On the Compressiveness of the Burrows-Wheeler Transform

Hideo Bannai
M&D Data Science Center, Institute of Integrated Research, Institute of Science Tokyo, Japan

Tomohiro I
Department of Artificial Intelligence, Kyushu Institute of Technology, Fukuoka, Japan

Yuto Nakashima
Department of Informatics, Kyushu University, Fukuoka, Japan

Abstract

The Burrows-Wheeler transform (BWT) is a reversible transform that converts a string w into another string $\mathsf{BWT}(w)$. The size of the run-length encoded BWT (RLBWT) can be interpreted as a measure of repetitiveness in the class of representations called dictionary compression which are essentially representations based on copy and paste operations. In this paper, we shed new light on the compressiveness of BWT and the bijective BWT (BBWT). We first extend previous results on the relations of their run-length compressed sizes r and r_B. We also show that the so-called "clustering effect" of BWT and BBWT can be captured by measures other than empirical entropy or run-length encoding. In particular, we show that BWT and BBWT do not increase the repetitiveness of the string with respect to various measures based on dictionary compression by more than a polylogarithmic factor. Furthermore, we show that there exists an infinite family of strings that are maximally incompressible by any dictionary compression measure, but become very compressible after applying BBWT. An interesting implication of this result is that it is possible to transcend dictionary compression in some cases by simply applying BBWT before applying dictionary compression.

2012 ACM Subject Classification Theory of computation → Data compression; Mathematics of computing → Combinatorics on words

Keywords and phrases Data Compression, Bijective Burrows-Wheeler Transform, Fibonacci words

Digital Object Identifier 10.4230/LIPIcs.CPM.2025.17

Funding *Hideo Bannai*: JSPS KAKENHI Grant Number JP24K02899.
Tomohiro I: JSPS KAKENHI Grant Number JP22K11907 and JP24K02899, JST AIP Acceleration Research JPMJCR24U4.
Yuto Nakashima: JSPS KAKENHI Grant Number JP21K17705 and JP23H04386.

1 Introduction

The Burrows-Wheeler Transform (BWT) [5] is a reversible mapping from a string to another string that enables compression and efficient pattern search, and is the theoretical cornerstone for essential tools in the field of bioinformatics [17, 18]. The compressibility of BWT has been studied in various contexts, but more recently, rather than statistical measures such as empirical entropy which are not helpful in highly repetitive datasets, the size r of the run-length encoded BWT (RLBWT) and its relation to the many other repetitiveness measures related to dictionary compression has become an important topic of study (See [21] for a comprehensive survey).

Dictionary compression is a family of compressed representations which are essentially based on copy and paste operations. Kempa and Prezza [15] proposed the notion of *string attractors* to view dictionary compression in a uniform way, and showed that the size γ of the smallest string attractor lower bounds all other measures of dictionary compression, since

36th Annual Symposium on Combinatorial Pattern Matching (CPM 2025).
Editors: Paola Bonizzoni and Veli Mäkinen; Article No. 17; pp. 17:1–17:15
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

they implicitly give a string attractor of the same size as their representations: the size b of the smallest bidirectional macro scheme (BMS) [23], the size z of the LZ77 parsing [25], the size g_{rl} of the smallest grammar with run-length rules, the size g of the smallest grammar, r, and more. Another measure δ based on substring complexity is known to lowerbound γ [16].

Concerning r in relation to the other measures, Navarro et al. [22] showed that given an RLBWT of size r, we can construct a BMS of size $2r$, thus showing $b = O(r)$. Kempa and Kociumaka [14] showed $r = O(z \log^2 n)$, and further $r = O(\delta \log^2 n)$[1]. Also, $z = O(b \log n)$ [22] has been shown, and therefore $z = O(r \log n)$ or, $r = \Omega(z/\log n)$. Recently, the size r_B of the run-length encoded bijective BWT (RLBBWT) has been studied [2, 1], and was shown, similarly to r, that $r_B = O(z \log^2 n)$ [1]. Also, the existence of an infinite family of strings such that $r = o(r_B)$, namely, $r = O(1)$ and $r_B = \Omega(\log n)$ was shown [1]. The existence of the opposite case, i.e., if there are string families for which $r_B = o(r)$, was left open.

The "clustering effect" of the BWT in terms of the run length encoding was studied by Mantaci et al. [19], where they proved that BWT can increase the size of the run-length encoding of the string by a factor of at most 2. The maximal clustering effect for run-length encoding can be seen for example with the Fibonacci words: Fibonacci words of length n have a run-length encoding of size $\Theta(n)$, while $r = 2$. While this is remarkable when viewed from run-length encoding since a maximally incompressible string (in terms of rle) is transformed into a compressible one, in terms of the smallest bidirectional macro scheme, Fibonacci words and their BWT have the same size 4, and one might argue that BWT is merely transforming a compressible string into another compressible string.

In this paper, we extend previous results on the relation between r and r_B. We also investigate the clustering effect of BWT and BBWT in terms of other repetitiveness measures and show that they can be much more powerful than previously perceived, giving an example where BBWT transforms a maximally incompressible string (in terms of dictionary compression) into a compressible one.

The contributions of this paper are as follows.

1. We show an infinite family of strings such that $r = \Omega(\log n)$ and $r_B = O(1)$, answering affirmatively an open question about the existence of a family of strings where $r_B = o(r)$ raised by Badkobeh et al. [1].
2. We show a polylogarithmic ($O(\log^4 n)$) upper bound on the multiplicative difference between r and r_B.
3. We analyze the clustering effect of BWT and BBWT in terms of other repetitiveness measures, and show that the application of BWT and BBWT can only increase various repetitiveness measures of a string by a polylogarithmic factor.
4. We show that BBWT on the strings whose BBWT images are Fibonacci words exhibit a maximal clustering effect in terms of the repetitiveness measures δ, γ, b, v, or r, from almost linear ($\Theta(n/\log n)$) to constant. The main combinatorial part of our proof is an elegant characterization of the LF mapping function on Fibonacci strings that uses the Zeckendorf representation of the positions.

   As a byproduct, this result shows that it is possible in some cases to transcend dictionary compression by simply applying BBWT before applying dictionary compression.

[1] More precisely, $r = O(\delta \log \delta \max(1, \log \frac{n}{\delta \log \delta}))$

2 Preliminaries

Let Σ denote a set of symbols called the alphabet. A string is an element of Σ^*. If $w = xyz$ for any strings w, x, y, z, then, x, y, z are respectively a *prefix*, *substring*, and *suffix* of w and are a *proper* prefix, substring, or suffix if they are not equal to w. The length of string x is denoted by $|x|$. The empty string, i.e., the string of length 0 is denoted by ε. The symbol at position i will be denoted by $x[i]$, and we will use a 0-based index, i.e., $x = x[0]\cdots x[|x|-1]$. For integers $i, j \in [0, |x|)$, let $x[i..j] = x[i]\cdots x[j]$ if $i \leq j$ and $x[i..j] = \varepsilon$ if $i > j$. We will also use $x[i..j) = x[i..j-1]$. For any string x, $x^0 = \varepsilon$, and for any integer $i \geq 1$, $x^i = x^{i-1}x$. A string w is *primitive* if it cannot be represented as $w = x^k$ for some string x and integer $k \geq 2$. For any symbol $c \in \Sigma$ let $|x|_c = |\{i \mid x[i] = c\}|$, i.e., the number of c's in x.

For any string x, let $\mathsf{rot}(x) = x[1..|x|)x[0]$, denote a left cyclic rotation of x by one symbol. Notice that for any integer i, $\mathsf{rot}^i(x)$ naturally corresponds to the cyclic rotation of x by i symbols to the left if $i > 0$, and to the right if $i < 0$. We will use $\mathsf{rot}^*(x)$ to denote the lexicographic smallest rotation of x. A *conjugacy class* is an equivalence class of the equivalence relation defined by $x \equiv y \iff \mathsf{rot}^*(x) = \mathsf{rot}^*(y)$.

