Common Substrings in Random Strings

Eric Blais* and Mathieu Blanchette**

McGill Centre for Bioinformatics and School of Computer Science
McGill University, Montréal, Québec, Canada
{eblais, blanchem}@mcb.mcgill.ca

Abstract. In computational biology, an important problem is to identify a word of length k present in each of a given set of sequences. Here, we investigate the problem of calculating the probability that such a word exists in a set of r random strings. Existing methods to approximate this probability are either inaccurate when $r > 2$ or are restricted to Bernoulli models. We introduce two new methods for computing this probability under Bernoulli and Markov models. We present generalizations of the methods to compute the probability of finding a word of length k shared among q of r sequences, and to allow mismatches. We show through simulations that our approximations are significantly more accurate than methods previously published.

1 Introduction

Many algorithms in biological sequence analysis are based on the identification of words that are present as substrings of a given set of DNA or protein sequences. Variants on this problem are used for identifying regulatory motifs in a set of co-regulated genes [20, 22], and for selecting seeds for sequence alignment [1, 2, 13]. These applications rely on the ability to estimate the statistical significance of the length of the common substring found: how surprising is it that a set of r strings of length n contain a common substring of length k?

The problem we consider in this paper is the following:

> **Common Substring in Random Strings (CSRS)**
> **Given:** A random process $\mathcal{P}$ generating r independent strings $S_1, \ldots, S_r$ of length n over the alphabet Σ, and a substring length k,
> **Find:** The probability that the r random strings $S_1, \ldots, S_r$ contain a common substring w of length k.

Various authors have studied his problem or its equivalent formulation as the longest common substring problem (see Section 2), but available methods are not accurate for finite length random sequences generated by a non-uniform Bernoulli or Markov process. In this article, we present a new approximation to the Common Substring in Random Strings (CSRS) problem and show that

* Partially supported by the André Courtemanche Fellowship in Bioinformatics.
** Partially supported by the National Science and Eng. Research Council of Canada.

M. Lewenstein and G. Valiente (Eds.): CPM 2006, LNCS 4009, pp. 129–140, 2006.

this new approximation is more accurate than previously published methods for the same problem. Moreover, we generalize our approach to solve the problems where matches are required in only q of the r strings, and where inexact matches are allowed.

The article is organized as follows. We review related work in Section 2. In Section 3, we present a model for solving the CSRS problem approximately using an assumption of independence between words. In Section 4, we present a second simplifying assumption of independence and show how the model obtained by combining both simplifying assumptions can compute approximations to CSRS in polynomial time (relative to k) for strings generated by Bernoulli and Markov processes. In Section 5, we present important generalizations for biological problems. Finally, in Section 6, we show through a set of Monte-Carlo simulations that our approach is quite accurate under all models considered.

2 Related Work

The probability that r strings contain a common *aligned* substring of length k can be determined by characterizing the length of the longest head run in a sequence of biased coin flips. This problem was examined by Feller [8] who provides a method for computing this probability with generating functions. The same problem was later studied by Erdős and Rényi [6] and Erdős and Révész [7] who provide tight asymptotic bounds on the distribution of the longest head run.

Arratia and Waterman [3, 4] generalize the results of Erdős and Rényi to examine the distribution of the longest common (unaligned) substring in two random strings. Karlin and Ost [12] provide further improvements on the bounds for the asymptotic behaviour of the longest common word in multiple random strings for a wide range of random processes.

Naus and Sheng [14] show that while the Karlin and Ost asymptotic equations provide very accurate approximations to CSRS on two random strings, the quality of the same equations deteriorates as the number of strings increases. They also provide refinements to the Karlin and Ost equations for the special case where the random strings are generated by a Bernoulli process, and show that those refinements offer very good approximations to the CSRS problem when the strings are generated by a uniform Bernoulli process. As we will show in Section 6, the quality of those approximations is not as good for strings generated by non-uniform Bernoulli processes, and their methods do not generalize to strings generated by Markov processes.

In a parallel effort, Guibas and Odlyzko [10] and many others since provide approximations and exact results for the distribution of the number of occurrences of a given word in a random text (see for instance [15, 18, 19] and the references therein). Those results allow us to compute the probability that a *given* word is a common substring to random strings. In the next section, we show how to apply these results to the computationi of the probability that a set of random strings contain *any* word as a common substring.

