Theoretical Computer Science 412 (2011) 3019–3032

Contents lists available at ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

Balancing and clustering of words in the Burrows–Wheeler transform

Antonio Restivo *, Giovanna Rosone

University of Palermo, Dipartimento di Matematica ed Applicazioni, Via Archirafi 34, 90123 Palermo, Italy

ARTICLE INFO

Keywords:
Combinatorics on words
Burrows–Wheeler transform
Data compression

ABSTRACT

Compression algorithms based on Burrows–Wheeler transform (BWT) take advantage of the fact that the word output of BWT shows a local similarity and then turns out to be highly compressible. The aim of the present paper is to study such "clustering effect" by using notions and methods from Combinatorics on Words.

The notion of balance of a word plays a central role in our investigation. Empirical observations suggest that balance is actually the combinatorial property of input word that ensure optimal BWT compression. Moreover, it is reasonable to assume that the more balanced the input word is, the more local similarity we have after BWT (and therefore the better the compression is). This hypothesis is here corroborated by experiments on "real" text, by using local entropy as a measure of the degree of balance of a word.

In the setting of Combinatorics on Words, a sound confirmation of previous hypothesis is given by a result of Mantaci et al. (2003) [27], which states that, in the case of a binary alphabet, there is an equivalence between circularly balanced words, words having a clusterized BWT, and the conjugates of standard words. In the case of alphabets of size greater than two, there is no more equivalence. The last section of the present paper is devoted to investigate the relationships between these notions, and other related ones (as, for instance, palindromic richness) in the case of a general alphabet.

1. Introduction

Michael Burrows and David Wheeler introduced in 1994 (cf. [9]) a reversible transformation on words that turns out to be an extremely useful tool for textual data compression. Compression algorithms based on the Burrows–Wheeler Transform (BWT) take advantage of the fact that the word output of BWT shows a local similarity (occurrences of a given symbol tend to occur in clusters) and then turns out to be highly compressible. Several authors refer to such property as the "clustering effect" of BWT. The aim of this paper is to approach in a formal setting, and in a quantitative way, the investigation of the "clustering effect" of BWT, and its consequences on the performances of the BWT-based compressors.

Several papers (cf. [30,14,23,24]) prove analytical upper bounds on the compression ratio of BWT-based compressors in terms of the kth order empirical entropy H_k of the input string. Recall that, under the hypothesis of the Markovian nature of the input string w, $H_k(w)$ gives a lower bound on the compression ratio of any encoder that is allowed to use only the context of length k preceding character σ in order to encode it. Kaplan et al. report in [23] some empirical results which seem to indicate that achieving good bounds with respect to H_k does not necessarily guarantee good compression results in practice. So they ask the question *whether there is another statistic (more appropriate than H_k) that actually capture the compressibility of the input text*.

Moreover, in [24] Kaplan and Verbin prove the non-optimality of the most of BWT-based algorithms, but they observe that such compressors work well in practice (in particular on English text). They believe that BWT-compressors work on

* Corresponding author. Tel.: +39 091 6040307; fax: +39 091 6040311.
E-mail addresses: restivo@math.unipa.it (A. Restivo), giovanna@math.unipa.it (G. Rosone).

0304-3975/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2010.11.040

English text better than Dictionary-based compressors because of the non-Markovian elements in English text and they ask the following question: *what kind of regularity is there in English text that compressors exploit*?

In this paper we propose an answer to the questions of [24] and [23]. In particular, we believe that "the regularity of the English text that BWT-compressors exploit" is related to the *balancing* properties of the text itself. We recall that a word w is *balanced* if, roughly speaking, the frequency of each symbol in different blocks of w is almost the same. Remark that this property is actually satisfied, with a very good approximation, by an English text, or by a text written in another natural language.

Moreover, empirical observations suggest the following hypothesis: *the more balanced the input word is, the more local similarity we have after* BWT, and, as a consequence, the better the compression is.

With the purpose to test in a quantitative way such hypothesis and to answer to the related question of [23] ("which is the statistic that actually captures the compressibility of the input text") we introduce the notion of *local entropy*, as a measure of the degree of balance of a text. Remark that as the BWT can be thought as a transformation acting on circular words, all the notions we introduce for our analysis are relative to circular words. So, in Section 5, we introduce the formal definitions concerning the balancing of a word w and the definition of Local Entropy $LE(w)$.

The most optimal situation for the balancing of a word occurs when the distance between any two consecutive occurrences of a given symbol is a constant throughout the word. Such words, called *constant gap words*, are known to be at the root of a more general class of words, called *balanced words*. We recall that a word w is balanced if, for any symbol a, the number of a's in two blocks of w of the same length differs by at most 1. The opposite extremal case, with respect to balancing properties, is represented by the class of *clustered words*, i.e. words where the number of runs (consecutive occurrences of the same symbol) is equal to the size of the alphabet.

For any word w, we prove tight lower and upper bounds of $LE(w)$ and, moreover, we prove that (i) $LE(w)$ reaches its maximum if and only if w is a constant gap word and (ii) $LE(w)$ reaches its minimum if and only if w is a clustered word. This shows that local entropy actually provides a measure of the degree of balance of a word. On the other hand, these results have an independent interest since they give a characterization of constant gap (and clustered) words in terms of local entropy.

By using $LE(w)$ as a measure of balance of a word w, we perform, in Section 6, some experiments on "real" text, in order to test the hypothesis that the more balanced the input word is, the more local similarity we have after BWT. The results of the experiments corroborate this hypothesis. Moreover they also suggest, as a practical application, a method to establish when to use a BWT-based compressor is more advantageous than to use a Dictionary-based compressor.

In the setting of Combinatorics on Words, a sound confirmation of our hypothesis is given, in the binary case, by a result proved in [27]. In this paper, the authors give a full characterization of words on a binary alphabet $\{a, b\}$ having the maximum amount of clustering under application of BWT, i.e. words w such that $bwt(w) = b^p a^q$, for some $p, q > 0$. Such words actually correspond to (circularly) balanced words.

It is interesting to remark that, in the binary case, (circularly) balanced words are closely related to the well known family of *Sturmian sequences* and, in particular, they correspond to the conjugates of *standard words* (cf. [26]).

In the case of words over alphabets with more than two letters, there is no more a tight equivalence between words having clusterized BWT and balancing. Remark that the structure of balanced words on arbitrary alphabets is to a large extent unknown; for instance, the Fraenkel conjecture (cf. [15]) corresponds to a special case of this general problem.

The relationship between balanced words and words having clusterized BWT are investigate in Section 8. Following [38], we introduce the class of *simple* BWT *words*, as a proper subclass of words having a clusterized BWT transform, and compare it with the class of (circularly) balanced words. Such investigation necessitates to introduce some supplementary notions from Combinatorics on Words. In particular, we need a generalization of the notion of standard word to a general alphabet.

The notion of standard word is closely related to that of Sturmian sequence. Numerous generalizations of Sturmian sequences have been introduced for an alphabet with more than 2 letters. Among them, one natural generalization are the *episturmian sequences* that are defined by using the palindromic closure property of Sturmian sequences (cf. [12]). Here we consider some special prefixes of episturmian sequences, that we call *finite epistandard* words: in the case of a binary alphabet they correspond to the finite standard words.

Another notion, related to the previous ones, that has been recently introduced is that of *(palindromic) rich word*: a word is rich if it contains the maximal number of distinct palindromic factor (cf. [18]). Rich words appear in many different contexts: they include sturmian and episturmian words, and also other families of words known in literature.

We remark that, in the case of alphabets of size greater than two, there is no more equivalence (as in the binary case) between the family of simple BWT words, the family of (circularly) balanced words and the conjugate of epistandard words. However, as a main results of Section 8, we prove that, under assumption of balancing, the following three conditions on a word w are equivalent: (i) w has simple BWT, (ii) w is a circularly rich word, and (iii) w is a conjugate of a finite epistandard word.

Apart from their interest for the study of the clustering effect of BWT (and of optimal performances of BWT-based compressors), the results of Section 8 can be considered as a contribution to combinatorics of episturmian sequences, and could provide new insight on the Fraenkel conjecture. A preliminary version of the results presented in Section 8 appear in [35].

In conclusion, the results presented in this paper deal with combinatorial properties of words applied to data compression. The main purpose of this investigation is to state a link between methods from Combinatorics on Words and techniques from data compression, in order to obtain a deeper comprehension of both research fields.

			F					L
			↓					↓
		1	a	a	b	r	a	c
I	→	2	a	b	r	a	c	a
		3	a	c	a	a	b	r
		4	b	r	a	c	a	a
		5	c	a	a	b	r	a
		6	r	a	c	a	a	b

Fig. 1. The matrix M of the word $w = abraca$.