A string w is a *Lyndon word* if it is lexicographically smaller than any of its, proper rotations, i.e., $w < \mathsf{rot}^k(w)$ for any $k \in [1, |w|)$. From the definition, it is clear that a Lyndon word must be primitive. A rotation of a primitive string x is always primitive and $\mathsf{rot}^*(x)$ is Lyndon and unique, which we will sometimes refer to as the Lyndon rotation of x. It is known that Lyndon words cannot have a non-empty proper border (a proper prefix that is also a proper suffix). The Lyndon factorization [6] of a string w is a unique partitioning of w into a sequence of non-increasing Lyndon words, i.e., $w = L_1^{e_1} \cdots L_{\ell(w)}^{e_{\ell(w)}}$ where each L_i ($i \in [1, \ell(w)]$), which we will call *Lyndon factors* of w, is a Lyndon word and $L_i > L_{i+1}$ for $i \in [1, \ell(w))$. It is known that any occurrence of a Lyndon word in w must be a substring of a Lyndon factor of w.

The ω-order $<_\omega$ between primitive strings or same length strings x, y is defined as $x <_\omega y \iff x^\infty < y^\infty$.[2] The ω-order can be different from the standard lexicographic order, but are identical when comparing strings of the same length or when comparing Lyndon words.

For a string $x[0..|x|)$, let $\mathsf{rank}_c(i, x) = |x[0..i)|_c$, i.e., the number of symbols c in $x[0..i)$. Let $\rho(x)$ denote the size of the run-length encoding of x, i.e., the maximal number of same symbol runs in x. Let $\rho_c(x)$ denote the number of runs of symbol c in the run-length encoding of x.

2.1 Repetitiveness Measures

A set of positions Γ is a string attractor [15] of w if any substring of w has an occurrence in w that covers a position in Γ. The size of the smallest string attractor of w is denoted by $\gamma(w)$. The measure δ [16] is defined as $\max_{k \in [1,|w|]} S_k/k$, where S_k is the number of distinct length-k substrings of w.

The Burrows-Wheeler transform (BWT) [5] $\mathsf{BWT}(w)$ of a string w is defined as the sequence of last (or equivalently, previous, in the cyclic sense) symbols of all rotations of w, in lexicographic order of the rotations. The size of the run-length encoding of $\mathsf{BWT}(w)$ will be denoted by $r(w)$, i.e., $r(w) = \rho(\mathsf{BWT}(w))$. The bijective BWT (BBWT) [11] $\mathsf{BBWT}(w)$

[2] As we will not be comparing non-primitive strings of different lengths in this paper, the definition here is a simplified version of the original.

of a string w is defined as the sequence of last (or again, previous, in the cyclic sense) symbols of all the rotations of all the Lyndon factors of w, in ω-order of the rotations. Actually, BWT can be understood as a special case of BBWT: $\mathsf{BWT}(w) = \mathsf{BBWT}(\mathsf{rot}^*(w))$ because $\mathsf{rot}^*(w) = L^e$ for some Lyndon word L and integer $e = |w|/|L|$, and the ω-order is equivalent to lexicographic order in this case since all the compared strings are of the same length. The size of the run-length encoding of $\mathsf{BBWT}(w)$ will be denoted by $r_B(w)$, i.e., $r_B(w) = \rho(\mathsf{BBWT}(w))$.

The inverse transform of BWT and BBWT on a string x can be defined by the *LF mapping*, which is a function $\Psi_x(i) = j$ over positions of x where $c = x[i] = s[j]$, $\mathsf{rank}_c(i, x) = \mathsf{rank}_c(j, s)$, and s is the string obtained by sorting the multiset of symbols of x in increasing order. Ψ_x is a permutation and thus forms cycles on the set of positions of x. A cycle $(i, \Psi^1(i), \cdots, \Psi^{k-1}(i))$ where k is the smallest positive integer such that $\Psi^k(i) = i$, corresponds to a (cyclic) string $x[\Psi^{k-1}(i)] \cdots x[\Psi^1(i)]x[i]$. This string is always primitive, and by concatenating, in non-increasing order, all the Lyndon rotations of the strings corresponding to all cycles, it can be shown that $\mathsf{BBWT}^{-1}(x)$ is obtained. Ψ_x can be interpreted as returning, given the ω-order rank of a given rotation of a cycle, the ω-order rank of the previous rotation of the cycle. Note that when x is a BWT image of a primitive string, Ψ_x consists of only a single cycle and that although the Lyndon rotation will give the string w for which $\mathsf{BBWT}(w) = x$, we have $\mathsf{BWT}(w') = x$ for any rotation w' of w.

A Bidirectional Macro Scheme (BMS) [23] of a string w is a partitioning of w into phrases, where each phrase is either a single symbol, or is a substring that has an occurrence elsewhere in w which we call the source, or the reference of the phrase. The references of the phrases must be such that the induced reference for each position in the phrases are acyclic, i.e., the referencing on the position forms a forest where the roots are the positions corresponding to single symbol phrases. The size of the smallest bidirectional macro scheme for w is denoted by $b(w)$.

The LZ77 factorization [25] of a string w is a BMS of w where all references are left-referencing, i.e., they must point to a smaller position, and the phrases are determined greedily from left-to-right. It is known that LZ77 is the smallest among left-referencing BMS. The size of LZ77 of w is denoted by $z(w)$.

The lex-parse [22] of a string w is a BMS of w where all references point to a rotation of smaller lexicographic (or equivalently ω-order) rotation, and the phrases are determined greedily from left-to-right. Similarly, it is known that lex-parse is the smallest among BMS with such constraint. The size of lex-parse of w is denoted by $v(w)$.

2.2 Fibonacci Words

Since we will later use Fibonacci words or their slight modifications to show some of our results, we introduce them here.

The Fibonacci words are defined recursively as: $F_0 = \mathtt{b}$, $F_1 = \mathtt{a}$, and for any integer $i \geq 2$, $F_i = F_{i-1}F_{i-2}$. Fibonacci words can also be defined via a morphism ϕ defined as $\phi(\mathtt{a}) = \mathtt{ab}$, and $\phi(\mathtt{b}) = \mathtt{a}$, and $F_i = \phi^i(\mathtt{b})$ for all $i \geq 0$. The lengths of the Fibonacci words correspond to the Fibonacci sequence, i.e., $f_0 = f_1 = 1$, and $f_i = f_{i-1} + f_{i-2}$. For technical reasons, we define $f_i = 0$ for $i < 0$. The observation below follows from a simple induction.

▶ **Observation 1.** *For any $i \geq 1$, $|F_i|_{\mathtt{a}} = f_{i-1}$, and $|F_i|_{\mathtt{b}} = f_{i-2}$.*

3 New results for r vs r_B

3.1 String family giving $r(w)/r_B(w) = \Omega(\log n)$

Here, we answer an open question raised by Badkobeh et al. [1], and show that there exists an infinite family of strings such that r_B is asymptotically strictly smaller than r.

