Theoretical Computer Science 1041 (2025) 115216

Contents lists available at ScienceDirect

Theoretical Computer Science

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

Subsequence covers of words

Panagiotis Charalampopoulos [a], Solon P. Pissis [b,c,*], Jakub Radoszewski [d,1], Wojciech Rytter [d], Tomasz Waleń [d,1], Wiktor Zuba [b]

[a] Birkbeck, University of London, London, UK
[b] CWI, Amsterdam, the Netherlands
[c] Vrije Universiteit, Amsterdam, the Netherlands
[d] Institute of Informatics, University of Warsaw, Warsaw, Poland

ARTICLE INFO

Section Editor: Pinyan Lu

Handling Editor: Marinella Sciortino

Keywords:
Text algorithms
Combinatorics on words
Covers
Shuffle powers
Subsequence

ABSTRACT

We introduce subsequence covers (s-covers, in short), a new type of covers of a word. A word C is an *s-cover* of a word S if the occurrences of C in S as subsequences cover all the positions in S. The s-covers seem to be computationally much harder than standard covers of words (cf. Apostolico et al. (1991) [1]), but, on the other hand, much easier than the related shuffle powers (Warmuth and Haussler (1984) [6]).

We give a linear-time algorithm for testing if a candidate word C is an s-cover of a word S over a polynomially-bounded integer alphabet. We also give an algorithm for finding a shortest s-cover of a word S, which in the case of a constant-sized alphabet, also runs in linear time.

The words without proper s-cover are called s-primitive. We complement our algorithmic results with explicit lower and an upper bound on the length of a longest s-primitive word. Both bounds are exponential in the size of the alphabet. The upper bound presented here improves the bound given in the conference version of this paper [SPIRE 2022].

1. Introduction

The problem of computing covers in a word is a classic one in text algorithms; see [1–3] and also [4,5] for recent surveys. In its most basic type, a word C is said to be a *cover* of a word S if every position of S lies within some occurrence of C as a factor (subword) in S [1].

In this paper, we introduce a new type of cover, in which instead of factors we take subsequences (scattered factors). Such covers turn out to be related to shuffle problems [6–8]. Formally the new type of cover is defined as follows:

Definition 1.1. A word C is a **subsequence cover** (s-cover, in short) of a word S if every position in S belongs to an occurrence of C as a subsequence in S. We also write $S \in C^{\otimes}$, where $C^{\otimes}$ is the set of words having C as an s-cover.

We say that an s-cover C of a word S is *non-trivial* if $|C| < |S|$. A word S is called *s-primitive* if it has no non-trivial s-cover.

* Corresponding author.
E-mail addresses: p.charalampopoulos@bbk.ac.uk (P. Charalampopoulos), solon.pissis@cwi.nl (S.P. Pissis), jrad@mimuw.edu.pl (J. Radoszewski), rytter@mimuw.edu.pl (W. Rytter), walen@mimuw.edu.pl (T. Waleń), wiktor.zuba@cwi.nl (W. Zuba).

[1] Supported by the Polish National Science Centre, grant no. 2018/31/D/ST6/03991.

https://doi.org/10.1016/j.tcs.2025.115216
Received 15 April 2024; Received in revised form 30 September 2024; Accepted 30 March 2025

Available online 3 April 2025

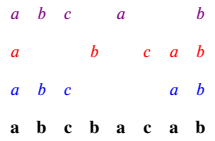

Fig. 1. An illustration of the fact that $C = abcab$ is an s-cover of $S = abcbacab$.

Remark 1.2. An example of an s-primitive word is the Zimin word Z_k [9], that is, a word over alphabet $\{1, \ldots, k\}$ given by recurrences

$$Z_1 = 1, \quad Z_i = Z_{i-1} i Z_{i-1} \text{ for } i \in \{2, \ldots, k\}. \tag{1}$$

The word Z_k has length $2^k - 1$. The fact that it is s-primitive can be shown by simple induction.

Clearly, if a word C is a (standard) cover of a word S, then C is an s-cover of S. However, the converse implication is false: ab is an s-cover of aab, but is not a standard cover. For another example of an s-cover, see below.

Example 1.3. Fig. 1 shows that $C = abcab$ is an s-cover of $S = abcbacab$. In fact C is a shortest s-cover of S.

We now provide some basic definitions and notation. An *alphabet* is a finite nonempty set of elements called *letters*. A *word* S is a sequence of letters over some alphabet. For a word S, by $|S|$ we denote its *length*, by $S[i]$, for $i = 0, \ldots, |S| - 1$, we denote its ith letter, and by $Alph(S)$ we denote the set of letters in S, i.e., $\{S[0], \ldots, S[|S| - 1]\}$. The *empty word* is the word of length 0.

For any two words U and V, by $U \cdot V = UV$ we denote their concatenation. For a word $S = PUQ$, where P, U, and Q are words, U is called a *factor* of S; it is called a *prefix* (resp. *suffix*) if P (resp. Q) is the empty word. By $S[i\,..\,j] = S[i\,..\,j+1) = S(i-1\,..\,j]$ we denote a factor $S[i] \ldots S[j]$ of S; we omit i if $i = 0$ and j if $j = |S| - 1$.

A word V is a *k-power* of a word U, for integer $k \geq 0$, if V is a concatenation of k copies of U, in which case we denote it by U^k. It is called a *square* if $k = 2$.

Remark 1.4. If a word contains a factor which is not s-primitive, then the whole word is not s-primitive. It is easy to see that any gapped repeat UVU (see [10]), where $Alph(V) \subseteq Alph(U)$, is not s-primitive and U is its non-trivial s-cover. This is the case because every position of V can be covered by an occurrence whose prefix belongs to the first copy of U and its suffix to the second copy of U with only the one position of V used. In particular nontrivial squares UU are of this type. Each word containing a factor of this type is not s-primitive.

A different version of covers, where we require that position-subsequences[2] are disjoint, is the *shuffle closure* problem. The shuffle closure of a word U, denoted by $U^{\odot}$, is the set of words resulting by interleaving many copies of U; see [6]. The words in $U^{\odot}$ are sometimes called *shuffle powers* of U.

The following problems are NP-hard for constant-sized alphabets:

(1) Given two words U and S, test if $S \in U^{\odot}$; see [6].
(2) Given a word S, check if there exists a word U such that $|U| = |S|/2$ and $S \in U^{\odot}$ (this was originally called the *shuffle square* problem); see [8]. An NP-hardness proof for a binary alphabet was recently given in [11].
(3) Given a word S, find a shortest word U such that $S \in U^{\odot}$; its hardness is trivially reduced from (2).

The following observation links s-covers and shuffle closures.

Observation 1.5. *Let S be a word of length n. Then*

$$S \in C^{\otimes} \;\Rightarrow\; \exists r_0, r_1 \ldots, r_{n-1} \in \mathbb{Z}_+ :\; S[0]^{r_0} S[1]^{r_1} \ldots S[n-1]^{r_{n-1}} \in C^{\odot}.$$

Proof. Word C is an s-cover of word S, so there exist position-subsequences $I_1, \ldots, I_k$ of length $|C|$ in S that correspond to occurrences of C and such that $I_1 \cup I_2 \cup \cdots \cup I_k = \{0, \ldots, n-1\}$. For $i \in \{0, \ldots, n-1\}$, we define r_i as the number of occurrences of i in the position-subsequences. Then, $S' := S[0]^{r_0} S[1]^{r_1} \ldots S[n-1]^{r_{n-1}}$ is a shuffle power of C.

