Theoretical Computer Science 571 (2015) 50–57

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Searching for Zimin patterns

Wojciech Rytter [a,1], Arseny M. Shur [b,*,2]

[a] Institute of Informatics, Warsaw University, Poland
[b] Institute of Mathematics and Computer Science, Ural Federal University, Ekaterinburg, Russia

ARTICLE INFO

Article history:
Received 22 May 2014
Received in revised form 12 November 2014
Accepted 10 January 2015
Available online 16 January 2015
Communicated by R. Giancarlo

Keywords:
Zimin word
Unavoidable pattern
On-line algorithm
Fibonacci word

ABSTRACT

In the area of pattern avoidability the central role is played by special words called Zimin patterns. The symbols of these patterns are treated as variables and the rank of the pattern is its number of variables. Zimin type of a word x is introduced here as the maximum rank of a Zimin pattern matching x. We show how to compute Zimin type of a word on-line in linear time. Consequently we get a quadratic time, linear-space algorithm for searching Zimin patterns in words. Then we demonstrate how the Zimin type of the length n prefix of the infinite Fibonacci word is related to the representation of n in the Fibonacci numeration system. Using this relation, we prove that Zimin types of such prefixes and Zimin patterns inside them can be found in logarithmic time. Finally, we give some upper bounds on the function $f(n,k)$ such that every k-ary word of length at least $f(n,k)$ has a factor that matches the rank n Zimin pattern.

1. Introduction

Pattern avoidability is a well-established area studying the problems involving words of two kinds: "usual" words over the alphabet of constants and *patterns* over the alphabet of variables.[3] A pattern X *embeds* in a word w if w has a factor of the form $h(X)$ where h is a non-erasing morphism. An *unavoidable* pattern is a pattern that embeds in any long enough word over any finite alphabet. In the problem of pattern (un)avoidability the crucial role is played by Zimin words [10]. The Zimin word (or *Zimin pattern*) of rank k is defined as follows:

$$\forall k > 1 \quad Z_k = Z_{k-1} \cdot x_k \cdot Z_{k-1}, \quad \text{and} \quad Z_1 = x_1.$$

Hence

$$Z_2 = x_1x_2x_1, \qquad Z_3 = x_1x_2x_1\, x_3\, x_1x_2x_1, \qquad Z_4 = x_1x_2x_1x_3x_1x_2x_1\, x_4\, x_1x_2x_1x_3x_1x_2x_1.$$

The seminal result in the area is the unavoidability theorem by Bean, Ehrenfeucht, McNulty, and Zimin ([2,10]; see [9] for an optimized proof). The theorem contains two conditions equivalent to unavoidability of a pattern X with k variables. The first condition is the existence of a successful computation in some nondeterministic reduction procedure on X, and the second, more elegant, condition says that X embeds in the word Z_k. On the other hand, it is still a big open problem whether

* Corresponding author.
E-mail addresses: rytter@mimuw.edu.pl (W. Rytter), arseny.shur@urfu.ru (A.M. Shur).

[1] Supported by the grant NCN2014/13/B/ST6/00770 of the Polish Research Center.
[2] Supported under the Agreement 02.A03.21.0006 of 27.08.2013 between the Ministry of Education and Science of the Russian Federation and Ural Federal University, and by the grant 13-01-00852 of the Russian Foundation of Basic Research.

[3] In a more general setting, which is not discussed in this paper, patterns may contain constants along with variables.

http://dx.doi.org/10.1016/j.tcs.2015.01.004
0304-3975/

unavoidability of a pattern can be checked in the time polynomial in its length [4, Problem 17]. This problem belongs to NP and is tractable for a fixed k. The general case is strongly suspected to be NP-complete, though no proof has been given.

Another natural computationally hard problem concerning avoidability is the embedding problem: given a word and a pattern, decide whether the pattern embeds in the word. This problem is NP-complete; Angluin [1] proved this fact for patterns with constants, but his proof can be adjusted for the patterns without constants as well. Note that the unavoidability problem does not refer to a (potentially long) Zimin word as a part of the input, and hence is not a particular case of the embedding problem. On the other hand, the inverse problem of embedding a Zimin pattern in a given word is a particular case of the embedding problem. Here we show that this particular case is quite simple.

In the first part of the paper (Section 2) we address the following decision problem:

Searching Zimin patterns
Input: a word w and an integer k;
Output: yes if Z_k embeds in w.

We give an algorithm solving this problem in quadratic time and linear space. The main step of the algorithm is an online linear-time computation of the characteristic we call *Zimin type* of a word. Zimin type of a finite word w is the maximum number k such that w is an image of the Zimin word Z_k under a non-erasing morphism. By definition, the empty word has Zimin type 0. In Lothaire's book, Zimin types of words are referred to as the orders of sesquipowers [6, Ch. 4].

Example 1.1. Zimin type of $u = adbadccccadbad$ is 3, because u is the image of Z_3 under the morphism

$$x_1 \to ad, \qquad x_2 \to b, \qquad x_3 \to cccc.$$

The Zimin decomposition of u is: $u = ad\,b\,ad\,cccc\,ad\,b\,ad$.