2. Preliminaries

Let $\mathcal{A} = \{a_1, a_2, \ldots, a_k\}$ be a finite ordered alphabet (with $a_1 < a_2 < \cdots < a_k$). We denote by $\mathcal{A}^*$ the set of words over A. Given a finite word $w = b_1 b_2 \cdots b_n \in A^*$ with each $b_i \in \mathcal{A}$, the length of w, denoted $|w|$, is equal to n. By convention, the empty word ε is the unique word of length 0. We denote by n_i the number of occurrences of the letter a_i in the word w. We denote by $\tilde{w}$ the reversal of w, given by $\tilde{w} = b_n \cdots b_2 b_1$. If w is a word that has the property of reading the same in either direction, i.e. if $w = \tilde{w}$, then w is called a *palindrome*. A word has the *two palindrome property* if it can be written as uv where u and v are palindromes or empty.

We say that two words $x, y \in \mathcal{A}^*$ are *conjugate*, if $x = uv$ and $y = vu$, where $u, v \in \mathcal{A}^*$. Conjugacy between words is an equivalence relation over $\mathcal{A}^*$. The *conjugacy class* $[x]$ of $x \in \mathcal{A}^n$ is the set of all words $b_i b_{i+1} \cdots b_n b_1 \cdots b_{i-1}$, for $1 \leq i \leq n$. A conjugacy class can also be represented as a circular word. Hence in what follows we will use "circular word" and "conjugacy class" as synonyms.

A nonempty word $w \in \mathcal{A}^*$ is *primitive* if $w = u^h$ implies $w = u$ and $h = 1$. Recall that (cf. [25]) every nonempty word $u \in \mathcal{A}^*$ can be written in a unique way as a power of a primitive word, i.e. there exists a unique primitive word w, called the *root* of u, and a unique integer k such that $u = w^k$.

If u is a word in $\mathcal{A}^*$, we denote by u^ω the infinite word obtained by infinitely iterating u, i.e. $u^\omega = uuuuu \ldots$. A word $w \in \mathcal{A}^\omega$ is *ultimately periodic* of period $n \in \mathbb{N}$ if $b_i = b_{i+n}$ for each $i \geq l$ and $l \in \mathbb{N}$. If $l = 1$, then w is *purely periodic*. An infinite word that is not ultimately periodic is said to be *aperiodic*.

A word $v \in \mathcal{A}^*$ is said to be a *factor* (resp. a *prefix*, resp. a *suffix*) of a word $w \in \mathcal{A}^*$ if there exist words $x, y \in \mathcal{A}^*$ such that $w = xvy$ (resp. $w = vy$, resp. $w = xv$). A factor (resp. the prefix, resp. the suffix) is *proper* if $xy \neq \varepsilon$ (resp. $y \neq \varepsilon$, resp. $x \neq \varepsilon$). A factor u of a finite or infinite word w is said to be *left special* (resp. *right special*) in w if there exist at least two distinct letters a, b such that au and bu (resp. ua, ub) are factors of w. For any finite or infinite word w, $F(w)$ denotes the set of all its factors. We say that $F(w)$ is closed under reversal if for any $u \in F(w)$, $\tilde{u} \in F(w)$. Moreover, if w is infinite, we denote by $Ult(w)$ the set of all letters occurring infinitely often in w. A factor of an infinite word x is *recurrent* in x if it occurs infinitely often in x, and x itself is said to be *recurrent* if all of its factors are recurrent in it. Given two palindromes w, v, we say that v is a *central factor* of w if $w = uv\tilde{u}$ for some $u \in \mathcal{A}^*$.

3. The Burrows–Wheeler transform

The Burrows–Wheeler transform was introduced in 1994 by Burrows and Wheeler [9] and represents an extremely useful tool for textual lossless data compression. The idea is to apply a reversible transformation in order to produce a permutation $bwt(w)$ of an input word w, defined over an ordered alphabet $\mathcal{A}$, so that the word becomes easier to compress. Actually the transformation tends to group characters together so that the probability of finding a character close to another instance of the same character is substantially increased. BWT transforms a word $w = b_1 b_2 \cdots b_n$ of length n by lexicographically sorting all the n conjugates of w and extracting the last character of each conjugate. The word $bwt(w)$ consists of the concatenation of these characters. We denote by M the matrix which consists of all conjugates $w_1, w_2, \ldots, w_n$ of w lexicographically sorted. In what follows we will refer to M as the "Burrows–Wheeler matrix" of w. Moreover the transformation computes the index I, that is the row containing the original word in the sorted list of the conjugates.

For instance, suppose we want to compute $bwt(w)$ where $w = abraca$. Consider the Burrows–Wheeler matrix M in Fig. 1.

The last column L of the matrix M represents $bwt(w) = caraab$ and $I = 2$ since the original word w appears in row 2. The first column F, instead, contains the word of the characters of w lexicographically sorted.
Next proposition is an easy consequence of the definition of BWT (cf. [9]).

Proposition 3.1. *The following properties hold:*

1. *For all $i = 1, \ldots, n$, $i \neq I$, the character $L[i]$ is followed in the original string by $F[i]$;*
2. *For each character α, the i-th occurrence of α in F corresponds to the i-th occurrence of α in L.*

From the above properties of the BWT, it follows that the transform is reversible in the sense that, given $bwt(w)$ and the index I, it is possible to recover the original string $w = b_1 b_2 \cdots b_n$.

Actually, according to Property 2 of Proposition 3.1, we can define a permutation

$$\tau : \{1, 2, \ldots, n\} \to \{1, 2, \ldots, n\} \tag{1}$$

giving the correspondence between the positions of characters of the first and the last column of the matrix M. For instance, the permutation τ of the word w in Fig. 2 is

$$\tau = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 2 & 4 & 5 & 6 & 1 & 3 \end{pmatrix}.$$

Starting from the position I, and by using Property 1 of Proposition 3.1, we can recover the word w as follows:

$$b_i = F[\tau^{i-1}(I)], \quad \text{where } \tau^0(x) = x, \text{ and } \tau^{i+1}(x) = \tau(\tau^i(x)). \tag{2}$$

Notice that the reconstruction algorithm corresponds to decompose the permutation τ into a product of cycles. In our case there is only one cycle. For instance, the permutation τ of the word $w = abraca$ can be decompose in this way:

$$\tau = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 2 & 4 & 5 & 6 & 1 & 3 \end{pmatrix} = (2\,4\,6\,3\,5\,1).$$

The permutation τ represents also the order in which we have to rearrange the elements of F to reconstruct the original word w. We show, for instance, how the reconstruction works for the example in Fig. 2:

$$\begin{aligned} b_1 &= F[2] = a \\ b_2 &= F[4] = b \\ b_3 &= F[6] = r \\ b_4 &= F[3] = a \\ b_5 &= F[5] = c \\ b_6 &= F[1] = a. \end{aligned}$$

Notice that if we except the index, all the mutual conjugate words have the same Burrows–Wheeler Transform. Actually the index has the only aim of denoting one representative in the conjugacy class. However this index is not necessary for the construction of the matrix M from L.

In what follows we consider the problem of characterizing the words w such that their BWT has a specific form. Since two conjugate words have the same BWT, our characterizations concern conjugacy classes or, equivalently, circularly words.

Notice also that BWT is not surjective on the set $\mathcal{A}^*$, that is, there exist some words in A^* that are not the image of any word by the BWT. Consider for instance the word $u = bccaaab$. It is easy to see that there exists no word w such that $bwt(w) = u$.

4. BWT and data compression

Compression algorithms based on BWT take advantage of the fact that the word output of BWT shows a local similarity (occurrences of a given symbol tend to occur in clusters) and then turns out to be highly compressible. Several authors refer to such property as the "clustering effect" of BWT. The aim of this paper is to approach in a formal setting, and in a quantitative way, the investigation of the "clustering effect" of BWT, and its consequences on the performances of the BWT-based compressors.

Several papers prove analytical upper bounds on the compression ratio of BWT-based compressors. In [30] Manzini gave the first worst-case upper bound on the compression ratio of several BWT-based algorithms in terms of the *empirical entropy* of the input string. It is well known that the zeroth order empirical entropy of a string w, $H_0(w)$, is a lower bound on the compression ratio of any order-0 compressor. Similarly, the kth order empirical entropy of the string w, $H_k(w)$ gives a lower bound on the compression ratio of any encoder that is allowed to use only the context of length k preceding the character σ in order to encode it. For this reason, the compression ratio of compression algorithms is traditionally compared to $H_k(w)$, for various values of k.

In [14] Ferragina et al. introduced a BWT-based compression booster and proved an upper bound of such a compression algorithm improving the result of [30]. Moreover, they proved that this upper bound is optimal (for details see [14]).

Kaplan et al. observed in [23] that the empirical results (see [23, Table 1]), obtained by implementing the algorithm in [14], surprisingly imply that while the algorithm of [14] is optimal with respect to H_k in a worst-case setting, its compression ratio in practice is comparable with that of algorithms with weaker worst-case guarantees. This seems to indicate that achieving good bounds with respect to H_k does not necessarily guarantee good compression results in practice. So they ask the question *whether there is another statistic (more appropriate than H_k) that actually capture the compressibility of the input text.*

In [24] Kaplan and Verbin prove the non-optimality of the most of BWT-based algorithms, but they observe that such compressors work well in practice. In particular, it is known that on English texts, BWT-based compressors work extremely well. The authors of [24] believe that BWT-compressors work on English text better than Dictionary-based compressors because of the non-Markovian elements in English text. In their opinion the discrepancy between the performance of BWT-based compressors on Markov sources that resembles on English texts and their performance on the text itself is yet to be

explored. They end their paper [24] with the following question: *what kind of regularity is there in English text that compressors exploit*?