To see that this holds, we transform the position-subsequences $I_1, \ldots, I_k$ into disjoint position-subsequences in S' by performing the following operation simultaneously for all $i \in \{0, \ldots, n-1\}$: Replace in any way the occurrences of i in $I_1, \ldots, I_k$ by distinct numbers in $\{s, \ldots, s + r_i - 1\}$, where $s = r_0 + \cdots + r_{i-1}$. By the definition of r_i, the resulting position-subsequences $I'_1, \ldots, I'_k$ in S' are disjoint, satisfy $I'_1 \cup \cdots \cup I'_k = \{0, \ldots, |S'| - 1\}$ and each of them corresponds to an occurrence of C in S'. □

[2] By a *position-subsequence* we mean the sorted set of positions that witness the occurrence of the subsequence.

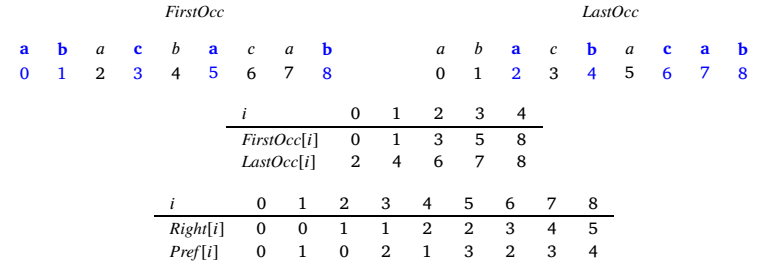

Fig. 2. *FirstOcc*, *LastOcc*, *Right*, *Pref* for $S = abacbacab$ and $C = abcab$.

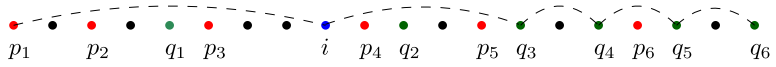

Fig. 3. Assume that for some words C and S the sequences $FirstOcc = \{p_1, p_2, p_3, p_4, p_5, p_6\}$ (red) and $LastOcc = \{q_1, q_2, q_3, q_4, q_5, q_6\}$ (green) are as in the figure. Further assume that $Pref[i] = 2$ (i.e., we have $S[i] = S[p_2] \neq S[p_3]$). As shown, we have $Right[i] = 2$. Thus, we have $Right[i] \leq Pref[i] + 1$ and consequently the position i is covered by an occurrence of C as a subsequence using positions $(p_1, i, q_3, q_4, q_5, q_6)$.

In this paper, we show that problems similar to (1) and (3) for s-covers, when we replace $\odot$ by $\otimes$, are tractable. Notably, the first one is solved in linear time for any polynomially-bounded integer alphabet; and the last one in linear time for any constant-sized alphabet.

Our results and paper organization

- In Section 2, we present a linear-time algorithm for checking if a word C is an s-cover of a word S, assuming that S is over a polynomially-bounded integer alphabet $\{0, \ldots, |S|^{\mathcal{O}(1)}\}$. We also discuss why an equally efficient algorithm for this problem without this assumption is unlikely.
- Let $\gamma(k)$ denote the length of a longest s-primitive word over an alphabet of size k. In Sections 3 to 5, we present general bounds on this function as well as its particular values for small values of k. Section 8 contains a proof of an auxiliary lemma.
- In Section 6, we show that computing a non-trivial s-cover is fixed parameter tractable for parameter $k = |Alph(S)|$. In particular, we obtain a linear-time algorithm for computing a shortest s-cover of a word over a constant-sized alphabet.
- Finally, in Section 7, we explore properties of s-covers that are significantly different from properties of standard covers. In particular, we show that a word can have exponentially many different shortest s-covers.

We conclude in Section 9 with a summary of our results and a list of open problems.

2. Testing a candidate s-cover

Let us consider words $C = C[0..m-1]$ and $S = S[0..n-1]$. We would like to check whether C is an s-cover of S.

Let sequences *FirstOcc* and *LastOcc* be the lexicographically first and last position-subsequences of S containing C. We assume that $FirstOcc[0] = 0$ and $LastOcc[m-1] = n-1$. If there are no such subsequences of positions then C is not an s-cover, so we assume they exist and are well defined.

For all $i \in \{0, \ldots, n-1\}$, we define

$$Right[i] = \min(\{j \,:\, LastOcc[j] > i\} \cup \{m\}),$$

$$Pref[i] = \max(\{j \,:\, FirstOcc[j] \leq i \ \wedge \ S[FirstOcc[j]] = S[i]\} \cup \{-1\}).$$

See also Fig. 2. Intuitively, if position i is in any subsequence occurrence of C in S, then there is a subsequence occurrence of C in S that consists of the prefix of *FirstOcc* of length $Pref[i]$ and an appropriate suffix of *LastOcc*. All we have to do is check, for all i, whether such a pair of prefix and suffix exists. See Fig. 3 for an illustration of the argument and Lemma 2.1 for a formal statement of the condition that needs to be satisfied.

The sequence *FirstOcc* can be computed with a simple left-to-right pass over S and C; the computation of *LastOcc* is symmetric. The table *Right* can be computed via a right-to-left pass. The table $Pref[i]$ is computed on-line using an additional table *PRED* indexed by the letters of the alphabet. The algorithm is formalized in the following pseudocode.

Algorithm 1: $TEST(C, S)$.

Input: word $C = C[0..m-1]$ and word $S = S[0..n-1]$
Output: *true* if and only if C is an s-cover of S

compute *FirstOcc* and *LastOcc*

▷ compute *Right*
  $k := m$
  for $i := n-1$ **down to** 0 **do**
    $Right[i] := k$
    if $k > 0$ **and** $i = LastOcc[k-1]$ **then** $k := k-1$

▷ compute *Pref*
  $PRED[c] := -1 \ \forall c \in \Sigma$
  $k := 0$
  for $i := 0$ **to** $n-1$ **do**
    if $i = FirstOcc[k]$ **then**
      $PRED[S[i]] := k$
      if $k < m-1$ **then** $k := k+1$
    $Pref[i] := PRED[S[i]]$

return $\forall_{i=0,\ldots,n-1}\,(Pref[i] \neq -1$ **and** $Right[i] \leq Pref[i]+1)$

Correctness of the algorithm follows from Lemma 2.1 (inspect also Fig. 3).

Lemma 2.1. *Let us assume that FirstOcc and LastOcc are well defined. Then C is an s-cover of S if and only if for each position $0 \leq i \leq n-1$ we have $Pref[i] \neq -1$ and $Right[i] \leq Pref[i]+1$.*

Proof. First, observe that if $Pref[i] = -1$ for any i, then C is not an s-cover of S. This follows from the greedy computation of *FirstOcc*, which implies that the prefix of C that precedes the first occurrence of $S[i]$ in C does not have a subsequence occurrence in $S[0..i-1]$; else, i would be in *FirstOcc*, a contradiction.

We henceforth assume that $Pref[i] \neq -1$ for every i and show that, in this case, C is an *s*-cover of S if and only if $Right[i] \leq Pref[i]+1$ for all $i \in \{0,\ldots,n-1\}$. Let $FirstOcc = \{p_0, p_1, \ldots, p_{m-1}\}$ and $LastOcc = \{q_0, q_1, \ldots, q_{m-1}\}$.

($\Leftarrow$). Assume that $Right[i] \leq Pref[i]+1$.

In this case, position i can be covered by a subsequence occupying positions $p_0, \ldots, p_{j-1}, i, q_{j+1}, \ldots, q_{m-1}$, for $j = Pref[i]$. As $S[p_j] = S[i]$, this subsequence is equal to C, and as $p_j \leq i$ and $q_{j+1} > i$ ($j+1 \geq Right[i]$), those positions form an increasing sequence (that is, we obtain a valid subsequence).

($\Rightarrow$). Assume that for some j there exists an increasing sequence

$$r_0, r_1, \ldots, r_{j-1}, i, r_{j+1}, \ldots, r_{m-1},$$

such that $S[r_0]S[r_1]\ldots S[r_{j-1}]S[i]S[r_{j+1}]\ldots S[r_{m-1}] = C$.