The answer of **Searching Zimin patterns** for $k = 3$ and the word $w = ccccadbadccccadbadccccc$ is *yes* (w has the word u of Zimin type 3 as a factor), but for $k = 3$ and $w = aaabbaabbaa$ the answer is *no*.

In the second part of the paper (Section 3) we study Zimin types and the embeddings of Zimin patterns for Fibonacci words. First we relate the type of the length n prefix of the infinite Fibonacci word to the representation of n in the Fibonacci numeration system (Theorem 3.2). This result and the fact that for Fibonacci words Zimin types of prefixes dominate Zimin types of other factors (Theorem 3.5) allow us to solve **Searching Zimin patterns** for this particular case in logarithmic time (Theorem 3.8).

In the last part of the paper (Section 4) we consider a couple of combinatorial problems. In Section 4.1 we analyze the fastest possible growth of the sequence of Zimin types for the prefixes of an infinite word. Finally, in Section 4.2 we give some results on the length such that the given Zimin pattern embeds in any word of this length over a given alphabet.

2. Algorithmic problems

2.1. Recurrence for Zimin types of prefixes of a word

Recall that a factor (prefix, suffix) of a word is called *proper* if it does not coincide with the word itself. A *border* of a word w is any word that is both a proper prefix and a proper suffix of w. Thus, any border of a border of w is a border of w as well. We call a border *short* if its length is $< \frac{|w|}{2}$. The notation $Bord(w)$ and $ShortBord(w)$ stand for the longest border of w and the longest short border of w, respectively. Clearly, any of these borders can coincide with the empty word.

Example 2.1. For $w = aabaabcaab\,aabaabcaab\,aab$ we have:

$$Bord(w) = aabaabcaab\,aab, \qquad ShortBord(w) = aabaab.$$

Observe that in this particular example $ShortBord(w)$ is the second longest border of w, but for any $k \geq 1$ there are examples where $ShortBord(w)$ is the kth longest border of w (the word a^{2k-1} is such an example).

For a given word x denote by $\mathsf{Ztype}[i]$ the Zimin type of $x[1..i]$.

Lemma 2.2. *Zimin type of a non-empty word can be computed iteratively through the equation*

$$\mathsf{Ztype}[i] = 1 + \mathsf{Ztype}[j], \quad \text{where } j = \left|ShortBord\big(x[1..i]\big)\right| \tag{2.1}$$

Proof. Since $u = ShortBord(w)$ implies $w = uvu$ for some non-empty word v, the left-hand part of (2.1) majorizes the right-hand part. At the same time, $\mathsf{Ztype}[i] = \mathsf{Ztype}[j] + 1$, where j is the length of *some* short border of $x[1..i]$. Hence, it suffices to show that increasing the length of the border within the interval $(0; i/2)$ cannot decrease its Zimin type.

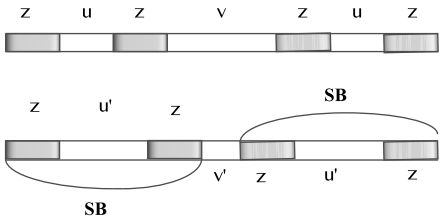


Fig. 1. Zimin decompositions using two different borders.

If $\mathsf{Ztype}[i] \le 2$, there is nothing to prove because all non-empty short borders of $x[1..i]$ have Zimin type 1. Thus we assume $\mathsf{Ztype}[i] \ge 3$. Then $x[1..i] = zuzvzuz$, where u, v are non-empty and

$$\mathsf{Ztype}\big[|z|\big] = \mathsf{Ztype}[i] - 2, \qquad \mathsf{Ztype}\big[|zuz|\big] = \mathsf{Ztype}[i] - 1.$$

Suppose that $x[1..i]$ has another bound which is longer than zuz but of length $< i/2$. The situation is depicted in Fig. 1.

Since zuz is both a prefix and a suffix of this new border, the new border begins and ends with z. Hence it has the form $zu'z$ for some non-empty u' and its Zimin type is at least $\mathsf{Ztype}[i] - 1$. The lemma is proved. □

We also mention the following property of the function *ShortBord*.

Lemma 2.3. *Let w be a non-empty word such that $ShortBord(w) = Bord(w)$. Then $ShortBord(ww) = ShortBord(w)$.*

Proof. Any border of ww of length strictly less than $|ww|/2 = |w|$ is a border of w and thus is no longer than $Bord(w)$. The result now follows from the definition of *ShortBord*. □

2.2. Algorithm

We show how to compute Zimin type of a given word x on-line in linear time. If necessary, for any $i \le |x|$ and any $k \le \mathsf{Ztype}[i]$ a morphism h such that $h(Z_k) = x[1..i]$ can be explicitly reconstructed in an obvious way from the table Ztype using Lemma 2.2. This reconstruction also takes linear time, but is not on-line.