▶ **Theorem 2.** *There exists an infinite family of strings such that $r = \Omega(r_B \log n)$.*

Proof. Consider string $w_k = \mathtt{b}F^*_{2k}$, where $F^*_{2k} = \mathsf{rot}^*(F_{2k})$. Then, for any $k \geq 3$, $r_B(w_k) = 3$ (Lemma 3), and $r(w_k) = 2k$ (Corollary 6). ◀

▶ **Lemma 3.** *Let $w_k = \mathtt{b}F^*_{2k}$, where $F^*_{2k} = \mathsf{rot}^*(F_{2k})$. Then, $r_B(w_k) = 3$ for any $k \geq 3$.*

Proof. The Lyndon factorization of w_k results in the factors $\mathtt{b}$ and F^*_{2k}, and it is known that $\mathsf{BBWT}(F^*_{2k}) = \mathsf{BWT}(F^*_{2k}) = \mathtt{b}^{f_{2k-2}}\mathtt{a}^{f_{2k-1}}$. Since $\mathtt{b}$ is greater, in ω-order, than any rotation of F^*_{2k}, we have $\mathsf{BBWT}(w_k) = \mathtt{b}^{f_{2k-2}}\mathtt{a}^{f_{2k-1}}\mathtt{b}$ and $r_B(w_k) = 3$. ◀

The rest of the proof will focus on showing $r(w_k) = 2k$. We show two proofs, one that relies heavily on previous results, and the other an alternate direct proof.

3.1.1 Proof for $r(w_k) = 2k$

Actually, it turns out that w_k is a rotation of the strings whose BWT are shown to have $2k$ runs in Proposition 3 of [12] and in Proposition 3 of [13]. The former defines it as $v_{2k}\mathtt{b}$ for $v_{2k} = (F_{2k})^R$, where w^R is the reverse of string w, i.e., $(w^R)[i] = w[|w| - 1 - i]$. The latter defines it as inserting $\mathtt{b}$ at position $f_{2k-1} - 2$ of F_{2k}, and mentions that it is a rotation of the former string. Our proof of $r(w_k) = 2k$ further connects these strings with the lexicographically smallest rotation F^*_{2k} of F_{2k}.

The following is known:

▶ **Theorem 4** (Theorem 1 in [7]). *The rotation of F_n with rank ρ in the lexicographically sorted list of all the rotations of F_n, for $n \geq 2$, $\rho \in [0..f_n)$ is the rotation $\mathsf{rot}^i(F_n)$ where*

$$i = \begin{cases} (\rho \cdot f_{n-2} - 1) \bmod f_n & \text{if } n \text{ odd} \\ (-(\rho + 1) \cdot f_{n-2} - 1) \bmod f_n & \text{if } n \text{ even.} \end{cases}$$

Therefore, we have

▶ **Corollary 5.** *For every integer $k \geq 1$, $F^*_{2k} = \mathsf{rot}^{f_{2k-1}-1}(F_{2k})$.*

Proof. From Theorem 4 with $\rho = 0$, $F^*_{2k} = \mathsf{rot}^i(F_{2k})$ where $i = -(f_{2k-2} + 1) \equiv f_{2k-1} - 1 \pmod{f_{2k}}$. ◀

which means that inserting a $\mathtt{b}$ at position $f_{2k-1} - 2$ of F_{2k} is a rotation of $\mathtt{b}F^*_{2k}$, leading to:

▶ **Corollary 6.** *Let $w_k = \mathtt{b} \cdot F^*_{2k}$, where $F^*_{2k} = \mathsf{rot}^*(F_{2k})$. Then, $r(w_k) = 2k$ for any $k \geq 3$.*

3.1.2 Alternate proof for $r(w_k) = 2k$

We also give an alternate direct proof based on morphisms, which is perhaps slightly simpler compared to the proof for $r(v_{2k}\mathtt{b}) = 2k$ presented in [12] which relies on additional results on *special factors* of standard words [4]. Results on morphisms and their effects on r have been studied by Fici et al. [9], but to the best of our knowledge, their results do not directly apply to our case.

We use a string morphism θ defined as: $\theta(\mathtt{a}) = \mathtt{aab}$, $\theta(\mathtt{b}) = \mathtt{ab}$. The following claim can be shown by a simple induction.

▷ Claim 7 (Claim 5 in [20]). For any string $w \in \{\mathtt{a}, \mathtt{b}\}^*$, $\theta(w) = \mathsf{rot}^2(\phi^2(w))$.

To prove Theorem 2, we first show the following lemma.

▶ **Lemma 8.** *For every integer $k \geq 0$, $F^*_{2k} = \theta^k(\mathtt{b})$.*

Proof. It clearly holds for $k = 0, 1$. For $k \geq 2$, we show that $\theta^k(\mathtt{b}) = \mathsf{rot}^{f_{2k-1}-1}(F_{2k})$ by induction on k. We can see that the statement holds for $k = 2$ since $\theta^2(\mathtt{b}) = \mathtt{aabab} = \mathsf{rot}^2(F_4)$. Suppose that $\theta^{k'}(\mathtt{b}) = \mathsf{rot}^{f_{2k'-1}-1}(F_{2k'})$ holds for some $k' \geq 2$. We have

$$\begin{aligned}\theta^{k'+1}(\mathtt{b}) &= \theta(\theta^{k'}(\mathtt{b})) = \theta(\mathsf{rot}^{f_{2k'-1}-1}(F_{2k'})) && \text{by induction hypothesis}\\ &= \theta(\mathsf{rot}^{-1}(F_{2k'-2}F_{2k'-1}))\\ &= \mathsf{rot}^2(\phi^2(\mathsf{rot}^{-1}(F_{2k'-2}F_{2k'-1})) && \text{by Claim 7}\end{aligned}$$

Since the last symbol of $F_{2k'}$ is $F_{2k'}[f_{2k'-1} - 1] = \mathtt{a}$ and $|\phi^2(\mathtt{a})| = 3$, we have

$$\begin{aligned}\theta^{k'+1}(\mathtt{b}) &= \mathsf{rot}^2(\mathsf{rot}^{-3}(F_{2k'}F_{2k'+1}))\\ &= \mathsf{rot}^{f_{2k'+1}-1}(F_{2k'+2})\\ &= \mathsf{rot}^{f_{2(k'+1)-1}-1}(F_{2(k'+1)})\end{aligned}$$

Hence, the statement also holds for $k = k' + 1$, and $\theta^k(\mathtt{b}) = \mathsf{rot}^{f_{2k-1}-1}(F_{2k})$ for every $k \geq 2$. By combining it with Corollary 5, we obtain $F^*_{2k} = \theta^k(\mathtt{b})$. ◀

We are ready to prove the following lemma.

▶ **Lemma 9.** *Let $w_k = \mathtt{b}F^*_{2k}$, where $F^*_{2k} = \mathsf{rot}^*(F_{2k})$. Then, $r(w_k) = 2k$ for any $k \geq 2$.*

Proof. We first prove that $r(y_k) = 2k + 2$ for $y_k = \mathtt{c}F^*_{2k}$ for any $k \geq 3$, where $\mathtt{c}$ is a symbol that is greater than $\mathtt{a}, \mathtt{b}$. From Lemma 8, we have $y_k = \mathtt{c}F^*_{2k} = \mathtt{c}\theta^k(\mathtt{b})$. Below, we extend the definition of θ so that $\theta(\mathtt{c}) = \mathtt{c}$, so $y_k = \theta^k(\mathtt{cb})$. We claim that

$$\mathsf{BWT}(y_k) = \mathtt{cb}^{f_{2k-2}-k} \prod_{j=1}^{k} (\mathtt{a}^{f_{2j-2}}\mathtt{b}) \tag{1}$$

We show this statement by induction on k. If $k = 3$, $y_3 = \mathtt{caabaababaabab}$ and $\mathsf{BWT}(y_3) = \mathtt{cb}^2\mathtt{aba}^2\mathtt{ba}^5\mathtt{b}$ hold. Assume that the statement holds for some $k' \geq 3$. We show the statement also holds for $k = k' + 1$. Figure 1 depicts $y_{k'+1} = \theta(y_{k'})$. We consider the three disjoint sets of the rotations of $y_{k'+1}$ as follows: the rotations (i) starting with $\mathtt{a}^2$, (ii) starting with $\mathtt{ab}$, (iii) starting with $\mathtt{b}$ or $\mathtt{c}$.