In this paper we propose an answer to the questions of [24,23]. In particular, we believe that "the regularity of the English text that BWT-compressors exploit" is related to the *balancing* properties of the text itself. We recall that a word w is *balanced* if, roughly speaking, the frequency of each symbol in different blocks of w is almost the same. Remark that this property is actually satisfied, with a very good approximation, by an English text, or by a text written in another natural language.

As to concern the related question of [23] ("which is the statistic that actually captures the compressibility of the input text") we propose the *local entropy*, as a measure of the degree of balance of a text.

In Section 5, we introduce the formal definitions relative to balancing properties of words, and the definition of local entropy of a word. We show that the maximum of local entropy is reached for words having maximum degree of balancing, and, moreover, the minimum of local entropy is reached for *clustered* words, i.e. words in which all occurrences of each symbol are grouped together. This shows that the local entropy actually provides a measure of the degree of balance of a word.

The introduction of local entropy as a measure of balancing of a word, allow us to test the general hypothesis which is at the base of the present paper: *the more balanced the input word is, the more local similarity we have after BWT*, and the better the compression is.

Actually, the experimental results reported in Section 6, and the combinatorial results given in Sections 7 and 8, strongly corroborate our hypothesis.

5. Balancing, clustering, and local entropy

The notion of balance of a word plays a central role in our investigation. Informally, a word w is balanced if each symbol occurs "equally spaced" in the word. The most tight situation occurs when the distance between two consecutive occurrences of a given symbol is constant throughout the word. Such words are called *constant gap words*.

We recall that an infinite sequence s is *constant gap* if, for any letter $\sigma \in \mathcal{A}$, there exists a constant $q(\sigma)$ such that the number of letters between two occurrences of successive letter σ in s is $q(\sigma)$. It is obvious that infinite constant gap words are periodic. A finite word w is *constant gap* if the infinite word w^ω is *constant gap*. So the notion of constant gap word is invariant by conjugacy and adapts to the notion of circular word. For instance, *abac* is a constant gap word and *ababc* is not a constant gap word. The unique primitive constant gap word on a binary alphabet is of the form *ab*. Whereas, on an arbitrary alphabet, the structure of constant gap words is to a large extent unknown (cf. [1]). In particular, given the distribution of letters (i.e. the number of occurrences of each letter), the problem to decide whether there exists a constant gap word with such distribution is NP-complete [4].

Constant gap words are known to be at the root of a more general class of words called *balanced words*. More precisely, a finite or infinite word is *balanced* if, for any two of its factors u, v with $|u| = |v|$, we have

$$||u|_a - |v|_a| \leq 1 \quad \text{for any letter } a \in \mathcal{A},$$

where $|u|_a$ denotes the number of distinct occurrences of the letter a in the factor u. Hence a word is balanced if the number of a's in each of u and v differs by at most 1. A finite word is *circularly balanced* if all its conjugates are balanced. Let denote by $\mathcal{B}$ the set of the circularly balanced finite words, so $u \in \mathcal{B}$ if and only if u^ω is balanced. For instance, the word $w = abacbabdabaebabcabadbabe$ is a circularly balanced word. This example also shows that there exist circularly balanced words that are not constant gap.

In the case of binary alphabet, balanced sequences correspond to Sturmian words, a family of words widely investigated in Combinatorics on Words (cf. [26]). Whereas, in the case of more than two-letter alphabets, balanced words appear in the statement of the Fraenkel conjecture (cf. [15]). As a direct consequence of a result of Graham, one can prove that balanced sequences on a set of letters having different frequencies must be periodic (cf. [19,41]). The *Fraenkel conjecture* states that the unique solution (up to a permutation of letters) of balanced word on each the $|\mathcal{A}| = k \geq 3$ letters with all distinct frequencies of letters is $(FR_k)^\omega = (FR_{k-1}kFR_{k-1})^\omega$ where $FR_3 = 1213121$. The sequence $(FR_k)^\omega$ is the *Fraenkel sequence*. This conjecture is true for $k = 3, 4, 5, 6$ (cf. [39,40]). The problem of characterizing balanced words over any alphabet has been developed by Altman et al. in [1] in the field of optimal routing in queuing networks. We refer the interested readers to Vuillon [41], for a survey and the references about balanced words.

Our idea is that one obtains a more compressible string as output of BWT if its input is very close to be balanced. The balanced words can be considered as one of the extremal cases of the "phenomenon" that we study. The opposite extremal case is represented by the clustered words.

We say that the word w is a *clustered word* if the number of runs (i.e. consecutive occurrences of the same symbol) of the circular word $[w]$ is equal to the size of alphabet. For instance, the words *ddddddccccaaaaabbb* and *ccddddaaaaaabbbcc* are two clustered words.

As a tool to measure the degree of balance of a word, we use the "Local Entropy" statistic (for shortly, LE) based on Distance Coding. Distance Coding (DC) (cf. [16,23]) is an encoding procedure which is relatively little-known, probably because it was originally described only on a Usenet post [6].

We define the distance between two characters as the number of character between them (so the distance is zero if the two characters are consecutive) and we give a circular version of the Distance Coding in the following way: for each

character σ, DC finds its circular previous occurrence and outputs the circular distance to it, that is DC encodes the character σ with an integer equal to the number of characters encountered since the circular previous occurrence of the symbol σ. According to this definition and for our purpose, we need to know neither the starting point of DC nor the starting position of the first occurrence of each symbol. Given a word w, we denote by $dc(w)$ the output of DC. It is a (circular) word of non-negative integers.

Example 5.1. In this example, we compute the distance coding of the word w.

$$\begin{array}{rcccccccc} w & = & a & c & b & c & a & a & b \\ dc(w) & = & 1 & 4 & 2 & 1 & 3 & 0 & 3. \end{array}$$

Given a word w of length n, the Local Entropy on Distance Coding is defined as follows.

$$LE(w) = \frac{1}{n}\sum_{i=1}^{n} \log(dc(w)[i] + 1)$$

where $dc(w)[i]$ denotes the ith symbol in $dc(w)$. That is, $LE(w)$ is the sum of the logarithms of the distance coding values plus 1 divided by the length n. We add the $+1$ in the logarithm so that a repeating symbol will contribute 0 to the sum. For example, for the word aa the DC value of the second a is 0. Moreover, when there is only one occurrence of the symbol σ, DC outputs $n - 1$.

Remark 5.2. The Local Entropy on Distance Coding was used by [23] for the first time. The concept of the "Local Entropy" statistic was implicitly considered by Bentley et al. [5] as well as by Manzini [29]. In particular, they defined the local entropy on Move-to-Front (for shortly, MTF). We recall that MTF keeps the list of the characters of the alphabet and stores each character in the input by outputting its position in the list, then moves it in the front of the list. Therefore, MTF is the same as DC, except that instead of counting the number of characters (with repeats) between two consecutive occurrences of σ, it counts the number of distinct symbols. We denote by $mtf(w)$ the output of MTF applied to a word w. Given a word w, one has:

$$LE_{MTF}(w) = \frac{1}{n}\sum_{i=1}^{n} \log(mtf(w)[i] + 1)$$

where $mtf(w)[i]$ denotes the ith symbol in $mtf(w)$. We observe that $mtf(w)[i]$ is less than or equal to $dc(w)[i]$, for any i, so one has that $LE_{MTF}(w) \leq LE(w)$.

In order to prove the bounds of LE, we recall that the order-zero empirical entropy (cf. [30]) of the word w on $\mathcal{A} = \{a_1, a_2, \ldots, a_k\}$ of length n is defined as

$$H_0(w) = \sum_{i=1}^{k} \frac{n_i}{n} \log \frac{n}{n_i} \tag{3}$$

where n_i denotes the number of occurrences of the letter a_i in w. The value $nH_0(w)$, represents the output size of an ideal compressor which uses $\log \frac{n}{n_i}$ bits for coding the symbol a_i. It is well known that this is the maximum compression we can achieve using a uniquely decodable code in which a fixed codeword is assigned to each alphabet symbol.

The following theorem is a refinement (cf. Remark 5.4) of a result in [5].