By induction for $k = 0, \ldots, j$, $r_k \geq p_k$ (including $r_j = i$) and for $k = m-1, \ldots, j+1$, $r_k \leq q_k$. But this means that $Pref[i] \geq j$ and $Right[i] \leq j+1$. Hence $Right[i] \leq Pref[i]+1$. This completes the proof. □

Note that, under the assumption of a polynomially-bounded integer alphabet, the table *PRED* can be initialized and updated deterministically in linear total time by first sorting the letters of S using Radix Sort. We thus arrive at the following result.

Theorem 2.2. *Given words C and S over an integer alphabet $\{0, \ldots, |S|^{\mathcal{O}(1)}\}$, we can check if C is an s-cover of S in $\mathcal{O}(|S|)$ time.*

In the standard setting (cf. [2]), one can check if a word C is a cover of a word S—what is more, find the shortest cover of S—in linear time for any (non-necessarily integer) alphabet. We show below that the existence of such an algorithm for testing a candidate s-cover is rather unlikely. Let us introduce a slightly more general version of the s-cover testing problem in which, if C is an s-cover of S, we are to say, for each position i in S, which position j of C is actually used to cover $S[i]$; if there is more than one such position j, any one of them can be output. Let us call this problem the *witness s-cover testing* problem. In particular, our algorithm solves the witness s-cover testing problem with the answers stored in the *Pref* array. We next give a comparison-based lower bound for the latter.

Theorem 2.3. *The witness s-cover testing problem for a word S of length n (and a word C) requires $\Omega(n \log n)$ time in the comparison model.*

Proof. Let us consider a word C of length m that is composed of m distinct letters and a family of words of the form $S = CTC$, where T is a word of length m such that $Alph(T) \subseteq Alph(C)$. Then C is an s-cover of each such word S. Each choice of the word T implies a different output to the witness s-cover testing problem on C and S. There are m^m different outputs, so a decision tree for this problem must have depth $\Omega(\log m^m) = \Omega(m \log m) = \Omega(n \log n)$. □

We can define the *coverage* of a word C in a word S as the number of positions in S that are covered by occurrences of C in S as subsequences. If C is not a subsequence of S, the coverage is equal to 0. If C is an s-cover of S, the coverage is equal to $|S|$.

Even if C turns out not to be an s-cover of S, Algorithm 1 actually computes the positions of S that can be covered using occurrences of C (they are exactly the positions i for which $Pref[i] \geq 0$ and $Right[i] \leq Pref[i] + 1$). This leads to the following corollary.

Corollary 2.4. *Given words C and S over an integer alphabet $\{0, \ldots, |S|^{\mathcal{O}(1)}\}$, we can compute the coverage of C in S in $\mathcal{O}(|S|)$ time.*

Hence, our algorithm can be useful to find partial variants of s-covers, defined analogously as for the standard covers [12–14].

3. Maximal lengths of s-primitive words over small alphabets

Let us recall that $\gamma(k)$ denotes the length of a longest s-primitive word over an alphabet of size k. It is obvious that $\gamma(1) = 1$ and $\gamma(2) = 3$ since there are no square-free words of length larger than 3 over a binary alphabet. In particular, the longest s-primitive binary words are aba and bab.

The case of ternary words is already more complicated; in this section, we study this case and the case of a quaternary alphabet. General bounds on the function $\gamma(k)$ are shown in Section 5 (weaker but simpler bounds are presented in Section 4).

Observation 3.1. *If a factor of the word S is not s-primitive, then S is also not s-primitive.*

Proof. If $S = PUQ$ and C is a non-trivial s-cover of U, then PCQ is a non-trivial s-cover of S. □

Fact 3.2. $\gamma(3) = 8$.

Proof. The word $S = abcabacb$ is of length 8 and it is s-primitive. Indeed, every s-cover of S needs to have a prefix abc, a suffix acb, and no further letters c. The only word of the form $abcXacb$, for word $|X| \leq 2$ over the alphabet $\{a, b\}$, that is an s-cover of S is S. Hence, $\gamma(3) \geq 8$.

In order to show that this bound is tight, we still have to show that each ternary word of length 9 is not s-primitive. (There are 19683 ternary words of length 9.) The number of words to consider is substantially reduced by observing that relevant words are square-free and do not contain the structure specified in the following claim.

Claim 3.3. *If a word S over a ternary alphabet contains a factor of the form $abXbc$ for some (maybe empty) word X and different letters a, b, c, then it is not s-primitive.*

Proof. The factor $abXbc$ has abc as its s-cover, and thus it is not s-primitive. Consequently, by Observation 3.1, the whole word S is not s-primitive. □

Fig. 4 shows a trie of all ternary square-free words starting with ab, truncated at words that are not s-primitive (in leaves). The words in all leaves but one contain the structure from the claim, and for the remaining word, a non-trivial s-cover can be easily given. The trie shows that words of length 9 over a ternary alphabet are not s-primitive. □

Fact 3.4. $\gamma(4) = 19$.

Proof. Using a computer-assisted verification we have checked that the word $S = abacadbabdcabcbadac$, of length 19, is s-primitive. Thus $\gamma(4) \geq 19$. We also checked experimentally that $\gamma(4) \leq 19$. Consequently, $\gamma(4) = 19$. □

An optimized C++ code used for the experiments can be found at https://github.com/craniac-swistak/s-covers. The program reads k and computes $\gamma(k)$.

Remark 3.5. There are $2 \cdot 3! = 12$ s-primitive words of length $\gamma(3) = 8$ over ternary alphabet (cf. Fig. 4, for each pair of distinct letters there are two s-primitive words starting with these letters). This accounts for less than 0.2% among all 3^8 ternary words of length 8. For a 4-letter alphabet, our program shows that the relative number of s-primitive words of length $\gamma(4) = 19$ is very small. There are exactly 2496 such words, out of 4^{19}, which gives a fraction less than 10^{-8}. This suggests that s-primitive 5-ary words of length $\gamma(5)$ are extremely sparse and finding an s-primitive word over a 5-letter alphabet of length $\gamma(5)$ could be a challenging task. (In particular, in the next subsection we show that $\gamma(5) \geq 39$.)

Table 1 summarizes this section.

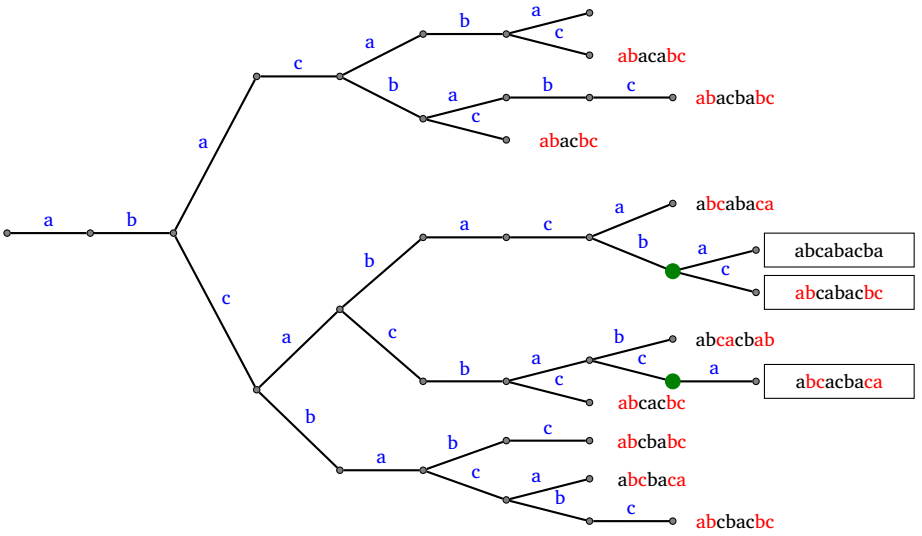

Fig. 4. A trie of all ternary square-free words starting with *ab*, truncated at words that are not s-primitive (in leaves). Only one word in a leaf (*abcabacba*) does not contain the structure specified in the claim inside the proof of Fact 3.2, but it still has a non-trivial s-cover *abcba*. The trie has depth 9 (the leaves with words of length 9 are shown in frames and the internal nodes of depth 8 corresponding to s-primitive words are drawn as big dots), so $\gamma(3) = 8$.