3 The Independent Words Model

Let ξ_w represent the event that the word w occurs in a random string S generated by a Bernoulli or Markov process. The value of $P(\xi_w)$ can be computed with the help of generating functions:

Theorem 1 (Régnier [18]). *The probability $P(\xi_w)$ that a string w of length k is found as a substring of the random string S of length n generated by a Bernoulli process or a stationary Markov process is*

$$P(\xi_w) = [s^n]\left(\frac{1}{1-s} \cdot \frac{P(w)s^k}{P(w)s^k + A_w(s)(1-s)}\right) \quad , \tag{1}$$

where $A_w(s)$ is the autocorrelation polynomial of w (see [18]) and $P(w)$ is the stationary probability of observing w at a given position in S.

There are various methods to implement numerical computations of $P(\xi_w)$. Régnier proposes a method to compute the value of $P(\xi_w)$ in $O(\log n)$ time [18]. The partial fractions method of Feller can also compute highly accurate approximations to $P(\xi_w)$ in $O(k)$ time [8].

The computation of $P(\zeta)$, where ζ represents the event that some word of length k occurs in all r random strings $S_1, \ldots, S_r$, is more problematic. Since the strings $S_1, \ldots, S_r$ are generated independently, we have $P(\zeta) = P(\cup_{w \in \Sigma^k} \xi_w^r)$. To compute $P(\zeta)$ exactly, we would need to account for the dependence between all the events ξ_w. However, as we will show in Section 6, we can get a very good approximation to $P(\zeta)$ even when we assume the independence of the events ξ_w. We therefore present the single assumption for the Independent Words Model:

Assumption 1. *For every words $w \neq w'$, we assume that the events ξ_w and $\xi_{w'}$ are independent.*

With this assumption, computing the value of $P(\zeta)$ is now straightforward.

Proposition 1. *When Assumption 1 holds, the probability $P(\zeta)$ of observing a substring of length k in each of the r random strings $S_1, \ldots, S_r$ generated by a Bernoulli or Markov process is*

$$P(\zeta) = 1 - \prod_{w \in \Sigma^k} \left(1 - P(\xi_w)^r\right) \quad . \tag{2}$$

Equation (2) can be implemented directly to provide an accurate approximation to $P(\zeta)$. However, since it requires the enumeration of every word w of length k in the alphabet Σ of size σ, its running time is exponential in k, taking $\Omega(\sigma^k)$ time even when a constant-time approximation algorithm is used to compute $P(\xi_w)$. In the next section, we present alternative algorithms for computing the approximation to $P(\zeta)$ in time polynomial in k.

4 The Double Independence Model

To develop efficient approximations to $P(\zeta)$, we want to reduce the number of terms that need to be enumerated during the computation. Already, we may note that some words have the same probability $P(\xi_w)$ of occurring in a random string. In fact, we can significantly augment the number of words that share the same probability $P(\xi_w)$ with a second simplifying assumption.

Assumption 2. *For every position $i \neq j$ in S, we assume that the probability that the events representing occurrences of the word w at position i and j are independent.*

The Assumption 2 gives the following approximation for $P(\xi_w)$:

Proposition 2. *When Assumption 2 is valid, the probability $P(\xi_w)$ of observing a word w of length k in a random string S of length n is*

$$\tilde{P}(\xi_w) = 1 - \big(1 - P(w)\big)^{n-k+1} \ , \tag{3}$$

where $P(w)$ is the stationary probability of observing the word w at a given position in S.

The approximation $\tilde{P}(\xi_w)$ has two advantages: it is the same for many different words, and can be computed in $O(1)$ time. However, we should be aware that it is not an accurate approximation to $P(\xi_w)$ for many words. Specifically, the probability $P(\xi_w)$ for words that have high self-overlap (e.g. `AAAAAA` or `CTCTCT`) is very poorly approximated by Proposition 2 [22]. Nevertheless, we will show in Section 4.3 that it is easy to correct this error in polynomial time.

We refer to the modified CSRS problem in which Assumptions 1 and 2 are valid as the Double Independence Model. In Sections 4.1 and 4.2, we show how the Double Independence Model can be used to obtain polynomial time algorithms for the computation of $P(\zeta)$ when the random string S is generated by a Bernoulli process or a Markov process, respectively.