Theorem 5.3. *For any word w, $LE(w) \leq H_0(w)$. Moreover the equality holds if and only if w is a constant gap word.*

Proof. We suppose that $w = b_1 b_2 \cdots b_n$ is a word on $\mathcal{A} = \{a_1, a_2, \ldots, a_k\}$ and each symbol a_i occurs n_i times in w. Given a symbol $a_i \in \mathcal{A}$, we define $\Gamma_{a_i}(w)$ as follows:

$$\Gamma_{a_i}(w) = \sum_{j:b_j=a_i} \log(dc(w)[j] + 1).$$

So $LE(w) = \frac{1}{n}\sum_{i=1}^{k} \Gamma_{a_i}(w)$. Clearly, the sum of the DC values plus 1 relative to symbol a_i is n. So, we can write:

$$\sum_{j:b_j=a_i} (dc(w)[j] + 1) = n.$$

First we prove that for any word w on $\mathcal{A}$, one has that $LE(w) \leq H_0(w)$. Let us recall that the Jensen inequality states that, if f is a concave function, for any $x_1, x_2, \ldots, x_h > 0$ and $y_1, y_2, \ldots, y_h > 0$ such that $\sum_{j=1}^{h} y_j = 1$, one has $\sum_{j=1}^{h} y_j f(x_j) \leq f(\sum_{j=1}^{h} y_j x_j)$. Since log is a concave function, by taking in the previous equality $f = \log$, $h = n_i$, $y_j = \frac{1}{n_i}$ and $x_j = dc(w)[j] + 1$, one has:

$$\frac{1}{n_i}\Gamma_{a_i}(w) = \sum_{j:b_j=a_i} \frac{1}{n_i} \log(dc(w)[j] + 1) \leq \log\left(\frac{1}{n_i}\sum_{j:b_j=a_i} (dc(w)[j] + 1)\right) = \log\frac{n}{n_i}.$$

Hence

$$\Gamma_{a_i}(w) \leq n_i \log \frac{n}{n_i}$$

and

$$LE(w) = \frac{1}{n}\sum_{i=1}^{k} \Gamma_{a_i}(w) \leq \frac{1}{n}\sum_{i=1}^{k} n_i \log \frac{n}{n_i} = H_0(w).$$

Now, we prove that $LE(w) = H_0(w)$ if and only if w is a constant gap word.

(if). $LE(w) = H_0(w)$ means that $\Gamma_{a_i}(w)$, for each $i = 1, \ldots, k$, reaches its maximum value. By the arithmetic–geometric means inequality, if one wishes to maximize $\Gamma_{a_i}(w)$, which is the sum of logarithms of n_i elements, under the constraint that the sum of these elements is n, then one needs to pick all the elements to be equal to n/n_i. This means that $dc(w)[j]+1 = \frac{n}{n_i}$ is a constant, i.e. w is a constant gap word.

(only if). Let w be a constant gap word. From definition, in the constant gap words, the distance between any two consecutive occurrences of a given symbol $a_i \in \mathcal{A}$ is constant throughout the word, in particular the DC values plus 1 relative to symbol a_i is $\frac{n}{n_i}$. We observe that $\frac{n}{n_i}$ is an integer and for each occurrence of the letter a_i, we have the same $\frac{n}{n_i}$. Thus, we have that:

$$\Gamma_{a_i}(w) = n_i \log \frac{n}{n_i}.$$

Hence

$$LE(w) = \frac{1}{n}\sum_{i=1}^{k} \Gamma_{a_i}(w) = \frac{1}{n}\sum_{i=1}^{k} n_i \log \frac{n}{n_i} = H_0(w). \quad \square$$

Remark 5.4. Theorem 5.3, in particular, states that $LE(w) \leq H_0(w)$. We observe that a similar bound has been proved in [5,23,30] by considering the words in linear (and not circular) way. Bentley et al. [5] define the local entropy on $MTF_{ignorefirst}$ and define $mtf_{ignorefirst}(w)$ to be a string which is identical to $mtf(w)$ except that they omit the integers representing the first occurrence of each symbol (so $mtf_{ignorefirst}(w)$ is of length less than n). Hence, they obtain that, for any word w, $LE_{MTF_{ignorefirst}}(w) \leq H_0(w)$. They ignore the first occurrence of each symbol, because if it is not omitted then we have to add the factor $k \log k$, where k is the cardinality of alphabet (cf. [23,30]). In this way they obtain the bound $LE_{MTF}(w) \leq H_0(w)+k \log k$.

In order to obtain the lower bound of the Local Entropy, for any word w on $\mathcal{A} = \{a_1, a_2, \ldots, a_k\}$ of length n, we define

$$G(w) = \frac{1}{n}\sum_{i=1}^{k} \log(n - n_i + 1)$$

where n_i denotes the number of occurrences of the letter a_i in w.

Theorem 5.5. *For any word w, $LE(w) \geq G(w)$. Moreover the equality holds if and only if w is a clustered word.*

Proof. We suppose that $w = b_1 b_2 \cdots b_n$ is a word on $\mathcal{A} = \{a_1, a_2, \ldots, a_k\}$ and each symbol a_i occurs n_i times in w. We first prove that $LE(w) = G(w)$ if and only if w is a clustered word. Given a symbol $a_i \in \mathcal{A}$, we recall that Γ_{a_i} is defined as follows:

$$\Gamma_{a_i}(w) = \sum_{j:b_j=a_i} \log(dc(w)[j] + 1).$$

Clearly, the sum of the DC values plus 1 relative to symbol a_i is n.

(only if). We suppose that w is a clustered word, i.e. for any i, $1 \leq i \leq k$, all occurrences of the symbol a_i are grouped together. When we apply DC to the word w, we have that, for each i, the first occurrence of the symbol a_i is coded by the integer $n - n_i$, whereas the other occurrences of a_i are coded by the integer 0. It follows that

$$\Gamma_{a_i}(w) = \log(n - n_i + 1).$$

Hence

$$LE(w) = \frac{1}{n}\sum_{i=1}^{k} \log(n - n_i + 1) = G(w)$$

(if). We know that if w is a clustered word then

$$\Gamma_{a_i}(w) = \log(n - n_i + 1).$$

Now we consider a word w' which is not a clustered word, it follows that w' contains at least two runs of a symbol a_i. We suppose that w' has exactly m runs of the symbol a_i, with $m > 1$. Now, we compute $\Gamma_{a_i}(w')$. By ordering the runs, denote

by r_j the distance between the first occurrence of a_i in the run of order j and the "circular" previous occurrence of a_i. We observe that $r_j > 0$, for all j, and $\sum_{j=1}^{m} r_j - n_i$. Thus

$$\Gamma_{a_i}(w') = \sum_{j=1}^{m} \log(r_j + 1) = \log \prod_{j=1}^{m} (r_j + 1).$$

We suppose now, by contradiction, that $\Gamma_{a_i}(w) > \Gamma_{a_i}(w')$, hence

$$\log(n - n_i + 1) > \sum_{j=1}^{m} \log(r_j + 1) = \log \prod_{j=1}^{m} (r_j + 1).$$

Because log is an increasing function then

$$n - n_i + 1 > \prod_{j=1}^{m} (r_j + 1).$$

From the arithmetic–geometric means inequality, we have that

$$\prod_{j=1}^{m} (r_j + 1) \geq \sum_{j=1}^{m} (r_j + 1) = n - n_i + m.$$

Hence, it follows that

$$n - n_i + 1 > \prod_{j=1}^{m} (r_j + 1) \geq \sum_{j=1}^{m} (r_j + 1) = n - n_i + m.$$

Since $m > 1$, it follows that $n - n_i + 1 < n - n_i + m$, which leads to a contradiction, hence $\Gamma_{a_i}(w) < \Gamma_{a_i}(w')$. This contradiction proves that the value $\frac{1}{n} \sum_{i=1}^{k} \log(n - n_i + 1)$ is only reached by the clustered words.

From this proof, it follows that, for any word w, one has $LE(w) \geq G(w)$. □

The following theorem collects together the results that were established.

Theorem 5.6. *For any word $w \in \mathcal{A}^*$,*

- $G(w) \leq LE(w) \leq H_0(w)$
- $LE(w) = H_0(w)$ *if and only if w is a constant gap word.*
- $LE(w) = G(w)$ *if and only if w is a clustered word.*

From this theorem, we note that the local entropy is useful to measure the degree of balance of a word. In particular, we can say that balanced words and clustered words correspond to the two extremal cases of the same "phenomenon", that is measured by local entropy.

6. Experimental results

We have introduced in previous section the notion of local entropy and we have proved that it provides a measure of the degree of balance of a word. So we are now able to perform some experiments on "real" text in order to evaluate the *clustering effect* of BWT, and, in particular, to test the hypothesis at the base of the present paper: the more balanced the input word is, the more local similarity we have after BWT, and the better the compression is.

For any word $w \in \mathcal{A}^*$, we introduce the following measures:

$$\delta(w) = \frac{H_0(w) - LE(w)}{H_0(w) - G(w)}$$

and

$$\tau(w) = \frac{LE(w) - G(w)}{H_0(w) - G(w)}.$$

By Theorem 5.6, if a word w has high degree of balance, we have that $\delta(w)$ is close to 0. On the contrary, strong local similarity in a word w corresponds to a value of $\tau(w)$ close to 0. Thus, by using $\delta(w)$ and $\tau(w)$, we test, in a quantitative way, on "real" text w, the following hypothesis:

(i) the more balanced the input word w is, the more local similarity is after BWT;
(ii) the more local similarity is found in the BWT of a word w, the better the compression is.

Table 1
The column denoted by "Size" represents the uncompressed size of the input text. The column denoted by $H_0(w)$ represents the order-zero entropy. The columns denoted by $bst(w)$ and $gzip(w)$ represent the compressed size of BST and Gzip, respectively. The column denoted by "Diff %" represents the difference between the *space savings* of BST and the *space savings* of Gzip. The columns denoted by $\delta(w)$ and $\tau(bwt(w))$ represent our measures.