(i) From the definition of θ, the number of rotations starting with $\mathtt{a}^2$ of $y_{k'+1}$ is $f_{2k'-1}$ (= $|F_{2k'}|_{\mathtt{a}}$). These rotations are the first $f_{2k'-1}$ rotations in the lexicographically increasingly sorted list of all rotations of $y_{k'+1}$. By the definition of y_k, the lexicographically smallest rotation is $\mathsf{rot}(y_{k'})$, and the preceding symbol of this rotation is $\mathtt{c}$. The other rotations of this case are preceded by $\mathtt{b}$. Hence, $\mathsf{BWT}(y_{k'+1})[0..f_{2k'-1} - 1] = \mathtt{cb}^{f_{2k'-1}-1}$.

(ii) Let pos be the set of positions in $y_{k'+1}$ that are preceded by an occurrence of $\mathtt{ab}$. From the definitions of y_k and θ, every occurrence of $\mathtt{ab}$ in $y_{k'+1}$ is a suffix of an occurrence of the substrings ($\mathtt{aab}$ or $\mathtt{ab}$) produced from θ and an occurrence of $\mathtt{a}$ or $\mathtt{b}$ in $y_{k'}$. Therefore, we have $|\mathsf{pos}| = |y_{k'}| - 1 = f_{2k'}$ and if $s(i)$ denotes the position in $y_{k'}$ that produces the corresponding symbol at position i in $y_{k'+1}$ by the morphism θ, then $\mathsf{rot}^i(y_{k'+1}) = \theta(\mathsf{rot}^{s(i)}(y_{k'}))$ for any $i \in \mathsf{pos}$. Thus, for any positions $i_1, i_2 \in \mathsf{pos}$,

$$\begin{aligned}\mathsf{rot}^{i_1-2}(y_{k'+1}) < \mathsf{rot}^{i_2-2}(y_{k'+1}) &\iff \mathsf{rot}^{i_1}(y_{k'+1}) < \mathsf{rot}^{i_2}(y_{k'+1})\\ &\iff \theta(\mathsf{rot}^{s(i_1)}(y_{k'})) < \theta(\mathsf{rot}^{s(i_2)}(y_{k'}))\\ &\iff \mathsf{rot}^{s(i_1)}(y_{k'}) < \mathsf{rot}^{s(i_2)}(y_{k'}). \qquad (2)\end{aligned}$$

where the last relation follows from the fact that θ is an order preserving morphism (i.e., $s < t \Leftrightarrow \theta(s) < \theta(t)$ holds for any s, t). We can also see that for any $i \in \mathsf{pos}$, the symbol $y_{k'+1}[i-3]$ preceding the occurrence of $\mathtt{ab}$ is $\mathtt{a}$ (resp. $\mathtt{b}$) iff $y_{k'}[s(i)-1]$ is $\mathtt{a}$ (resp. $\mathtt{b}$). Thus, together with Equation (2), the sequence of entries in $\mathsf{BWT}(y_{k'+1})$ corresponding to rotations that start with $\mathtt{ab}$ are equivalent to the sequence of entries in $\mathsf{BWT}(y_{k'})$ for rotations that start with $\mathtt{a}$ or $\mathtt{b}$, i.e., $\mathsf{BWT}(y_{k'+1})[f_{2k'-1}..f_{2k'+1}-1] = \mathsf{BWT}(y_{k'})[1..f_{2k'}] = \mathtt{b}^{f_{2k'-2}-k'} \prod_{j=1}^{k'}(\mathtt{a}^{f_{2j-2}}\mathtt{b})$. See also Figure 1.

(iii) From the definition of θ, the number of rotations starting with $\mathtt{b}$ in $y_{k'+1}$ is $f_{2k'}$, and the lexicographically largest rotation is $y_{k'+1}$ itself. These rotations are the last $f_{2k'}+1$ rotations in the lexicographically increasingly sorted list of all rotations of $y_{k'+1}$. The preceding symbol of the largest rotation, or equivalently, the last symbol of y_k is $\mathtt{b}$. All rotations starting with $\mathtt{b}$ are preceded by $\mathtt{a}$. Hence, $\mathsf{BWT}(y_{k'+1})[f_{2k'+1}..f_{2k'+2}] = \mathtt{a}^{f_{2k'}}\mathtt{b}$.

All together, we have

$$\begin{aligned}\mathsf{BWT}(y_{k'+1}) &= \overbrace{\mathtt{cb}^{f_{2k'-1}-1}}^{\text{(i)}} \cdot \overbrace{\mathtt{b}^{f_{2k'-2}-k'} \prod_{j=1}^{k'}(\mathtt{a}^{f_{2j-2}}\mathtt{b})}^{\text{(ii)}} \cdot \overbrace{\mathtt{a}^{f_{2k'}}\mathtt{b}}^{\text{(iii)}}\\ &= \mathtt{cb}^{f_{2k'}-(k'+1)} \prod_{j=1}^{k'+1}(\mathtt{a}^{f_{2j-2}}\mathtt{b})\end{aligned}$$

and the statement holds for $k = k'+1$, proving Equation (1). Therefore, $r(y_k) = 2k+2$. We note that Equation (1) holds for $k = 2$, but since $f_{2k-2} - 2 = 0$, $r(y_2) = 5$.

The last symbol of F^*_{2k} is $\mathtt{b}$, so changing the $\mathtt{c}$ in y_k to $\mathtt{b}$ changes the unique occurrence of $\mathtt{bc}$ in all rotations of y_k except for y_k itself, to the unique occurrence of $\mathtt{bb}$ in all rotations of w_k except for w_k itself. Therefore, the lexicographic order of rotations of w_k are equivalent to those in y_k, except for w_k itself. We can see that $\mathsf{BWT}(w_k)$ will have the following changes compared to $\mathsf{BWT}(y_k)$: 1) the lexicographically smallest rotation $\mathsf{rot}(w_k)$ is preceded by $\mathtt{b}$ instead of $\mathtt{c}$, and 2) w_k is the lexicographically smallest rotation that starts with $\mathtt{b}$ (rather than $\mathtt{c}$). 1) changes $\mathtt{c}$ to $\mathtt{b}$ and thus decreases the run-length by 1. 2) moves the last $\mathtt{b}$ right after the second to last $\mathtt{b}$ in $\mathsf{BWT}(y_k)$ (the preceding symbol of the largest rotation starting with $\mathtt{a}$) and thus decreases the run-length by 1. Therefore, in total, $r(w_k) = 2k$ which can be confirmed to hold for $k = 2$ as well. ◀

3.2 Bounds for $r(w)/r_B(w)$

We show poly-logarithmic upper and lower bounds on the ratio $r(w)/r_B(w)$ for any string.

We first show a simple upper bound on the multiplicative rotation sensitivity of z.

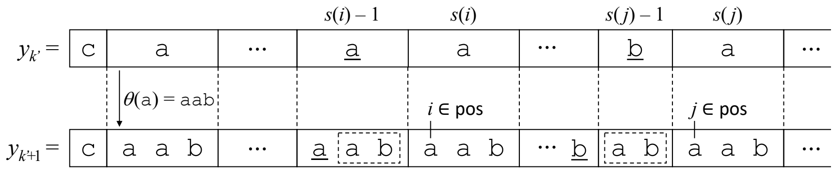

Figure 1 Illustration of the proof of Lemma 9. Every position in the set pos is preceded by an occurrence of ab. We can obtain the lexicographic rank of the rotation $\mathsf{rot}^{i-2}(y_{k'+1})$ that starts with ab by using the corresponding rotation $\mathsf{rot}^{s(i)}(y_{k'})$. The preceded symbols (underlined symbols) of $y_{k'+1}[i-3]$ and $y_{k'}[s(i)-1]$ are the always same. The figure shows the case of $y_{k'+1}[j-3] = y_{k'}[s(j)-1] = \mathtt{a}$ by the position i and the case of $y_{k'+1}[j-3] = y_{k'}[s(j)-1] = \mathtt{b}$ by the position j.