Theorem 2.4. *Zimin type can be computed on-line in linear time using no arithmetic operations other than the shift by one bit and the increment.*

Proof. For a given word x, let $\mathsf{B}[i] = |Bord(x[1..i])|$, $\mathsf{SB}[i] = |ShortBord(x[1..i])|$. It is known since Morris and Pratt [8] that the array B can be computed on-line in linear time.

The following modification of the Morris–Pratt function computes the array Ztype on-line in linear time for the word $x = x[1..m]$.

Algorithm Compute-Zimin-Type-Sequence.

$t := s := \mathsf{B}[1] := 0$; $\mathsf{B}[0] := -1$;
$\mathsf{Ztype}[0] := 0$; $\mathsf{Ztype}[1] := 1$;
`for` $i = 2$ to m `do begin`

Compute $\mathsf{B}[i]$**:**

`while` $t \ge 0$ `and` $x[t+1] \ne x[i]$ `do`
$t := \mathsf{B}[t]$;
$t := t + 1$; $\mathsf{B}[i] := t$;

Compute s $(= \mathsf{SB}[i])$**:**

`while` $s \ge 0$ `and` ($2s + 1 \ge i$ `or` $x[s+1] \ne x[i]$) `do`
$s := \mathsf{B}[s]$;
$s := s + 1$;

Compute $\mathsf{Ztype}[i]$**:**

$\mathsf{Ztype}[i] := \mathsf{Ztype}[s] + 1$;

Complexity analysis The first part (computing $\mathsf{B}[i]$) is a classical computation of the border array. The complexity of the next part (computing $s = \mathsf{SB}[i]$) takes in total linear time, since the number of executed assignments "$s := \mathsf{B}[s]$" (decrements

of s) is bounded by the number of assignments "$s := s + 1$", which is linear. Therefore the algorithm works on-line in linear time.

Correctness The only thing to be proved is that s indeed equals $\mathsf{SB}[i]$. Let us prove this by induction; the base case is trivial. For the inductive step note that if $x[1..i]$ has a (short) border of length $k > 0$ then the word $x[1..i-1]$ has a (short) border of length $k-1$. By the inductive hypothesis, we have $s = \mathsf{SB}[i-1]$ at the beginning of the ith iteration. During this iteration, all short borders of $x[1..i-1]$ are examined in the order of decreasing length until a border extending to a short border of $x[1..i]$ is found. The border found is $\mathit{ShortBord}(x[1..i])$ by definition, whence the result. □

Remark 2.5. In general, the number $\mathsf{SB}[i]$ cannot be computed from the previous values of SB instead of B. Indeed, let $x[1..i] = www = \mathit{ababa\ ababa}$. Then $\mathsf{SB}[i-1] = i/2 - 1$. This border extends to $x[1..i]$ then it is no longer short. Hence, $\mathit{ShortBord}(x[1..i]) = \mathit{Bord}(w)$ (compare to Lemma 2.3). So, $\mathsf{B}[i/2] = 3$ (not $\mathsf{SB}[i/2] = 1$) should be precomputed.

Theorem 2.6. *An embedding of a Zimin pattern of a given type in a word can be found in quadratic time and linear space.*

Proof. One can apply the previous algorithm to each suffix of the word. □

3. Zimin types of Fibonacci factors

In stringology, Fibonacci words are frequently used to demonstrate certain algorithms and constructions. In this section we show that Fibonacci words possess quite interesting properties related to Zimin patterns. First, Zimin types of prefixes of the infinite Fibonacci word and the related function SB are very closely related to the Fibonacci numeration system. Second, the problem **Searching Zimin patterns** for these prefixes has extremely low time complexity.

We recall that Fibonacci words can be defined by the recurrence relation $F_n = F_{n-1}F_{n-2}$ with the base values $F_0 = a$, $F_{-1} = b$. These words correspond to Fibonacci numbers: $\Phi_n = |F_n|$. We write F_∞ for the infinite Fibonacci word, which is a unique word having all F_n as prefixes, and let $\mathsf{ZFib}[n]$ denote the Zimin type of $F_\infty[1..n]$. The notation $\mathsf{B}[n]$ and $\mathsf{SB}[n]$ in this section refers to the border array and the short-border array of F_∞, respectively.

Recall that any positive integer n can be uniquely written as the sum of one or more non-consecutive Fibonacci numbers:

$$n = \Phi_{n_1} + \Phi_{n_2} + \cdots + \Phi_{n_k}, \tag{3.1}$$

where $k \geq 1$, $n_k \geq 0$, and $n_i > n_{i+1} + 1$ for each $i = 1, \ldots, k-1$. This simple but nice observation is usually referred to as Zeckendorf's theorem. The sum (3.1) can be converted into a binary positional notation, called the *Fibonacci representation* of n, for example:

$$28 = 21 + 5 + 2 = 1 \cdot \Phi_6 + 0 \cdot \Phi_5 + 0 \cdot \Phi_4 + 1 \cdot \Phi_3 + 0 \cdot \Phi_2 + 1 \cdot \Phi_1 + 0 \cdot \Phi_0 = (1001010)_{\mathit{Fib}}.$$

This way of writing numbers is usually called *Fibonacci numeration system*. The array SB admits an easy description in terms of Fibonacci representation, as the following lemma shows. Let λ denote the empty word, $\Sigma = \{0, 1\}$.