File name	Size	$H_0(w)$	$bst(w)$	$gzip(w)$	Diff %	$\delta(w)$	$\tau(bwt(w))$
bible	4,047,392	4.343	796,231	1,191,071	9.755	0.117	0.233
english	52,428,800	4.529	11,533,171	19,672,355	15.524	0.136	0.238
etext99	105,277,340	4.596	24,949,871	39,493,346	13.814	0.141	0.264
english	104,857,600	4.556	23,993,810	39,437,704	14.728	0.143	0.250
dblp.xml	52,428,800	5.230	4,871,450	9,034,902	7.941	0.152	0.093
dblp.xml	104,857,600	5.228	9,427,936	17,765,502	7.951	0.153	0.090
dblp.xml	209,715,200	5.257	18,522,167	35,897,168	8.285	0.162	0.088
dblp.xml	296,135,874	5.262	25,597,003	50,481,103	8.403	0.164	0.086
world192	2,473,400	4.998	430,225	724,606	11.902	0.174	0.183
sprot34.dat	109,617,186	4.762	18,850,472	26,712,981	7.173	0.215	0.206
jdk13c	69,728,899	5.531	3,187,900	7,525,172	6.220	0.224	0.041
howto	39,886,973	4.857	8,713,851	12,638,334	9.839	0.231	0.229
rfc	116,421,901	4.623	17,565,908	26,712,981	7.857	0.239	0.163
w3c2	104,201,579	5.954	7,021,478	15,159,804	7.810	0.246	0.058
chr22.dna	34,553,758	2.137	8,015,707	8,870,068	2.473	0.341	0.575
pitches	52,428,800	5.633	18,651,999	16,884,651	−3.371	0.530	0.344
pitches	55,832,855	5.628	19,475,065	16,040,370	−6.152	0.533	0.337

To evaluate the efficiency of the BWT-based compressor, we compare BWT-based compressor against dictionary-based compressor, in particular, we use a compressor based on "LZ" method (see [42,43]). The LZ family of methods became popular because it gave excellent compression.

For experiments, we use "MtfRleMth" algorithm implemented in Booster Library (BST) (see [14,13]) to compress the text by using BWT, where "MtfRleMth" algorithm executes the same steps as Bzip2 [37] operating on the whole input instead that on fixed length blocks. We denote by $bst(w)$ the output of BST on w. We use the Gzip compressor, with "compress better" option, to compress the text with the "LZ" method. We denote by $gzip(w)$ the output of Gzip on w.

As a testbed, we use some files included in "Large Corpus" [2], in "Pizza & Chili Corpus" [10] and in "Manzini Corpus" [28]. We use these files instead of the classical Calgary corpus since this corpora contains only relatively small files which provide a poor indication of the behavior of our measures.

In the experiments we also compute the *space savings*. The *space savings* of a compressor C applied to a text w is the reduction in size relative to the uncompressed size $|w|$. So, if $|C(w)|$ is the compressed size of w after that the compressor C has been applied to w, the space savings of C on w is defined as follows:

$$\textit{Space Savings}(C(w)) = \left(1 - \frac{|C(w)|}{|w|}\right) * 100. \tag{4}$$

The experiments reported in Table 1 corroborate our hypothesis. In particular, they show that, when $\delta(w)$ is less than 0.23, then $\tau(bwt(w))$ is less than 0.27, in agreement with hypothesis (i). Moreover, when $\tau(bwt(w))$ is less than 0.27, then the BWT-based compressor has good performances (by "good performance" here we mean that BST is more advantageous than Gzip by percentage greater than 5%), and this is in agreement with hypothesis (ii).

Remark that the experiments reported suggest, as a practical application, a method to establish when to use a BWT-based compressor is more advantageous than to use a Gzip. We say that BST is better than Gzip on a text w, if the difference between the *space savings* of BST on w and the *space savings* of Gzip on w is at least 5%. The effectiveness of this application is based on the fact that the computation of $\delta(w)$ is actually a fast test for the choice between BWT-based compressor and Gzip compressor. For instance, from the results reported in Table 1, we can state that, when $\delta(w)$ is less than 0.23, then to use a BWT compressor is more advantageous than to use a Gzip.

7. Balancing and clustering on two letters alphabets

The experimental results reported in previous section corroborate the hypothesis that the more balanced the input word is, the more local similarity one has after BWT. From the theoretical point of view, a sound confirmation of such clustering effect of BWT is given, in the case of a binary alphabet, by a result proved in [27]. In this paper, the authors give a full characterization of words on a binary alphabet $\{a, b\}$ having the maximum amount of clustering under application of BWT, i.e. words w such that $bwt(w) = b^p a^q$, for some $p, q > 0$. Remark that, if $a < b$, the words of the form $a^q b^p$ cannot be obtained as output of BWT.

It turns out that the family of words w such that $bwt(w) = b^p a^q$, for some $p, q > 0$, corresponds to the family of (circularly) balanced words. Observe that, in the binary case, the subfamily of constant gap words is trivial: indeed the only constant gap words over $\{a, b\}$ are the powers of the word ab.

It is interesting to remark that, in the binary case, balanced words are closely related to Sturmian words, that were introduced by Morse and Hedlund (cf. [31]). Sturmian words can be defined in several different but equivalent ways (cf. [26, Chapter 2]). Some definitions are "combinatorial" and others of "geometrical" nature. From the "geometrical" point of view, the Sturmian words code discrete lines. In particular, a Sturmian word can be defined by considering the intersections with a squared-lattice of a semiline having a slope which is an irrational number. A vertical intersection is denoted by the letter a, a horizontal intersection by b and the intersection with a corner by ab or ba (cf. [26,27]). If the semiline starts from the origin the corresponding Sturmian words is called *characteristic*. Characteristic Sturmian words can be constructed by a family of finite words called *standard words*, in the sense that every characteristic word is the limit of a sequence of standard words (cf. [26]).

Classical examples of Standard words are the Fibonacci words f_i, $i \geq 0$, defined as follows: $f_0 = b$, $f_1 = a$, and $f_{n+1} = f_n f_{n-1}$, where $n \geq 1$. A classical example of the characteristic Sturmian word is the infinite Fibonacci word obtained as the limit of the sequence of Fibonacci words.

The formal definition of Standard words can be considered as a generalization of the definition of the Fibonacci words.

Let $d_1, d_2, \ldots, d_n, \ldots$ be a sequence of natural numbers, with $d_i \geq 0$ and $d_i > 0$ for $i = 2, \ldots, n, \ldots$. Consider the following sequence $\{s_n\}_{n\geq 0}$ of words over the binary alphabet $\{a, b\}$: $s_0 = b$, $s_1 = a$, and $s_{n+1} = s_n^{d_n} s_{n-1}$ for $n \geq 1$. The sequence $\{s_n\}_{n\geq 0}$ converges to a limit s that is a *characteristic Sturmian* word. Moreover any characteristic Sturmian word is obtained in this way. The sequence $\{s_n\}_{n\geq 0}$ is called the *approximating sequence* of s and $(d_1, d_2, \ldots, d_n, \ldots)$ is the *directive sequence* of s. Each finite word s_n in the sequence is called a *standard* word. It is univocally determined by the (finite) directive sequence $(d_1, d_2, \ldots, d_{n-1})$. For instance, the infinite Fibonacci word f is the characteristic Sturmian word whose directive sequence is $[1, 1, 1, \ldots]$. The characterization proved in [27] (cf. also [20]) is given by the following theorem.

Theorem 7.1. *Let w a word over the alphabet $\mathcal{A} = \{a, b\}$. Then following conditions are equivalent:*

1. *$bwt(w) = b^p a^q$, for some $p, q > 0$;*
2. *w is a conjugate of a power of a standard word;*
3. *w is circularly balanced.*

8. The case of an arbitrary alphabet

In this section, we study the words on larger alphabets with the aim to generalize the result over binary alphabets stated in Theorem 7.1. We show that, when the words are over alphabets of more than two letters, there is no more a complete equivalence between words having clusterized Burrows–Wheeler transform and the balancing. The study of their relationship in the general case necessitates to introduce some supplementary notions from Combinatorics on Words and, in particular, a generalization of the notion of standard word and the notion of palindromic richness. A preliminary version of the results presented in this section appears in [35].

8.1. Words with simple Burrows–Wheeler Transforms

A first generalization of the words w such that $bwt(w) = b^p a^q$ has been given by Simpson and Puglisi in [38]. They define the set $\mathcal{S}$ of the words w over a totally ordered alphabet $\mathcal{A} = \{a_1, a_2, \ldots, a_k\}$, with $a_1 < a_2 < \cdots < a_k$, for which

$$bwt(w) = a_k^{n_k} a_{k-1}^{n_{k-1}} \cdots a_2^{n_2} a_1^{n_1}$$

for some non-negative integers $n_1, n_2, \ldots, n_k$. They called these words: *words with simple Burrows–Wheeler Transforms* and here such words are denoted by *simple* BWT *words*.

In the notion of simple BWT word, the order used to perform the BWT is the inverse of the order in which the symbols appears in the output of BWT. So the set $\mathcal{S}$ of simple BWT words is a proper subset of the set of words whose BWT is a clustered word.

