Linear-Time Sequence Comparison Using Minimal Absent Words & Applications

Maxime Crochemore¹, Gabriele Fici², Robert Mercas¹³, and Solon P. Pissis¹

¹ Department of Informatics, King’s College London, UK
² Dipartimento di Matematica e Informatica, Università di Palermo, Italy
³ Department of Computer Science, Kiel University, Germany
maxime.crochemore@kcl.ac.uk, gabriele.fici@unipa.it,
rgm@informatik.uni-kiel.de, solon.pissis@kcl.ac.uk

Abstract. Sequence comparison is a prerequisite to virtually all comparative genomic analyses. It is often realised by sequence alignment techniques, which are computationally expensive. This has led to increased research into alignment free techniques, which are based on measures referring to the composition of sequences in terms of their constituent patterns. These measures, such as q-gram distance, can usually be computed in time linear with respect to the length of the sequences. In this article, we focus on the complementary idea: how two sequences can be efficiently compared based on information about what does not occur in the sequences. In particular, we present a linear-time and linear-space algorithm to compare two sequences by considering all their minimal absent words. An absent word is minimal if all its proper factors occur in the sequence. In the process, we present some results of combinatorial interest, and also extend the proposed techniques to circular sequences.

1 Introduction

Sequence comparison is an important step in many basic tasks in bioinformatics; from phylogenies reconstruction to genome assembly. It is often realised by sequence alignment techniques, which are computationally expensive, requiring quadratic time in the length of the sequences. This has led to increased research into alignment free techniques. Hence standard notions for sequence comparison are gradually being complemented and in some cases replaced by alternative ones [1]. One such notion is based on comparing the words that are absent in each sequence [2]. A word is an absent word (or a forbidden word) of some sequence if it does not occur in the sequence. Absent words represent a type of negative information: information about what does not occur in the sequence.

Given a sequence of length n, the number of absent words of length at most n is exponential in n. However, the number of certain classes of absent words is only linear in n. A minimal absent word of a sequence is an absent word whose proper factors all occur in the sequence [3]. An upper bound on the number of minimal absent words is known to be O(σn) [4], where σ is the size of the alphabet. Hence it may be possible to compare sequences in time proportional to
their lengths, for a fixed-sized alphabet, instead of proportional to the product of their lengths. In what follows, we consider sequences on a fixed-sized alphabet.

From an algorithmic perspective, an $O(n)$-time and $O(n)$-space algorithm for computing all minimal absent words on a fixed-sized alphabet based on the construction of suffix automata was presented in [4]. A space-efficient method to compute all minimal absent words in time $O(n)$ using succinct representations of the bidirectional BWT was recently presented in [6]. The computation of minimal absent words based on the construction of suffix arrays was considered in [7]; although this algorithm has a linear-time performance in practice, the worst-case time complexity is $O(n^2)$. A new $O(n)$-time and $O(n)$-space suffix-array-based algorithm was recently presented in [8] to bridge this unpleasant gap. An implementation of this algorithm is currently, and to the best of our knowledge, the fastest available for the computation of minimal absent words.

In this article, we consider the problem of comparing two sequences $x$ and $y$ of respective lengths $m$ and $n$, using their sets of minimal absent words. In [9], Chairungsee and Crochemore presented an alternative linear-time and linear-space algorithm for finding minimal absent words of length at most $\ell = O(1)$, based on the construction of tries of bounded-length factors. The authors also introduced a measure of similarity between two sequences based on the notion of minimal absent words. They made use of a length-weighted index to provide a measure of similarity between two sequences, using sample sets of their minimal absent words, by considering the length of each member in the symmetric difference of these sample sets. This measure can be trivially computed in time and space $O(m + n)$ provided that these sample sets contain minimal absent words of bounded length $\ell$. For unbounded length, it can be trivially computed in time $O(m^2 + n^2)$: for a given word, the cumulative length of all its minimal absent words can grow quadratically with respect to the length of the sequence.

The same problem can be considered for two circular sequences. The measure of similarity of Chairungsee and Crochemore can be used in this setting provided that one extends the definition of minimal absent words to circular sequences. In Section 4, we give a definition of minimal absent words for a circular sequence from the Formal Language Theory point of view. We believe that this definition may also be of interest from the point of view of Symbolic Dynamics, which is the original context in which minimal absent words have been defined [3].