Lemma 3.1. *Let $n \geq 3$ be an integer, $n = (w)_{\mathit{Fib}}$. Then either $w = 101\alpha$ for some $\alpha \in \Sigma^*$, or $w = 1001\alpha$ for some $\alpha \in \Sigma^*$, or $w = 100\alpha$ for some $\alpha \in \{\lambda\} \cup 0\Sigma^*$. In each case, $\mathsf{SB}[n] = (1\alpha)_{\mathit{Fib}}$.*

Proof. Let us compute the formula for $\mathsf{SB}[n]$. A folklore result says that the sequence of minimal periods of prefixes of F_∞ looks like

$$1\ 2^2\ 3^3\ 5^5\ 8^8\ 13^{13}\ 21^{21}\ \ldots,$$

i.e., each Φ_k appears in it exactly Φ_k times. Thus, Φ_{k-1} is the minimal period of $F_\infty[1..n]$ for $n = \Phi_k - 1, \Phi_k, \ldots, \Phi_{k+1} - 2$. The knowledge of the period immediately gives us $\mathsf{B}[n] = n - \Phi_{k-1}$. If $n < 2\Phi_{k-1}$, one has $\mathsf{SB}[n] = \mathsf{B}[n]$. It remains to find $\mathsf{SB}[n]$ for $n = 2\Phi_{k-1}, \ldots, \Phi_{k+1} - 2$. Since

$$F_\infty[1..2\Phi_{k-1}] = F_{k-1}F_{k-1} = F_{k-3}F_{k-4}F_{k-3}F_{k-3}F_{k-4}F_{k-3},$$

by Lemma 2.3 we have $\mathsf{SB}[2\Phi_{k-1}] = |F_{k-3}| = \Phi_{k-3}$. This length of border corresponds to the period Φ_k. All prefixes with the length in the considered range have the period Φ_k as well. Hence, $\mathsf{SB}[n+i] = \mathsf{SB}[n] + i$ if both n and $n+i$ belong to the range. As a result, for any $k \geq 2$ we can restore the whole picture of periods and borders for the prefixes of length between Φ_k and $\Phi_{k+1} - 1$, see Table 1.

Splitting the rows of Table 1 into three ranges, we write the following recursive formula for $\mathsf{SB}[n]$ where $n \in \{\Phi_k, \ldots, \Phi_{k+1} - 1\}$:

$$\mathsf{SB}[n] = \begin{cases} \Phi_{k-2} + j & \text{if } n = \Phi_k + j,\ j < \Phi_{k-3} \text{ (top range)}, \\ \Phi_{k-3} + j & \text{if } n = \Phi_k + \Phi_{k-3} + j,\ j < \Phi_{k-4} \text{ (middle range)}, \\ \Phi_{k-2} + j & \text{if } n = \Phi_k + \Phi_{k-2} + j,\ j < \Phi_{k-3} \text{ (bottom range)}. \end{cases} \tag{3.2}$$

Table 1
Periods and borders of the prefixes of the infinite Fibonacci word.

Length	Min period	B	SB
$\Phi_k + 0$	Φ_{k-1}	Φ_{k-2}	Φ_{i-2}
$\Phi_k + 1$	Φ_{k-1}	$\Phi_{k-2} + 1$	$\Phi_{k-2} + 1$
$\cdots$	$\cdots$	$\cdots$	$\cdots$
$\Phi_k + \Phi_{k-3} - 1$	Φ_{k-1}	$\Phi_{k-2} + \Phi_{k-3} - 1$	$\Phi_{k-2} + \Phi_{k-3} - 1$
$\Phi_k + \Phi_{k-3}$	Φ_{k-1}	Φ_{k-1}	Φ_{k-3}
$\cdots$	$\cdots$	$\cdots$	$\cdots$
$\Phi_k + \Phi_{k-2} - 1$	Φ_{k-1}	$2\Phi_{k-2} - 1$	$\Phi_{k-2} - 1$
$\Phi_k + \Phi_{k-2}$	Φ_{k-1}	$2\Phi_{k-2}$	Φ_{k-2}
$\cdots$	$\cdots$	$\cdots$	$\cdots$
$\Phi_k + \Phi_{k-1} - 2$	Φ_{k-1}	$\Phi_{k-1} + \Phi_{k-2} - 2$	$\Phi_{k-1} - 2$
$\Phi_k + \Phi_{k-1} - 1$	Φ_k	$\Phi_{k-1} - 1$	$\Phi_{k-1} - 1$

