1$), and $d = j$ if there is no such occurrence. Since $d < \sqrt{n}$, we can shift the position of A in the alignment d positions to the right, and 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 for 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 $\frac{l-r+2}{(r+1)l-r^2+2}$, then we build the string A as follows: We begin by taking $A = 0^n$. 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 ones. Furthermore, for $j = 1, \ldots, l-r+1$, we change block number $j \cdot \frac{l-r+2}{(r+1)l-r^2+2} \cdot n^{1/r}$ in level $r-1$ to consists of all ones. The blocks that were changed are called the *ones blocks*.

For every string B of length m with

$$\textsc{Ones}(B) \leq (r + 1 - \frac{r}{l-r+2}) \cdot \frac{1}{n} \cdot m - O(\frac{1}{n^{1+1/r}}),$$

we have that either there is a substring B' of B of length $2 \cdot \frac{l-r+2}{(r+1)l-r^2+2} n + o(n)$ that contains at most r ones, or the prefix of B of length $n + O(n^{1-1/r})$ contains at most $r-1$ ones. We describe below the proof for the former case (the proof for the latter case is similar). Suppose w.l.o.g. that B' contains exactly r ones. We create an alignment of A and B that covers B as follows: First, we align the rightmost one in B' with the rightmost character of the appropriate ones block of level $r-1$. Then, for $i = 2, \ldots, r$, we move A to the right, until the i-th one from the right in B' is aligned against the rightmost character of some ones block of level $r-i$. 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 $j \leq i-1$, after the i-th movement, the j-th one from the right in B' is aligned against a character of some ones block of level j', where $r-i-1 \leq j' \leq r-j$. In particular, at the end, all the ones in B' are aligned against ones in A, and therefore A covers B.

The case when $n^{1/r}$ is not divisible by $3l-2$, or when $n^{1/r}$ is not an integer, is handled the same as before. $\square$

The following theorem gives an upper bound that matches the lower bound in Theorem 3.

Theorem 4. *For every integer r, if $g \leq n^{1-1/r}$ then $\alpha(n, g, m) \leq \frac{r}{n} + O(\frac{1}{m})$.*

Proof. An *indices vector* is a vector $(i_1, \ldots, i_{r-1})$ such that $0 < i_1 < \cdots < i_{r-1} < n$. The *shift* of an indices vector $v = (i_1, \ldots, i_{r-1})$ is the vector $(i_2 - i_1, i_3 - i_1, \ldots, i_{r-1} - i_1, n - i_1)$, 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 ones. Define Y to be the set of all the vectors $(i_1, \ldots, i_{r-1})$ such that $A[x] = A[x + i_1] = \cdots = A[x + i_{r-1}] = 1$ for some integer x. As

$|Y| \le \binom{g}{r} < \binom{n-1}{r-1}/l$, we have that there is an indices vector $v = (i_1, \ldots, i_{r-1})$ such that no s-shift of v is in Y (for every $s \ge 0$).

Let C be a string of length n such that $C[1] = C[i_1 + 1] = \cdots = C[i_{r-1} + 1] = 1$, and the other characters of C are zeros, and let B be the prefix of length m of C^m.

For every substring B' of B of length n, there is an r-shift $v' = (i'_1, \ldots, i'_{r-1})$ of v and an integer x, such that $B'[x] = B'[x + i'_1] = \cdots = B'[x + i'_{r-1}]$. Therefore, A does not cover B, and it follows that

$$\alpha(n, g, m) \le \frac{\textsc{Ones}(B) - 1}{m} \le \frac{r\lceil m/n \rceil}{m} \le \frac{r}{n} + \frac{r}{m}. \qquad \Box$$

References

[1] S. Altschul, W. Gisch, W. Miller, E. Myers, D. Lipman, *Basic Local Alignment Search Tool*, Journal of Molecular Biology, 215(3) pp 403–410, 1990.

[2] A. Amir, M. Lewenstein, E. Porat, *Faster algorithms for string matching with k-mismatches*, Proc. ACM-SIAM SODA, pp 794–803, 2000.

[3] J. Buhler, *Provably sensitive indexing strategies for biosequence similarity search*, Proc. ACM RECOMB, pp 90–99, 2002.

[4] J. Buhler, U. Keich, Y. Sun, *Designing seeds for similarity search in genomic DNA*, Proc. ACM RECOMB, pp 67–75, 2003.

[5] S. Burkhardt, J. Karkkainen, *Better filtering with gapped q-grams*, Proc. CPM, pp 73–85, 2001.

[6] R. Cole, R. Hariharan, *Approximate String Matching, a Simpler, Faster Algorithm*, Proc. ACM-SIAM SODA, pp 463–472, 1997.

[7] A. Califano, I. Rigoutsos, *Flash: a fast look-up algorithm for string homology*, Proc. ISMB, pp 56–64, 1993.

[8] G.M. Landau, U. Vishkin, *Fast parallel and serial approximate string matching*, Journal of Algorithms, 10(2) pp 157–169, 1989.

[9] B. Ma, J. Tromp, M. Li, *PatternHunter: Faster and more sensitive homology search*, Bioinformatics, 18, pp 440–445, 2002.

[10] P. Pevzner, M. Waterman, *Multiple filtration and approximate pattern matching*, Algorithmica, 13, pp 135–154, 1995.

[11] S.C. Sahinalp, U. Vishkin, *Efficient Approximate and Dynamic Matching of Patterns Using a Labeling Paradigm*, Proc. IEEE FOCS, pp 320–328, 1996.