Our Contribution. Here we make the following threefold contribution:

a) We present an $O(m + n)$-time and $O(m + n)$-space algorithm to compute the similarity measure introduced by Chairungsee and Crochemore by considering all minimal absent words of two sequences $x$ and $y$ of length $m$ and $n$, respectively; thereby showing that it is possible to compare two sequences in time proportional to their lengths (Section 3).

b) We show how this algorithm can be applied to compute this similarity measure for two circular sequences $x$ and $y$ of length $m$ and $n$, respectively, in the same time and space complexity as a result of the extension of the definition of minimal absent words to circular sequences (Section 4).

c) We provide an open-source code implementation of our algorithms and investigate potential applications of our theoretical findings (Section 5).
2 Preliminaries

To provide an overview of our result and algorithms, we begin with basic definitions. Let \( y = y[0]y[1] \ldots y[n-1] \) be a word of length \( n = \vert y \vert \) over a finite ordered alphabet \( \Sigma \) of size \( \sigma = \vert \Sigma \vert = O(1) \). For two position \( i \) and \( j \) on \( y \), we denote by \( y[i \ldots j] = y[i] \ldots y[j] \) the factor (sometimes called substring) of \( y \) that starts at position \( i \) and ends at position \( j \) (it is empty if \( j < i \)), and by \( \epsilon \) the empty word, word of length 0. We recall that a prefix of \( y \) is a factor that starts at position 0 \( (y[0 \ldots j]) \) and a suffix is a factor that ends at position \( n-1 \) \( (y[i \ldots n-1]) \), and that a factor of \( y \) is a proper factor if it is not the empty word or \( y \) itself. The set of all the factors of the word \( y \) is denoted by \( \mathcal{F}_y \).

Let \( x \) be a word of length \( 0 < m \leq n \). We say that there exists an occurrence of \( x \) in \( y \), or, more simply, that \( x \) occurs in \( y \), when \( x \) is a factor of \( y \). Every occurrence of \( x \) can be characterised by a starting position in \( y \). Thus we say that \( x \) occurs at the starting position \( i \) in \( y \) when \( x = y[i \ldots i + m - 1] \). Opposingy, we say that the word \( x \) is an absent word of \( y \) if it does not occur in \( y \). The absent word \( x, m \geq 2 \), of \( y \) is minimal if and only if all its proper factors occur in \( y \). The set of all minimal absent words for a word \( y \) is denoted by \( \mathcal{M}_y \). For example, if \( y = abaab \), then \( \mathcal{M}_y = \{aaa, aaba, bab, bb\} \). In general, if we suppose that all the letters of the alphabet appear in \( y \) of length \( n \), the length of a minimal absent word of \( y \) lies between 2 and \( n + 1 \). It is equal to \( n + 1 \) if and only if \( y \) is the concatenation of \( n \) copies of the same letter. So, if \( y \) contains occurrences of at least two different letters, the length of a minimal absent word for \( y \) is bounded by \( n \).

A language over the alphabet \( \Sigma \) is a set of finite words over \( \Sigma \). A language is regular if it is recognised by a finite state automaton. A language is factorial if it contains all the factors of its words. A language is antifactorial if no word in the language is a proper factor of another word in the language. Given a word \( x \), the language generated by \( x \) is the language \( x^* = \{x^k \mid k \geq 0\} = \{\epsilon, x, xx, xxx, \cdots\} \). The factorial closure of a language \( L \) is the language \( \mathcal{F}_L = \{y \mid y \in L\} \). Given a factorial language \( L \), one can define the antifactorial language of minimal absent words for \( L \) as \( \mathcal{M}_L = \{aub \mid aub \notin L, au, ub \in L\} \). Notice that \( \mathcal{M}_L \) is not the same language as the union of \( \mathcal{M}_x \) for \( x \in L \).

We denote by \( \text{SA} \) the suffix array of \( y \), that is the array of length \( n \) of the starting positions of all sorted (in lexicographic order) suffixes of \( y \), i.e., for all \( 1 \leq r < n \), we have \( y[\text{SA}[r-1] \ldots n-1] < y[\text{SA}[r] \ldots n-1] \). Let \( \text{lcp}(r, s) \) denote the length of the longest common prefix between the words \( y[\text{SA}[r] \ldots n-1] \) and \( y[\text{SA}[s] \ldots n-1] \), for all positions \( r, s \) on \( y \), and 0 otherwise. We denote by \( \text{LCP} \) the longest common prefix array of \( y \) defined by \( \text{LCP}[r] = \text{lcp}(r-1, r) \), for all \( 1 \leq r < n \), and \( \text{LCP}[0] = 0 \). The inverse \( i\text{SA} \) of the array \( \text{SA} \) is defined by \( i\text{SA}[\text{SA}[r]] = r \), for all \( 0 \leq r < n \). \( \text{SA} \) \cite{11}, \( i\text{SA} \), and \( \text{LCP} \) \cite{12} of \( y \) can be computed in time and space \( O(n) \).

In what follows, as already proposed in \cite{8}, for every word \( y \), the set of minimal words associated to \( y \), denoted by \( \mathcal{M}_y \), is represented as a set of tuples \( \langle a, i, j \rangle \), where the corresponding minimal absent word \( x \) of \( y \) is defined by \( x[0] = a, a \in \Sigma, \) and \( x[i \ldots m-1] = y[i \ldots j], \) where \( j - i + 1 = m \geq 2 \). It is known that if \( \vert y \vert = n \) and \( \vert \Sigma \vert = \sigma \), then \( \vert \mathcal{M}_y \vert \leq \sigma n \), see \cite{13}.
In [9], Chairungsee and Crochemore introduced a measure of similarity between two sequences \( x \) and \( y \) based on the notion of minimal absent words. Let \( M_\ell^x \) (resp. \( M_\ell^y \)) denote the set of minimal absent words of length at most \( \ell \) of \( x \) (resp. \( y \)). The authors made use of a length-weighted index to provide a measure of the similarity between \( x \) and \( y \), using their sample sets \( M_\ell^x \) and \( M_\ell^y \), by considering the length of each member in the symmetric difference \( (M_\ell^x \triangle M_\ell^y) \) of the sample sets. For sample sets \( M_\ell^x \) and \( M_\ell^y \), they defined this index to be

\[
LW(M_\ell^x, M_\ell^y) = \sum_{w \in M_\ell^x \triangle M_\ell^y} \frac{1}{|w|^2}.
\]

This work considers the following generalised version of the same problem.