Table 1
The values of γ for small alphabet size k.

k	$\gamma(k)$	examples of longest s-primitive words
1	1	a
2	3	aba
3	8	$abcabacb$
4	19	$abacadbabdcabcbadac$

4. General alphabet, preliminary upper bound

We start with a simple proof of a preliminary upper bound. We use two characteristic subsequences defined below. For a word S over alphabet $Alph(S)$ of size k, let $first(S)$ (resp. $last(S)$) denote the length-k word containing all the letters of $Alph(S)$ in the order of their first (resp. last) occurrence in S.

Example 4.1. We have

$$first(\underline{aba}\underline{d}\underline{bc}\underline{d}) = abdc,\ last(ab\underline{a}\underline{d}\underline{bcd}) = abcd.$$

Lemma 4.2. *For a word S, let $F = first(S)$, $L = last(S)$. If the word $FLFL$ is a subsequence of S, then FL is an s-cover of S.*

Proof. We need to show that each position i of S is covered by an occurrence of FL as a subsequence.

There exists a position j in S such that FL is a subsequence of each of the words $S[..j]$ and $S(j..]$. We can assume that $i \le j$; the other case is symmetric.

Let p be the index such that $F[p] = S[i]$. It suffices to argue that:

(1) $F[..p)$ is a subsequence of $S[..i)$
(2) $(FL)(p..]$ is a subsequence of $S(i..]$.

Point (1) follows by the definition of $F = first(S)$. As for point (2), $S(i..]$ has a subsequence FL by the fact that $i \le j$ and the definition of j, and $(FL)(p..]$ is a suffix of FL. □

The next observation follows from Observation 3.1 and the definition of $\gamma(k)$.

Observation 4.3. *If word S is s-primitive and $|S| > \gamma(k-1)$, then every factor of S of length $\gamma(k-1)+1$ contains at least k different letters.*

Fact 4.4. *For $k > 1$, $\gamma(k) \le (2k+2)(\gamma(k-1)+1) - 1$.*

Proof. It is enough to show that if a word S satisfies $|Alph(S)| = k$ and $|S| = (2k+2)(\gamma(k-1)+1)$, then S is not s-primitive. Assume to the contrary, that such a word S is s-primitive. Let $F = first(S)$, $L = last(S)$. We will show that $FLFL$ is a subsequence of S. By Lemma 4.2, this will yield a contradiction as $|FL| = 2k < |S|$.

Let us partition S into blocks $S_1, S_2, \ldots, S_{2k+2}$ of length $\gamma(k-1)+1$. By Observation 4.3, each of the blocks contains all the k different letters of S.

Hence, F is a subsequence of S_1, L (as a length-k word) is a subsequence of $S_2 S_3 \cdots S_{k+1}$, F is a subsequence of $S_{k+2} S_{k+3} \cdots S_{2k+1}$, and L is a subsequence of S_{2k+2}. Consequently, $FLFL$ is a subsequence of S. □

In the next section, we show how to improve the preliminary bound on $\gamma(k)$ from Fact 4.4.

5. General alphabet, lower bound and improved upper bound

5.1. Lower bound

We need the following straightforward fact.

Observation 5.1. *If a letter a occurs in a word $S = S'aS''$ only once, then C is an s-cover of S if and only if $C = C'aC''$, where C', C'' are s-covers of S', S'', respectively.*

First we obtain a recurrence showing the relation between $\gamma(k)$ and $\gamma(k-1)$ in general case. Then we use the word from Fact 3.4 as a base of the recurrence.

Lemma 5.2. *For $k \geq 2$ we have $\gamma(k) \geq 2 \cdot \gamma(k-1) + 1$.*

Proof. Let S be an s-primitive word of length $\gamma(k-1)$ with $|Alph(S)| = k-1$ and $a \notin Alph(S)$. Then, by Observation 5.1, $S' = SaS$ is s-primitive. We have $|Alph(S')| = k$ and $|S'| = 2 \cdot \gamma(k-1) + 1$. □

Theorem 5.3 *(Lower bound). For $k \geq 4$ we have $\gamma(k) \geq 5 \cdot 2^{k-2} - 1$.*

Proof. Due to Fact 3.4, $\gamma(4) = 19 = 5 \cdot 2^{4-2} - 1$. By Lemma 5.2 and induction, we have for $k > 4$,

$$\gamma(k) \geq 2 \cdot \gamma(k-1) + 1 \geq 2 \cdot (5 \cdot 2^{k-3} - 1) + 1 = 5 \cdot 2^{k-2} - 1. \quad \square$$

5.2. Two combinatorial facts

We define the following condition for two words X, Y of length at least 2.

$\Phi(X,Y)$**:** there exist factors a_1a_2 of X and b_1b_2 of Y, where a_1, a_2, b_1, b_2 are letters, satisfying

$$a_1 \notin \{b_1, b_2\} \ \wedge \ b_2 \notin \{a_1, a_2\}. \tag{2}$$

Example 5.4. We have $\Phi(X,Y)$ for $X = abca$, $Y = aacddd$.

The proof of the following technical lemma is deferred to Section 8.

Lemma 5.5 *(XY-Lemma). If X, Y are square-free words of length 4 and 6, respectively, then $\Phi(X,Y)$.*

For a word $S = a_1 a_2 .. a_n$ of length $n \geq 4$ we introduce yet another, similar condition:

$$\Psi(S) \ \Leftrightarrow \ \Phi(a_1a_2, \ a_{n-1}a_n).$$

We assume that $\Psi(S)$ holds if $n \leq 3$.

Example 5.6. We have $\Psi(abbc)$ but not $\Psi(abcb)$.

The following fact is a consequence of the previous lemma; see Fig. 5.

Corollary 5.7. *If W is square-free, then W contains a factor W' such that $|W'| \geq |W| - 6$ and $\Psi(W')$.*

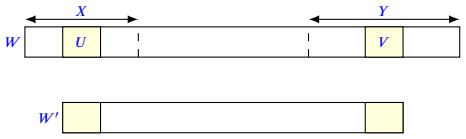

Fig. 5. Illustration of Corollary 5.7. $|X| = 4$, $|Y| = 6$ and $|U| = |V| = 2$, consequently $|W'| \geq |W| - 6$. If $\Phi(U, V)$ then $\Psi(W')$.

Proof. We assume that $|W| \geq 10$ since for $|W| < 10$, $\Psi(W')$ is true for $W' = W[0\,..\,\min(3, |W|))$.

If $|W| \geq 10$, then consider the prefix X of W of length 4 and the suffix Y of W of length 6; see Fig. 5. By Lemma 5.5, $\Phi(X, Y)$ holds. Let $U = a_1a_2$ and $V = b_1b_2$ be the factors of X and Y, respectively, that satisfy the corresponding condition (2), with $U = X[i\,..\,i+1]$ and $V = Y[j\,..\,j+1]$. Then $\Psi(W')$ holds for the factor $W' = W[i\,..\,(|W| - 6) + j + 1]$; we have $|W'| = |W| - 4 + j - i \geq |W| - 6$ as $i \leq 2$, $j \geq 0$. □

Example 5.8. The number 6 in the above corollary cannot be decreased. The word $W = abcabadabacba$ is square-free (and s-primitive, and a palindrome). For every factor W' of W of length $|W'| \geq |W| - 5$, $\Psi(W')$ does not hold. Let us note that $W'' = badabac$ is a factor of W of length $|W| - 6$ such that $\Psi(W'')$ holds.

5.3. Upper bound

In this section, we give a refined upper bound $\gamma(k) \leq (2k-2)\gamma(k-1) + 6$ for $k \geq 4$.