Now compare the Fibonacci representations of n and $\mathsf{SB}[n]$ in all three cases. For the top range, the Fibonacci representation of n starts with 1 corresponding to Φ_k, while the next three digits (if all exist) are zeroes. Hence, $n = (100\alpha)_{Fib}$ for some word $\alpha \in \{\lambda\} \cup 0\Sigma^*$, and we see that $\mathsf{SB}[n] = n - \Phi_k + \Phi_{k-2} = (1\alpha)_{Fib}$. In a similar way, for the middle range one has $n = (1001\alpha)_{Fib}$ and $\mathsf{SB}[n] = (1\alpha)_{Fib}$; for the bottom range, $n = (101\alpha)_{Fib}$ and $\mathsf{SB}[n] = (1\alpha)_{Fib}$. The lemma is proved. □

Lemma 3.1 implicitly mentions the following parameter of Fibonacci representation. For $n = (w)_{Fib}$ define $\psi(n)$ to be the positive integer k such that

$$w = 1x_1 \cdots x_{k-1}z, \quad \text{where } x_1, \ldots, x_{k-1} \in \{00, 001, 01\},\ z \in \{\lambda, 0\}. \tag{3.3}$$

For example, $\psi(28) = 3$ since $1001010 = 1 \cdot 001 \cdot 01 \cdot 0 \in 1\{00, 001, 01\}^2 0$. The function $\psi(n)$ is well defined. Indeed, assuming a standard notation for regular expressions, we have $w \in 1(00^*1)^*0^* = 1(00 + 001 + 01)^*(\lambda + 0)$. Hence, the decomposition (3.3) exists, and its uniqueness follows from the fact that $\{00, 001, 01\}$ is a code.

Theorem 3.2. $\mathsf{ZFib}[n] = \psi(n)$.

Proof. Let $n = (w)_{Fib}$, $k = \psi(n)$, and compute the corresponding representation $w = 1x_1 \cdots x_{k-1}z$. Then by Lemma 3.1 we have $\mathsf{SB}[n] = (1x_2 \cdots x_{k-1}z)$, $\mathsf{SB}[\mathsf{SB}[n]] = (1x_3 \cdots x_{k-1}z)$, and so on. After $k-1$ such steps we arrive at the number $(1z)_{Fib}$ which equals 1 or 2. Since $\mathsf{ZFib}[1] = \mathsf{ZFib}[2] = 1$, we obtain $\mathsf{ZFib}[n] = k$ by Lemma 2.2. □

Corollary 3.3. *Let $a_n = \mathsf{ZFib}[n]/\log_\phi n$, where ϕ is the* golden ratio. *Then the sequence $\{a_n\}_1^\infty$ has no limit, $\limsup_{n\to\infty} a_n = 1/2$, $\liminf_{n\to\infty} a_n = 1/3$, and any number between $1/3$ and $1/2$ is a limit point of a_n.*

Proof. One can take $n_i = \Phi_i = (10^i)_{Fib}$ for lim sup and $n_i = (1(001)^i)_{Fib}$ for lim inf. Any intermediate limit point α can be obtained by taking a sequence of numbers n_i having Fibonacci representations of the form $1x_1 \cdots x_i z$ with the fraction of the factors 001 approaching $3 - 6\alpha$. □

Corollary 3.4. *Suppose that n is an arbitrary number such that copying, addition, subtraction, and comparison of numbers up to n can be performed in constant time. Then*

(1) *the value* $\mathsf{ZFib}[n]$ *can be computed in* $O(\log n)$ *time and space;*
(2) *the array with n elements* $\mathsf{ZFib}[1], \ldots, \mathsf{ZFib}[n]$ *can be computed in sublinear time, namely, in time* $O(n \log\log n/\log n)$.

Proof. For statement 1, one can compute the Fibonacci representation $(w)_{Fib}$ of n as follows. First, Fibonacci numbers are calculated in ascending order until the last number Φ_k exceeds n; second, the length of $(w)_{Fib}$ is set to k, the leading digit is set to 1, and Φ_{k-1} is subtracted from n to get the remainder n'; third, Fibonacci numbers are calculated in descending order and $(w)_{Fib}$ is filled with zeroes until the number $\Phi_{k'} < n'$ is found; then 1 is appended to w, n' is set to $n' - \Phi_{k'}$, and the previous step is repeated until $n' > 0$; finally, the rest of w is filled with zeroes. At any moment, only two Fibonacci numbers are stored (those last computed). Clearly, $|w| = O(\log n)$ and the whole computation takes $O(|w|)$ time and space. After this, $\psi(n)$ is calculated from w; since the code $\{00, 001, 01\}$ has bounded delay, this can be done in $O(|w|)$ time also.