**MAW-SequenceComparison**

**Input:** a word \( x \) of length \( m \) and a word \( y \) of length \( n \)

**Output:** \( LW(M_x, M_y) \), where \( M_x \) and \( M_y \) denote the sets of minimal absent words of \( x \) and \( y \), respectively.

We also consider the aforementioned problem for two circular sequences. A circular word of length \( m \) can be viewed as a traditional linear word which has the left- and right-most letters wrapped around and stuck together in some way. Under this notion, the same circular word can be seen as \( m \) different linear words, which would all be considered equivalent. More formally, given a word \( x \) of length \( m \), we denote by \( x^{(i)} = x[i\ldots m-1]x[0\ldots i-1] \), \( 0 \leq i < m \), the \( i \)-th rotation of \( x \), where \( x^{(0)} = x \). Given two words \( x \) and \( y \), we define \( x \sim y \) if and only if there exist \( i, 0 \leq i < |x| \), such that \( y = x^{(i)} \). A **circular word** \( \tilde{x} \) is a conjugacy class of the equivalence relation \( \sim \). Given a circular word \( \tilde{x} \), any (linear) word \( x \) in the equivalence class \( \tilde{x} \) is called a linearization of the circular word \( \tilde{x} \). Conversely, given a linear word \( x \), we say that \( \tilde{x} \) is a circularization of \( x \) if and only if \( x \) is a linearization of \( \tilde{x} \). The set \( F_x \) of factors of the circular word \( \tilde{x} \) is equal to the set \( F_{x^x} \cap \Sigma^{\leq |x|} \) of factors of \( xx \) whose length is at most \( |x| \), where \( x \) is any linearization of \( \tilde{x} \).

Note that if \( x^{(i)} \) and \( x^{(j)} \) are two rotations of the same word, then the factorial languages \( F_{x^{(i)}} \) and \( F_{x^{(j)}} \), coincide, so one can unambiguously define the (infinite) language \( F_{\tilde{x}^*} \) as the language \( F_{\tilde{x}^*} \), where \( x \) is any linearization of \( \tilde{x} \).

In Section 4, we give the definition of the set \( M_{\tilde{x}} \) of minimal absent words for a circular word \( \tilde{x} \). We will prove that the following problem can be solved with the same complexity as its counterpart in the linear case.

**MAW-CircularSequenceComparison**

**Input:** a word \( x \) of length \( m \) and a word \( y \) of length \( n \)

**Output:** \( LW(M_x, M_y) \), where \( M_x \) and \( M_y \) denote the sets of minimal absent words of the circularizations \( \tilde{x} \) of \( x \) and \( \tilde{y} \) of \( y \), respectively.
3 Sequence Comparison

The goal in this section is to provide a linear-time algorithm that computes the similarity measure between two sequences defined over a fixed-sized alphabet. For this, consider two sequences \( x \) and \( y \) of lengths \( m \) and \( n \), respectively, and their associated sets of minimal absent words, \( M_x \) and \( M_y \), respectively. Next, we give a linear-time solution for the MAW-SEQUENCECOMPARISON problem. It is known from [4] and [5] that we can compute the sets \( M_x \) and \( M_y \) in linear time and space with respect to the two lengths \( m \) and \( n \), respectively. The idea of our strategy is not difficult and it basically consists of a merge sort on the sets \( M_x \) and \( M_y \), after they were ordered with the help of suffix arrays.

To this end, we construct the suffix array associated to the word \( w = xy \), together with the implicit LCP array corresponding to it. All of these structures can be constructed in time and space \( O(m + n) \), as mentioned earlier. Furthermore, we can preprocess the array LCP for range minimum queries, which we denote by \( \text{RMQ}_{\text{LCP}} \) [13]. With the preprocessing complete, the longest common prefix \( \text{LCE} \) of two suffixes of \( w \) starting at positions \( p \) and \( q \) can be computed in constant time [14], using the formula \( \text{LCE}(w, p, q) = \text{LCP}[\text{RMQ}_{\text{LCP}}(\text{iSA}[p] + 1, \text{iSA}[q])] \).

