Optimal Computation of Overabundant Words

Yannis Almirantis[1], Panagiotis Charalampopoulos[2], Jia Gao[3], Costas S. Iliopoulos[4], Manal Mohamed[5], Solon P. Pissis[6], and Dimitris Polychronopoulos[7]

1 **National Center for Scientific Research Demokritos, Athens, Greece**
yalmir@bio.demokritos.gr
2 **Department of Informatics, King's College London, London, UK**
panagiotis.charalampopoulos@kcl.ac.uk
3 **Department of Informatics, King's College London, London, UK**
jia.gao@kcl.ac.uk
4 **Department of Informatics, King's College London, London, UK**
costas.iliopoulos@kcl.ac.uk
5 **Department of Informatics, King's College London, London, UK**
manal.mohamed@kcl.ac.uk
6 **Department of Informatics, King's College London, London, UK**
solon.pissis@kcl.ac.uk
7 **Computational Regulatory Genomics Group, MRC London Institute of Medical Sciences, Imperial College London, Hammersmith Hospital Campus, London, UK**
dpolychr@imperial.ac.uk

Abstract

The observed frequency of the longest proper prefix, the longest proper suffix, and the longest infix of a word w in a given sequence x can be used for classifying w as *avoided* or *overabundant*. The definitions used for the expectation and deviation of w in this statistical model were described and biologically justified by Brendel et al. (J Biomol Struct Dyn 1986). We have very recently introduced a time-optimal algorithm for computing all avoided words of a given sequence over an integer alphabet (Algorithms Mol Biol 2017). In this article, we extend this study by presenting an $\mathcal{O}(n)$-time and $\mathcal{O}(n)$-space algorithm for computing all overabundant words in a sequence x of length n over an integer alphabet. Our main result is based on a new *non-trivial* combinatorial property of the suffix tree $\mathcal{T}$ of x: the number of distinct factors of x whose longest infix is the label of an explicit node of $\mathcal{T}$ is no more than $3n-4$. We further show that the presented algorithm is time-optimal by proving that $\mathcal{O}(n)$ is a tight upper bound for the number of overabundant words. Finally, we present experimental results, using both synthetic and real data, which justify the *effectiveness* and *efficiency* of our approach in practical terms.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases overabundant words, avoided words, suffix tree, DNA sequence analysis

Digital Object Identifier 10.4230/LIPIcs.WABI.2017.4

1 Introduction

Brendel et al. in [6] initiated research into the linguistics of nucleotide sequences that focused on the concept of words in continuous languages – languages devoid of blanks – and introduced an operational definition of words. The authors suggested a method to measure, for each possible word w of length k, the deviation of its observed frequency $f(w)$ from the expected frequency $E(w)$ in a given sequence x. The observed frequency of the longest proper prefix,

the longest proper suffix, and the longest infix of w in x were used to measure $E(w)$. The values of the deviation, denoted by $dev(w)$, were then used to identify words that are *avoided* or *overabundant* among all possible words of length k. The typical length of avoided (or of overabundant) words of the nucleotide language was found to range from 3 to 5 (tri- to pentamers). The statistical significance of the avoided words was shown to reflect their biological importance. That work, however, was based on the very limited sequence data available at the time: only DNA sequences from two viral and one bacterial genomes were considered. Also note that the range of typical word length k might change when considering eukaryotic genomes, the complex dynamics and function of which are expected to impose more demanding roles to avoided or overabundant words of nucleotides.

To this end, in [1], we presented an $\mathcal{O}(n)$-time and $\mathcal{O}(n)$-space algorithm for computing all avoided words of length k in a sequence of length n over a fixed-sized alphabet. For words over an integer alphabet of size σ, the algorithm requires time $\mathcal{O}(\sigma n)$, which is optimal for sufficiently large σ. We also presented a time-optimal $\mathcal{O}(\sigma n)$-time algorithm to compute all avoided words (of any length) in a sequence of length n over an integer alphabet of size σ. We provided a tight asymptotic upper bound for the number of avoided words over an integer alphabet and the expected length of the longest one. We also proved that the same asymptotic upper bound is tight for the number of avoided words of fixed length k when the alphabet is sufficiently large. The authors in [3, 2, 4] studied a similar notion of *unusual words* – based on different definitions than the ones Brendel et al. use for expectation and deviation – focusing on the factors of a sequence; based on Brendel et al.'s definitions, we focus on any word over the alphabet. More recently, space-efficient detection of unusual words has also been considered [5]; such avoidances is becoming an interesting line of research [18].

In this article, we wish to complement our study in [1] by focusing on overabundant words. The motivation comes from molecular biology. Genome dynamics, i.e. the molecular mechanisms generating random mutations in the evolving genome, are quite complex, often presenting self-enhancing features. Thus, it is expected to often give rise to words of nucleotides which will be overabundant, i.e. being present at higher amounts than expected on the basis of their longest proper prefix, longest proper suffix, and longest infix frequencies. One specific such mechanism, which might generate overabundant words, is the following: it is well-known that in a genomic sequence of an initially random composition, the existing relatively long homonucleotide tracts present a higher frequency of further elongation than the frequency expected on the basis of single nucleotide mutations [15]; that is, they present a sort of autocatalytic self-elongation. This feature, in combination with the much higher frequency of transition *vs.* transversion mutation events, generates overabundant words which are homopurinic or homopurimidinic tracts. It is also anticipated that the overabundance of homonucleotide tracts will strongly differentiate between conserved and non-conserved parts of the genome. While this phenomenon is largely free to act within the non-conserved genomic regions, and thus it is expected to generate there large amounts of overabundant words, it is hindered in the conserved genomic regions due to selective constraints.

Our Contributions. Analogously to avoided words [6, 11, 1], many different models and algorithms exist for identifying words that are in abundance in a given sequence; see for instance [7, 9]. In this article, we make use of the biologically justified model introduced by Brendel et al. [6] and, by proving non-trivial combinatorial properties, we show that it admits *efficient* computation for overabundant words as well. We also present experimental results, using both synthetic and real data, which further highlight the *effectiveness* of this model. The computational problem can be described as follows. Given a sequence x of length

n and a real number $\rho > 0$, compute the set of ρ-overabundant words, i.e. all words w for which $dev(w) \geq \rho$. We present an $\mathcal{O}(n)$-time and $\mathcal{O}(n)$-space algorithm for computing all ρ-overabundant words (of any length) in a sequence x of length n over an integer alphabet. This result is based on a combinatorial property of the suffix tree $\mathcal{T}$ of x that we prove here: the number of distinct factors of x whose longest infix is the label of an explicit node of $\mathcal{T}$ is no more than $3n - 4$. We further show that the presented algorithm is time-optimal by proving that $\mathcal{O}(n)$ is a tight upper bound for the number of ρ-overabundant words. Finally, we pose an open question of combinatorial nature on the maximum number $\mathsf{OW}(n, \sigma)$ of overabundant words that a sequence of length n over an alphabet of size $\sigma > 1$ can contain.