To prove it we need to show that if a word S satisfies $|Alph(S)| = k$ and $|S| > (2k-2)\gamma(k-1) + 6$, then S is not s-primitive. By Corollary 5.7 and Observation 3.1 we know that it is enough to prove this fact for S such that $\Psi(S)$ and $|S| > (2k-2)\gamma(k-1)$ (a factor of the original S).

Let $F = first(S)$. If the last letter of F and the first letter of $last(S)$ are different, then let $L = last(S)$, otherwise let L equal $last(S)$ with the first letter deleted.

Observation 5.9. *FL does not contain any equal adjacent letters.*

Words F, L have the following useful property.

Observation 5.10. *If $S[i]$ equals the jth letter of F, then there is a sequence of j positions $i_1 < i_2 < \cdots < i_j = i$ such that $S[i_1]S[i_2]\cdots S[i_j]$ is a prefix of F. A symmetric "reverse" property holds for L.*

Lemma 5.11. *If $\Psi(S)$ and S is s-primitive then $|S| \leq (2k-2)\gamma(k-1)$.*

Proof. The proof is by contradiction. We show that if S is s-primitive and $|S| > (2k-2)\gamma(k-1)$, then FL is an s-cover of S.

Let us cover S with $2k-2$ blocks, each of length $\gamma(k-1)+1$, with overlaps of one position between consecutive blocks. The final block can be longer if $|S| > (2k-2)\cdot\gamma(k-1) + 1$.

Let $B_1, B_2, \ldots, B_{2k-2}$ denote the respective blocks and $\mathbf{T} = B_1B_2\cdots B_{2k-2}$ be their concatenation. Observation 5.9 implies the following.

Observation 5.12. *FL is an s-cover of S if and only if it is an s-cover of $\mathbf{T}$.*

Proof. Word $\mathbf{T}$ is the word S with some letters duplicated. If FL is an s-cover of S, then the duplicates of letters that occur in $\mathbf{T}$ can be covered using the same subsequences as the originals. Conversely, if FL is an s-cover of $\mathbf{T}$, each subsequence of $\mathbf{T}$ equal to FL transforms to a subsequence equal to FL in S by removing the duplicated letters. Indeed, by Observation 5.9, a subsequence equal to FL of $\mathbf{T}$ uses at most one of each pair of equal adjacent letters. □

Therefore, it suffices to show that FL is an s-cover of $\mathbf{T}$. By Observation 4.3, each block in $\mathbf{T}$ contains all the k letters. The first and the last block in $\mathbf{T}$ contain F and L as a subsequence, respectively. Any $k-3$ consecutive blocks contain F_- or L^- as a subsequence, where X_- (respectively X^-) is the word obtained from X by deleting the first (last, respectively) **three letters**.

Hence there is a decomposition $\mathbf{T} = A\,B\,U\,V\,C\,D$, where U, V are *middle* blocks, A is the first block and contains F, B contains L^-, U and V each contain every distinct letter, C contains F_-, and D contains L.

Assume we want to cover a position i with an occurrence of FL as a subsequence. We start by showing that each position i inside the UV block is covered. It is enough to prove the following claim for the simplified version $\mathbf{T}'$ of $\mathbf{T}$ in which the outer blocks are replaced only by the letters used when a position inside UV is covered with an occurrence of FL.

We notice that $\Psi(\mathbf{T}) \Leftrightarrow \Psi(\mathbf{T}')$, since the first two and last two letters in both words match.

Claim 5.13 *(Middle-blocks). Let $k \geq 4$ and $\mathbf{T}' = F\cdot L^-\cdot U\cdot V\cdot F_-\cdot L$, where F is a permutation of $Alph(\mathbf{T}')$ and either L or $F[|F|-1]L$ is a permutation of $Alph(\mathbf{T}')$. Assume that $\Psi(\mathbf{T}')$ holds and each of U, V contains all letters from $Alph(\mathbf{T}')$. Then FL is an s-cover of $\mathbf{T}'$.*

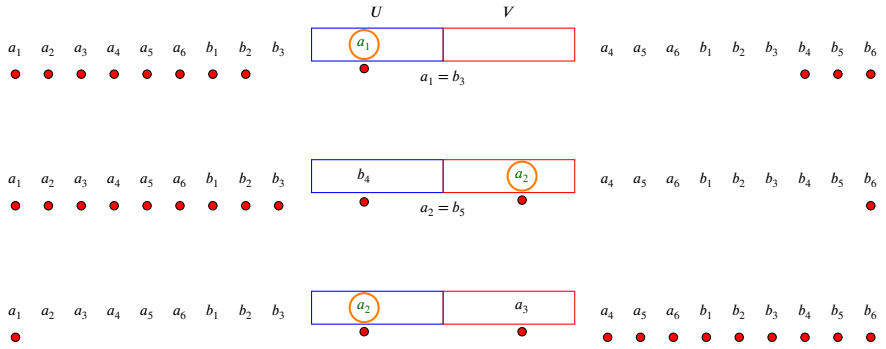

Fig. 6. We have here $\Psi(\mathbf{T}')$, where $\mathbf{T}' = F \cdot L^- \cdot U \cdot V \cdot F_- \cdot L$, $F = a_1 \cdots a_6$, $L = b_1 \cdots b_6$. In the construction, we use the fact that $a_1 \neq b_5$, $a_1 \neq b_6$ and $a_2 \neq b_6$. The big dots show the FL-subsequences covering a chosen position i in UV containing a_1 or a_2. In the last two cases, we find an *additional* position in U, V containing b_4, a_3, respectively. The case $\mathbf{T}'[i] = b_6 \vee \mathbf{T}'[i] = b_5$ is symmetric.

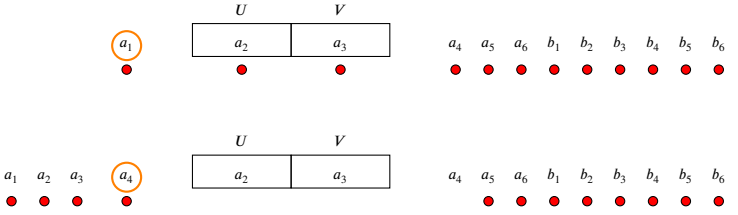

Fig. 7. Illustration of covering a position $i \in AB$ (encircled; to the left of UV). We show the cases $\mathbf{T}[i] = a_1$ and $\mathbf{T}[i] = a_4$.

Proof. It suffices to show that each position in $U \cdot V$ is covered by FL. Let $F = a_1 a_2 \cdots a_k$, $L = b_1 b_2 \cdots b_{k'}$ ($k' \in \{k-1, k\}$). Let i be a position in $\mathbf{T}'$ that is located in UV. There are three critical cases when $\mathbf{T}'[i] \in \{a_1, a_2\}$. Fig. 6 illustrates the proof for each of these cases.

Case 1: $\mathbf{T}'[i] = a_1$. Due to condition $\Psi(\mathbf{T}')$ we have $a_1 = b_j$, $j < k' - 1$. Then FL covers position i as follows: $a_1 a_2 \cdots a_k b_1 b_2 \cdots b_{j-1}$ is a prefix of FL^-, $b_j = \mathbf{T}'[i]$, and $b_{j+1} \cdots b_{k'}$ is a suffix of L.

Case 2: $(\mathbf{T}'[i] = a_2) \wedge (i \in V)$. Again, due to $\Psi(\mathbf{T}')$, $a_2 = b_j$, $j < k'$. Then FL covers position i as follows: $a_1 a_2 \cdots a_k b_1 b_2 \cdots b_{j-2}$ is a prefix of FL^-, $b_{j-1} \in U$, $b_j = \mathbf{T}'[i]$, and $b_{j+1} \cdots b_{k'}$ is a suffix of L.

Case 3: $(\mathbf{T}'[i] = a_2) \wedge (i \in U)$. Now we can select $a_3 \in V$ and FL covers i.