4.1 Bernoulli Process

When the random string S is generated by a Bernoulli process, each character of S is generated independently and takes the value $x \in \Sigma$ with probability p_x. Under the double independence model, words that share the same composition, as defined below, will have the same probability $\tilde{P}(\xi_w)$ of occurring in S.

Definition 1. *The* Bernoulli composition *of a string w is the multiset γ of characters in w.*

Example 1. The Bernoulli composition of the string $w =$ `ACCATA` is the multiset $\gamma = \{\mathtt{A}, \mathtt{A}, \mathtt{A}, \mathtt{C}, \mathtt{C}, \mathtt{T}\}$.

We define P_γ to be equal to the probability $\tilde{P}(\xi_w)$ for any word w with composition γ. We also define $N_\gamma(x)$ as the number of copies of the character x in γ. We

let $\Omega(\gamma)$ represent the number of words w with composition γ, and we represent the set of all possible Bernoulli compositions for words of length k with $\mathcal{C}_k$. We then obtain the following theorem.

Theorem 2. *Let $S_1, \ldots, S_r$ represent r random strings of length n generated independently by a Bernoulli process over the alphabet Σ. When Assumptions 1 and 2 hold, the probabilty $P(\zeta)$ that the strings $S_1, \ldots, S_r$ share a common substring of length k is defined by*

$$P(\zeta) = 1 - \prod_{\gamma \in \mathcal{C}_k} \left(1 - {P_\gamma}^r\right)^{\Omega(\gamma)} \quad , \tag{4}$$

where the probability that a word w with composition γ is found in a random string S_i is $P_\gamma = 1 - (1 - \prod_{\alpha \in \Sigma} {p_\alpha}^{N_\gamma(\alpha)})^{n-k+1}$ and the number of words that have the composition γ is

$$\Omega(\gamma) = \frac{k!}{\prod_{\alpha \in \Sigma} N_\gamma(\alpha)!} \quad . \tag{5}$$

Proof. The equation for P_γ follows directly from Proposition 2, by noting that the probability $P(w)$ of observing a word w with composition γ in a given position in S is $P(w) = \prod_{\alpha \in \Sigma} {p_\alpha}^{N_\gamma(\alpha)}$. The number $\Omega(\gamma)$ of words with composition γ is equal to the number of distinct strings that can be formed from the symbols in γ, i.e. the multinomial coefficient in (5) [8]. The final result in (4) follows from Proposition 1 and the observation that every word w of length k is represented in exactly one composition γ in $\mathcal{C}_k$. □

To implement the result from Theorem 2 in an efficient algorithm, we need an efficient method to enumerate all the Bernoulli compositions in $\mathcal{C}_k$. This can be done through a simple recursive algorithm or by using the iterative algorithm of Nijenhuis and Wilf [16]. Either of these approaches uses a constant amount of computation to generate each composition. Assuming a constant alphabet size, the values of P_γ and $\Omega(\gamma)$ can both be computed in $O(1)$ time, and the total running time of a simple algorithm that implements Theorem 2 is $O(|\mathcal{C}_k|)$. Since the number of Bernoulli compositions in $\mathcal{C}_k$ is equal to the number of different compositions of the integer k into a maximum of σ parts, we get $|\mathcal{C}_k| = \binom{k+\sigma-1}{\sigma-1} \in O(k^{\sigma-1})$ and the computation of $P(\zeta)$ can be done in $O(k^{\sigma-1})$ time.

Uniform Bernoulli Process. In the special case where the random strings $S_1, \ldots, S_r$ are generated by a uniform Bernoulli process, the value $\tilde{P}(\xi_w)$ is identical for every word $w \in \Sigma^k$. In this case, the probability $P(\zeta)$ can be computed in constant time with

$$P(\zeta) = 1 - \left(1 - \left(1 - \left(1 - 1/\sigma^k\right)^{n-k+1}\right)^r\right)^{\sigma^k} \quad . \tag{6}$$

This result is equivalent to the derivation obtained by Naus and Sheng [14] in their Equation (10) under the same special case of uniform Bernoulli processes.