2 Terminology and Technical Background

2.1 Definitions and Notation

We begin with basic definitions and notation, generally following [8]. Let $x = x[0]x[1] \dots x[n-1]$ be a *word* of *length* $n = |x|$ over a finite ordered *alphabet* Σ of size σ, i.e. $\sigma = |\Sigma|$. In particular, we consider the case of an *integer alphabet*; in this case each letter is replaced by its rank such that the resulting word consists of integers in the range $\{1, \dots, n\}$. In what follows we assume without loss of generality that $\Sigma = \{0, 1, \dots, \sigma - 1\}$. We also define Σ_x to be the alphabet of word x and $\sigma_x = |\Sigma_x|$. For two positions i and j on x, we denote by $x[i\,..\,j] = x[i] \dots x[j]$ the *factor* (sometimes called *subword*) of x that starts at position i and ends at position j (it is empty if $j < i$), and by ε the *empty word*, word of length 0. We recall that a *prefix* of x is a factor that starts at position 0 ($x[0\,..\,j]$) and a *suffix* is a factor that ends at position $n - 1$ ($x[i\,..\,n-1]$), and that a factor of x is a *proper* factor if it is not x itself. A factor of x that is neither a prefix nor a suffix of x is called an *infix* of x. We denote the *reverse* word of x by $\mathsf{rev}(x)$, i.e. $\mathsf{rev}(x) = x[n-1]x[n-2] \dots x[1]x[0]$. We say that x is *a power* of a word y if there exists a positive integer k, $k > 1$, such that x is expressed as k consecutive concatenations of y; we denote that by $x = y^k$.

Let $w = w[0]w[1] \dots w[m-1]$ be a word, $0 < m \leq n$. We say that there exists an *occurrence* of w in x, or, more simply, that w *occurs in* x, if w is a factor of x, which we denote by $w \preceq x$. Every occurrence of w can be characterised by a starting position in x. Thus we say that w occurs at position i in x when $w = x[i\,..\,i+m-1]$. Further, let $f(w)$ denote the *observed frequency*, that is, the number of occurrences of a non-empty word w in word x. If $f(w) = 0$ for some word w, then w is called *absent* (which is denoted by $w \not\preceq x$), otherwise, w is called *occurring*.

By $f(w_p)$, $f(w_s)$, and $f(w_i)$ we denote the observed frequency of the longest proper prefix w_p, suffix w_s, and infix w_i of w in x, respectively. We can now define the *expected frequency* of word w, $|w| > 2$, in x as in Brendel et al. [6]:

$$E(w) = \frac{f(w_p) \times f(w_s)}{f(w_i)}, \text{ if } f(w_i) > 0; \text{ else } E(w) = 0. \tag{1}$$

The above definition can be explained intuitively as follows. Suppose we are given $f(w_p)$, $f(w_s)$, and $f(w_i)$. Given an occurrence of w_i in x, the probability of it being preceded by $w[0]$ is $\frac{f(w_p)}{f(w_i)}$ as $w[0]$ precedes exactly $f(w_p)$ of the $f(w_i)$ occurrences of w_i. Similarly, this occurrence of w_i is also an occurrence of w_s with probability $\frac{f(w_s)}{f(w_i)}$. Although these two events are not always independent, the product $\frac{f(w_p)}{f(w_i)} \times \frac{f(w_s)}{f(w_i)}$ gives a good approximation of the probability that an occurrence of w_i at position j implies an occurrence of w at position $j - 1$. It can be seen then that by multiplying this product by the number of occurrences of w_i we get the above formula for the expected frequency of w.

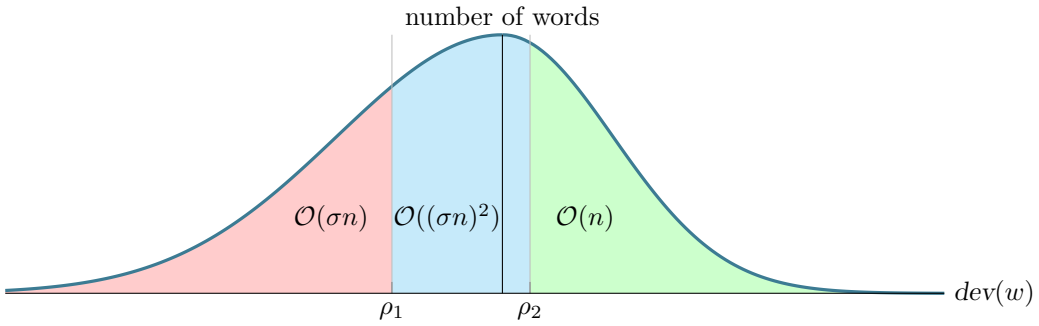

Figure 1 For a word x, the words for which $dev(w)$ is defined are the ones of the form $w = aub$, where u is a factor of x and $a, b \in \Sigma$, not necessarily distinct. There are $\mathcal{O}(n^2)$ distinct factors in a word of length n and for each of these we obtain σ^2 words of this form. We have shown that the ρ_1-avoided words are $\mathcal{O}(\sigma n)$ [1]. In this article, we show that the ρ_2-overabundant ones are $\mathcal{O}(n)$.

Moreover, to measure the deviation of the observed frequency of a word w from its expected frequency in x, we define the *deviation* (χ^2 test) of w as:

$$dev(w) = \frac{f(w) - E(w)}{\max\{\sqrt{E(w)}, 1\}}. \tag{2}$$

For more details on the *biological* justification of these definitions see [6] and [1].

Using the above definitions and two given thresholds, we can classify a word w as either *avoided*, *common*, or *overabundant* in x. In particular, for two given thresholds $\rho_1 < 0$ and $\rho_2 > 0$, a word w is called ρ_1-*avoided* if $dev(w) \leq \rho_1$, ρ_2-*overabundant* if $dev(w) \geq \rho_2$, and (ρ_1, ρ_2)-*common* otherwise (see Figure 1). We have very recently shown that the number of ρ_1-avoided words is $\mathcal{O}(\sigma n)$, and have introduced a time-optimal algorithm for computing all of them in a given sequence over an integer alphabet [1]. In this article, we show that the number of ρ_2-overabundant words is $\mathcal{O}(n)$, and study the following computational problem.