Using these data structures, it is straightforward to sort the tuples in the sets \( M_x \) and \( M_y \) lexicographically. That is, two tuples \( x_1, x_2 \in M_x \), are ordered according to the letter following their longest common prefix, or when it is not the case, with the one being the prefix, coming first. To do this, we simply go once through the suffix array associated to \( w \) and assign to each tuple in \( M_x \), respectively, \( M_y \), the rank of the suffix starting at the position indicated by its second component, in the suffix array. Obviously, since sorting an array of \( n \) distinct integers, such that each it is in \([0, n-1] \), can be done in time \( O(n) \) (using bucket sort, for example), we can sort now each of the sets of minimal absent words, taking into consideration the letter on the first position and these ranks. Thus, from now on, we assume that \( M_x = (x_0, x_1, \ldots, x_k) \) where \( x_i \) is smaller lexicographically than \( x_{i+1} \), for \( 0 \leq i < k \leq \sigma m \), and \( M_y = (y_0, y_1, \ldots, y_t) \), where \( y_j \) is smaller lexicographically than \( y_{j+1} \), for \( 0 \leq j < \ell \leq \sigma n \).

Provided these tools, we now proceed to do the merge. Thus, considering that we are analysing the \((i+1)\)th tuple in \( M_x \) and the \((j+1)\)th tuple in \( M_y \), we note that the two are equal, if and only if \( x_i[0] = y_j[0] \), and

\[
\text{LCE}(w, x_i[1], |x| + y_j[1]) \geq \ell, \quad \text{where} \quad \ell = x_i[2] - x_i[1] = y_j[2] - y_j[1].
\]

In other words, the two minimal absent words are equal if and only if their first letters coincide, they have equal length \( \ell + 1 \), and the longest common prefix of the suffixes of \( w \) starting at the positions indicated by second components of the tuples has length at least \( \ell \).

Obviously, such a strategy will empower us with the means for constructing a new set \( M_{xy} = M_x \cup M_y \). At each step, when analysing tuples \( x_i \) and \( y_j \) we proceed as following:

\[
M_{xy} = \begin{cases} M_{xy} \cup \{x_i\}, & \text{and increment } i, \quad \text{if } x_i < y_j; \\ M_{xy} \cup \{y_j\}, & \text{and increment } j, \quad \text{if } x_i > y_j; \\ M_{xy} \cup \{x_i = y_j\}, & \text{and increment both } i \text{ and } j, \quad \text{if } x_i = y_j. \end{cases}
\]
Observe that the last condition is saying that basically each common tuple is added only once to their union.

Furthermore, simultaneously with this construction we can also calculate the similarity between the words, given by $LW(\mathcal{M}_x, \mathcal{M}_y)$. Thus, at each step, when comparing the tuples $x_i$ and $y_j$, we update

$$LW(\mathcal{M}_x, \mathcal{M}_y) = \begin{cases} LW(\mathcal{M}_x, \mathcal{M}_y) + \frac{|x_i|}{|y_j|^2}, & \text{and increment } i, \text{ if } x_i < y_j; \\ LW(\mathcal{M}_x, \mathcal{M}_y) + \frac{|y_j|^2}{|x_i|^2}, & \text{and increment } j, \text{ if } x_i > y_j; \\ LW(\mathcal{M}_x, \mathcal{M}_y), & \text{and increment both } i \text{ and } j, \text{ if } x_i = y_j. \end{cases}$$

We impose the increment of both $i$ and $j$ in the case of equality as in this case we only look at the symmetric difference between the sets of minimal absent words.

As all these operations take constant time, once per each tuple in $\mathcal{M}_x$ and $\mathcal{M}_y$, it is easily concluded that the whole operation takes in the case of a fixed-sized alphabet time and space $O(m + n)$. Thus we can compute the symmetric difference between the sets of minimal absent words of two sequences defined over a fixed-sized alphabet in linear time and space with respect to the lengths of the two sequences.

**Theorem 1.** Problem MAW-SEQUENCECOMPARISON can be solved in time and space $O(m + n)$.

### 4 Circular Sequence Comparison

To the best of our knowledge, there is no definition of a minimal absent word for a circular word in literature. In this section, we discuss two possible definitions and the differences between them.

We start by recalling some basic facts about minimal absent words. For further details and references the reader is recommended [15]. Every factorial language $L$ is uniquely determined by its (antifactorial) language of minimal absent words $\mathcal{M}_L$, through the equation $L = \Sigma^* \setminus \Sigma^* \mathcal{M}_L \Sigma^*$. The converse is also true, since by the definition of a minimal absent word we have $\mathcal{M}_L = \Sigma L \cap L \Sigma \cap (\Sigma^* \setminus L)$. The previous equations define a bijection between factorial and antifactorial languages. Moreover, this bijection preserves regularity. In the case of a single (linear) word $x$, the set of minimal absent words for $x$ is indeed the antifactorial language $\mathcal{M}_F^x$. Furthermore, we can retrieve $x$ from its set of minimal absent words in linear time and space.