4.2 Markov Process

Biological sequences rarely follow a Bernoulli model, but have been shown to be accurately modeled by low-order Markov models [20, 22]. To be applicable in a biological context, our probability calculations have to be accurate and tractable under such models. In this section, we present an algorithmic approach to approximate $P(\zeta)$ in polynomial time for 1st order Markov models. With minor changes, the approach described below can also be extended to mth order Markov models, for any fixed $m \geq 1$ [5].

Let S represent a random string generated by a stationary 1st order Markov process. For $x, y \in \Sigma$, the probability that the ith character of S takes the value y is $p_{x \to y}$, where x is the value of the $(i-1)$th character in S. The stationary probability of observing the value y at the position i in S if we do not know the values of any other character in S is $p_{\Lambda \to y}$, where Λ is a special start character. We now define the concept of *Markov composition* of a word to identify words that will share the same probability $\tilde{P}(\xi_w)$ of occurring in S.

Definition 2. *The* 1st order Markov composition *of a string w is the multiset γ of transitions between consecutive characters in w, along with a transition from the start state Λ to the first character in w.*

Example 2. The 1st order Markov composition of the strings `AACAT` and `ACAAT` is $\gamma = \{(\Lambda \to \mathtt{A}), (\mathtt{A} \to \mathtt{A}), (\mathtt{A} \to \mathtt{C}), (\mathtt{A} \to \mathtt{T}), (\mathtt{C} \to \mathtt{A})\}$.

We let P_γ be equal to $\tilde{P}(\xi_w)$ for any word w with the Markov composition γ, we let $N_\gamma(x, y)$ represent the multiplicity of the transition $(x \to y)$ in the multiset γ, and we let $\Omega(\gamma)$ represent the number of words with the composition γ. Defining $\mathcal{C}_k^1$ to be the set of all 1st order Markov compositions for words of length k, we get the following result for $P(\zeta)$.

Theorem 3. *Let $S_1, \ldots, S_r$ represent r random strings of length n generated independently by a 1st order Markov process over the alphabet Σ. When Assumptions 1 and 2 hold, the probabilty $P(\zeta)$ that the strings $S_1, \ldots, S_r$ share a common substring of length k is defined by*

$$P(\zeta) = 1 - \prod_{\gamma \in \mathcal{C}_k^1} \left(1 - {P_\gamma}^r\right)^{\Omega(\gamma)} , \tag{7}$$

where the probability that a word w with composition γ is found in a random string S_i is $P_\gamma = 1 - \left(1 - \prod_{(x,y) \in \{\Sigma, \Lambda\} \times \Sigma} {p_{x \to y}}^{N_\gamma(x,y)}\right)^{n-k+1}$ and the number $\Omega(\gamma)$ of words that have the composition γ is defined below in Theorem 4.

Proof. The result again follows directly from Propositions 1 and 2. □

To compute an approximation of $P(\zeta)$ with Theorem 3, we still need to define a method for determining $\Omega(\gamma)$ and for iterating efficiently through the different Markov compositions in $\mathcal{C}_k^1$. We first turn to the problem of evaluating $\Omega(\gamma)$, and introduce a new structure that will help us in this task.

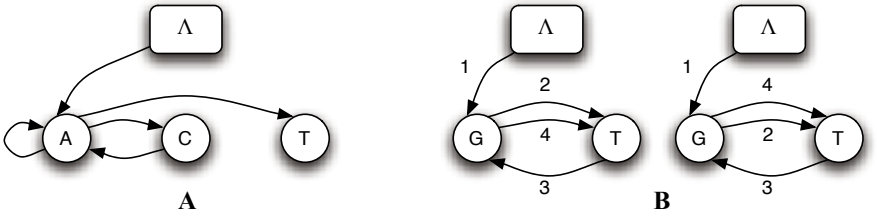

Fig. 1. (**A**) The 1st order Markov graph for the composition of the words `AACAT` and `ACAAT`, and (**B**) two distinct Eulerian trails that both represent the word `GTGT`

Definition 3. *The* 1st order Markov composition graph G_γ *for the Markov composition* γ *is the directed multigraph* $G_\gamma = \{V_\gamma, E_\gamma\}$*, where* V_γ *contains one vertex for every character in* $\Sigma \cup \{\Lambda\}$ *and* E_γ *contains one edge for every transition in* γ*.*

Example 3. The 1st order Markov composition graph for the composition of the strings `AACAT` and `ACAAT` is shown in Fig. 1A.