Now let us prove statement 2. Combining (2.1) with (3.2), we see that the only operations used in the construction of the array $\mathsf{ZFib}[1..n]$ are "copy a block" and "increment all elements of a block". Each element of the array $\mathsf{ZFib}[1..n]$ is $O(\log n)$ and hence can be written in $O(\log\log n)$ bits. By the conditions of the corollary, one can operate in one step with numbers written in $\log n$ bits. According to this, one can pack up to $\log n/\log\log n$ array values, belonging to one block, into one cell. Copying [resp., adding specific constant to] a cell, one copies [resp., increments] all array values inside it. In this way, the total number of operations will be linear in the number of cells, which is $O(n \log\log n/\log n)$. □

Next we analyze Zimin types of arbitrary factors of the infinite Fibonacci word. In what follows, we refer to such factors as *Fibonacci factors*. The following theorem shows that the type of any Fibonacci factor is majorized by the type of a relatively short Fibonacci word.

Theorem 3.5. *Suppose that w is a Fibonacci factor, n is the last position of the leftmost occurrence of w in F_∞, and k is such that $\Phi_{2(k-1)} \le n < \Phi_{2k}$. Then* $\mathsf{Ztype}(w) \le k = \mathsf{Ztype}(F_{2(k-1)})$.

The proof is based on two lemmas. Lemma 3.6(1) is Corollary 4 of [5]. Lemma 3.6(2) is immediate from the (unnumbered) Lemma and the proof of Proposition 6 of [7].

Lemma 3.6. *(1) The minimal period of any Fibonacci factor is a Fibonacci number. (2) The length of a Fibonacci factor of period Φ_k is at most $\Phi_{k+1} + 2\Phi_k - 2$.*

Lemma 3.7. *Any Fibonacci factor w such that $|w| < \Phi_k$ satisfies the inequality $|ShortBord(w)| < \Phi_{k-2}$.*

Proof. Let p be the minimal period of w and let $x = ShortBord(w)$. Since $|w| - |x|$ is a period of w, one has $|w| - p \ge |x|$. By the definition of *ShortBord*,

$$|x| = |w| - p \quad \text{if } p > |w|/2 \quad \text{and} \quad |x| < |w| - p \quad \text{otherwise.} \tag{3.4}$$

Consider the case $p > |w|/2$. By Lemma 3.6(1), p is a Fibonacci number. If $p = \Phi_{k-1}$ then by (3.4)

$$|x| = |w| - \Phi_{k-1} < \Phi_k - \Phi_{k-1} = \Phi_{k-2},$$

as required. If $p \le \Phi_{k-2}$, then $|x| < \Phi_{k-2}$ since $|x| < |w|/2 < p$.

Now let $p \le |w|/2$. Then clearly $p < \Phi_{k-1}$. Let u be the prefix of w of length p. By minimality of p, u is *primitive* (not a power of a shorter word). A basic characterization of primitive words is that the word uu contains no "internal" occurrences of u. Let $p = \Phi_{k-2}$. If x has u as a prefix, then w should have the prefix uux due to the above mentioned property of uu. But $|uux| \ge |uuu| = 3\Phi_{k-2} \ge \Phi_k$, a contradiction. Hence, x is a proper prefix of u, i.e., $|x| < p$. Finally, let $p = \Phi_{k-l}$ for some $l \ge 3$. Once again, if u is a prefix of x, then uux is a prefix of w. Thus, $|w| \ge 2\Phi_{k-l} + |x|$. On the other hand, Lemma 3.6(2) says that $|w| \le \Phi_{k-l+1} + 2\Phi_{k-l} - 2$. Comparing the two inequalities, we get $|x| < \Phi_{k-l+1} \le \Phi_{k-2}$, as required. □

Proof of Theorem 3.5. By the conditions of the theorem, $|w| \le n < \Phi_{2k}$. Define a finite sequence of words by putting $w_0 = w$ and $w_{t+1} = ShortBord(w_t)$ for all $t \ge 0$ such that $w_t \ne \lambda$. Assume that $\bar{t}$ is such that $w_{\bar{t}} = \lambda$. Then $\mathsf{Ztype}(w) = \bar{t}$ by (2.1). On the other hand, $\bar{t} \le k$. Indeed, if w_k exists, then a k-fold application of Lemma 3.7 implies that w_k is shorter then $F_0 = a$, i.e., $w_k = \lambda$. Thus, $\mathsf{Ztype}(w) \le k$. It remains to note that $\Phi_{2(k-1)} = (1 \cdot (00)^{k-1})_{Fib}$, and hence $\mathsf{Ztype}(F_{2(k-1)}) = k$ by Theorem 3.2. □

In general, it is easier to find Zimin type of a word w than to give an embedding of the Zimin pattern of a maximal possible rank into w, see Section 2. But for the prefixes of the Fibonacci words both problems have the same complexity, and the algorithm for the latter problem is even simpler than the algorithm for the former one. Indeed, Theorem 3.5 implies that the maximum of Zimin types of the factors for any word $F_\infty[1..n]$ is achieved on its prefix $F_{2(k-1)}$, where $\Phi_{2(k-1)} \le n < \Phi_{2k}$, and is equal to k. The embedding of Z_k into $F_{2(k-1)}$ can be immediately obtained from the observation that $ShortBord(F_j) = F_{j-2} : x_k \to F_{2k-5}, x_{k-1} \to F_{2k-7}, \ldots, x_2 \to F_{-1}(= b), x_1 \to a$. Since k can be computed from n in logarithmic time (cf. Corollary 3.4), we get the following

Theorem 3.8. *Suppose that n is an arbitrary number such that addition and comparison of numbers up to n can be performed in constant time. Then the maximal rank of a Zimin pattern embeddable in $F_\infty[1..n]$ and a morphism for such an embedding can be found in $O(\log n)$ time.*

4. Some combinatorial issues

4.1. More on Zimin type sequences

By Zimin type sequence of an infinite word w we mean the sequence $\{\mathsf{Ztype}[i]\}_{i=1}^{\infty}$ of Zimin types of its prefixes.