AllOverabundantWordsComputation
Input: A word x of length n and a real number $\rho > 0$
Output: All ρ-overabundant words in x

2.2 Suffix Trees

In our algorithms, suffix trees are used extensively as computational tools. For a general introduction to suffix trees see [8].

The *suffix tree* $\mathcal{T}(x)$ of a non-empty word x of length n is a compact trie representing all suffixes of x. The nodes of the trie which become nodes of the suffix tree are called *explicit* nodes, while the other nodes are called *implicit*. Each edge of the suffix tree can be viewed as an upward maximal path of implicit nodes starting with an explicit node. Moreover, each node belongs to a unique path of that kind. Then, each node of the trie can be represented in the suffix tree by the edge it belongs to and an index within the corresponding path.

We use $\mathcal{L}(v)$ to denote the *path-label* of a node v, i.e., the concatenation of the edge labels along the path from the root to v. We say that v is path-labelled $\mathcal{L}(v)$. Additionally, $\mathcal{D}(v) = |\mathcal{L}(v)|$ is used to denote the *word-depth* of node v. Node v is a *terminal* node if and only if $\mathcal{L}(v) = x[i \,..\, n-1]$, $0 \leq i < n$; here v is also labelled with index i. It should be clear that each occurring word w in x is uniquely represented by either an explicit or an implicit

node of $\mathcal{T}(x)$. The *suffix-link* of a node v with path-label $\mathcal{L}(v) = \alpha y$ is a pointer to the node path-labelled y, where $\alpha \in \Sigma$ is a single letter and y is a word. The suffix-link of v exists if v is a non-root internal node of $\mathcal{T}(x)$.

In any standard implementation of the suffix tree, we assume that each node of the suffix tree is able to access its parent. Note that once $\mathcal{T}(x)$ is constructed, it can be traversed in a depth-first manner to compute the word-depth $\mathcal{D}(v)$ for each node v. Let u be the parent of v. Then the word-depth $\mathcal{D}(v)$ is computed by adding $\mathcal{D}(u)$ to the length of the label of edge (u, v). If v is the root then $\mathcal{D}(v) = 0$. Additionally, a depth-first traversal of $\mathcal{T}(x)$ allows us to count, for each node v, the number of terminal nodes in the subtree rooted at v, denoted by $\mathcal{C}(v)$, as follows. When internal node v is visited, $\mathcal{C}(v)$ is computed by adding up $\mathcal{C}(u)$ of all the nodes u, such that u is a child of v, and then $\mathcal{C}(v)$ is incremented by 1 if v itself is a terminal node. If a node v is a leaf then $\mathcal{C}(v) = 1$.

We assume that the terminal nodes of $\mathcal{T}(x)$ have suffix-links as well. We can either store them while building $\mathcal{T}(x)$ or just traverse it once and construct an array $node[0\,..\,n-1]$ such that $node[i] = v$ if $\mathcal{L}(v) = x[i\,..\,n-1]$. We further denote by $\textsc{Parent}(v)$ the parent of a node v in $\mathcal{T}(x)$ and by $\textsc{Child}(v, \alpha)$ the explicit node that is obtained from v by traversing the outgoing edge whose label starts with $\alpha \in \Sigma$. A batch of q $\textsc{Child}(v, \alpha)$ queries can be answered off-line in time $\mathcal{O}(n + q)$ for a word x over an integer alphabet (via radix sort).

3 Combinatorial Properties

In this section, we prove some properties that are useful for designing the time-optimal algorithm presented in the next section.

▶ **Fact 1.** *Given a word x of length n over an alphabet of size σ, the number of words w for which $dev(w)$ is defined is $\mathcal{O}((\sigma n)^2)$.*

Proof. For a word w over Σ, $dev(w)$ is only defined if $w_i \preceq x$. Hence the words w for which $dev(w)$ is defined are of the form aub for some non-empty $u \preceq x$ and $a, b \in \Sigma$. For each distinct factor $u \neq \varepsilon$ of x there are σ^2 words of the form aub, $a, b \in \Sigma$. Since there are $\mathcal{O}(n^2)$ distinct factors in a word of length n, the fact follows. ◀

▶ **Fact 2.** *Every word w that does not occur in x and for which $dev(w)$ is defined has $dev(w) \leq 0$.*

Proof. For such a word we have that $E(w) \geq 0$ and that $f(w) = 0$ and hence $dev(w) = \frac{f(w)-E(w)}{\max\{\sqrt{E(w)},1\}} \leq 0$. ◀

Naïve algorithm. By using Fact 2, we can compute $dev(w)$, for each factor w of x, thus solving Problem AllOverabundantWordsComputation. There are $\mathcal{O}(n^2)$ such factors, however, which make this computation inefficient.

▶ **Fact 3.** *Given a factor w of a word x, if w_i corresponds to an implicit node in the suffix tree $\mathcal{T}(x)$, then so does w_p.*

Proof. A factor w' of x corresponds to an implicit node $\mathcal{T}(x)$ if and only if every occurrence of it in x is followed by the same unique letter $b \in \Sigma$. Hence, since $w_p = aw_i$ for some $a \in \Sigma$, if w_i is always followed by, say, $b \in \Sigma$, every occurrence of w_p in x must also always be followed by b. Thus w_p corresponds to an implicit node as well. ◀

▶ **Lemma 4.** *If w is a factor of a word x and w_i corresponds to an implicit node in $\mathcal{T}(x)$, then $dev(w) = 0$.*

Proof. If a word $w' \preceq x$ corresponds to an implicit node along the edge (u, v) in $\mathcal{T}(x)$ and $\mathcal{L}(v) = w$ then the number of occurrences of w' in x is equal to that of w.

If w_i corresponds to an implicit node on edge (u, v) it follows immediately that $f(w_i) = f(w_s)$, as either w_s also corresponds to an implicit node in the same edge or $w_s = \mathcal{L}(v)$. In addition, from Fact 3 we have that w_p is an implicit node as well and it similarly follows that $f(w_p) = f(w)$. We thus have $E(w) = \frac{f(w_p) \times f(w_s)}{f(w_i)} = f(w)$ and hence $dev(w) = \frac{f(w) - E(w)}{\max\{\sqrt{E(w)}, 1\}} = 0$. ◀

Based on these properties, the aim of the algorithm in the next section is to find the factors of x whose longest infix corresponds to an explicit node and check if they are ρ-overabundant. More specifically, for each explicit node v in $\mathcal{T}(x)$, such that $\mathcal{L}(v) = y$, we aim at identifying the factors of x that have y as their longest infix (i.e. factors of the form ayb, $a, b \in \Sigma$). We will do that by identifying the factors of x that have y as their longest proper suffix (i.e. factors of the form ay, $a \in \Sigma$) and then checking for each of these the different letters that succeed it in x. Then we can check in time $\mathcal{O}(1)$ if each of these words is ρ-overabundant.