An Eulerian trail on the graph G_γ is a walk through the graph where we traverse each edge in E_γ exactly once. Each Eulerian trail on G_γ corresponds to a word w with composition γ, so we use some results on the enumeration of Eulerian trails on a directed multigraph to compute the number $\Omega(\gamma)$ of words with the composition γ.

Theorem 4. *The number of words with the Markov composition* γ *is*

$$\Omega(\gamma) = \frac{c_\gamma \cdot \prod_{v \in V_\gamma} (d(v) - 1)!}{\prod_{(u,v) \in V_\gamma^2} M(u,v)!} \quad , \tag{8}$$

where $G_\gamma = \{V_\gamma, E_\gamma\}$ *is the Markov composition graph of* γ*,* c_γ *is the cofactor of* G_γ *(see [11]),* $d(v)$ *is the out-degree of the vertex* v*, and* $M(u,v)$ *is the number of edges going from* u *to* v *in* E_γ*.*

Proof. By the BEST theorem [21], the total number of Eulerian trails in the graph G_γ is $c_\gamma \cdot \prod_{v \in V_\gamma} (d(v) - 1)!$ (see [11]). This number overestimates the number of distinct words with composition γ since two trails that represent the same word are counted separately if the edges between two vertices $u, v \in V_\gamma$ are taken in different order (see Fig. 1 for an example). The result in (8) follows from the fact that there are $\prod_{(u,v) \in V_\gamma^2} M(u,v)!$ different ways to order the edges in a Eulerian trail without affecting the sequence of vertices in the trail. □

We now turn to the problem of efficiently enumerating all the different Markov compositions in $\mathcal{C}_k^1$. To do this, we first present the definition of *canonical* words.

Definition 4. *A word* w *with Markov composition* γ *is* canonical *if and only if no other word with composition* γ *is lexicographically inferior to* w*.*

Example 4. As we saw in Example 2, the words `AACAT` and `ACAAT` share the same 1st order Markov composition. Since `AACAT` $<_{lex}$ `ACAAT` and there are no other words with the same composition, the word `AACAT` is canonical while the word `ACAAT` is not.

We can enumerate the canonical word for each Markov composition using the following proposition.

Proposition 3. *The word w of length k with the 1st order Markov composition γ and ending with the character z is canonical if and only if its prefix of length $k-1$ is canonical and all the transitions in γ that go from z to another character are present in lexicographic order in w.*

We can enumerate all the canonical words of length k with a recursive algorithm by enumerating every canonical word of length $k-1$ and using Proposition 3 to determine which characters can be appended to these words. There are at most $|C_k^1|$ canonical words for each length $l \leq k$, and the test for each potential character to append can be accomplished in $O(k)$ time, so the entire recursive enumeration algorithm runs in $O(k^2|C_k^1|)$ time, assuming a constant alphabet size. The complexity of computation for P_γ and $\Omega(\gamma)$ also depends only on the alphabet size, so they can both be computed in $O(1)$ time when the alphabet size is constant. There are $(\sigma+1)\cdot\sigma$ different transitions possible in 1st order Markov models, so the size of $|C_k^1| \in O(k^{\sigma^2})$, and the running time of an algorithm that implements Theorem 3 is $O(k^2|C_k^1|) \in O(k^{\sigma^2+2})$.

4.3 Correcting for Auto-correlation

The *basic period* of a word w is the smallest positive integer i such that the word w overlaps with a copy of itself shifted by i positions to the right. Words with a small basic period are also the ones for which $\tilde{P}(\xi_w)$ gives a poor approximation. Let $\mathcal{W}_{k,c}$ represent the set of all words of length k with a basic period of at most c. A simple method for improving the approximation of $P(\zeta)$ obtained with (4) or (7) is to enumerate all the words w in $\mathcal{W}_{k,c}$ and to replace the inaccurate approximation $\tilde{P}(\xi_w)$ with the better approximations $P(\xi_w)$ for these words. Specifically, we let

$$P(\zeta) = 1 - \prod_{\gamma \in \mathcal{C}_k^m} \left(1 - {P_\gamma}^r\right)^{\Omega(\gamma)} \cdot \prod_{w \in \mathcal{W}_{k,c}} \left(\frac{1 - P(\xi_w)^r}{1 - \tilde{P}(\xi_w)^r}\right) \quad . \tag{9}$$