▶ **Lemma 10.** *For any strings w, w' of length n in the same conjugacy class, $z(w') = O(z(w)\log n)$.*

Proof. It is easy to see that $b(w') = \Theta(b(w))$. This is because, given any BMS of w of size b, we can reuse the parsing and referencing, except for the following changes, to construct a BMS for w' of size $2b+1$: (1) if w' starts in the middle of a BMS phrase of w, we split the phrase, and (2) if a phrase references a substring of w but is split in w', we split the phrase. Since only one of the phrases of the split phrase in (1) is split by (2), the total number of phrases is at most $2b+1$.

Since $b(w) \leq z(w) = O(b(w)\log n)$ and $b(w') \leq z(w') = O(b(w')\log n)$, we have $z(w')/z(w) = O(\log n)$. ◀

▶ **Lemma 11.** *For any strings w, w' of length n in the same conjugacy class, $r_B(w') = O(r_B(w)\log^4 n)$.*

Proof. Badkobeh et al. [1] showed (1) $b(w) = O(r_B(w))$, which implies $z(w) = O(r_B(w)\log n)$, and (2) $r_B(w) = O(z(w)\log^2 n)$. Therefore, $r_B(w')/r_B(w) = O(z(w')/z(w)\log^3 n) = O(\log^4 n)$ from Lemma 10. ◀

▶ **Corollary 12.** *For any string w of length n, $r_B(w) = O(r(w)\log^4 n)$ and $r(w) = O(r_B(w)\log^4 n)$.*

4 Clustering effects of BWT and BBWT

4.1 In terms of ρ

The clustering effect of BWT was measured by the length ρ of the run-length encoding. Namely, the following statement was claimed by Mantaci et al. [19].

▶ **Theorem 13** (Theorem 3.3 in [19])**.** *For any string w, $\rho(\mathsf{BWT}(w)) \leq 2\rho(w)$.*

While the statement is true, we believe there is a non-trivial case that was not covered in their original proof. Here, we address this case and also show that the same statement holds for BBWT. Note that the following Theorem 14 implies Theorem 13 because $\mathsf{BWT}(w) = \mathsf{BBWT}(\mathsf{rot}^*(w))$ and $\rho(\mathsf{rot}^*(w)) \leq \rho(w)$.

▶ **Theorem 14.** *For any string w, $\rho(\mathsf{BBWT}(w)) \leq 2\rho(w)$.*

Proof. We first recall the arguments of [19] for BWT. Consider a range $[i..j]$ of $x = \mathsf{BWT}(w)$ that corresponds to rotations of w that start with a symbol $c \in \Sigma$. Then, $x[i..j]$ will consist of $|w|_c - \rho_c(w)$ occurrences of c's that correspond to those preceding a c in the same run of c's in w, and $\rho_c(w)$ occurrence of symbols not equal to c that precede a maximal run of c's in w. Thus, $\rho(x[i..j])$ is maximized when the $\rho_c(w)$ non-c symbols in $x[i..j]$ are not adjacent to each other. Mantaci et al. argued that this implies $\rho(x[i..j]) \leq 2\rho_c(w)$ summing up to $2\rho(w)$ for all c. However, they did not give a reason to why $\rho(x[i..j])$ could not be $2\rho_c(w)+1$ (summing up to $\rho(w)+\sigma$), which is possibly the case when $x[i..j]$ starts and ends with the symbol c. We show that in such a case, an LF mapping with a single cycle cannot be defined, and ultimately, $\rho(x[i..j]) \leq 2\rho_c(w)$ holds.

Assume that $x[i] = x[j] = c$ and that all $\rho_c(w)$ non-c symbols in $x[i..j]$ are separated by one or more c's. We claim that in such a case, there exists a position $k \in [i..j]$ s.t. $\Psi_x(k) = k$ resulting in a cycle of a single symbol. If there was no c in $x[0..i)$, then, since $x[i]$ is the first c in x, $\Psi_x(i) = i$ must hold. Thus, consider the case that there exist c's in $x[0..i)$. For $i \leq k \leq j$, since the difference in the number of c's in $x[0..k]$ and the length of the range $[i..k]$ (corresponding to the $k-i+1$ lexicographically (or in ω order) smallest rotations that start with c) is monotonically non-increasing in k and changes by at most 1, there exists some $i \leq k' \leq j$ such that $|x[0..k')|_c = |x[0..k']|_c = k'-i+1$. Note that such k' must necessarily satisfy $x[k'] \neq c$, and therefore $k' < j$. However, from the assumption of $x[i..j]$, $x[k'+1] = c$ holds and since $|x[0..k'+1]|_c = k'-i+2$, this implies that $\Psi_x(k'+1) = k'+1$. Therefore $\rho(x[i..j]) \leq 2\rho_c(w)$ holds.

For $x = \mathsf{BBWT}(w)$, similarly consider the range $[i..j]$ of $\mathsf{BBWT}(w)$ such that the corresponding rotations of the Lyndon factors of w start with the symbol $c \in \Sigma$. Since it is known that Lyndon factors of w (except for single symbol factors) must start and end at run-boundaries of w [10], a maximal run of c's in w is either a substring of a Lyndon factor of w, or a maximal run of a single symbol Lyndon factors. If it is a substring of a Lyndon factor but not a prefix, then there is one non-c symbol per such run that precedes the run and occurs in $x[i..j]$. If it is a prefix of a Lyndon factor longer than 1, then the previous symbol (or the last symbol of the Lyndon factor) is non-c due to the Lyndon factor not having a proper border. Otherwise, the maximal run is a maximal run of single symbol Lyndon factors c, in which case the previous symbol for these occurring in $x[i..j]$ will also be c. Thus, the arguments in the previous paragraph for BWT hold for BBWT as well, except for the last part: it can be that $x[i] = x[j] = c$ and $\Psi(k') = k'$ for some $k' \in [i, j]$. However, this implies that this corresponds to a single symbol Lyndon factor which would not have introduced a non-c symbol in $x[i..j]$. Therefore $\rho(x[i..j]) \leq 2\rho_c(w)$ still holds. ◀

4.2 In terms of other repetitiveness measures

We consider measuring the clustering effect of BWT and BBWT other than ρ. Here, we show that BWT or BBWT can only increase the repetitiveness of the string by a polylogarithmic factor.

▶ **Theorem 15.** *For repetitiveness measure $\mu \in \{\delta, \gamma, b, v, z, g_{\mathrm{rl}}\}$ and any string w of length n, it holds that $\mu(\mathsf{BWT}(w)) = O(\mu(w)\log^2 n)$. Moreover, $\mu(\mathsf{BBWT}(w)) = O(\mu(w)\log^2 n)$ for $\mu \in \{z, g_{\mathrm{rl}}\}$, and $\mu(\mathsf{BBWT}(w)) = O(\mu(w)\log^6 n)$ for $\mu \in \{\delta, \gamma, b, v\}$.*

Proof. The proof for BWT follows from a simple observation that for any string w, $\mu(w) = O(\rho(w))$ implying $\mu(\mathsf{BWT}(w)) = O(\rho(\mathsf{BWT}(w))$. Since $r(w) = \rho(\mathsf{BWT}(w)) = O(\delta(w)\log^2 n)$, we have $\mu(\mathsf{BWT}(w)) = O(\delta(w)\log^2 n) = O(\mu(w)\log^2 n)$, where the last relation follows from the fact that δ lower bounds all the repetitiveness measures considered.