The condition $\Psi(\mathbf{T}')$ is crucial here. The remaining cases, that is, $\mathbf{T}'[i] \notin \{a_1, a_2\}$, are straightforward (the condition $\Psi(\mathbf{T}')$ does not play a role).

Indeed, let $\mathbf{T}'[i] = a_j$ with $j > 2$. Then FL covers position i as follows: $a_1 a_2 \cdots a_{j-1}$ is a prefix of F (hence, of FL^-), $a_j = \mathbf{T}'[i]$, and $a_{j+1} \cdots a_k L$ is a suffix of $F_- L$ (as $j + 1 > 3$). This completes the proof of the claim. □

By the claim, every position in the middle blocks of $\mathbf{T} = ABUVCD$ is covered by an occurrence of FL. Now it remains to show how we can cover the remaining positions of $\mathbf{T}$.

Case 1: i is to the left of UV. Let $\mathbf{T}[i] = a_j$. Then, due to Observation 5.10, there is a sequence of positions $i_1 < i_2 < \cdots < i_j = i$ such that $\mathbf{T}[i_1]\mathbf{T}[i_2] \cdots \mathbf{T}[i_j] = a_1 a_2 \cdots a_j$. The word UVC has a subsequence $a_{j+1} a_{j+2} \cdots a_k$ and the word D has a subsequence L. Hence, FL covers i. Fig. 7 shows the case when $k = 6$ and $\mathbf{T}_i = a_1$ or $\mathbf{T}_i = a_4$.

Case 2: i is to the right of UV. The proof is symmetric to Case 1.

Assuming that S is s-primitive, we derived a contradiction: FL is an s-cover of $\mathbf{T}$, and consequently, due to Observation 5.12, of S. This completes the proof of the lemma. □

Lemma 5.11 and XY-Lemma (Lemma 5.5) directly imply the following upper bound for general case.

Theorem 5.14 *(Upper bound). For $k \geq 4$, $\gamma(k) \leq (2k-2)\gamma(k-1) + 6$.*

Let us define

$$\Delta(4) = 19, \ \Delta(k) = (2k-2) \cdot \Delta(k-1) + 6 \text{ for } k > 4. \tag{3}$$

In the conference version of the paper [15], we showed

$$\gamma(k) \le P(k), \text{ where } P(k) = 2k \cdot P(k-1) \text{ for } k > 5,\ P(4) = 19.$$

Here we improve this bound by replacing $P(k)$ by $\Delta(k)$. We have

$$\Delta(5) = 158,\ \Delta(6) = 1586,\ \Delta(7) = 19038,\ \Delta(8) = 266538.$$

$$P(5) = 190,\ P(6) = 2280,\ P(7) = 32046,\ P(8) = 512736.$$

Furthermore, it can be shown by induction that for $k \ge 4$, $\Delta(k) < 2^{k-1}\, k!$. Therefore, Theorem 5.3, Fact 3.4 and Theorem 5.14 imply the following bounds.

Corollary 5.15. *For $k \ge 4$, we have*

$$5 \cdot 2^{k-1} - 1 \le \gamma(k) < k! \cdot 2^{k-1}.$$

6. Computing s-covers

The following observation is a common property of s-covers and standard covers.

Observation 6.1. *If C is an s-cover of S and C' is an s-cover of C, then C' is an s-cover of S.*

Theorem 6.2. *Let S be a length-n word over an integer alphabet of size $k = n^{\mathcal{O}(1)}$.*

(a) *A shortest s-cover of S can be computed in $\mathcal{O}(n \cdot \min(2^n, (k-1)^{\gamma(k)}))$ time.*
(b) *One can check if S is s-primitive and, if not, return a non-trivial s-cover of S in $\mathcal{O}(n + 2^{\gamma(k)}\gamma(k))$ time.*
(c) *An s-cover of S of length at most $\gamma(k)$ can be computed in $\mathcal{O}(n2^{\gamma(k)}\gamma(k))$ time.*

Proof. We prove each of the three points separately.

Point (a). By Observation 6.1, a shortest s-cover of S is s-primitive. By the definition of $\gamma(k)$, every shortest s-cover C of S is a word of length up to $\gamma(k)$. Moreover, every such C starts with the same letter as S and $C[i] \neq C[i-1]$ holds for all $i \in \{1, \ldots, |C| - 1\}$. Thus the number of words that need to be tested is bounded by $\sum_{\ell=0}^{\gamma(k)-1}(k-1)^{\ell} = \mathcal{O}((k-1)^{\gamma(k)})$. On the other hand, there are 2^n subsequences of S. Hence, there are $\min(2^n, (k-1)^{\gamma(k)})$ candidates to be checked. With the aid of the algorithm from Theorem 2.2 we can generate and check each candidate in $\mathcal{O}(n)$ time. This gives the desired complexity.

Point (b). If $n \le \gamma(k)$, we can use the algorithm from (a) which works in $\mathcal{O}(2^{\gamma(k)}\gamma(k))$ time. Otherwise, S is not s-primitive. We can find a non-trivial s-cover of S as follows.

Let $S = S'S''$ where $|S'| = \gamma(k) + 1$. We can use the algorithm from (a) to compute a shortest s-cover C of S' in $\mathcal{O}(2^{\gamma(k)}\gamma(k))$ time. Then C is a non-trivial s-cover of S'. Then, we can output CS'' as a non-trivial s-cover of S (cf. Observation 3.1). This takes $\mathcal{O}(n + 2^{\gamma(k)}\gamma(k))$ time.

Point (c). By Observation 6.1, any s-cover of an s-cover of S will be an s-cover of S. We can thus repeatedly apply the algorithm underlying (b), apart from outputting the computed non-trivial s-cover. More precisely, in each step the current word is represented as a concatenation of a word of length up to $\gamma(k)$ and a suffix of S.

As each application of this algorithm removes at least one letter of S, the number of steps is at most $n - \gamma(k)$. Each step takes $\mathcal{O}(2^{\gamma(k)}\gamma(k))$ time and hence the conclusion follows. □

Corollary 6.3. *A shortest s-cover of a word over a constant-sized alphabet can be computed in linear time.*

The algorithm from Theorem 6.2(a) can be slightly improved to work in

$$\mathcal{O}(\min(2^n n,\ (n + \gamma(k))\alpha(k,2)^{\gamma(k)}))$$

time, where

$$\alpha(k,2) = k - 1 - \tfrac{1}{k-1} - \tfrac{1}{(k-1)^3} + \mathcal{O}\left(\tfrac{1}{k^5}\right)$$

is an upper bound on the growth rate of square-free words over an alphabet of size k; see [16].

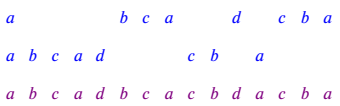

Fig. 8. $C_1 = abca\ d\ cba$ is a shortest s-cover of $S = abca\ d\ bcacb\ d\ acba$ (a palindrome). Hence $C_1^R = C_2 = abc\ d\ acba$ is also an s-cover.

7. The number of distinct shortest s-covers

In the case of standard covers, if a word S has two covers C, C', then one of C, C' is a cover of the other. This property implies, in particular, that a word has exactly one shortest cover. In this section, we show that analogous properties do not hold for s-covers. There exist words S having two s-covers C, C' such that none of C, C' is an s-cover of the other; e.g. $S = abcabcabcb$, $C = abcb$ and $C' = abcacb$. Moreover, a word can have many different shortest s-covers, as shown in Theorem 7.3.

We start with an example of a word S of length 15 (see Fig. 8) with two different shortest s-covers and then extend it recursively.

Lemma 7.1. *The word $S = abca\,d\,bcacb\,d\,acba$ has two different s-covers of length 8, $C_1 = abca\,d\,cba$ and $C_2 = abc\,d\,acba$ (cf. Fig. 8). It does not have any shorter s-cover.*

Proof. Let $S = abcadbcacbdacba$. Fig. 8 illustrates that S has two s-covers of length 8. We show they are the shortest s-covers.