Remark further that, in the binary case, the set of simple BWT words coincides with the set of words whose BWT is a clustered word. Indeed, the unique clustered words w over $\{a, b\}$ with $a < b$, which we can obtain by applying the BWT is of the form $b^p a^q$, where p is the number of occurrences of the letter b and q is the number of occurrences of the letter a in the word w. Whereas, when the words are over alphabets of more than two letters, there exist words whose BWT are clustered words which are not simple BWT. For instance, the BWT of the word $w = abacad$ over $\{a, b, c, d\}$ with $a < b < c < d$, is a clustered word, indeed we have that $bwt(w) = dbca^3$. However it is not simple.

A characterization of words having simple Burrows–Wheeler transform in the case of three letters alphabets has been given by Simpson and Puglisi in [38], where they also report some preliminary results in the general case.

The following result provides a characterization of the words in $\mathcal{S}$ in terms of the Burrows–Wheeler matrix M. For completeness, we report the proof, as given in [36]. We denote by R the matrix obtained from M by a rotation of 180°. Notice that the rows of R correspond to the conjugates of $\tilde{w}$.

Remark 8.1. By construction, the properties 1 and 2 stated in Proposition 3.1 for the matrix M hold true also for the matrix R:

1. For all $i, j = 1, \ldots, n$, $i \neq j$, the character $L_R[i]$ is followed by $F_R[i]$ in the j-th row of R.
2. For each character α, the i-th occurrence of α in F_R corresponds to the i-th occurrence of α in L_R.

As a consequence, given F_R and L_R, one can uniquely reconstruct the matrix R by the same procedure used for reversing BWT.

M						R					
a	a	b	r	a	c	b	a	a	c	a	r
a	b	r	a	c	a	a	r	b	a	a	c
a	c	a	a	b	r	a	a	c	a	r	b
b	r	a	c	a	a	r	b	a	a	c	a
c	a	a	b	r	a	a	c	a	r	b	a
r	a	c	a	a	b	c	a	r	b	a	a

Fig. 2. The matrix M and R of the sequence $w = abraca$.

Theorem 8.2. *A word $w \in \mathcal{S}$ if and only if $M = R$.*

Proof. Let w be a word in $\mathcal{S}$ and let M be the corresponding Burrows–Wheeler matrix. Since $bwt(w) = L_M = a_k^{n_k} a_{k-1}^{n_{k-1}} \cdots a_2^{n_2} a_1^{n_1}$, one has $L_M = \widetilde{F_M}$. Since, by definition of R, $L_R = \widetilde{F_M}$ and $F_R = \widetilde{L_M}$, it follows that

$$L_R = L_M \quad \text{and} \quad F_R = F_M. \tag{5}$$

By the Remark 8.1, $M = R$.

Conversely, if $M = R$, it follows trivially that $bwt(w) = a_k^{n_k} a_{k-1}^{n_{k-1}} \cdots a_2^{n_2} a_1^{n_1}$, i.e. $w \in \mathcal{S}$. □

For instance, in Fig. 2, M and R are distinct and the word $w = abraca$ does not belong to $\mathcal{S}$. We mention that a result equivalent of Theorem 8.2 has been obtained, with a different proof, by Simpson and Puglisi [38, Theorem 4.3]. They also derive the following corollary (cf. [38, Corollary 4.4]).

Corollary 8.3. *Each conjugate of $w \in \mathcal{S}$ has the two palindrome property.*

For alphabet of cardinality greater than two, the equivalence of 1 and 3 of Theorem 7.1 is no longer true. For example, there exist circularly balanced words that do not have simple BWT, for instance $v = ababc$ (in fact $bwt(v) = cbaab$) and there exist unbalanced words having simple BWT, for instance $u = bbacacacaca$ (in fact $bwt(u) = c^4b^2a^5$). Moreover, in order to study the relationship between the conditions 1, 2 and 3 of Theorem 7.1 in the case of larger alphabets, we need to extend the notion of (finite) standard word.

8.2. Finite epistandard words

Finite standard words are particular prefixes of Sturmian infinite words. In order to extend the notion of standard words to larger alphabets we need to generalize the notion of Sturmian word. A first generalization of *Sturmian words* to an arbitrary alphabet is the family of *Arnoux–Rauzy sequences* (cf. [34,3]). Another generalization of Sturmian sequences, which also is a slight generalization of Arnoux–Rauzy sequences, is the set of infinite episturmian sequences. These sequences are not necessarily balanced, nor are they necessarily aperiodic (cf. [12]). An infinite word $t \in \mathcal{A}^\omega$ is *episturmian* if $F(t)$ is closed under reversal and t has at most one right (or equivalently left) special factor of each length. Moreover, an episturmian word is *standard* if all of its left special factors are prefixes of it. Sturmian words are exactly the aperiodic episturmian words over a 2-letter alphabet. For a recent survey on the theory of episturmian words see [17].

In order to give the formal definition we need some notations. The *palindromic right-closure* $w^{(+)}$ of a finite word w is the (unique) shortest palindrome having w as a prefix (see [11]). For example, $(abcd)^{(+)} = abcdcba$. The *iterated palindromic closure* function [21], denoted by Pal, is defined recursively as follows. Set $\mathrm{Pal}(\varepsilon) = \varepsilon$ and, for any word w and letter x, define $\mathrm{Pal}(wx) = (\mathrm{Pal}(w)x)^{(+)}$. For example, $\mathrm{Pal}(abc) = (\mathrm{Pal}(ab)c)^{(+)} = (abac)^{(+)} = abacaba$. The following definition has been given by Droubay et al. in [12].

Definition 8.4. An infinite sequence s is *standard episturmian* if it satisfies one of the following equivalent conditions:

(i) For every prefix u of s, $u^{(+)}$ is also prefix of s.
(ii) Every leftmost occurrence of a palindrome in s is a central factor of a palindromic prefix of s.
(iii) There exists an infinite sequence $u_1 = \varepsilon, u_2, u_3, \ldots$ of palindromes and an infinite sequence $\Delta(s) = x_1x_2\cdots, x_i \in \mathcal{A}$, such that $u_{n+1} = (u_nx_n)^{(+)}$ for all $n \geq 1$ and that all the u_n are prefixes of s.

$\Delta(s)$ is called the *directive sequence* of the standard episturmian sequence s.

Example 8.5. We consider the *directive sequence* $\Delta(s) = (abc)^\omega$. We obtain the following sequence:

$$s = \underline{aba}\underline{c}\underline{abaa}\underline{babacaba}\underline{bacabaabacaba}\underline{c}\underline{abaabaca}\cdots,$$

where each palindromic prefix $\mathrm{Pal}(x_1\cdots x_n)$ is followed by an underlined letter x_n. This sequence is called *Tribonacci sequence* (or *Rauzy word* [34]).

Remark 8.6. An episturmian sequence s is periodic if and only if $|Ult(\Delta(s))| = 1$ (see [22, Proposition 2.9]).

A standard episturmian sequence s can also be obtained by the *Rauzy rules* (see [12, Theorem 8]), where if $\Delta(s) = a_{i_1}a_{i_2}a_{i_3}\cdots$ then the sequence of the labels of the applied Rauzy rules is $i_1, i_2, i_3, \ldots$. We recall that a sequence $(R_n)_{n\in\mathbb{N}}$ of Rauzy k-tuples $R_n = (A_n^{(1)}, A_n^{(2)}, \ldots, A_n^{(k)})$ is defined as follows: $R_0 = (a_1, a_2, \ldots, a_k)$, R_{n+1} is obtained from R_n by applying one of the Rauzy rules, labelled $1, 2, \ldots, k$, with the rule $i \in [1, k]$ defined by
$A_{n+1}^{(i)} = A_n^{(i)}$
$A_{n+1}^{(j)} = A_n^{(i)}A_n^{(j)}$ for $j \in [1, k]\backslash\{i\}$.
There exists a unique (infinite) sequence u such that every prefix of u is a prefix of infinitely many of the $A_n^{(q)}, n \in \mathbb{N}, q \in [1, k]$. So any Rauzy sequence $(R_n)_{n\in\mathbb{N}}$ defines an infinite standard episturmian sequence.

Definition 8.7. A word $w \in \mathcal{A}^*$ is called *finite epistandard* if it is the element of a k-tuples R_n, for some $n \geq 1$.

It is easy to see that, in the case of binary alphabets, the notion of finite epistandard word corresponds to the notion of (finite) standard word given in Section 7. We observe that the notion of finite epistandard word is also connected to the notion of epichristoffel word defined in [32].

Let us remark that a finite epistandard word is primitive. In the sequel we will denote by $\mathcal{EP}$ the set of words that are a power of a conjugate of a finite epistandard word. From previous construction and Remark 8.6 follows lemma stated below.

Lemma 8.8. *A periodic standard episturmian sequence is of the form t^ω, where t is a finite epistandard word.*

Contrary to the case of two letters alphabet, a standard episturmian sequence over an alphabet of size greater than two is not in general balanced. For example the word $w = aadaacaad$ is epistandard, but it is not balanced, whereas the word $w = abcabdabcabe$ is balanced, but it is not epistandard. The following important theorem of Paquin and Vuillon (cf. [33]) gives a characterization of balanced standard episturmian sequences.