For BBWT, $\mu(\mathsf{BBWT}(w)) = O(\rho(\mathsf{BBWT}(w)))$ holds, but only $r_B(w) = \rho(\mathsf{BBWT}(w)) = O(z(w)\log^2 n)$ has been proved, from which $r_B(w) = O(g_{\mathrm{rl}}(w)\log^2 n)$ and the statement follows for $\mu \in \{z, g_{\mathrm{rl}}\}$. For the other measures, Corollary 12 gives us $r_B(w) = O(r(w)\log^4 n) = O(\delta(w)\log^6 n)$ from which the statement follows. ◀

4.3 A family of significantly "clustered" strings by BBWT

Finally, we show an infinite family of strings for which BBWT exhibits an asymptotically maximal clustering effect in terms of δ, γ, b, v, r, and slightly weaker in terms of z, g_{rl}, g. Namely, we consider the family of strings whose BBWT image are Fibonacci words.

▶ **Theorem 16.** *Let w be a string of length n where $\mathsf{BBWT}(w) = F_k$, where k is prime. Then, for $\mu \in \{\delta, \gamma, b, v, r\}$, $\mu(\mathsf{BBWT}(w)) = O(\frac{\log n}{n}\mu(w))$ and for $\mu \in \{z, g_{\mathrm{rl}}, g\}$, $\mu(\mathsf{BBWT}(w)) = O(\frac{\log^2 n}{n}\mu(w))$.*

In order to understand w, we first introduce the tools we use to characterize the LF mapping Ψ_{F_k} on Fibonacci words.

The Zeckendorf representation [24] of a non-negative integer is a unique (sub-)set of distinct non-consecutive Fibonacci numbers $\{f_k \mid k \geq 1\}$ that sum up to the integer. We represent a non-negative integer i as a bit string $Z(i)$, where $i = ||Z(i)|| = \sum_{j\geq 0} Z(i)[j] \cdot f_{j+1}$. We use $Z_k(i)$ to denote the length-k prefix $Z(i)[0..k-1]$ of $Z(i)$. Note that for any $i \in [0, f_k)$, it suffices that a subset of $\{f_1, \ldots, f_{k-1}\}$ is used, and thus $Z(i)$ requires only up to $(k-1)$ bits, i.e., 1's will only occur in $Z(i)[0..k-2]$, and the rest $Z(i)[k-1..]$ will be 0.

We observe that length $i \geq 0$ prefix of a Fibonacci word can be uniquely factorized into a sequence of distinct non-consecutive Fibonacci words in order of decreasing length, where the length of each Fibonacci word corresponds to an element in the Zeckendorf representation of i.

▶ **Lemma 17.** *For any $i \geq 0$,*

$$F_\infty[0..i) = \prod_{j=k-2}^{0} F_{j+1}^{Z(i)[j]} = F_{k-1}^{Z(i)[k-2]} \cdots F_1^{Z(i)[0]}, \tag{3}$$

where k is the smallest integer such that $i < f_k$.

Proof. Consider a greedy factorization of $F_\infty[0..i)$ that takes the longest prefix of the remaining string that is a Fibonacci word. Let k be the smallest integer such that $i \leq f_k$. If $i = f_k$ then we simply take F_k as the last element. Otherwise, we take F_{k-1}. Notice that in this case, the remaining string is $F_\infty[f_{k-1}..i) = F_k[f_{k-1}..i) = F_{k-2}[0..i - f_{k-1})$ of length $i' = i - f_{k-1} < f_{k-2}$, and we repeat the process to find a greedy factorization of $F_{k-2}[0..i')$. The remaining string is a proper prefix of F_{k-2} and thus F_{k-2} will not be chosen next, implying that consecutive Fibonacci words are not chosen. Thus, the lengths of the sequence of strings will be a set of non-consecutive set of distinct Fibonacci numbers that sum up to i. The lemma follows from the uniqueness of the Zeckendorf representation of i. ◀

The following relation about the least significant bit in the Zeckendorf representation and the ith symbol in the (infinite) Fibonacci word is known.

▶ **Lemma 18** (Problem 6 in [8]). *For $i = 0, 1, \ldots,$*

$$F_\infty[i] = \begin{cases} \mathtt{a} & \text{if } Z(i)[0] = 0, \\ \mathtt{b} & \text{if } Z(i)[0] = 1. \end{cases}$$

Now, we make observations on the LF-mapping for Fibonacci words.

▶ **Lemma 19.** *The LF-mapping function for the Fibonacci word F_k is, for any $0 \leq i < f_k$,*

$$\Psi_{F_k}(i) = \begin{cases} \mathsf{rank}_{\mathsf{a}}(i, F_k) & \text{if } F_k[i] = \mathsf{a}, \\ f_{k-1} + \mathsf{rank}_{\mathsf{b}}(i, F_k) & \text{if } F_k[i] = \mathsf{b}. \end{cases}$$

Proof. Straightforward from the definition of Ψ_{F_k} and Observation 1. ◀

The next lemma shows that the LF mappings on F_k can be interpreted as rotation operations on the Zeckendorf representation of the position to be mapped.

▶ **Lemma 20.** *For any $i \in [0, f_k)$,*

$$Z_k(\Psi_{F_k}(i)) = \begin{cases} \mathsf{rot}(Z_k(i)) & \text{if } F_k[i] = \mathsf{a}, \\ \mathsf{rot}^2(Z_k(i)) & \text{if } F_k[i] = \mathsf{b}. \end{cases}$$

Proof. See Figure 2 for a concrete example. Let $j = \Psi_{F_k}(i)$. Since $i, j \in [0, f_k)$, the most significant bits of $Z_k(i)$ and $Z_k(j)$, corresponding to f_k, are always zero, i.e., $Z_k(i)[k-1] = Z_k(j)[k-1] = 0$.

From Lemma 18, $F_k[i] = \mathsf{a}$ implies that $Z_k(i)[0] = 0$. Thus, $\mathsf{rot}(Z_k(i)) = Z_k(i)[1..k-1] \cdot 0$. Therefore, we have

$$\begin{aligned} \Psi_{F_k}(i) &= \mathsf{rank}_{\mathsf{a}}(i, F_k) && \text{by Lemma 19} \\ &= \left| \prod_{j=k-2}^{0} F_{j+1}^{Z(i)[j]} \right|_{\mathsf{a}} && \text{by Lemma 17} \\ &= \sum_{j=0}^{k-2} Z_k(i)[j] \cdot f_j = \sum_{j=0}^{k-1} Z_k(i)[j] \cdot f_j \\ &= Z_k(i)[0] \cdot f_k + \sum_{j=0}^{k-2} Z_k(i)[j+1] \cdot f_{j+1} \\ &= \sum_{j=0}^{k-1} Z_k(i)[(j+1) \bmod k] \cdot f_{j+1} \\ &= ||\mathsf{rot}(Z_k(i))|| \end{aligned}$$

On the other hand, again from Lemma 18, $F_k[i] = \mathsf{b}$ implies that $Z_k(i)[0] = 1$ and since the Zeckendorf representation does not use consecutive Fibonacci numbers, $Z_k(i)[1] = 0$. Thus, $\mathsf{rot}^2(Z_k(i)) = Z_k(i)[2..k-1] \cdot 1 \cdot 0$. Therefore, we have

$$\begin{aligned} \mathsf{rank}_{\mathsf{b}}(i, F_k) &= \left| \prod_{j=k-2}^{0} F_{j+1}^{Z(i)[j]} \right|_{\mathsf{b}} && \text{by Lemma 17} \\ &= \sum_{j=0}^{k-2} Z_k(i)[j] \cdot f_{j-1} = \sum_{j=0}^{k-1} Z_k(i)[j] \cdot f_{j-1} \\ &= Z_k(i)[0] \cdot f_{-1} + Z_k(i)[1] \cdot f_0 + \sum_{j=0}^{k-3} Z_k(i)[j+2] \cdot f_{j+1} \\ &= \sum_{j=0}^{k-3} Z_k(i)[j+2] \cdot f_{j+1} \end{aligned}$$