Any s-cover of S must contain the letter d and before its first occurrence letters a, b, c (in that order) must appear. Symmetrically, after this letter, letters c, b, a must appear. The only word of length smaller than 8 which satisfies this property is $abc\ d\ cba$; however, this is not an s-cover of S (as it does not cover the middle letter a in S). □

Remark 7.2. Assume $a_1, a_2, a_3, \cdots, a_m$ are new letters which are pairwise distinct. Let us take the word of the form

$$S\,a_1\,S\,a_2\,S\,a_3\,S\,\cdots\,S\,a_m,$$

of length n.

This word has $2^m = 2^{\frac{n}{16}}$ distinct shortest s-covers, due to a straightforward extension of Observation 5.1. However, the number of distinct letters is here $\Omega(n)$. In the following theorem, we show that it can be lowered to $O(\log n)$.

Theorem 7.3. *For every positive integer n there exists a word of length n over an alphabet of size $\mathcal{O}(\log n)$ that has at least $2^{\lfloor \frac{n+1}{16} \rfloor}$ different shortest s-covers.*

Proof. Assume $a_1, a_2, a_3 \cdots$ are new distinct letters. We now construct a sequence of words T_i such that $T_0 = S$ and $T_i = T_{i-1} a_i T_{i-1}$ for $i > 0$. The word T_i has length $16 \cdot 2^i - 1 = 2^{i+4} - 1$. Let us consider an infinite word T containing each T_i as its prefix (it is well defined as each T_i is a prefix of T_{i+1}). We show by induction, using Observation 5.1, the following fact.

$T[0..n)$ has exactly $2^{\lfloor \frac{n+1}{16} \rfloor}$ different shortest covers.

The base case for $n \le 15$ holds as every word has a shortest s-cover and for $n = 15$ we apply Lemma 7.1 as $T[..15) = S$.

Assume that $n > 15$ and $2^{i+4} \le n < 2^{i+5}$. Then

$$T[0..n) = T_{i+1}[0..n) = T_i\,a_{i+1}\,T_i[0..n - 2^{i+4}).$$

By Observation 5.1, the number of shortest s-covers of $T[..n)$ is the number of shortest s-covers of T_i times the number of shortest s-covers of $T[0..n - 2^{i+4})$, that is

$$2^{\frac{2^{i+4}}{16}} \cdot 2^{\lfloor \frac{n-2^{i+4}+1}{16} \rfloor} = 2^{\lfloor \frac{n+1}{16} \rfloor},$$

as desired. □

8. Proof of XY-Lemma (Lemma 5.5)

It will be convenient in the proof to use a "reversed" version of Φ. We say that X *matches* Y if $\Phi(X, Y^R)$, where Y^R is the reverse of Y. In other words, X matches Y if there are length-2 factors a_1a_2 and b_1b_2 of X and Y, respectively, satisfying the inequalities represented by edges in Fig. 9. Let A, B, C, D be pairwise distinct letters.

Observation 8.1.

(1) *If Z_1, Z_2, Z are three different letters, then Z_1Z matches Z_2Z.*
(2) *If $V \in \{AC, BA, CB, AD, BD, CD\}$, then $ABCA$ matches V.*

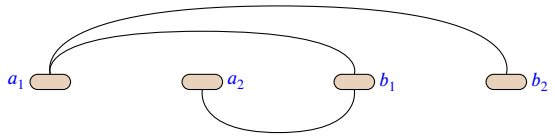

Fig. 9. a_1a_2 matches b_1b_2. The edges mean that two connected symbols are different.

Lemma 5.5 is readily equivalent to the following fact:

Claim 8.2. *Assume $|X| = 4$, $|Y| = 6$ and both X, Y are square-free. Then X matches Y.*

The proof is split into proofs of two claims. The first one concerns the situation when Y satisfies one of several simple conditions. The second one is when none of these conditions is satisfied. Let $\Gamma = \{\lambda_1, \lambda_2, \lambda_3, \lambda_4\}$,

$$\lambda_1 = ABAC,\ \lambda_2 = ABCA,\ \lambda_3 = ABCB,\ \lambda_4 = ABCD.$$

Observation 8.3. *Each λ_i contains AB and contains BA or BC.*

Each length-4 square-free word X is equal, after one-to-one renaming of letters, to one of the words in Γ. The mapping of letters assigns, in a left-to-right order, to each letter of X that does not have a previous occurrence the first unused letter in A, B, C, D. For example, $ADAB$ is "isomorphic" to $ABAC$. Let Γ' be the set of all length-6 square-free words Y satisfying at least one of the following conditions:

1. the length-5 prefix of Y contains a letter outside $\{A, B, C\}$
2. Y contains CB
3. the length-5 prefix of Y contains AC
4. the length-4 prefix of Y contains BA.

Claim 8.4. *If $X \in \Gamma$ and $Y \in \Gamma'$, then X matches Y.*

Proof of the claim. We consider each condition separately. We find matching length-2 factors U and V of X and Y, respectively.

Condition 1. If $Y[i] \notin \{A, B, C\}$ for $i < 5$, then let $V = Y[i..i+1]$. For each $X \in \Gamma$, we will identify a corresponding length-2 factor U such that U matches V. If $Y[i+1] = A$, we can take as U any of the two words BA, BC, since each word in Γ contains one of BA, BC as a factor. If $Y[i+1] \neq A$, then take $U = AB$.

Condition 2. If Y contains $V = CB$, we take $U = AB$. Each word in Γ contains AB.

Condition 3. If Y contains $V = AC$, then it matches $\lambda_2, \lambda_3, \lambda_4$ since each of them contains $U = BC$.

Hence we can assume that $X = \lambda_1 = ABAC$. Denote by Z the next letter after C in Y. If $Z = B$ then $V = CB$ was already considered in the previous condition. If $Z \notin \{A, B, C\}$, then we can take $U = AB$, $V = CZ$. Hence we can assume $Z = A$ and take $V = CA$, which matches BA which is contained in λ_1.

Condition 4. If BA appears in the length-4 prefix of Y, then we can assume that BA is followed by BC or BZ, where $Z \notin \{A, B, C\}$ (otherwise we would have one of the previous conditions or a square in Y). Then Y contains $BABC$ or $BABZ$. Each λ_i matches $BABC$ and $BABZ$. Hence X matches Y. □

Claim 8.5. *If $X \in \Gamma$ and $Y \notin \Gamma'$ is a square-free word, then X matches Y.*

Proof. The length-4 prefix Y' of Y contains only letters A, B, C and contains none of BA, AC, CB, AA, BB, CC as its factor. Hence

$$Y' \in \{ABCA,\ BCAB,\ CABC\}.$$

This implies that each possible word Y' contains each of AB, BC, CA as a factor.

Let us note that AC in λ_1 matches BC (see point (1) of Observation 8.1), CB in λ_3 matches AB, and CD in λ_4 matches AB. Hence, each λ_i, for $i \neq 2$, matches one of AB, BC, CA. Consequently, if $i \neq 2$, $X = \lambda_i$ matches Y since Y contains each of AB, BC, CA.

We are left with the case $X = \lambda_2 = ABCA$. Observe that Y' has period 3. We also know that Y has length 6 and is square-free. Hence the period 3 of Y' is broken at position $i = 5$ or $i = 6$. Let $V = Y[i-1]Y[i]$. We have that

$$V \in \{AC,\ BA,\ CB\} \text{ or } V \in \{A, B, C\} \cdot Z \text{ where } Z \notin \{A, B, C\}.$$

Table 2
Our algorithmic results.