Note that the algorithm presented in Section 4 is fundamentally different and in a sense more involved than the one presented in [1] for the computation of *occurring* ρ-avoided words (note that a ρ-avoided word can be *absent*). This is due to the fact that for occurring ρ-avoided words we have the stronger property that w_p must correspond to an explicit node.

▶ **Theorem 5.** *Given a word x of length n, the number of distinct factors of x of the form ayb, where $a, b \in \Sigma$ and $y \neq \varepsilon$ is the label of an explicit node of $\mathcal{T}(x)$, is no more than $3n - 2 - 2\sigma_x$.*

Proof. Let S be the set of all explicit or implicit nodes in $\mathcal{T}(x)$ of the form yb such that y is represented by an explicit node other than the root. We have at most $2n - 2 - \sigma_x$ of them; there are at most $2n - 2$ edges in $\mathcal{T}(x)$, but σ_x of them are outgoing from the root. For such a word yb, the number of factors of x of the form ayb is equal to the degree of the node representing $\mathsf{rev}(yb)$ in $\mathcal{T}(\mathsf{rev}(x))$.

For every node in S, we obtain a distinct node in $\mathcal{T}(\mathsf{rev}(x))$. Let us suppose that k_1 of these nodes are non-root internal explicit nodes, k_2 are leaves, and the rest $2n-2-\sigma_x-k_1-k_2$ are implicit nodes. Each internal explicit node u contributes at most $deg(u)$ factors, where $deg(u)$ is the number of outgoing edges of node u, each leaf contributes 0 factors, and each implicit node contributes at most 1 factor.

Hence the number of such factors would be maximised if we obtained all the non-root internal explicit nodes and no leaves in $\mathcal{T}(\mathsf{rev}(x))$. Let $\mathcal{T}(\mathsf{rev}(x))$ have m non-root internal explicit nodes. The resulting upper bound then is $\sum_{u \in \mathcal{T}(\mathsf{rev}(x)) \setminus \{root\}} deg(u) + (2n - 2 - \sigma_x - m) \leq n + m - \sigma_x + (2n - 2 - \sigma_x - m) = 3n - 2 - 2\sigma_x$.

Note that $\sum_{u \in \mathcal{T}(\mathsf{rev}(x)) \setminus \{root\}} deg(u) \leq n + m - \sigma_x$ since there are at most n edges from explicit internal nodes to leaves and m edges to other internal nodes; σ_x of these are outgoing from the root. ◀

▶ **Corollary 6.** *The ρ-overabundant words in a word x of length n are at most $3n - 2 - 2\sigma_x$.*

Proof. By Fact 2, Lemma 4, and symmetry, it follows that the ρ-overabundant words in x are factors of x of the form ayb, where $a, b \in \Sigma$, such that $y \neq \varepsilon$ is represented by an explicit node in $\mathcal{T}(x)$ and $\mathsf{rev}(y)$ represented by an explicit node in $\mathcal{T}(\mathsf{rev}(x))$. Hence they are a subset of the set of words considered in Theorem 5. ◀

▶ **Lemma 7.** *The ρ-overabundant words in a word x of length n over a binary alphabet (e.g. $\Sigma = \{\mathtt{a}, \mathtt{b}\}$) are no more than $2n - 4$.*

Proof. For every internal explicit node u of $\mathcal{T}(x)$, other than the root, let $deg'(u)$ be $deg(u)+1$ if node u is terminal and $deg(u)$ otherwise. The sum of $deg'(u)$ over the internal explicit non-root nodes of $\mathcal{T}(x)$ is no more than $2n-4$ (ignoring the case when $x=\alpha^n, \alpha\in\Sigma$). We will show that, for each such node, the ρ-overabundant words with $w_i=\mathcal{L}(u)$ as their longest proper infix are at most $deg'(u)$.

- *Case I:* $deg'(u)=2$.
  - *Subcase 1:* $deg(u)=1$. Node u is terminal and it has an edge with label α. We can then have at most 2 ρ-overabundant words with w_i as their longest proper infix: $\mathtt{a}w_i\alpha$ and $\mathtt{b}w_i\alpha$.
  - *Subcase 2:* $deg(u)=2$. Node u is not terminal and it has an edge with label $\mathtt{a}$ and an edge with label $\mathtt{b}$. If only one of $\mathtt{a}w_i$ and $\mathtt{b}w_i$ occurs in x we are done. If both of them occur in x we argue as follows (irrespective of whether w_i is also a prefix of x):
    If $\mathtt{a}w_i\mathtt{a}$ is ρ-overabundant, then
    $f(\mathtt{a}w_i\mathtt{a}) - f(\mathtt{a}w_i)\times f(w_i\mathtt{a})/f(w_i)\geq\rho>0\Rightarrow f(\mathtt{a}w_i\mathtt{a})/f(\mathtt{a}w_i)>f(w_i\mathtt{a})/f(w_i)\Leftrightarrow 1-f(\mathtt{a}w_i\mathtt{a})/f(\mathtt{a}w_i)<1-f(w_i\mathtt{a})/f(w_i)\Leftrightarrow f(\mathtt{a}w_i\mathtt{b})/f(\mathtt{a}w_i)<f(w_i\mathtt{b})/f(w_i)\Leftrightarrow f(\mathtt{a}w_i\mathtt{b})-f(\mathtt{a}w_i)\times f(w_i\mathtt{b})/f(w_i)<0$
    and hence $\mathtt{a}w_i\mathtt{b}$ is not ρ-overabundant. (Similarly for $\mathtt{b}w_i\mathtt{a}$ and $\mathtt{b}w_i\mathtt{b}$.)
- *Case II:* $deg'(u)=3$. Node u is terminal and it has an edge with label $\mathtt{a}$ and an edge with label $\mathtt{b}$. If only one of $\mathtt{a}w_i$ and $\mathtt{b}w_i$ occurs in x or if both of them occur in x, but w_i is not a prefix of x, we can have at most 2 ρ-overabundant words with w_i as the proper longest infix; this can be seen by looking at the node representing $\mathsf{rev}(w_i)$ in $\mathcal{T}(\mathsf{rev}(x))$, which falls in *Case I*.
  So we only have to consider the case where both $\mathtt{a}w_i$ and $\mathtt{b}w_i$ occur in x and w_i is a prefix of x. For this case, we assume without loss of generality that $\mathtt{a}w_i$ is a suffix of x.
  If $\mathtt{a}w_i\mathtt{a}$ is ρ-overabundant, then
  $f(\mathtt{a}w_i\mathtt{a}) - f(\mathtt{a}w_i)\times f(w_i\mathtt{a})/f(w_i)\geq\rho>0\Rightarrow f(\mathtt{a}w_i\mathtt{a})/f(\mathtt{a}w_i)>f(w_i\mathtt{a})/f(w_i)\Leftrightarrow 1-f(\mathtt{a}w_i\mathtt{a})/f(\mathtt{a}w_i)<1-f(w_i\mathtt{a})/f(w_i)\Leftrightarrow (f(\mathtt{a}w_i\mathtt{b})+1)/f(\mathtt{a}w_i)<(f(w_i\mathtt{b})+1)/f(w_i)\Rightarrow f(\mathtt{a}w_i\mathtt{b})/f(\mathtt{a}w_i)<(f(w_i\mathtt{b})/f(w_i)\Leftrightarrow f(\mathtt{a}w_i\mathtt{b})-f(\mathtt{a}w_i)\times f(w_i\mathtt{b})/f(w_i)<0$
  and hence $\mathtt{a}w_i\mathtt{b}$ is not ρ-overabundant. Thus in this case we can have at most $3=deg'(u)$ ρ-overabundant words.