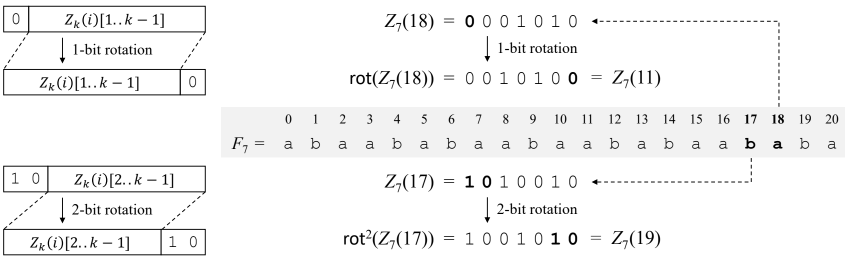

Figure 2 Example of Lemma 20 for F_7. The `a` at position 18 is the 12th `a` in $F_7[0..18]$, and thus the LF mapping should point to position 11, whose Zeckendorf representation $Z_7(11)$ is a 1-bit left rotation of $Z_7(18)$. The `b` at position 17 is the 7th `b` in $F_7[0..17]$, and since there are 13 `a`'s in F_7, the LF mapping should point to position $13 + 7 - 1 = 19$, whose Zeckendorf representation $Z_7(19)$ is a 2-bit left rotation of $Z_7(17)$.

Furthermore,

$$\begin{aligned}
\Psi_{F_k}(i) &= \mathsf{rank}_{\mathtt{b}}(i, F_k) + f_{k-1} && \text{by Lemma 19}\\
&= \sum_{j=0}^{k-3} Z_k(i)[j+2] \cdot f_{j+1} + f_{k-1}\\
&= \sum_{j=0}^{k-3} Z_k(i)[j+2] \cdot f_{j+1} + Z_k(i)[0] \cdot f_{k-1} + Z_k(i)[1] \cdot f_k\\
&= \sum_{j=0}^{k-1} Z_k(i)[(j+2) \bmod k] \cdot f_{j+1}\\
&= ||\mathsf{rot}^2(Z_k(i))||.
\end{aligned}$$

thus proving the lemma (see also Figure 2). ◀

We are now ready to prove Theorem 16:

▶ **Theorem 16.** *Let w be a string of length n where $\mathsf{BBWT}(w) = F_k$, where k is prime. Then, for $\mu \in \{\delta, \gamma, b, v, r\}$, $\mu(\mathsf{BBWT}(w)) = O(\frac{\log n}{n}\mu(w))$ and for $\mu \in \{z, g_{\mathrm{rl}}, g\}$, $\mu(\mathsf{BBWT}(w)) = O(\frac{\log^2 n}{n}\mu(w))$.*

Proof. For any Fibonacci word F_k, $\delta(F_k) \le \gamma(F_k) \le b(F_k) \le v(F_k) \le r(F_k) = 2$ and $z(F_k), g_{\mathrm{rl}}(F_k), g(F_k) = O(\log n)$ are known. In the rest of the proof, we show $\delta(w) = \Omega(n/\log n)$ implying the same lower bound for all other measures which finishes the proof.

From Lemma 20, Ψ_{F_k} can be interpreted as a rotation operation on a k-bit string corresponding to the Zeckendorf representation of the given position. Therefore, the number of cycles that Ψ_{F_k} produces is equivalent to the number of conjugacy classes among bit strings corresponding to the Zeckendorf representations of the set of positions $[0, F_k)$. Since the Zeckendorf representations do not use adjacent Fibonacci numbers, this is equivalent to the number $C(k)$ of binary necklaces of length k that do not contain "11". This corresponds to entry A000358 in the on-line encyclopedia of integer sequences, which is known [3] to be:

$$C(k) = \frac{1}{k}\sum_{d|k} \varphi(k/d)(f_{d-2} + f_d), \tag{4}$$

where $d|k$ represents that d is a divisor of k, and φ is Euler's totient function which returns, for integer n, the number of positive integers less than n that are relatively prime to n (i.e., having gcd of 1). For prime k, Equation (4) becomes:

$$C(k) = \frac{1}{k}\left(\varphi(1)(f_{k-2} + f_k) + \varphi(k)(f_{-1} + f_1)\right) = \frac{1}{k}(f_{k-2} + f_k + k - 1) \geq f_k/k.$$

As seen in the proof of Lemma 20, an $\mathtt{a}$ in F_k implies a 1-bit left rotation on Z_k for Ψ_{F_k}, and $\mathtt{b}$ in F_k implies a 2-bit left rotation on Z_k for Ψ_{F_k}. Each necklace will then correspond to some cyclic string w', where

$$|w'|_{\mathtt{a}} + 2|w'|_{\mathtt{b}} = k, \tag{5}$$

except for the string $\mathtt{a}$ corresponding to the bit string 0^k, since all other necklaces must be primitive (due to k being prime) and a total of k-bits must be rotated in order to come back to the same bit string. Consider the lexicographically smallest rotations of all of them, which are Lyndon words. All of them are distinct, and any two of them (except for the single $\mathtt{a}$) cannot be a substring of the other due to the constraint on their lengths (Equation (5)). The string $w = \mathsf{BBWT}^{-1}(F_k)$ is a concatenation (in non-increasing order) of all these strings with the single $\mathtt{a}$ appended at the end, corresponding to the Lyndon factorization of w. Since Lyndon words can only occur as a substring of a Lyndon factor, it follows that there are $\Omega(C(k))$ distinct uniquely occurring strings (Lyndon factors) that do not overlap. Therefore, $\gamma(w) = \Omega(C(k)) = \Omega(n/\log n)$ holds.

Furthermore, since the uniquely occurring Lyndon factors of w satisfy Equation (5) and have length at most $k-1$, any length $2k$ substring of w will contain at least one full occurrence of a Lyndon factor of w. Since all such strings must also have unique occurrences and therefore be distinct, $\delta(w) \geq (f_k - 2k)/2k = \Omega(n/\log n)$ holds. ◀

For example, for $\mathsf{BBWT}^{-1}(F_7)$, the binary necklaces of length 7 that do not contain "11" are: 0000000, 0000001, 0000101, 0001001, 0010101, which respectively correspond to `a`, `aaaaab`, `aaabb`, `aabab`, and `abbb`, and therefore, $\mathsf{BBWT}^{-1}(F_7) =$ `abbbaababaaabbaaaaaba`.

We note that $\mathsf{BBWT}^{-1}(F_k)$ is asymptotically maximally incompressible by dictionary compression since it is known that for any string, $\delta \leq \gamma \leq z = O(n/\log_\sigma n)$ holds, which is $O(n/\log n)$ for binary strings. We also note that Theorem 16 holds for general k, which will be shown in the full version of the paper.

5 Discussion

The log factors in our bounds, especially in Theorem 15 are most likely not tight; Our focus here was on showing the existence of such bounds. A natural open question is how many log factors can be shaved.

References

1 Golnaz Badkobeh, Hideo Bannai, and Dominik Köppl. Bijective BWT based compression schemes. In Zsuzsanna Lipták, Edleno Silva de Moura, Karina Figueroa, and Ricardo Baeza-Yates, editors, *String Processing and Information Retrieval - 31st International Symposium, SPIRE 2024, Puerto Vallarta, Mexico, September 23-25, 2024, Proceedings*, volume 14899 of *Lecture Notes in Computer Science*, pages 16–25. Springer, 2024. doi:10.1007/978-3-031-72200-4_2.