Theorem 8.9. *Any balanced standard episturmian sequence s over an alphabet with 3 or more letters is of the form $s = t^\omega$, where t is a finite epistandard word that belongs to one of the following three families (up to letter permutation):*

(i) $t = pa_2$, *with* $p = \mathrm{Pal}(a_1^m a_k a_{k-1}\cdots a_3)$, *where* $k \geq 3$ *and* $m \geq 1$;
(ii) $t = pa_2$, *with* $p = \mathrm{Pal}(a_1 a_k a_{k-1}\cdots a_{k-\ell}a_1 a_{k-\ell-1}a_{k-\ell-2}\cdots a_3)$, *where* $0 \leq \ell \leq k-4$ *and* $k \geq 4$;
(iii) $t = \mathrm{Pal}(a_1 a_k a_{k-1}\cdots a_2)$, *where* $k \geq 3$.

We observe that the sequences of the last family of Theorem 8.9 correspond to the *Fraenkel sequence*.

Since s is periodic and balanced, as a direct consequence, one can prove the following corollary.

Corollary 8.10. *A finite epistandard word is circularly balanced if and only if it belongs to one of the three families described in Theorem 8.9.*

In order to state a relation among the notions of simple BWT words, finite epistandard word and circularly balanced word, we need a further definition, that of (palindromic) rich word, which is introduced in next subsection.

8.3. Rich words

In [12], it was proved that any word w of length $|w|$ contains at most $|w| + 1$ distinct palindromic factors (including the empty word). The episturmian sequences, which include the Sturmian sequences, are "rich" in palindromes, in the sense that they contain the maximum number of different palindromic factors. Specifically, in [12], it was proved that if an infinite word w is episturmian, then any factor u of w contains exactly $|u| + 1$ distinct palindromic factors.

Glen et al. in [18] introduced and studied rich words, that constitute a new class of finite and infinite words characterized by containing the maximal number of distinct palindromes. More precisely, a finite word w is *rich* if it has exactly $|w| + 1$ distinct palindromic factors. A finite or infinite word is rich if all of its factors are rich. Rich words have been recently investigated in several papers (cf. [18,7,8]). In particular, we refer to [18] for a complete survey on the subject, including very recent results. We say that a finite word w is *circularly rich* if the infinite word w^ω is rich. We denote by $\mathcal{R}$ the set of the circularly rich words.

We first remark that the set of circularly balanced words over more than two letters alphabets does not coincide with the set of circularly rich words. For example, the word $w = bbbbbacaca$ is circularly rich, but it is not circularly balanced, whereas the word $u = abcabdabcabe$ is circularly balanced, but it is not circularly rich. In order to study the relationship between the circularly rich words and the circularly balanced words, we mention some results from [18].

Theorem 8.11. *Recurrent balanced rich infinite words are precisely the balanced episturmian words.*

Since, by Theorems 8.9 and 8.11, recurrent balanced rich infinite words are periodic, the following characterization (cf. [18]) of rich infinite periodic words is useful.

Proposition 8.12. *For a finite word w, the following properties are equivalent:*

1. *w^ω is rich;*
2. *w^2 is rich;*
3. *w is the product of two palindromes and all of the conjugates of w (including itself) are rich.*

The following key result, proved in [36], relates rich words and words having simple Burrows–Wheeler transform.

Theorem 8.13. *If the word w belongs to $\mathcal{S}$ then w is circularly rich.*

We here observe that the proof of Theorem 8.13 makes use of Proposition 8.12 and of Theorem 8.2 (in particular Corollary 8.3). Indeed, if a word belongs to $\mathcal{S}$, then, by Corollary 8.3, it has the two palindromic property. So, by using statement 3 of Proposition 8.12, in order to prove that the word is circularly rich, we need to prove that all its conjugates are rich. This last step in the proof, which is however long and complex, is the main contribution in [36]. The following example shows that the converse of Theorem 8.13 is false.

Example 8.14. The word $w = ccaaccb$ is circularly rich, but $bwt(w) = cacccba$, hence $w \notin \mathcal{S}$.

8.4. Relating previous notions on alphabets of more than two letters

Now, we show the following relationship:

$$\mathcal{B} \cap \mathcal{R} = \mathcal{B} \cap \mathcal{EP} = \mathcal{B} \cap \mathcal{S}.$$

We observe that the notions of epistandard, circularly balanced and circularly rich word are invariant under letter permutation. On the contrary the property that the word has simple BWT depends on the order of the alphabet. Hence the equivalence that we state between some of these notions holds true up to letter permutation.

Theorem 8.15. *Any conjugate of a word in one of the three families defined in Theorem 8.9 belongs (up to letter permutation) to the set $\mathcal{S}$.*

Proof. We consider the words t of the three families in the form given in Theorem 8.9, and the alphabet order $a_1 < a_2 < \cdots < a_k$. In three cases, by the structure of t, we can determine the factor that follows each occurrence of letters of $\mathcal{A}$ in each conjugate of t. Then we prove that the letters of the last column L of the Burrows–Wheeler matrix M of t are non-increasing.

Type (i): $t = pa_2$, with $p = \mathrm{Pal}(a_1^m a_k a_{k-1} \cdots a_3)$.
Each occurrence of letter a_k is followed by the factor $a_1^m a_j$ with $1 < j < k$. Each occurrence of letter a_i, with $2 < i < k$, is followed by the factor $\mathrm{Pal}(a_1^m a_k \cdots a_{i+1})a_j$, with $1 < j < i$. The unique occurrence of letter a_2 is followed by the factor $\mathrm{Pal}(a_1^m a_k \cdots a_3)$. Finally, each occurrence of letter a_1 is followed either by the factor $a_1^h a_j$ (only in the case $m > 1$), with $1 \le h \le m-1$, or by the letter a_j, with $2 \le j \le k$.

Type (ii): $t = pa_2$, with $p = \mathrm{Pal}(a_1 a_k a_{k-1} \cdots a_{k-\ell} a_1 a_{k-\ell-1} a_{k-\ell-2} \cdots a_3)$.
Each occurrence of letter a_k is followed by the factor $a_1 a_j$, with $1 \le j < k$. Each occurrence of letter a_i, with $k-\ell \le i \le k-1$, is followed by the factor $\mathrm{Pal}(a_1 a_k \cdots a_{i+1})a_j$, with $1 \le j < i$. Each occurrence of letter $a_{k-\ell-1}$ is followed by the factor $\mathrm{Pal}(a_1 a_k \cdots a_{k-\ell} a_1)a_j$, with $1 < j < k-\ell-1$. Each occurrence of letter a_i, with $2 < i < k-\ell-1$, is followed by the factor $\mathrm{Pal}(a_1 a_k \cdots a_{k-\ell} a_1 a_{k-\ell-1} \cdots a_{i+1})a_j$, with $1 < j < i$. The unique occurrence of letter a_2 is followed by the factor $\mathrm{Pal}(a_1 a_k a_{k-1} \cdots a_{k-\ell} a_1 a_{k-\ell-1} \cdots a_3)$. Finally, each occurrence of letter a_1 is followed either by the factor $\mathrm{Pal}(a_1 a_k \cdots a_{k-\ell})a_j$, with $2 \le j \le k-\ell-1$, or by letter a_j, with $2 \le j \le k$.

Type (iii): $t = \mathrm{Pal}(a_1 a_k a_{k-1} \cdots a_2)$.
Each occurrence of letter a_k is followed by the factor $a_1 a_j$, with $1 \le j < i$, each occurrence of letter a_i, with $1 < i < k$, is followed by the factor $\mathrm{Pal}(a_1 a_k \cdots a_{i+1})a_j$, with $1 \le j < i$. Finally, each occurrence of letter a_1 is followed either by the factor $\mathrm{Pal}(a_1 a_k \cdots a_3)a_2$ or by letter a_j with $2 \le j \le k$.

Clearly, in all cases, the smallest rows (in the lexicographic order) of M end with a_k; moreover the greatest rows end with a_1. Hence the intermediate rows end with a_i for $1 < i < k$. Now we consider two conjugates t' and t'' of t, where the last letter of t', say a_i, is greater than the last letter of t'', say a_j, where $1 < j < i < k$. By the relation $a_j < a_i$ we derive that $t' < t''$. Indeed, by structure of t, the longest common prefix of t' and t'' is a palindromic factor with a_{i+1} as central letter. Moreover such a factor is followed by a letter $a_h < a_i$ in t' and by the letter a_i in t''. Since $a_h < a_i$, we obtain $t' < t''$. □

Previous results culminate in the following theorem that relates the notions introduced in this section.