Recall that given a circular word $\tilde{x}$, the set $\mathcal{F}_\tilde{x}$ of factors of $\tilde{x}$ is equal to the set $\mathcal{F}_{xx} \cap \Sigma^* \Sigma^{|x|}$ of factors of $xx$ whose length is at most $|x|$, where $x$ is any linearization of $\tilde{x}$. Since a circular word $\tilde{x}$ is a conjugacy class containing all the rotations of a linear word $x$, the language $\mathcal{F}_\tilde{x}$ can be seen as the factorial closure of the set $\{x^{(i)} \mid i = 0, \ldots, |x| - 1\}$. This leads to the first definition of the set of minimal absent words for $\tilde{x}$, that is the set $\mathcal{M}_{\tilde{x}} = \{aub \mid a, b \in \Sigma, aub \notin \mathcal{F}_\tilde{x}, a, ub \in \mathcal{F}_\tilde{x}\}$. For instance, if $x = abab$, we have

$$\mathcal{M}_{\tilde{x}} = \{aaa, aabaa, aababa, abaaba, ababaa, baabab, babaab, babab, bb\}.$$
The advantage of this definition is that we can retrieve uniquely \( \tilde{x} \) from \( \mathcal{M}_{\tilde{x}} \). However, the total size of \( \mathcal{M}_{\tilde{x}} \) (that is, the sum of the lengths of its elements) can be very large, as the following lemma suggests.

**Lemma 2.** Let \( \tilde{x} \) be a circular word of length \( m > 0 \). The set \( \mathcal{M}_{\tilde{x}} \) contains precisely \( \ell \) words of maximal length \( m+1 \), where \( \ell \) is the number of distinct rotations of any linearization \( x \) of \( \tilde{x} \), that is, the cardinality of \( \{ x^{(i)} \mid i = 0, \ldots, |x| - 1 \} \).

**Proof.** Let \( x = x[0]x[1] \ldots x[m-1] \) be a linearization of \( \tilde{x} \). The word obtained by appending to \( x \) its first letter, \( x[0]x[1] \ldots x[m-1]x[0] \), belongs to \( \mathcal{M}_{\tilde{x}} \), since it has length \( m + 1 \), hence it cannot belong to \( F_{\tilde{x}} \), but its maximal proper prefix \( x = x^{(0)} \) and its maximal proper suffix \( x^{(1)} = x[1] \ldots x[m-1]x[0] \) belong to \( F_{\tilde{x}} \).

The same argument shows that for any rotation \( x^{(i)} = x[i]x[i+1] \ldots x[m-1]x[0] \ldots x[i] \) of \( x \), the word \( x[i]x[i+1] \ldots x[m-1]x[0] \ldots x[i] \) obtained appending to \( x^{(i)} \) its first letter, belongs to \( \mathcal{M}_{\tilde{x}} \).

Conversely, if a word of maximal length \( m+1 \) is in \( \mathcal{M}_{\tilde{x}} \), then its maximal proper prefix and its maximal proper suffix are words of length \( m \) in \( F_{\tilde{x}} \), so they must be consecutive rotations of \( x \).

Therefore, the number of words of maximal length \( m+1 \) in \( \mathcal{M}_{\tilde{x}} \) equals the number of distinct rotations of \( x \), hence the statement follows. \(_\square_\)

This is in sharp contrast with the situation for linear words, where the set of minimal absent words can be represented on a trie having size linear in the length of the word. Indeed, the algorithm \( \text{MF-Trie} \), introduced in [1], builds the tree-like deterministic automaton accepting the set of minimal absent words for a word \( x \) taking as input the factor automaton of \( x \), that is the minimal deterministic automaton recognising the set of factors of \( x \). The leaves of the trie correspond to the minimal absent words for \( x \), while the internal states are those of the factor automaton. Since the factor automaton of a word \( x \) has less than \( 2|x| \) states (for details, see [16]), this provides a representation of the minimal absent words of a word of length \( n \) in space \( O(\sigma n) \).

This algorithmic drawback leads us to our second definition. First, we give a combinatorial result which shows that when considering circular words it does not make sense to look at absent words obtained from more than two rotations.

**Lemma 3.** For any positive integer \( k \) and any word \( u \), the set \( V = \{ v \mid k|u|+1 < |v| \leq (k+1)|u| \} \subseteq \mathcal{M}_{u^{k+1}} \setminus \mathcal{M}_{u^k} \) is empty.

**Proof.** This obviously holds for all words \( u \) of length 1. Assume towards a contradiction that this is not the case in general. Hence, there must exist a word \( v \) of length \( m \) that fulfills the conditions in the lemma, thus \( v \in V \) and \( m > 2 \). Furthermore, there must exist positions \( i < j \leq n = |u| \) such that

\[
v[i \ldots i+m-2] = u^{k+1}[i + j + m - 2] = u^{k+1}[j + i + m - 2]. \quad (1)
\]

Obviously, following Equation (1), since \( m - 2 \geq kn \), we have that \( v[i \ldots i+m-2] \) is \((j-i)\)-periodic. But, we know that \( v[i \ldots i+m-2] \) is also \( n \)-periodic. Thus, following a direct application of the periodicity lemma we have that \( v[i \ldots i+m-2] \)

\[
= u^{k+1}[i + j + m - 2] = u^{k+1}[j + i + m - 2].
\]
is \( p = \gcd(j - i, n) \)-periodic. But, in this case we have that \( u \) is \( p \)-periodic, and, therefore, \( u[i] = u[j] \), which leads to a contradiction with the fact that \( v \) is a minimal absent word, whenever \( i \) is defined. Thus, it must be the case that \( i = -1 \). Using the same strategy and looking at positions \( u[i + m - 2] \) and \( u[j + m - 2] \), we conclude that \( j + m - 2 = (k + 1)n \). Therefore, in this case, we have that \( m = kn + 1 \), which is a contradiction with the fact that the word \( v \) fulfills the conditions of the lemma. This concludes the proof.