We can thus have at most $deg'(u)$ ρ-overabundant words for each internal explicit non-root node of $\mathcal{T}(x)$. This concludes the proof. ◀

▶ **Lemma 8.** *The ρ-overabundant words in a word of length n are $\mathcal{O}(n)$ and this bound is tight. There exists a word over the binary alphabet with $2n-6$ ρ-overabundant words.*

Proof. The asymptotic bound follows directly from Corollary 6. The tightness of the asymptotic bound can be seen by considering word $x=ba^{n-2}b$, $a,b\in\Sigma$, of length n and some ρ such that $0<\rho<1/n$. Then for every prefix w of x of the form ba^k and for every suffix w' of x of the form a^kb, $2\leq k\leq n-2$, we have that $f(w_p)=f(w'_s)=1$, $f(w_s)=f(w'_p)=n-k-1$, and $f(w_i)=f(w'_i)=n-k$. Hence for any w we have $dev(w)=1-\frac{1\times(n-k-1)}{n-k}=\frac{1}{n-k}>\rho$. For instance, for $w=ba^{n-2}$, we have $dev(w)=1/2$. There are $2n-6=\Omega(n)$ such factors and hence at least these many ρ-overabundant words. ◀

▶ **Corollary 9.** *The (ρ_1,ρ_2)-common words in a word of length n over an alphabet of size σ are $\mathcal{O}((\sigma n)^2)$.*

Proof. By Fact 1 we know that $dev(w)$ is defined for $\mathcal{O}((\sigma n)^2)$ words. The ρ_1-avoided ones are $\mathcal{O}(\sigma n)$ [1], while the ρ_2-overabundant are $\mathcal{O}(n)$ by Corollary 6. Hence the (ρ_1,ρ_2)-common words are $\mathcal{O}((\sigma n)^2)$. ◀

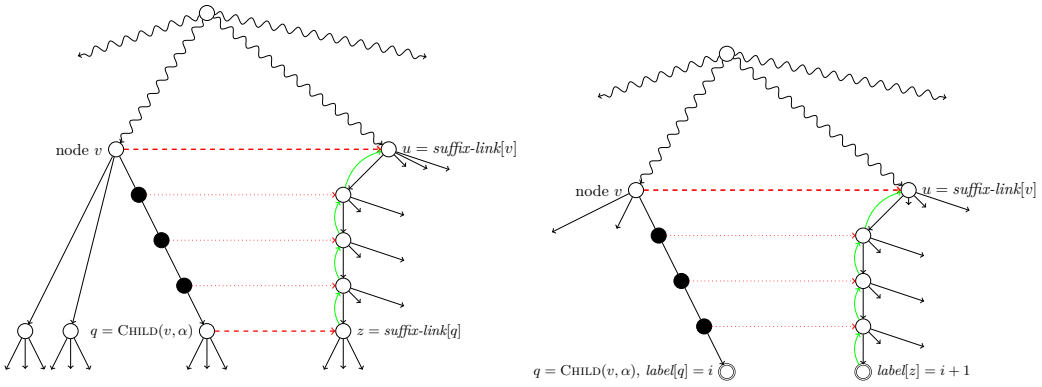

Figure 2 The above figures illustrate the nodes (implicit or explicit) considered in a step (lines 6–37) of Algorithm 1. The figure on the left presents the case where $\text{CHILD}(v, \alpha)$ is an internal node, while the right one the case that it is a leaf. Black nodes represent implicit nodes along the edge (v, q) that we have to consider as potential w_p, and the red dotted line joins them with the respective (white) explicit node that represents the longest suffix of this w_p, i.e. w_i.

4 Algorithm

Based on Fact 2 and Lemma 4 all ρ-overabundant words of a word x are factors of x of the form ayb, where $a, b \in \Sigma$ and y is the label of an explicit node of $\mathcal{T}(x)$. It thus suffices to consider these words and check for each of them whether it is ρ-overabundant. We can find the ones that have their longest proper prefix represented by an explicit node in $\mathcal{T}(x)$ easily, by taking the suffix-link from that node during a traversal of the tree. To find the ones that have their longest proper prefix represented by an implicit node we use the following fact, which follows directly from the definition of the suffix-links of the suffix tree.

▶ **Fact 10.** *Suppose aw, where $a \in \Sigma$ and $w \in \Sigma^*$, is a factor of a word x and that w is represented by an explicit node v in $\mathcal{T}(x)$, while aw by an implicit node along the edge (u_1, u_2) in $\mathcal{T}(x)$. The suffix-link from u_2 points to a node in the subtree of $\mathcal{T}(x)$ rooted at v.*

The algorithm first builds the suffix tree of word x, which can be done in time and space $\mathcal{O}(n)$ for words over an integer alphabet [10]. It is also easy to compute $\mathcal{D}(v)$ and $\mathcal{C}(v)$, for each node v of $\mathcal{T}(x)$, within the same time complexity (lines 2–5 in Algorithm 1).

