Quasi-Periodicity Under Mismatch Errors

Amihood Amir
Bar-Ilan University and Johns Hopkins University
Ramat-Gan, Israel
amir@cs.biu.ac.il

Avivit Levy
Shenkar College
Ramat-Gan, Israel
avivitlevy@shenkar.ac.il

Ely Porat
Bar-Ilan University
Ramat-Gan, Israel
porately@cs.biu.ac.il

Abstract

Tracing regularities plays a key role in data analysis for various areas of science, including coding and automata theory, formal language theory, combinatorics, molecular biology and many others. Part of the scientific process is understanding and explaining these regularities. A common notion to describe regularity in a string T is a *cover* or *quasi-period*, which is a string C for which every letter of T lies within some occurrence of C. In many applications finding exact repetitions is not sufficient, due to the presence of errors. In this paper we initiate the study of *quasi-periodicity persistence* under mismatch errors, and our goal is to characterize situations where a given quasi-periodic string remains quasi-periodic even after substitution errors have been introduced to the string. Our study results in proving necessary conditions as well as a theorem stating sufficient conditions for quasi-periodicity persistence. As an application, we are able to close the gap in understanding the complexity of *Approximate Cover Problem* (ACP) relaxations studied by [5, 4] and solve an open question.

2012 ACM Subject Classification Mathematics of computing → Combinatorics on words, Theory of computation → Pattern matching

Keywords and phrases Periodicity, Quasi-Periodicity, Cover, Approximate Cover

Digital Object Identifier 10.4230/LIPIcs.CPM.2018.4

1 Introduction

Tracing regularities plays a key role in data analysis for various areas of science, including coding and automata theory, formal language theory, combinatorics, molecular biology and many others. Part of the scientific process is understanding and explaining these regularities. A typical form of regularity is *periodicity*, meaning that a "long" string T can be represented as a concatenation of copies of a "short" string P, possibly ending in a prefix of P. Periodicity has been extensively studied in Computer Science over the years (see [26]).

For many phenomena the definition of periodicity is too restrictive and it is necessary to study wider classes of repetitive patterns. One common such notion is that of a *cover* or a *quasi-period*, defined as follows.

▶ **Definition 1** (Cover). A length m substring C of a string T of length n, is said to be a *cover* of T, if $n > m$ and every letter of T lies within some occurrence of C.

Note that by the definition of cover, the string C is both a prefix and a suffix of the string T. For example, consider the string $T = abaababaaba$. Clearly, T is "almost" periodic with period aba, however, as it is not completely periodic, the algorithms that exploit repetitions cannot be applied to it. On the other hand, the string $C = aba$ is a cover of T, which allows applying to T cover-based algorithms. In this paper we study quasi-periodicity under mismatch errors.

Quasi-periodicity was introduced by Ehrenfeucht in 1990 (according to [8]). The earliest paper in which it was studied is by Apostolico, Farach and Iliopoulos [9], which defined the *quasi-period* of a string to be the length of its shortest cover and presented an algorithm for computing the quasi-period of a given string in $O(n)$ time and space. The new notion attracted immediately several groups of researchers (e.g. [10], [27, 28], [25], [11]). An overview on the first decade of the research on covers can be found in the surveys [8, 18, 31].

While covers are a significant generalization of the notion of periods as formalizing regularities in strings, they are still restrictive, in the sense that it remains unlikely that an arbitrary string has a cover shorter than the word itself. One direction to deal with this is to study different variants of quasi-periodicity. Variants that were introduces include *seeds* [20], *maximal quasi-periodic substring* [7], the notion of *k-covers* [19], *λ-cover* [32], *enhanced covers* [16], *partial cover* [21]. Since the notion of a seed is necessary to our study, we give its formal definition here.

▶ **Definition 2** (Seed). A length m substring C of a string T of length n, is said to be a *seed* of T, if $n > m$ and there exists a superstring T' of T such that C is a cover of T'.