Observe now that the set \( V \) consists in fact of all extra minimal absent words generated whenever we look at more than one rotation, that do not include the length arguments. That is, \( V \) does not include the words bounding the maximum length that a word is allowed, nor the words created, or lost, during a further concatenation of an image of \( u \). However, when considering an iterative concatenation of the element these extra elements determined by the length constrain cancel each other.

As observed in Section 2, two rotations of the same word \( x \) generate two languages that have the same set of factors. So, we can unambiguously associate to a circular word \( \tilde{x} \), the (infinite) factorial language \( F_{\tilde{x}} \). It is therefore natural to define the set of minimal absent words for the circular word \( \tilde{x} \) as the set \( M_{F_{\tilde{x}}} \). For instance, if \( \tilde{x} = abaab \), then we have

\[
M_{F_{\tilde{x}}} = \{aaa, aabaa, babab, bb\}.
\]

This second definition is much more efficient in terms of space, as we show below. In particular, the length of the words in \( M_{F_{\tilde{x}}} \) is bounded by \( |x| \), and therefore \( M_{F_{\tilde{x}}} \) is a finite set.

Recall that a word \( x \) is a power of a word \( y \) if there exists a positive integer \( k > 1 \) such that \( x \) is expressed as \( k \) consecutive catenations of \( y \), denoted by \( x = y^k \). Conversely, a word \( x \) is primitive if \( x = y^k \) implies \( k = 1 \). Notice that a word is primitive if and only if any of its rotation is. We can therefore extend the definition of primitivity to circular words. The definition of \( M_{F_{\tilde{x}}} \) does not allow one to uniquely reconstruct \( \tilde{x} \) from \( M_{F_{\tilde{x}}} \), unless \( \tilde{x} \) is known to be primitive, since it is readily verified that \( F_{\tilde{x}} = F_{\tilde{x}\tilde{x}} \) and therefore also the minimal absent words of these two languages coincide. However, from the algorithmic point of view, this issue can be easily managed by storing the length \( |x| \) of a linearization \( x \) of \( \tilde{x} \) together with the set \( M_{F_{\tilde{x}}} \). Moreover, in most practical cases, for example when dealing with biological sequences, it is highly unlikely that the circular word considered is not primitive.

The difference between the two definitions above is presented in the next lemma.

**Lemma 4.** \( M_{F_{\tilde{x}}} = M_{F_{\tilde{x}}} \cap \Sigma^{\leq |x|} \).

**Proof.** Clearly, \( F_{\tilde{x}} \cap \Sigma^{\leq |x|} = F_{\tilde{x}} \). The statement then follows from the definition of minimal absent words. \( \square \)

Based on the previous discussion, we set \( M_{\tilde{x}} = M_{F_{\tilde{x}}} \). The following lemma comes straightforwardly from the definition and Lemma 4.
Lemma 5. Let $\tilde{x}$ be a circular word. Then $M_{\tilde{x}} = M_{x}^{[x]}$.

Using Lemma 5, we can easily extend the algorithm described in the previous section to the case of circular words. That is, given two circular sequences $\tilde{x}$ of length $m$ and $\tilde{y}$ of length $n$, we can compute in time and space $O(m+n)$ the quantity $LW(M_{\tilde{x}}, M_{\tilde{y}})$.

Theorem 6. Problem MAW-CircularSequenceComparison can be solved in time and space $O(m+n)$.

5 Implementation and Application

We implemented the presented algorithms as programme scMAW to perform pairwise sequence comparison for a set of sequences using minimal absent words. scMAW uses programme MAW [8] for linear-time computation of minimal absent words using suffix array. scMAW was implemented in the C programming language and developed under GNU/Linux operating system. It takes, as input argument, a file in MultiFASTA format with the input sequences, and then any of the two methods, for linear or circular sequence comparison, can be applied. It then produces a file in PHYLIP format with the distance matrix as output. Cell $[x, y]$ of the matrix stores $LW(M_{x}, M_{y})$ (or $LW(M_{\tilde{x}}, M_{\tilde{y}})$). The implementation is distributed under the GNU General Public License (GPL), and it is available at http://github.com/solonas13/maw, which is set up for maintaining the source code and the man-page documentation. Notice that all input datasets and the produced outputs referred to in this section are publicly maintained at the same web-site.

An important feature of the proposed algorithms is that they require space linear in the length of the sequences (see Theorem 1 and Theorem 6). Hence, we were also able to implement scMAW using the Open Multi-Processing (OpenMP) PI for shared memory multiprocess programming to distribute the workload across the available processing threads without a large memory footprint.