Theorem 8.16. *Let $\mathcal{A} = \{a_1, a_2, \ldots, a_k\}$ be a totally ordered alphabet and let $w \in \mathcal{A}^*$ be a circularly balanced word over $\mathcal{A}$. The following statements are equivalent up to a letter permutation:*

(i) *w belongs to $\mathcal{S}$;*
(ii) *w is a circularly rich word;*
(iii) *w is a conjugate of a power of a finite epistandard word.*

Proof. (i) ⇒ (ii): it the Theorem 8.13.
(ii) ⇔ (iii): it follows from Theorem 8.11 and Lemma 8.8.
(iii) ⇒ (i): it follows from Theorem 8.15. □

Example 8.17. The circularly balanced word $w = adacadabadacada$ belongs to $\mathcal{S}$, is a finite epistandard word and is circularly rich.

The following examples show that the notions coincide only under assumption of balancing.

Example 8.18. The non-circularly balanced word $w = bbbbbacaca$ belongs to $\mathcal{S}$ (clearly it is circularly rich), but it is not a finite epistandard word. The non-circularly balanced word $w = (adac)^2adab(adac)^2ada(adac)^2adab(adac) \notin \mathcal{S}$ and it is a finite epistandard word.

The following example shows that there exist non-circularly balanced words which belong to $\mathcal{EP} \cap \mathcal{S}$.

Example 8.19. The non-circularly balanced word $w = aadaacaad$ is a finite epistandard word and belongs to $\mathcal{S}$.

Acknowledgements

The authors thank the anonymous referees for helpful comments that improved the presentation of this paper.

References

[1] Eitan Altman, Bruno Gaujal, Arie Hordijk, Balanced sequences and optimal routing, Journal of the ACM 47 (4) (2000) 752–775.
[2] Ross Arnold, Tim Bell, The Canterbury corpus home page. http://corpus.canterbury.ac.nz.
[3] Pierre Arnoux, Gérard Rauzy, Représentation géométrique de suites de complexité $2n + 1$, Bulletin de la Société Mathématique de France. Soc. Math. France, Paris 119 (1991) 199–215.
[4] Amotz Bar-Noy, Randeep Bhatia, Joseph (Seffi) Naor, Baruch Schieber, Minimizing service and operation costs of periodic scheduling, Mathematics of Operations Research 27 (3) (2002) 518–544.
[5] Jon Louis Bentley, Daniel D. Sleator, Robert E. Tarjan, Victor K. Wei, A locally adaptive data compression scheme, Commun. ACM 29 (4) (1986) 320–330.
[6] Edgar Binder, Distance coder. Usenet group: comp.compression, 2000.
[7] Michelangelo Bucci, Alessandro De Luca, Amy Glen, Luca Q. Zamboni, A connection between palindromic and factor complexity using return words, Advances In Applied Mathematics 42 (2009) 60–74.
[8] Michelangelo Bucci, Alessandro De Luca, Amy Glen, Luca Q. Zamboni, A new characteristic property of rich words, Theoretical Computer Science 410 (30–32) (2009) 2860–2863.
[9] Michael Burrows, David J. Wheeler, A block sorting data compression algorithm. Technical report, DIGITAL System Research Center, 1994.
[10] Pizza&Chili Corpus. http://pizzachili.di.unipi.it/texts.html.
[11] Aldo de Luca, Sturmian words: structure, combinatorics, and their arithmetics, Theoretical Computer Science 183 (1) (1997) 45–82.
[12] Xavier Droubay, Jacques Justin, Giuseppe Pirillo, Episturmian words and some constructions of de Luca and Rauzy, Theoretical Computer Science 255 (1–2) (2001) 539–553.
[13] Paolo Ferragina, Raffaele Giancarlo, Giovanni Manzini, The engineering of a compression boosting library: theory vs practice in bwt compression, in: ESA'06: Proceedings of the 14th Conference on Annual European Symposium, Springer-Verlag, London, UK, 2006, pp. 756–767.
[14] Paolo Ferragina, Raffaele Giancarlo, Giovanni Manzini, Marinella Sciortino, Boosting textual compression in optimal linear time, Journal of the ACM 52 (4) (2005) 688–713.
[15] Aviezri S. Fraenkel, Complementing and exactly covering sequences, Journal of Combinatorial Theory. Series A 14 (1) (1973) 8–20.
[16] Travis Gagie, Giovanni Manzini, Move-to-front, distance coding, and inversion frequencies revisited, in: Bin Ma, Kaizhong Zhang (Eds.), Combinatorial Pattern Matching, 18th Annual Symposium, CPM 2007, London, Canada, July 9–11, 2007, Proceedings, in: Lecture Notes in Computer Science, vol. 4580, Springer, 2007, pp. 71–82.
[17] Amy Glen, Jacques Justin, Episturmian words: a survey, in: RAIRO Theoretical Informatics and Applications, 2009.
[18] Amy Glen, Jacques Justin, Steve Widmer, Luca Q. Zamboni, Palindromic richness, European Journal of Combinatorics 30 (2) (2009) 510–531.
[19] Ronald L. Graham, Covering the positive integers by disjoint sets of the form $\{[n\alpha + \beta] : n = 1, 2, \ldots\}$, Journal of Combinatorial Theory. Series A 15 (3) (1973) 354–358.
[20] Oliver Jenkinson, Luca Q. Zamboni, Characterisations of balanced words via orderings, Theoretical Computer Science 310 (1–3) (2004) 247–271.
[21] Jacques Justin, Episturmian morphisms and a Galois theorem on continued fractions, Theoretical Informatics and Applications 39 (1) (2005) 207–215.
[22] Jacques Justin, Giuseppe Pirillo, Episturmian words and episturmian morphisms, Theoretical Computer Science 276 (1–2) (2002) 281–313.
[23] Haim Kaplan, Shir Landau, Elad Verbin, A simpler analysis of Burrows–Wheeler-based compression, Theoretical Computer Science 387 (3) (2007) 220–235.
[24] Haim Kaplan, Elad Verbin, Most Burrows–Wheeler based compressors are not optimal, in: Bin Ma, Kaizhong Zhang (Eds.), Combinatorial Pattern Matching, 18th Annual Symposium, CPM 2007, London, Canada, July 9–11, 2007, Proceedings, in: Lecture Notes in Computer Science, vol. 4580, Springer, 2007, pp. 107–118.
[25] M. Lothaire, Combinatorics on Words, in: Encyclopedia of Mathematics, vol. 17, Addison-Wesley, Reading, Mass, 1983, Reprinted in the Cambridge Mathematical Library, Cambridge University Press, 1997.
[26] M. Lothaire, Algebraic Combinatorics on Words, Cambridge University Press, 2002.
[27] Sabrina Mantaci, Antonio Restivo, Marinella Sciortino, Burrows–Wheeler transform and Sturmian words, Information Processing Letters 86 (2003) 241–246.
[28] Giovanni Manzini, http://web.unipmn.it/~manzini/lightweight/corpus/.
[29] Giovanni Manzini, The Burrows–Wheeler transform: Theory and practice, in: Proc. of the 24th International Symposium on Mathematical Foundations of Computer Science, MFCS'99, in: LNCS, vol. 1672, Springer-Verlag, 1999, pp. 34–47.
[30] Giovanni Manzini, An analysis of the Burrows–Wheeler transform, Journal of the ACM 48 (3) (2001) 407–430.
[31] Marston Morse, Gustav A. Hedlund, Symbolic dynamics II. Sturmian trajectories, American Journal of Mathematics 62 (1) (1940) 1–42.
[32] Geneviève Paquin, On a generalization of christoffel words: epichristoffel words, Theoretical Computer Science 410 (38–40) (2009) 3782–3791.
[33] Geneviève Paquin, Laurent Vuillon, A characterization of balanced episturmian sequences, Electronic Journal of Combinatorics 14 (2006) 12.
[34] Gérard Rauzy, Suites à termes dans un alphabet fini. In Séminaire de théorie des Nombres de Bordeaux. Exposé 25, 1982–1983, pp. 1–16.
[35] Antonio Restivo, Giovanna Rosone, Balanced words having simple Burrows–Wheeler transform, in: DLT'09: Proceedings of the 13th International Conference on Developments in Language Theory, in: Lecture Notes in Computer Science, vol. 5583, Springer-Verlag, 2009, pp. 431–442.
[36] Antonio Restivo, Giovanna Rosone, Burrows–Wheeler transform and palindromic richness, Theoretical Computer Science 410 (30–32) (2009) 3018–3026.
[37] Julian Seward, The BZIP2 home page, 2006. http://www.bzip.org.
[38] Jamie Simpson, Simon J. Puglisi, Words with simple Burrows–Wheeler transforms, Electronic Journal of Combinatorics 15 (2008) article R83.
[39] Robert Tijdeman, Exact covers of balanced sequences and Fraenkel's conjecture, 2000, pp. 467–483.
[40] Robert Tijdeman, Fraenkel's conjecture for six sequences, Discrete Mathematics 222 (1–3) (2000) 223–234.
[41] Laurent Vuillon, Balanced words, Bulletin of the Belgian Mathematical Society 10 (5) (2003) 787–805.
[42] Jacob Ziv, Abraham Lempel, A universal algorithm for sequential data compression, IEEE Transactions on Information Theory 23 (3) (1977) 337–343.
[43] Jacob Ziv, Abraham Lempel, Compression of individual sequences via variable-rate coding, IEEE Transactions on Information Theory 24 (5) (1978) 530–536.