There are $O(\sigma^c)$ words over the alphabet Σ of size σ, and the computation of $P(\xi_w)$, as we saw in Section 3, can be accomplished in $O(k)$. Therefore, the correction for the auto-correlation presented in (9) adds a factor of $O(k\sigma^c)$ to the running time of the algorithm. In practice, c does not need to be large to provide significant improvements to the estimates obtained with (4) or (7). In Section 6, we show that even a correction with $c = 1$ is enough to provide a significant improvement in accuracy.

5 Generalizations

The approach for solving the CSRS problem presented in this article can be generalized to handle many variations on the CSRS problem. For instance, we can immediately see that the above framework can be modified to handle random strings $S_1, \ldots, S_r$ that are generated by different random processes and have different lengths $n_1, \ldots, n_r$. We present two other useful generalizations below.

Common Substrings in q of r Random Strings. A common modification to the CSRS problem is to ask for the probability $P(\zeta_{q,r})$ that q of the r random strings $S_1, \ldots, S_r$ share a common substring. The equation for $P(\zeta_{q,r})$ can be obtained with the following straightforward modification of (2):

$$P(\zeta_{r,s}) = 1 - \prod_{w \in \Sigma^k} \left(1 - \sum_{j=q}^{r} \binom{r}{j} P(\xi_w)^j (1 - P(\xi_w))^{r-j}\right) . \tag{10}$$

A similar modification can also be applied to (4) and (7) for Bernoulli and Markov processes, respectively, with the running time of their corresponding algorithms increasing only by a factor of r.

Allowing Imperfect Matches. In many biologically realistic situation, one may want to allow a small number δ of mismatches in each occurrence of a word w in the random strings $S_1, \ldots, S_r$ [17]. Let $\Delta(w, \delta)$ represent the set of words that have at most δ mismatches to w. The probability that a word w occurs with at most δ mismatches in each of the strings $S_1, \ldots, S_r$ is

$$P(\xi_{w,\delta}) = 1 - \prod_{w' \in \Delta(w,\delta)} \left(1 - P(\xi_{w'})\right) . \tag{11}$$

This result can again be incorporated in (2), (4) and (7). The number of words in $\Delta(w, \delta)$ is $\sum_{i=0\ldots\delta} \binom{k}{i}(\sigma - 1)^i \in O(k^\delta)$ when the size σ of the alphabet is constant, so the running time of algorithms that incorporate (11) is exponential in k. However, δ is generally quite small in practice so the algorithm remains practical.

6 Results

We tested the accuracy of our approximations for $P(\zeta)$ against a set of hit-or-miss Monte-Carlo simulations. For each configuration of parameters tested, we generated 1,500,000 independent sets of r random strings of length n and counted what fraction of the sets contained a common substring of length k. The number of trials was selected to give a maximum error of the true probability $P(\zeta)$ of ± 0.0005, 99 times out of 100, according to the normal error bounds [9].

In Table 1, we compare different approximations to $P(\zeta)$ under a non-uniform Bernoulli model that simulates the distribution of nucleotides in the human genome ($p_A, p_T \approx 0.30$, and $p_C, p_G \approx 0.20$). For the different values of r we selected the string lengths n that give $P(\zeta) \approx 0.05$ and $P(\zeta) \approx 0.01$ to observe the quality of the approximations over the most significant range for statistical

Table 1. Approximations for the probability of observing a common substring of length $k = 6$ in r random strings of length n generated by the Bernoulli model representing the human genome ($p_A = 0.2962$, $p_C = 0.2037$, $p_G = 0.2035$, $p_T = 0.2966$)

r	n	(2)	(4)	(9)	K.O.	N.S.(a)	N.S.(b)	Simulation
2	18	0.0490	0.0491	0.0490	0.0368	0.0367	0.0600	0.0386
4	197	0.0497	0.0500	0.0497	0.0585	0.0322	0.0521	0.0477
6	467	0.0499	0.0508	0.0502	0.1041	0.0164	0.0543	0.0490
8	727	0.0500	0.0514	0.0505	0.2458	0.0075	0.0576	0.0493
10	958	0.0501	0.0519	0.0509	0.6065	0.0034	0.0609	0.0495
2	11	0.0107	0.0107	0.0107	0.0080	0.0079	0.0126	0.0088
4	131	0.0100	0.0101	0.0100	0.0111	0.0062	0.0104	0.0096
6	346	0.0099	0.0101	0.0100	0.0176	0.0029	0.0107	0.0098
8	569	0.0100	0.0103	0.0101	0.0384	0.0012	0.0113	0.0099
10	774	0.0100	0.0104	0.0102	0.1034	0.0005	0.0119	0.0100