Note that the definition of a seed allows the first and last occurrence of the seed C in T to be incomplete. Other recently explored directions include the inverse problem for cover arrays [14], extensions to strings in which not all letters are uniquely defined, such as *indeterminate strings* [6] or *weighted sequences* [33]. Some of the related problems are $\mathcal{NP}$-hard (see e.g., [6, 12, 21]).

Another direction to deal with the restrictiveness of quasi-periodicity definition is to consider the presence of errors. This direction was motivated by some applications, such as molecular biology and computer-assisted music analysis, were finding exact repetitions is not sufficient. In these applications, a more appropriate notion is that of *approximate repetitions*, where errors are allowed (see, e.g., [13, 15]). This notion was first studied in 1993 by Landau and Schmidt [23, 24] who concentrated on approximate tandem repeats. Note that, the natural definition of an approximate repetition is not clear. One possible definition is that the distance between any two adjacent repeats is small. Another possibility is that all repeats lie at a small distance from a single "original". Such a definition of *approximate seeds* is studied in [12, 29, 17]. Indeed, all these definitions along with other ones were proposed and studied (see [1, 22, 30]). Yet another possibility is that all repeats must be equal, but we allow a fixed total number of mismatches. The possibility presented in [1] is a global one, assuming that an original unknown string is a sequence of repeats without errors, but the process of sequence creation or transmission incurs errors to the sequence of repeats, and, thus, the examined input string is not a sequence of repeats. [5] extend this approach to quasi-periodicity and study the *approximate cover problem* (ACP), in which the input text is a sequence of some cover repetitions with possible mismatch errors, and a string that covers the text with the minimum number of errors is sought. We continue this line of research by studying quasi-periodic strings that have been introduced to substitution errors.

Our Results. In this paper we initiate the study of *quasi-periodicity persistence* under mismatch errors, and our goal is to characterize situations where a given quasi-periodic string remains quasi-periodic even after substitution errors have been introduced to the string. An implicit study of this question was introduced by [3], while proving that two strings with Hamming distance 1 cannot be both quasi-periodic. In our terminology this means that quasi-periodicity does not persist under a single substitution error. Broadening the study to situations where more than one substitution error may happen necessitates a deep understanding of the structure of quasi-periodic strings as well as their behaviour under mismatch errors. Our study results in proving necessary conditions as well as a theorem stating sufficient conditions for quasi-periodicity persistence. As an application, we are able to close the gap in understanding the complexity of ACP relaxations studied by [5, 4] and solve an open question [5] regarding the complexity of the full-tiling relaxation of the ACP.

Paper Contributions. The main contributions of this paper are:

- Giving a first explicit study of quasi-periodicity persistence, while broadening the knowledge on quasi-periodic strings under mismatch errors.
- Proving that the full-tiling relaxation of the ACP is polynomial-time computable, which was beyond the reach of current research prior to this paper. This result closes the gap in understanding the complexity hierarchy of ACP relaxations, where the ACP itself was proven to be $\mathcal{NP}$-hard [5].
- Proving properties of covers, seeds and quasi-periodic strings under mismatch errors that can serve future research of approximate regularities.

The paper is organized as follows. In Section 2, we give formal definitions and basic lemmas. Section 3 is devoted to the study of quasi-periodicity persistence under mismatch errors. In Section 4, we give a description of the full-tiling relaxation of the ACP and prove it is polynomial-time computable by applying the results from Section 3.

2 Preliminaries

In this section we give the needed formal definitions and prove some basic combinatorial properties of covers and seeds.

▶ **Definition 3** (Tiling). Let T be a string over alphabet Σ such that the string C over alphabet Σ is a cover of T. Then, the sorted list of indices representing the start positions of occurrences of the cover C in the text T is called the *tiling* of C in T.

In this paper we have a text T which may contain substitution errors and, therefore, is not coverable. However, we would like to refer to a retained tiling of an unknown string C in T although C does not cover T because of mismatch positions. The following definition makes a distinction between a list of indices that may be assumed to be a tiling of the text before mismatch errors occurred and a list of indices that cannot be such a tiling.