Application

Recently, there has been a number of studies on the biological significance of absent words in various species [2,17,18,19]. For instance, dendrograms from dinucleotide relative abundances in sets of minimal absent words for prokaryotes and eukaryotic genomes were presented in [17]. The analyses support the hypothesis that minimal absent words are inherited through a common ancestor, in addition to lineage-specific inheritance, only in vertebrates. Very recently, in [18], it was shown that there exist three minimal words in the Ebola virus genomes which are absent from human genome. The authors suggest that the identification of such species-specific sequences may prove to be useful for the development of both diagnosis and therapeutics.

In this section, we show a potential application of our results for the construction of dendrograms for DNA sequences with circular structure. Circular DNA
sequences can be found in viruses, as plasmids in archaea and bacteria, and in the mitochondria and plastids of eukaryotic cells. Circular sequence comparison thus finds applications in several contexts such as reconstructing phylogenies using viroids RNA [20] or Mitochondrial DNA (MtDNA) [21]. Conventional tools to align circular sequences could yield an incorrectly high genetic distance between closely-related species. Indeed, when sequencing molecules, the position where a circular sequence starts can be totally arbitrary. Due to this arbitrariness, a suitable rotation of one sequence would give much better results for a pairwise alignment [22]. In what follows, we demonstrate the power of minimal absent words to pave a path to resolve this issue by applying Lemma 5 and Theorem 6. In what follows, we do not claim that a solid phylogenetic analysis is presented but rather an investigation for potential applications of our theoretical findings.

We performed the following experiment with synthetic data. First, we simulated a basic dataset of DNA sequences using INDELible [23]. The number of taxa, denoted by \( \alpha \), was set to 12; the length of the sequence generated at the root of the tree, denoted by \( \beta \), was set to 2500bp; and the substitution rate, denoted by \( \gamma \), was set to 0.05. We also used the following parameters: a deletion rate, denoted by \( \delta \), of 0.06 relative to substitution rate of 1; and an insertion rate, denoted by \( \epsilon \), of 0.04 relative to substitution rate of 1. The parameters were chosen based on the genetic diversity standard measures observed for sets of MtDNA sequences from primates and mammals [22]. We generated another instance of the basic dataset, containing one arbitrary rotation of each of the \( \alpha \) sequences from the basic dataset. We then used this randomised dataset as input to scMAW by considering \( \text{LW}(\mathcal{M}_x, \mathcal{M}_y) \) as the distance metric. The output of scMAW was passed as input to NINJA [24], an efficient implementation of neighbor-joining [25], a well-established hierarchical clustering algorithm for inferring dendrograms (trees). We thus used NINJA to infer the respective tree \( T_1 \) under the neighbor-joining criterion. We also inferred the tree \( T_2 \) by following the same pipeline, but by considering \( \text{LW}(\mathcal{M}_x, \mathcal{M}_y) \) as distance metric, as well as the tree \( T_3 \) by using the basic dataset as input of this pipeline and \( \text{LW}(\mathcal{M}_x, \mathcal{M}_y) \) as distance metric. Hence, notice that \( T_3 \) represents the original tree. Finally, we computed the pairwise Robinson-Foulds (RF) distance [26] between: \( T_1 \) and \( T_3 \); and \( T_2 \) and \( T_3 \). Let us define accuracy as the difference between 1 and the relative pairwise RF distance. We repeated this experiment by simulating different datasets \( < \alpha, \beta, \gamma, \delta, \epsilon > \) and measured the corresponding accuracy. The results in Table 1 suggest that by considering \( \text{LW}(\mathcal{M}_x, \mathcal{M}_y) \) we can always re-construct the original tree even if the sequences have first been arbitrarily rotated (Lemma 5). This is not the case if we consider \( \text{LW}(\mathcal{M}_x, \mathcal{M}_y) \). Notice that 100% accuracy denotes a (relative) pairwise RF distance of 0.

6 Final Remarks

In this article, complementary to measures that refer to the composition of sequences in terms of their constituent patterns, we considered sequence compar-
ison using minimal absent words, information about what does not occur in the sequences. We presented a linear-time and linear-space algorithm to compare two sequences by considering all their minimal absent words (Theorem 1). In the process, we presented some results of combinatorial interest, and also extended the proposed techniques to circular sequences. The power of minimal absent words is that they provide a tool for sequence comparison that is as efficient for circular as it is for linear sequences (Lemma 4 and Theorem 6); whereas, this is not the case, for instance, using the edit distance model [27]. Finally, a preliminary experimental study shows the potential of our theoretical findings.

Acknowledgements

We warmly thank Alice Heliou for her inestimable code contribution and Antonio Restivo for useful discussions. Gabriele Fici’s work was supported by the PRIN 2010/2011 project “Autonomy e Linguaggi Formali: Aspetti Matematici e Applicativi” of the Italian Ministry of Education (MIUR). Robert Mercas’s work was supported by the P.R.I.M.E. programme of DAAD co-funded by BMBF and EU’s 7th Framework Programme (grant 605728). Solon P. Pissis’s work was supported by a Research Grant (#RG130720) awarded by the Royal Society.