Table 2. (**A**) Approximations for $P(\zeta)$ in the 1st order Markov model representing the human genome, when $k = 6$. (**B**) Approximation for $P(\zeta)$ when allowing imperfect matches ($k = 8$, $\delta = 1$) in the human Bernoulli model.

r	n	(2)	(7)	(9)	Simulation
2	17	0.0496	0.0497	0.0495	0.0388
4	175	0.0495	0.0522	0.0493	0.0474
6	410	0.0497	0.0549	0.0493	0.0486
8	633	0.0500	0.0607	0.0494	0.0493
10	828	0.0502	0.0674	0.0494	0.0493
2	10	0.0088	0.0088	0.0088	0.0073
4	117	0.0101	0.0105	0.0101	0.0098
6	304	0.0100	0.0113	0.0099	0.0099
8	494	0.0100	0.0128	0.0099	0.0099
10	666	0.0100	0.0148	0.0098	0.0099

A

r	n	(2), (11)	Simulation
2	9	0.0451	0.0176
4	68	0.0502	0.0417
6	187	0.0502	0.0452
8	315	0.0496	0.0446
10	436	0.0504	0.0443
2	8	0.0115	0.0050
4	47	0.0098	0.0085
6	141	0.0098	0.0091
8	251	0.0099	0.0090
10	358	0.0100	0.0088

B

analysis. As the results in Table 1 show, the approximation (2) obtained with the Independent Words Model is very accurate when $r > 2$ and converges to the true value of $P(\zeta)$ as the number of strings increases. The approximation (4) obtained with the Double Independence Model also offers a good approximation to $P(\zeta)$, although the correction of $P(\xi_w)$ for words with a basic period of 1 provided by (9) improves the approximation significantly. The table also shows that the approximation of Karlin and Ost (K.O.: (2.12) in [12]) and the first approximation of Naus and Sheng (N.S.(a): (8) in [14]) diverge significantly from the true value of $P(\zeta)$ when there are $r > 2$ strings. The alternative approximation of Naus and Sheng that incorporates exact results (N.S.(b): (14) in [14]) diverges more slowly from the value of $P(\zeta)$ as the number of strings increases, but is still not as accurate as our approximations.

We also tested our approximations on a 1st order Markov model that approximates the human genome. As we see in Table 2A, the Independent Model again offers an accurate approximation to $P(\zeta)$. However, in this case the approximation of the Double Independence Model diverges from the true value of $P(\zeta)$ as the number of strings increases and highlights the importance of the correction of Section 4.3. With the correction of $P(\xi_w)$ for words with a basic period of 1, we see a significant improvement in the approximation accuracy.

In Table 2B, we show the results of the approximation for $P(\zeta)$ for words of length $k = 8$ when a mismatch is allowed ($\delta = 1$) for each occurrence of a word in a string, under the Bernoulli model of the human genome described above. We see that the approach described in (11) provides an acceptable approximation to the true value of $P(\zeta)$ in this model, although the values diverge slowly as the number of strings increases.

7 Discussion and Future Work

We propose two new methods for approximating the probability $P(\zeta)$ that r random strings contain a common substring of length k. The approximation obtained under the Independent Words Model offers an accurate estimate for $P(\zeta)$ when $r > 2$ and works well on both the Bernoulli and Markov models. The approximations obtained under the Double Independence Model are also quite accurate when the correction for auto-correlation proposed in Section 4.3 is applied, and can be computed in polynomial time (relative to k). Both methods are shown to be more accurate than previously published approaches on random strings generated by a non-uniform Bernoulli model.