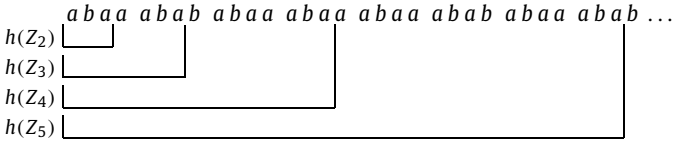
Remark 4.1. Zimin type sequence of a word is unbounded if and only if this word is recurrent (i.e., any its factor occurs in it infinitely often). This is Proposition 4.3.1 of [6].

Table 2
Lengths guaranteeing the embedding of Z_n into every word. A cell with the coordinates (n,k) corresponds to $f(n,k)$.

$n \setminus k$	2	3	4	5	…	r	…
1	1	1	1	1	…	1	…
2	5	7	9	11	…	$2r+1$	…
3	29	≤ 319	≤ 3169	≤ 37991	…	$\le \sqrt{e}\cdot 2^r(r+1)!+2r+1$	…
4	≤ 236489						

Fibonacci words provide extremal examples for many problems, but Zimin type sequences can grow faster than the sequence for the infinite Fibonacci word. Indeed, an example of the fastest asymptotic growth is given by any word $g[1..\infty]$ such that for any n the word $g[1..2^n-1]$ is an image of Z_n. We call such infinite words *Zimin encodings* because they are images of the infinite Zimin word under a letter-to-letter morphism (a *coding*). The following example shows that the class of Zimin encodings is far from being trivial and contains, e.g., aperiodic binary words generated by morphisms.

Example 4.2. The word generated by the binary morphism $h: a \to abaa,\ b \to abab$ is a Zimin encoding:



Since $|Z_n| = 2^n - 1$, any Zimin type sequence is less or equal (in the coordinate-wise order) than the sequence

$$1^2\, 2^4\, 3^8 \cdots n^{2^n} \cdots \tag{4.1}$$

Lemma 4.3. *Any infinite word reaching the maximum (4.1) is unary.*

Proof. The Zimin type of the word of length 2^n can be equal to n only if the image of x_n has length 2 while all other images of letters have length 1. Thus, a word w having the Zimin type sequence (4.1) satisfies the equalities $w[1..2^{n-1}-1] = w[2^{n-1}+1..2^n-1]$ and $w[1..2^{n-1}-1] = w[2^{n-1}+2..2^n]$ for any $n > 1$. Hence, all letters in the word $w[2^{n-1}+1..2^n]$ are equal (to $a = w[1]$). Using this observation for all n, we see that all letters in w are equal to a. □

4.2. Length bounds on unavoidability of Zimin patterns

Let $f(n,k)$ be the minimum number such that the pattern Z_n can be embedded in every word of length at least $f(n,k)$ over a size k alphabet. Here we prove initial facts about this astronomically growing function.

Theorem 4.4. *The function $f(n,k)$ satisfies the restrictions given in Table 2.*

Proof. The equality $f(1,k) = 1$ is trivial. To justify other figures in Table 2, we need some lemmas.

Lemma 4.5. *The pattern Z_2 embeds in a word w if and only if some letter occurs in w in two non-consecutive positions.*

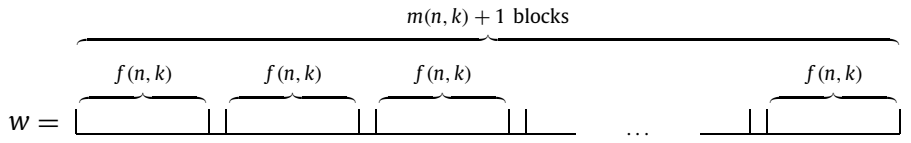
Proof. If w has a factor $h(x_1x_2x_1)$, then any letter occurring in $h(x_1)$ satisfies the required condition. Conversely, if w has a factor ava for a letter a and a non-empty word v, then this factor is an image of Z_2. □

Lemma 4.5 shows that $a_1^2 \cdots a_k^2$, where $a_1, \ldots, a_k$ are distinct letters, is the longest k-ary word containing no images of Z_2. This fact explains the second row of Table 2 (this particular result is not new, see [6, Prop. 3.1.2]). In order to explain the values in the last two rows, we introduce a new notion. We say that a word of Zimin type n is *minimal* if any of its proper factors has Zimin type $< n$. Obviously, if Z_n embeds in w, then w contains a minimal word of Zimin type n as a factor.