The algorithm then performs a traversal of $\mathcal{T}(x)$. When it first reaches a node v, it considers $\mathcal{L}(v)$ as a potential longest proper prefix of ρ-overabundant words – i.e. $\mathcal{L}(v) = w_p = aw_i$, where $a \in \Sigma$. By following the suffix-link to node u, which represents the respective w_i, and based on the first letter of the label of each outgoing edge (v, q) from v, it computes the deviation for all possible factors of x of the form $w_p b$, where $b \in \Sigma$. (Note that we can answer all the $\text{CHILD}(u, \alpha)$ queries off-line in time $\mathcal{O}(n)$ in total for integer alphabets.) It is clear that this procedure can be implemented in time $\mathcal{O}(n)$ in total (lines 7–19).

Then, while on node v and based on Fact 10, the algorithm considers for every outgoing edge (v, q), the implicit nodes along this edge that correspond to words (potential w_p's) whose proper longest suffix (the respective w_i) is represented by an explicit node in $\mathcal{T}(x)$.

Hence, when $\mathcal{D}(q) - \mathcal{D}(v) > 1$ the algorithm follows the suffix-link from node q to node z. It then checks whether $\text{PARENT}(z) = u$. If not, then the word $\mathcal{L}(q)[0\,..\,\mathcal{D}(\text{PARENT}(z))]$ is represented by an implicit node along the edge (v, q) and hence $\mathcal{L}(q)[0\,..\,\mathcal{D}(\text{PARENT}(z)) + 1]$ has to be checked as a potential ρ-overabundant word. After the check is completed, the algorithm sets $z = \text{PARENT}(z)$ and iterates until $\text{PARENT}(z) = u$. This is illustrated in

Algorithm 1 Compute all ρ-overabundant words

```
1: procedure ComputeOverabundantWords(word x, real number ρ)
2:     T(x) ← BuildSuffixTree(x)
3:     for each node v ∈ T(x) do
4:         D(v) ← word-depth of v
5:         C(v) ← number of terminal nodes in the subtree rooted at v
6:     for each node v ∈ T(x) do                                  ▷ prefix node
7:         ▷ Report ρ-overabundant words w such that w_p is explicit
8:         u ← suffix-link[v]                                     ▷ infix node
9:         if D(v) > 1 and IsInternal(v) then
10:            f_p ← C(v), f_i ← C(u)
11:            if f_i > f_p and u ≠ Root(T(x)) then
12:                for each child y of node v do
13:                    if not(IsTerminal(y) and D(y) = D(v) + 1) then
14:                        f_w ← C(y)
15:                        α ← L(y)[D(v) + 1]
16:                        f_s ← C(Child(u, α))
17:                        E ← f_p × f_s/f_i
18:                        if (f_w − E)/(max{1, √E}) ≥ ρ then
19:                            Report(L(y)[0 .. D(v)])
20:        ▷ Report ρ-overabundant words w such that w_p is implicit
21:        for each child y of node v do
22:            if D(y) > D(v) + 1 then
23:                if IsInternal(y) then
24:                    z ← suffix-link[y]
25:                else                                           ▷ y is a terminal node
26:                    i ← label[y]
27:                    z ← node[i + 1]
28:                    if D(z) = D(Parent(z)) + 1 then
29:                        z ← Parent(z)
30:                f_w ← f_p ← C(y)
31:                while Parent(z) ≠ u do
32:                    f_i ← C(Parent(z))
33:                    f_s ← C(z)
34:                    E ← f_p × f_s/f_i
35:                    if (f_w − E)/(max{1, √E}) ≥ ρ then
36:                        Report(L(y)[0 .. D(Parent(z)) + 1])
37:                    z ← Parent(z)
```

Figure 2. By Theorem 5, the $\textsc{Parent}(z) = u$ check will fail $\mathcal{O}(n)$ times in total. All other operations take time $\mathcal{O}(1)$ and hence this procedure takes time $\mathcal{O}(n)$ in total (lines 20–37).

We formalise this procedure in Algorithm 1, where we assume that the suffix tree of $x\$$ is built, where $\$$ is a special letter, $\$ \notin \Sigma$. This forces all terminal nodes in $\mathcal{T}(x)$ to be leaf nodes. We thus obtain the following result; optimality follows directly from Lemma 8.

▶ **Theorem 11.** *Algorithm 1 solves problem* AllOverabundantWordsComputation *in time and space* $\mathcal{O}(n)$*, and this is time-optimal.*

5 Experimental Results: Effectiveness, Efficiency, and Applications

Algorithm 1 was implemented as a program to compute the ρ-overabundant words in one or more input sequences. The program was implemented in the `C++` programming

Table 1 The deviation of the randomly generated inserted word w, as well as the word $w_{\max}$ with the maximum deviation. The length of each of the 25 randomly generated sequences over $\Sigma = \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}$ was $n = 80,000$, the length of w was $m = 6$, and $\rho = 0.000001$. In green are the cases when the word with the maximum deviation was w itself or one of its factors.

Times t of inserting w	20	40	80	160	320
w	TTACAA	GTGCCC	CACTTT	AGTTAC	AAACAG
$dev(w)$	2.233313	4.143015	5.623615	6.010327	5.674220
$w_{\max}$	CTCCTATG	GTGCCC	CACTTT	AGTTA	ACAG
$dev(w_{\max})$	3.354102	4.143015	5.623615	6.900740	9.617803
w	AATCTG	AGTCGA	GAAGTC	TATCTT	CAAAAA
$dev(w)$	2.034233	2.888529	4.456468	5.073860	11.071170
$w_{\max}$	ATTGGGG	TCTGTATG	GAAGTC	ATCTT	CAAAAA
$dev(w_{\max})$	3.265609	3.272727	4.456468	6.115612	11.071170
w	GTACCA	GGCGTG	AAGGAT	GGGTCC	TTCCGG
$dev(w)$	2.187170	3.658060	4.428189	5.467296	5.256409
$w_{\max}$	TCTGTGCG	ACGATACC	AAGGAT	GGTCC	TTCCG
$dev(w_{\max})$	3.548977	4.000000	4.428189	6.787771	9.105009
w	CCATAG	GTTGAT	TGAGCG	ACATTT	CTTGTA
$dev(w)$	2.470681	2.467858	4.214544	5.755475	5.362435
$w_{\max}$	CAGTGGTC	TTTTCCT	TGAGC	ACATT	TTGTA
$dev(w_{\max})$	3.333333	3.368226	5.072968	6.376277	9.467110
w	TCGACA	CGCTTT	TACAAC	TATTAG	TGAGAT
$dev(w)$	1.531083	2.789220	3.552902	4.959926	5.124976
$w_{\max}$	CTTTGCT	ATTACC	ACAAC	ATTAG	GACAT
$dev(w_{\max})$	3.308195	3.322163	5.653479	6.837628	10.012316

language and developed under GNU/Linux operating system. Our program makes use of the implementation of the *compressed suffix tree* available in the Succinct Data Structure Library [12]. The input parameters are a (Multi)FASTA file with the input sequence(s) and a real number $\rho > 0$. The output is a file with the set of ρ-overabundant words per input sequence. The implementation is distributed under the GNU General Public License, and it is available at `http://github.com/solonas13/aw`. The experiments were conducted on a Desktop PC using one core of Intel Core i5-4690 CPU at 3.50GHz under GNU/Linux. The program was compiled with `g++` version 4.8.4 at optimisation level 3 (-O3). We also implemented a brute-force approach to confirm the correctness of our implementation. Here we do not plot the results of the brute-force approach as it is easily understood that it is orders of magnitude slower than our linear-time approach.