References

1. Domazet-Lošo, M., Haubold, B.: Efficient estimation of pairwise distances between genomes. Bioinformatics 25(24) (2009) 3221–3227
2. Acquisti, C., Poste, G., Curtis, D., Kumar, S.: Nullomers: Really a matter of natural selection? PLoS ONE 2(10) (2007)
3. Béal, M., Mignosi, F., Restivo, A., Sciortino, M.: Forbidden words in symbolic dynamics. Advances in Applied Mathematics 25(2) (2000) 163–193
4. Crochemore, M., Mignosi, F., Restivo, A.: Automata and forbidden words. Information Processing Letters 67 (1998) 111–117
5. Mignosi, F., Restivo, A., Sciortino, M.: Words and forbidden factors. Theoretical Computer Science 273(1-2) (2002) 99–117
6. Belazzougui, D., Cunial, F., Kärkkäinen, J., Mäkinen, V.: Versatile succinct representations of the bidirectional Burrows–Wheeler transform. In: ESA. Volume 8125 of LNCS., Springer (2013) 133–144
7. Pinho, A.J., Ferreira, P.J.S.G., Garcia, S.P.: On finding minimal absent words. BMC Bioinformatics 11 (2009)
8. Barton, C., Heliou, A., Mouchard, L., Pissis, S.P.: Linear-time computation of minimal absent words using suffix array. BMC Bioinformatics 15 (2014) 388
9. Chairungsee, S., Crochemore, M.: Using minimal absent words to build phylogeny. Theoretical Computer Science 450(0) (2012) 109–116
10. Manber, U., Myers, E.W.: Suffix arrays: A new method for on-line string searches. SIAM J. Comput. 22(5) (1993) 935–948
11. Nong, G., Zhang, S., Chan, W.H.: Linear suffix array construction by almost pure induced-sorting. In: Data Compression Conference. IEEE (2009) 193–202
12. Fischer, J.: Inducing the LCP-array. In Dehne, F., Iacono, J., Sack, J.R., eds.: Algorithms and Data Structures. Volume 6844 of LNCS., Springer (2011) 374–385
13. Fischer, J., Heun, V.: Space-efficient preprocessing schemes for range minimum queries on static arrays. SIAM J. Comput. 40(2) (2011) 465–492
14. Ilie, L., Navarro, G., Tinta, L.: The longest common extension problem revisited and applications to approximate string searching. Journal of Discrete Algorithms 8(4) (2010) 418–428
15. Fici, G.: Minimal Forbidden Words and Applications. PhD thesis, Université de Marne-la-Vallée (2006)
16. Crochemore, M., Hancart, C., Lecroq, T.: Algorithms on Strings. Cambridge University Press, New York, NY, USA (2007)
17. Garcia, S.P., Pinho, O.J., Rodrigues, J.M.O.S., Bastos, C.A.C., G, P.J.S.: Minimal absent words in prokaryotic and eukaryotic genomes. PLoS ONE 6 (2011)
18. Silva, R.M., Pratas, D., Castro, L., Pinho, A.J., Ferreira, P.J.S.G.: Three minimal sequences found in ebola virus genomes and absent from human DNA. Bioinformatics (2015)
19. Garcia, S.P., Pinho, A.J.: Minimal Absent Words in Four Human Genome Assemblies. PLoS ONE 6(12) (2011)
20. Mosig, A., Hofacker, I.L., Stadler, P.F.: Comparative Analysis of Cyclic Sequences: Viroids and other Small Circular RNAs. In Huson, D.H., Kohlbacher, O., Lupas, A.N., Nieselt, K., Zell, A., eds.: German Conference on Bioinformatics. Volume 83 of LNI., GI (2006) 93–102
21. Goios, A., Pereira, L., Bogue, M., Macaulay, V., Amorim, A.: mtDNA phylogeny and evolution of laboratory mouse strains. Genome Res. 17(3) (2007) 293–298
22. Barton, C., Iliopoulos, C.S., Kundu, R., Pissis, S.P., Retha, A., Vayani, F.: Accurate and efficient methods to improve multiple circular sequence alignment. In: 14th SEA. Volume 9125 of LNCS. Springer (2015) 247–258
23. Fletcher, W., Yang, Z.: INDELible: A flexible simulator of biological sequence evolution. Molecular Biology and Evolution 26(8) (2009) 1879–1888
24. Wheeler, T.: Large-scale neighbor-joining with NINJA. In Salzberg, S.L., Warnow, T., eds.: Algorithms in Bioinformatics. Volume 5724 of LNCS. (2009) 375–389
25. Saitou, N., Nei, M.: The neighbor-joining method: a new method for reconstructing phylogenetic trees. Molecular Biology and Evolution 4(4) (1987) 406–425
26. Robinson, D., Foulds, L.: Comparison of phylogenetic trees. Mathematical Biosciences 53(1-2) (1981) 131–147
27. Maes, M.: On a cyclic string-to-string correction problem. IPL 35(2) (1990) 73–78