Lemma 4.6. *Let $m(n,k)$ be the number of k-ary minimal words of Zimin type n. Then the following inequality holds for any $n, k \ge 2$:*

$$f(n+1,k) \le \big(f(n,k)+1\big)\cdot m(n,k) + f(n,k). \tag{4.2}$$

Proof. Consider a word w of length $(f(n,k)+1)\cdot m(n,k)+f(n,k)$, partitioned as follows:



Each block of length $f(n,k)$ contains a minimal word of type n. By the pigeonhole principle, some minimal word z occurs twice. Then the factor of w containing z as a short border has Zimin type $\geq n+1$, whence the result. □

Lemma 4.7. *The number of k-ary minimal words of Zimin type 2 is*

$$m(2,k)=k!\cdot\sum_{i=0}^{k-1}\frac{2^{k-1-i}}{i!}\,. \tag{4.3}$$

Proof. By Lemma 4.5 and the definition of minimality, k-ary minimal word w of Zimin type 2 has a unique pair of equal letters in non-consecutive positions: the first and the last letter. Thus, either $w=aaa$ for a letter a, or $w=ab_1^{j_1}\cdots b_r^{j_r}a$, where $r<k$, all letters b_i are distinct from each other and from a, and $j_i\in\{1,2\}$ for any i. Counting such words is a mere combinatorial exercise. For example, if $r=k-1$, there are $k!$ ways to choose the letters $a,b_1,\ldots,b_r$ and 2^{k-1} ways to choose the numbers $j_1,\ldots,j_r$; this gives us the first summand in (4.3). The other cases are similar, so we omit the rest of the computation. □

Lemmas 4.6 and 4.7 give the bounds for the values $f(3,k)$. The general bound $f(3,r)\leq\sqrt{e}\cdot 2^r(r+1)!+2r+1$ is obtained by computing an infinite sum instead of the finite one in (4.3). As for the binary alphabet, $m(2,2)=6$, implying $f(3,2)\leq 41$ by (4.2). This bound means that a direct computer search to compute the exact values of $f(3,2)$ and $m(3,2)$ is feasible. Implementing this search, we learned that $f(3,2)=29$ and $m(3,2)=7882$. Then (4.2) gives us an upper bound for $f(4,2)$. The theorem is proved. □

5. Some open questions

We hope that each section of this paper will give rise to some further research. The main question concerning the results of Section 2 is whether the quadratic bound of Theorem 2.6 is optimal. It is likely that this bound can be improved, because linear-time algorithms are known for the search of many types of repetitions (squares, overlaps, runs, palindromes, etc.).

For the results of Section 3, it is quite interesting whether other characteristic Sturmian words possess properties similar to Corollary 3.4 and Theorem 3.8. As for Section 4, it would be nice to strengthen and generalize upper bounds given in Theorem 4.4. When this paper was under review, a preprint [3] appeared with a double-exponential *lower* bound $f(n,k)>k^{2^{n-1}(1+o(1))}$ and a particular bound $f(4,2)>10483$. Since the general upper bound stemming from Lemma 4.6 is a tower of $n-1$ exponents, there is enough room for improvement.

Acknowledgements

The authors are grateful to the organizers and participants of the Dagstuhl seminar 14111 “Combinatorics and algorithmics of strings” for stimulating discussions: this paper arose from one of them. Our special thanks to the anonymous referees for a number of useful remarks and comments.

References

[1] Dana Angluin, Finding patterns common to a set of strings, J. Comput. System Sci. 21 (1) (1980) 46–62.
[2] D.A. Bean, A. Ehrenfeucht, G. McNulty, Avoidable patterns in strings of symbols, Pacific J. Math. 85 (1979) 261–294.
[3] Joshua Cooper, Danny Rorabaugh, Bounds on Zimin word avoidance, arXiv:1409.3080 [math.CO], 2014.
[4] J.D. Currie, Pattern avoidance: themes and variations, Theoret. Comput. Sci. 339 (2005) 7–18.
[5] J.D. Currie, K. Saari, Least periods of factors of infinite words, RAIRO Theor. Inform. Appl. 43 (2009) 165–178.
[6] M. Lothaire, Algebraic Combinatorics on Words, Encyclopedia of Mathematics and Its Applications, vol. 90, Cambridge University Press, 2002.
[7] F. Mignosi, G. Pirillo, Repetitions in the Fibonacci infinite word, RAIRO Theor. Inform. Appl. 26 (1992) 199–204.
[8] J.H. Morris, V.R. Pratt, A linear pattern-matching algorithm, Tech. Rep. 40, University of California, Berkeley, 1970.
[9] M.V. Sapir, Combinatorics on words with applications, LITP report, 32, 1995.
[10] A.I. Zimin, Blocking sets of terms, Mat. Sb. 119 (1982) 363–375, 447, in Russian. English translation in Math. USSR Sbornik, 47 (1984), 353–364.