Over all the configurations of parameters tested, we find that our approximations are always slightly conservative. This is to be expected, since the most important dependency that is ignored with Assumption 1 is the positive correlation between the events ξ_w and $\xi_{w'}$ for words w and w' that overlap each other. Importantly, the fact that our estimates are always conservative means that a user who applies our approximation to assess the significance of a common substring will never be mislead into thinking that a result is more significant than it actually is.

The approach presented in this article lends itself to a number of biologically important generalizations, in particular those allowing for the substrings to be found in only a subset of the strings and allowing for imperfect matches.

There are many areas in which the research presented in this article could be further developed. Significantly, theoretical bounds on the error induced by the assumptions in our models would be very useful. Development of new methods that weaken the assumptions presented in this article could also lead to interesting and more accurate approximation algorithms.

Acknowledgments

The authors wish to thank Uri Keich for helpful discussions, and the anonymous referees for many insightful and valuable comments.

References

1. S. F. Altschul, W. Gish, W. Miller, E. W. Myers and D. J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215:403-410, 1990.
2. S. F. Altschul, T. L. Madden, A. A. Schaffer, J. Zhang, Z. Zhang, W. Miller and D. J. Lipman. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Research*, 25:3389-3402, 1997.
3. R. Arratia and M. S. Waterman. An Erdős-Rényi law with shifts. *Advances in Mathematics*, 55:13-23, 1985.
4. R. Arratia and M. S. Waterman. Critical Phenomena in sequence matching. *The Annals of Probability*, 13:1236-1249, 1985.
5. E. Blais. *Computing Probabilities for Common Substrings in Random Strings.* M. Sc. Thesis, McGill University, 2006.
6. P. Erdős and A. Rényi. On a new law of large numbers. *Journal d'Analyse Mathématique*, 22:103-111, 1970.
7. P. Erdős and P. Révész. On the length of the longest head run. *Topics in Information Theory, Coll. Math. Soc. János Bolyai No. 16*, 219-228, 1975.
8. W. Feller. *An Introduction to Probability Theory and its Applications*, Volume 1 (3rd Edition), John Wiley & Sons, 1968.
9. G. S. Fishman. *Monte Carlo: Concepts, Algorithms, and Apps.*, Springer, 1996.
10. L. J. Guibas and A. M. Odlyzko. String overlaps, pattern matching, and nontransitive games. *Journal of Combinatorial Theory, Series A*, 30:183-208, 1981.
11. F. Harary. *Graphical Enumeration*, Academic Press, 1973.
12. S. Karlin and F. Ost. Maximal length of common words among random letter sequences. *The Annals of Probability*, 16:535-563, 1988.
13. B. Morgenstern, K. Frech, A. Dress and T. Werner. DIALIGN: Finding local similarities by multiple sequence alignment. *Bioinformatics* 14:290-294, 1998.
14. J. Naus and K.-N. Sheng. Matching among multiple random sequences. *Bulletin of Mathematical Biology*, 59:483-496, 1997.
15. P. Nicodème, B. Salvy and P. Flajolet. Motif statistics. *Theoretical Computer Science*, 287:593-617, 2002.
16. A. Nijenhuis and H. Wilf *Combinatorial Algorithms for Computers and Calculators*, Academic Press, 1978.
17. P. A. Pevzner and S. Sze. Combinatorial approaches to finding subtle signals in DNA sequences. *Proc. 8th Inter. Conf. on Int. Sys. for Mol. Biol.*, 269-278, 2000.
18. M. Régnier. A unified approach to word occurrence probabilities. *Discrete Applied Mathematics*, 104:259-280, 2000.
19. M. Régnier and W. Szpankowski. On pattern frequency occurrences in a Markovian sequence. *Algorithmica*, 22:631-649, 1998.
20. S. Sinha and M. Tompa. Discovery of novel transcription factor binding sites by statistical overrepresentation. *Nucleic Acids Research*, 30:5549-5560, 2002.
21. T. van Aardenne-Ehrenfest and N. G. de Bruijn. Circuits and trees in oriented linear graphs. *Simon Stevin*, 28:203-217, 1951.
22. J. van Helden, B. André and J. Collado-Vides. Extracting regulatory sites from the upstream region of yeast genes by computational analysis of oligonucleotide frequencies. *Journal of Molecular Biology*, 281:827-842, 1998.