Experiment I (Effectiveness). In the first experiment, our task was to establish the effectiveness of the statistical model in identifying overabundant words. To this end, we generated 25 random sequences of length $n = 80,000$ over the DNA alphabet $\Sigma = \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}$ (uniform distribution). Then for each of these sequences, we inserted a random word w of length $m = 6$ in t random positions. We varied the value of t based on the fact that in a random sequence of length n over an alphabet of size $\sigma = |\Sigma|$, where letters are independent, identically uniformly distributed random variables, a specific word of length m is expected to occur roughly $r = n/\sigma^m$ times. We hence considered t equal to r, $2r$, $4r$, $8r$, and $16r$. We then ran our program for each resulting sequence to identify the ρ-overabundant words with

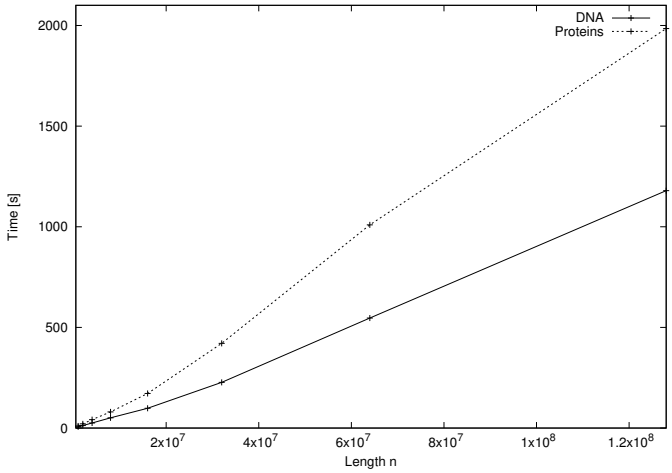

Figure 3 Elapsed time of Algorithm 1 using synthetic DNA ($\sigma = 4$) and proteins ($\sigma = 20$) sequences of length 1M to 128M.

$\rho = 0.000001$, and output the deviation of the inserted word w, as well as the word $w_{\max}$ with the maximum deviation. The inserted word w was reported as a ρ-overabundant word in *all* cases. Furthermore, in many cases the word with the maximum deviation was w itself and in many other cases one of its factors; this was true in *all* cases for $t \geq 80 \approx 4r$. Hence, the model is effective in identifying words that are overabundant. The full results of this experiment are presented in Table 1.

Experiment II (Efficiency). Our task here was to establish the fact that the elapsed time of the implementation grows linearly with n, the length of the input sequence. As input datasets, for this experiment, we used synthetic DNA ($\sigma = 4$) and proteins ($\sigma = 20$) sequences ranging from 1 to 128 M (Million letters). For each sequence we used a constant value of $\rho = 10$. The results are plotted in Fig. 3. It becomes evident from the results that the elapsed time of the program grows linearly with n. The longer time required for the proteins sequences compared to the DNA sequences for increasing n is explained by the dependence of the time required to answer queries of the form $\textsc{Child}(v, \alpha)$ on the size of the alphabet ($\sigma = 20$ *vs.* $\sigma = 4$) in the implementation of the compressed suffix tree we used.

Experiment III (Real Application). Here we proceed to the examination of seven collections of Conserved Non-coding Elements (CNEs) obtained through multiple sequence alignment between the human and other genomes. Despite being located at the non-coding part of genomes, CNEs can be extremely conserved on the sequence level across organisms. Their genesis, functions and evolutionary dynamics still remain enigmatic [16, 13]. The detailed description of how those CNEs were identified can be found in [17]. For each CNE of these datasets, a sequence stretch (surrogate sequence) of non-coding DNA of equal length and equal `GC` content was taken at random from the repeat-masked human genome. The CNEs of each collection were concatenated into a single long sequence and the same procedure was followed for the corresponding surrogates. We have determined through the proposed algorithm the overabundant words for $k = 10$ (decamers) and $\rho = 3$ for these fourteen datasets and the results are presented in Table 2. Likewise, in Table 3, we show all overabundant words (i.e. $k > 2$) for $\rho = 3$.

Table 2 Number of overabundant words for $k = 10$ and $\rho = 3$.

$k = 10$, $\rho = 3$	CNEs 75–80	CNEs 80–85	CNEs 85–90	CNEs 90–95	CNEs 95–100	Mammalian	Amniotic
Surr	1,144	718	473	297	469	15,470	2,874
CNEs	331	181	100	59	71	491	149
Ratio	**3.46**	**3.97**	**4.73**	**5.03**	**6.61**	**31.51**	**19.29**

Table 3 Number of overabundant words for $k > 2$ and $\rho = 3$.

$k > 2$, $\rho = 3$	CNEs 75–80	CNEs 80–85	CNEs 85–90	CNEs 90–95	CNEs 95–100	Mammalian	Amniotic
Surr	5,925	3,798	2,770	1,948	2,405	69,022	12,913
CNEs	1,373	778	512	390	403	7,549	1,401
Ratio	**4.32**	**4.88**	**5.41**	**4.99**	**5.97**	**9.14**	**9.22**

The first five CNE collections have been composed through multiple sequence alignment of the same set of genomes (human vs. chicken; mapped on the human genome) and they differ only in the thresholds of sequence similarity applied between the considered genomes: from 75% to 80% (the least conserved CNEs, which thus are expected to serve less demanding functional roles) to 95–100% which represent the extremely conserved non-coding elements (UCNEs or CNEs 95–100) [17]. The remaining two collections have been composed under different constraints and have been derived after alignment of Mammalian and Amniotic genomes. In Tables 2 and 3, the last line shows the ratios formed by the numbers of overabundant words of each concatenate of surrogates divided by the numbers of overabundant words of the corresponding CNE dataset.