Problem	Complexity	Note
Testing if a single candidate is an s-cover	$\mathcal{O}(n)$	Theorem 2.2
Finding a shortest s-cover	$\mathcal{O}(n \cdot \min(2^n, (k-1)^{\gamma(k)}))$	Theorem 6.2
Finding any s-cover	$\mathcal{O}(n + 2^{\gamma(k)}\gamma(k))$	
Finding an s-cover of length at most $\gamma(k)$	$\mathcal{O}(n2^{\gamma(k)}\gamma(k))$	

Then, due to point (2) in Observation 8.1, $\lambda_2 = ABCA$ matches V. Consequently, X matches Y. This completes the proof of the claim. □

The thesis of Lemma 5.5 follows from the three claims above.

9. Conclusions and open problems

Subsequence covers are algorithmically and combinatorially more complicated than standard covers. We solved several basic problems and left open several others. The main difficulty is the behavior of the function $\gamma(k)$, which grows exponentially and in an irregular way.

Our combinatorial results These results are mainly related to the function $\gamma(k)$.

- Exact values of the function γ for small arguments: $\gamma(1) = 1$, $\gamma(2) = 3$, $\gamma(3) = 8$ (see Fact 3.2), $\gamma(4) = 19$ (Fact 3.4).
- For $k \geq 4$, a recursive lower bound $\gamma(k) \geq 2 \cdot \gamma(k-1) + 1$ (Lemma 5.2) and a compact lower bound $\gamma(k) \geq 5 \cdot 2^{k-2} - 1$ (Theorem 5.3).
- For $k \geq 4$, a recursive upper bound $\gamma(k) \leq (2k-2)\gamma(k-1) + 6$ (Theorem 5.14) and a compact upper bound $\gamma(k) < k! \cdot 2^{k-1}$ (Corollary 5.15).
- For every $n > 0$, a construction of a word of length n over an alphabet of size $\mathcal{O}(\log n)$ that has at least $2^{\lfloor \frac{n+1}{16} \rfloor}$ different shortest s-covers (Theorem 7.3).

Our algorithmic results Table 2 summarizes our algorithmic results, where n denotes the length of the input word and k the size of alphabet.

Open problems There are several natural algorithmic and combinatorial questions:

- What is the complexity of checking if a given word is s-primitive? We do not even know if it is in NP. We know it is in co-NP, because we can guess a witness (a non-trivial s-cover) and test it in linear time.
- What is the complexity of computing the shortest s-cover of a word?
- What is the complexity of computing the number of different s-covers of a word?
- What is the exact value of $\gamma(5)$? We know only that $39 \leq \gamma(5) \leq 158$.
- Let us define $\gamma'(1) = 1$, $\gamma'(k+1) = 2\gamma'(k) + k$ for $k > 1$. We have $\gamma(k) = \gamma'(k)$ for $1 \leq k < 5$. Is it always true?
- Is there a short and computer-free proof that $abacadbabdcabcbadac$ is s-primitive?

We conjecture that the first three (algorithmic) problems are NP-hard for general alphabets. The last three (combinatorial) problems seem to be elusive.

CRediT authorship contribution statement

Panagiotis Charalampopoulos: Writing – review & editing, Writing – original draft, Methodology, Investigation, Formal analysis, Conceptualization. **Solon P. Pissis:** Writing – review & editing, Writing – original draft, Methodology, Investigation, Formal analysis, Conceptualization. **Jakub Radoszewski:** Writing – review & editing, Writing – original draft, Methodology, Investigation, Funding acquisition, Formal analysis. **Wojciech Rytter:** Writing – review & editing, Writing – original draft, Methodology, Investigation, Formal analysis, Conceptualization. **Tomasz Waleń:** Writing – review & editing, Writing – original draft, Methodology, Investigation, Formal analysis, Conceptualization. **Wiktor Zuba:** Writing – review & editing, Writing – original draft, Methodology, Investigation, Formal analysis, Conceptualization.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Jakub Radoszewski reports financial support was provided by National Science Centre Poland, grant no. 2018/31/D/ST6/03991. Tomasz Waleń reports financial support was provided by National Science Centre Poland, grant no. 2018/31/D/ST6/03991. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

We thank Juliusz Straszyński for his help in conducting computer experiments.

Data availability

An explicit link to supplementary code on github is presented in the paper.

References

[1] A. Apostolico, M. Farach, C.S. Iliopoulos, Optimal superprimitivity testing for strings, Inf. Process. Lett. 39 (1) (1991) 17–20, https://doi.org/10.1016/0020-0190(91)90056-N.
[2] D. Breslauer, An on-line string superprimitivity test, Inf. Process. Lett. 44 (6) (1992) 345–347, https://doi.org/10.1016/0020-0190(92)90111-8.
[3] D.W.G. Moore, W.F. Smyth, A correction to "An optimal algorithm to compute all the covers of a string", Inf. Process. Lett. 54 (2) (1995) 101–103, https://doi.org/10.1016/0020-0190(94)00235-Q.
[4] P. Czajka, J. Radoszewski, Experimental evaluation of algorithms for computing quasiperiods, Theor. Comput. Sci. 854 (2021) 17–29, https://doi.org/10.1016/j.tcs.2020.11.033.
[5] N. Mhaskar, W.F. Smyth, String covering: a survey, Fundam. Inform. 190 (1) (2022) 17–45, https://doi.org/10.3233/FI-222164.
[6] M.K. Warmuth, D. Haussler, On the complexity of iterated shuffle, J. Comput. Syst. Sci. 28 (3) (1984) 345–358, https://doi.org/10.1016/0022-0000(84)90018-7.
[7] R. Rizzi, S. Vialette, On recognising words that are squares for the shuffle product, Theor. Comput. Sci. 956 (2023) 111156, https://doi.org/10.1016/J.TCS.2017.04.003.
[8] S. Buss, M. Soltys, Unshuffling a square is NP-hard, J. Comput. Syst. Sci. 80 (4) (2014) 766–776, https://doi.org/10.1016/j.jcss.2013.11.002.
[9] M. Lothaire, Algebraic Combinatorics on Words, Encyclopedia of Mathematics and Its Applications, Cambridge University Press, 2002.
[10] R. Kolpakov, M. Podolskiy, M. Posypkin, N. Khrapov, Searching of gapped repeats and subrepetitions in a word, J. Discret. Algorithms 46–47 (2017) 1–15, https://doi.org/10.1016/J.JDA.2017.10.004.
[11] L. Bulteau, S. Vialette, Recognizing binary shuffle squares is NP-hard, Theor. Comput. Sci. 806 (2020) 116–132, https://doi.org/10.1016/j.tcs.2019.01.012.
[12] T. Flouri, C.S. Iliopoulos, T. Kociumaka, S.P. Pissis, S.J. Puglisi, W.F. Smyth, W. Tyczyński, Enhanced string covering, Theor. Comput. Sci. 506 (2013) 102–114, https://doi.org/10.1016/j.tcs.2013.08.013.
[13] T. Kociumaka, S.P. Pissis, J. Radoszewski, W. Rytter, T. Waleń, Fast algorithm for partial covers in words, Algorithmica 73 (1) (2015) 217–233, https://doi.org/10.1007/s00453-014-9915-3.
[14] J. Radoszewski, Linear time construction of cover suffix tree and applications, in: 31st Annual European Symposium on Algorithms, ESA 2023, in: LIPIcs, vol. 274, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023, pp. 89:1–89:17, https://doi.org/10.4230/LIPICS.ESA.2023.89.
[15] P. Charalampopoulos, S.P. Pissis, J. Radoszewski, W. Rytter, T. Waleń, W. Zuba, Subsequence covers of words, in: String Processing and Information Retrieval - 29th International Symposium, SPIRE 2022, in: Lecture Notes in Computer Science, vol. 13617, Springer, 2022, pp. 3–15, https://doi.org/10.1007/978-3-031-20643-6_1.
[16] A.M. Shur, Growth of power-free languages over large alphabets, Theory Comput. Syst. 54 (2) (2014) 224–243, https://doi.org/10.1007/S00224-013-9512-X.