2 Elena Biagi, Davide Cenzato, Zsuzsanna Lipták, and Giuseppe Romana. On the number of equal-letter runs of the bijective Burrows-Wheeler transform. In Giuseppa Castiglione and Marinella Sciortino, editors, *Proceedings of the 24th Italian Conference on Theoretical Computer Science, Palermo, Italy, September 13-15, 2023*, volume 3587 of *CEUR Workshop Proceedings*, pages 129–142. CEUR-WS.org, 2023. URL: `https://ceur-ws.org/Vol-3587/4564.pdf`.

3 M. Bona. *Handbook of Enumerative Combinatorics*. Discrete Mathematics and Its Applications. CRC Press, 2015. URL: `https://books.google.co.jp/books?id=j3kZBwAAQBAJ`.

4 Jean-Pierre Borel and Christophe Reutenauer. On Christoffel classes. *RAIRO Theor. Informatics Appl.*, 40(1):15–27, 2006. `doi:10.1051/ITA:2005038`.

5 Michael Burrows and David J. Wheeler. A block-sorting lossless data compression algorithm. Technical report, Digital Equipment Corporation, Systems Reseach Center, 1994. SRC Research Report 124.

6 Kuo Tsai Chen, Ralph H. Fox, and Roger C. Lyndon. Free differential calculus, IV. The quotient groups of the lower central series. *Annals of Mathematics*, 68(1):81–95, 1958.

7 Manolis Christodoulakis, Costas S. Iliopoulos, and Yoan José Pinzón Ardila. Simple algorithm for sorting the fibonacci string rotations. In Jirí Wiedermann, Gerard Tel, Jaroslav Pokorný, Mária Bieliková, and Julius Stuller, editors, *SOFSEM 2006: Theory and Practice of Computer Science, 32nd Conference on Current Trends in Theory and Practice of Computer Science, Merín, Czech Republic, January 21-27, 2006, Proceedings*, volume 3831 of *Lecture Notes in Computer Science*, pages 218–225. Springer, 2006. `doi:10.1007/11611257_19`.

8 Maxime Crochemore, Thierry Lecroq, and Wojciech Rytter. *Fibonacci Words and Fibonacci Numeration System*, pages 17–52. Cambridge University Press, 2021.

9 Gabriele Fici, Giuseppe Romana, Marinella Sciortino, and Cristian Urbina. On the impact of morphisms on BWT-runs. In Laurent Bulteau and Zsuzsanna Lipták, editors, *34th Annual Symposium on Combinatorial Pattern Matching, CPM 2023, June 26-28, 2023, Marne-la-Vallée, France*, volume 259 of *LIPIcs*, pages 10:1–10:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. `doi:10.4230/LIPICS.CPM.2023.10`.

10 Yuta Fujishige, Yuto Nakashima, Shunsuke Inenaga, Hideo Bannai, and Masayuki Takeda. Almost linear time computation of maximal repetitions in run length encoded strings. In Yoshio Okamoto and Takeshi Tokuyama, editors, *28th International Symposium on Algorithms and Computation, ISAAC 2017, December 9-12, 2017, Phuket, Thailand*, volume 92 of *LIPIcs*, pages 33:1–33:12. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017. `doi:10.4230/LIPICS.ISAAC.2017.33`.

11 Joseph Yossi Gil and David Allen Scott. A bijective string sorting transform. *CoRR*, abs/1201.3077, 2012. `arXiv:1201.3077`.

12 Sara Giuliani, Shunsuke Inenaga, Zsuzsanna Lipták, Nicola Prezza, Marinella Sciortino, and Anna Toffanello. Novel results on the number of runs of the Burrows-Wheeler-transform. In Tomás Bures, Riccardo Dondi, Johann Gamper, Giovanna Guerrini, Tomasz Jurdzinski, Claus Pahl, Florian Sikora, and Prudence W. H. Wong, editors, *SOFSEM 2021: Theory and Practice of Computer Science - 47th International Conference on Current Trends in Theory and Practice of Computer Science, SOFSEM 2021, Bolzano-Bozen, Italy, January 25-29, 2021, Proceedings*, volume 12607 of *Lecture Notes in Computer Science*, pages 249–262. Springer, 2021. `doi:10.1007/978-3-030-67731-2_18`.

13 Sara Giuliani, Shunsuke Inenaga, Zsuzsanna Lipták, Giuseppe Romana, Marinella Sciortino, and Cristian Urbina. Bit catastrophes for the Burrows-Wheeler transform. In Frank Drewes and Mikhail Volkov, editors, *Developments in Language Theory - 27th International Conference, DLT 2023, Umeå, Sweden, June 12-16, 2023, Proceedings*, volume 13911 of *Lecture Notes in Computer Science*, pages 86–99. Springer, 2023. `doi:10.1007/978-3-031-33264-7_8`.

14 Dominik Kempa and Tomasz Kociumaka. Resolution of the Burrows-Wheeler transform conjecture. *Commun. ACM*, 65(6):91–98, 2022. `doi:10.1145/3531445`.

15 Dominik Kempa and Nicola Prezza. At the roots of dictionary compression: string attractors. In Ilias Diakonikolas, David Kempe, and Monika Henzinger, editors, *Proceedings of the 50th*

Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018, pages 827–840. ACM, 2018. doi:10.1145/3188745.3188814.

16 Tomasz Kociumaka, Gonzalo Navarro, and Nicola Prezza. Toward a definitive compressibility measure for repetitive sequences. *IEEE Transactions on Information Theory*, 69(4):2074–2092, 2023. doi:10.1109/TIT.2022.3224382.

17 Ben Langmead, Cole Trapnell, Mihai Pop, and Steven L. Salzberg. Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. *Genome Biology*, 10:R25, 2009.

18 Heng Li and Richard Durbin. Fast and accurate short read alignment with burrows–wheeler transform. *Bioinformatics*, 25(14):1754–1760, May 2009. doi:10.1093/bioinformatics/btp324.

19 Sabrina Mantaci, Antonio Restivo, Giovanna Rosone, Marinella Sciortino, and Luca Versari. Measuring the clustering effect of BWT via RLE. *Theor. Comput. Sci.*, 698:79–87, 2017. doi:10.1016/J.TCS.2017.07.015.

20 Takuya Mieno, Shunsuke Inenaga, and Takashi Horiyama. RePair grammars are the smallest grammars for Fibonacci words. In Hideo Bannai and Jan Holub, editors, *33rd Annual Symposium on Combinatorial Pattern Matching, CPM 2022, June 27-29, 2022, Prague, Czech Republic*, volume 223 of *LIPIcs*, pages 26:1–26:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPICS.CPM.2022.26.

21 Gonzalo Navarro. Indexing highly repetitive string collections, part II: compressed indexes. *ACM Comput. Surv.*, 54(2):26:1–26:32, 2022. doi:10.1145/3432999.

22 Gonzalo Navarro, Carlos Ochoa, and Nicola Prezza. On the approximation ratio of ordered parsings. *IEEE Trans. Inf. Theory*, 67(2):1008–1026, 2021. doi:10.1109/TIT.2020.3042746.

23 James A. Storer and Thomas G. Szymanski. Data compression via textual substitution. *J. ACM*, 29(4):928–951, 1982. doi:10.1145/322344.322346.

24 Edouard Zeckendorf. Representations des nombres naturels par une somme de nombres de Fibonacci on de nombres de Lucas. *Bulletin de La Society Royale des Sciences de Liege*, pages 179–182, 1972. URL: https://cir.nii.ac.jp/crid/1570009749187075840.

25 Jacob Ziv and Abraham Lempel. A universal algorithm for sequential data compression. *IEEE Trans. Inf. Theory*, 23(3):337–343, 1977. doi:10.1109/TIT.1977.1055714.