Inspecting data contained in Tables 2 and 3, first we observe in all cases that absolute numbers of overabundant words drop from low- to high-conserved CNE concatenates. This feature is shared by the corresponding concatenates of surrogate sequences as evidenced along table rows from CNEs 75–80 to CNEs 95–100. This is due to the considerable decrease in absolute numbers of the corresponding elements in the human genome, which is reflected to the length of their concatenates. Note that in genomic sequences, extreme conservation is always clearly less frequent than medium conservation. As the studied sequences decrease in length, the numbers of overabundant words also drop in each category (CNEs or surrogates). Consequently, the important quantity is the ratio of these numbers between CNE and surrogate dataset. As amniotic and mammalian CNEs are classes characterized by different conservation thresholds (the former being much more conserved), they also present disparate overabundant word numbers, again the corresponding ratios being the relevant quantities.

Two results directly related to our analysis stem from inspection of Tables 2 and 3:

1. In all cases, the number of overabundant words from the surrogate concatenate of sequences *far exceeds* the corresponding number derived from the CNE dataset.
2. In the case of datasets with increasing degree of similarity between aligned genomes (from 75–80 to 95–100), the ratios of the numbers of overabundant words show a clear, *increasing trend*.

Both these findings can be understood on the basis of the difference in functionality between CNE and surrogate datasets. As we briefly describe in Section 1, this systematic difference (finding 1 above) is expected on the basis of the self-enhancing elongation of relatively long homonucleotide tracts [14, 15], which occurs mainly in the non-constrained

parts of the genome, here the surrogate datasets. Therefore, we expect and we do find that CNE datasets always have less overabundant words than their corresponding surrogate. Moreover, finding 2 corroborates the proposed mechanism of overabundance, as in CNE datasets 1–5 depletion in overabundant words quantitatively follows the degree of sequence conservation. Inspection of the individual overabundant words found in the surrogate datasets verifies that they largely consist of short repeats of the types described in [14] and in [15]. There is an analogy of this finding with a corresponding one, concerning the occurrence of avoided words in the same sequence sets, which is described in [1].

6 Open Question

By Corollary 6 and Lemma 8, we have the following bounds on the maximum number $\mathsf{OW}(n,\sigma)$ of overabundant words in a sequence of length n over an alphabet of size $\sigma > 1$:

$$2n - 6 \leq \mathsf{OW}(n,\sigma) \leq 3n - 2 - 2\sigma.$$

We have conducted computational experiments, and for $\sigma > 2$ we obtained sequences with more than $2n$ overabundant words. An open problem is thus to find $\mathsf{OW}(n,\sigma)$.

References

1 Yannis Almirantis, Panagiotis Charalampopoulos, Jia Gao, Costas S. Iliopoulos, Manal Mohamed, Solon P. Pissis, and Dimitris Polychronopoulos. On avoided words, absent words, and their application to biological sequence analysis. *Algorithms for Molecular Biology*, 12(1):5, 2017.

2 Alberto Apostolico, Mary Ellen Bock, and Stefano Lonardi. Monotony of surprise and large-scale quest for unusual words. *Journal of Computational Biology*, 10(3-4):283–311, 2003.

3 Alberto Apostolico, Mary Ellen Bock, Stefano Lonardi, and Xuyan Xu. Efficient detection of unusual words. *Journal of Computational Biology*, 7(1-2):71–94, 2000.

4 Alberto Apostolico, Fang-Cheng Gong, and Stefano Lonardi. Verbumculus and the discovery of unusual words. *Journal of Computer Science and Technology*, 19(1):22–41, 2004.

5 Djamal Belazzougui and Fabio Cunial. Space-efficient detection of unusual words. In *SPIRE*, volume 9309 of *LNCS*, pages 222–233. Springer, 2015.

6 Volker Brendel, Jacques S Beckmann, and Edward N Trifonov. Linguistics of nucleotide sequences: morphology and comparison of vocabularies. *Journal of Biomolecular Structure and Dynamics*, 4(1):11–21, 1986.

7 Chris Burge, Allan M. Campbello, and Samuel Karlin. Over- and under-representation of short oligonucleotides in DNA sequences. *Proc Natl Acad Sci USA*, 89(4):1358–1362, 1992.

8 Maxime Crochemore, Christophe Hancart, and Thierry Lecroq. *Algorithms on strings*. 2007.

9 Alain Denise, Mireille Régnier, and Mathias Vandenbogaert. Assessing the statistical significance of overrepresented oligonucleotides. In *WABI*, volume 2149 of *LNCS*, pages 85–97. Springer Berlin Heidelberg, 2001.

10 Martin Farach. Optimal suffix tree construction with large alphabets. In *FOCS*, pages 137–143. IEEE, 1997.

11 Mikhail S. Gelfand and Eugene V. Koonin. Avoidance of palindromic words in bacterial and archaeal genomes: a close connection with restriction enzymes. *Nucleic Acids Research*, 25(12):2430–2439, 1997.

12 Simon Gog, Timo Beller, Alistair Moffat, and Matthias Petri. From theory to practice: Plug and play with succinct data structures. In *SEA*, volume 8504 of *LNCS*, pages 326–337. Springer, 2014.

13 Nathan Harmston, Anja Barešić, and Boris Lenhard. The mystery of extreme non-coding conservation. *Phil. Trans. R. Soc. B*, 368(1632):20130021, 2013.

14 Suzanne E. Hile and Kristin A. Eckert. Positive correlation between DNA polymerase α-primase pausing and mutagenesis within polypyrimidine/polypurine microsatellite sequences. *Journal of Molecular Biology*, 335(3):745–759, 2004.

15 G. Levinson and G. A. Gutman. Slipped-strand mispairing: a major mechanism for DNA sequence evolution. *Molecular Biology and Evolution*, 4(3):203–221, 1987.

16 Dimitris Polychronopoulos, Diamantis Sellis, and Yannis Almirantis. Conserved noncoding elements follow power-law-like distributions in several genomes as a result of genome dynamics. *PloS One*, 9(5):e95437, 2014.

17 Dimitris Polychronopoulos, Emanuel Weitschek, Slavica Dimitrieva, Philipp Bucher, Giovanni Felici, and Yannis Almirantis. Classification of selectively constrained DNA elements using feature vectors and rule-based classifiers. *Genomics*, 104(2):79–86, 2014.

18 Ivan Rusinov, Anna Ershova, Anna Karyagina, Sergey Spirin, and Andrei Alexeevski. Lifespan of restriction-modification systems critically affects avoidance of their recognition sites in host genomes. *BMC Genomics*, 16(1):1, 2015.