▶ **Definition 4** (A Valid Tiling). Let T be an n-length string over alphabet Σ and let L be a sorted list of indices $L \subset \{1, ..., n\}$. Let $m = n + 1 - L_{last}$, where L_{last} is the last index in L. Then, L is called *a valid tiling* of T, if $i_1 = 1$ and for every $i_k, i_{k+1} \in L$, it holds that $i_{k+1} - i_k \leq m$.

▶ **Notation 5.** *Let C be an m length string over alphabet Σ. Denote by $S(C)$ a string of length n, $n > m$, such that C is a cover of $S(C)$.*

Note that $S(C)$ is not uniquely defined even for a fixed $n > m$, since different valid tilings of the m-length string C may generate a different n-length string $S(C)$. A unique version is obtained if a unique appropriate valid tiling L is also given.

▶ **Notation 6.** *Let T be an n-length string over alphabet Σ and let L be a valid tiling of T. Let $m = n + 1 - L_{last}$, where L_{last} is the last index in the tiling L. For any m-length string C', let $S_L(C')$ be the n-length string obtained using C' as a cover and L as the tiling as follows: $S_L(C')$ begins with a copy of C' and for each index i in L a new copy of C' is concatenated starting from index i of $S_L(C')$ (maybe running over a suffix of the last copy of C').*

▶ **Definition 7.** Let T be a string of length n over alphabet Σ. Let H be the Hamming distance. The *distance of T from being covered* is:

$$dist = \min_{C \in \Sigma^*, |C| < n, S(C) \in \Sigma^n} H(S(C), T).$$

We will also refer to *dist* as *the number of errors in T*.

▶ **Definition 8.** Let T be a string of length n over alphabet Σ. An m-long string C over Σ, $m \in \mathbb{N}$, $m < n$, is called *an m-length approximate cover of T*, if for every string C' of length m over Σ, $\min_{S(C') \in \Sigma^n} H(S(C'), T) \geq \min_{S(C) \in \Sigma^n} H(S(C), T)$, where H is the Hamming distance of the given strings.
We refer to $\min_{S(C) \in \Sigma^n} H(S(C), T)$ as the *number of errors of an m-length approximate cover of T*.

▶ **Definition 9** (Approximate Cover)**.** Let T be a string of length n over alphabet Σ. A string C over alphabet Σ is called an *approximate cover of T* if:

1. C is an m-length approximate cover of T for some $m \in \mathbb{N}$, $m < n$, for which

$$\min_{S(C) \in \Sigma^n} H(S(C), T) = dist.$$

2. for every m'-length approximate cover of T, C', s.t. $\min_{S(C') \in \Sigma^n} H(S(C'), T) = dist$, it holds that: $m' \geq m$.

Primitivity. By definition, an approximate cover C should be *primitive*, i.e., it cannot be covered by a string other than itself (otherwise, T has a cover with a smaller length). Note that a periodic string can be covered by a smaller string (not necessarily the period), and therefore, is not primitive.

▶ **Definition 10.** The *Approximate Cover Problem* (ACP) is the following:
INPUT: String T of length n over alphabet Σ.
OUTPUT: An approximate cover of T, C, and the number of errors in T.

2.1 Properties of Covers and Seeds

Our analysis of quasi-periodicity persistence under mismatch errors in Section 3 requires a preliminary study of properties of covers and seeds. We use the following easy observations:

▶ **Observation 11.** *If a string W is coverable but non-periodic then the shortest cover c of W satisfies $|c| < |W|/2$.*

▶ **Observation 12.** *Coverability is transitive, i.e., a cover of a cover of W is a cover of W.*

▶ **Observation 13.** *If a string c is a cover of a string W then c is a seed of every factor of W of length at least $|c|$.*

▶ **Observation 14.** *If a string s is a seed of a string W then s is a seed of every factor of W of length at least $|s|$.*

▶ **Lemma 15.** *A periodic string W is coverable, and one of the following must hold:*

1. *Its shortest cover is of length more than $|W|/2$. In this case, $W = p^2p'$, where p' is a non-empty prefix of p.*
2. *Its shortest cover c is of length at most $|W|/2$ and $|c| \neq |p|$, where p is the period of W.*
3. *Its shortest cover c is of length at most $|W|/2$ and $|c| = |p|$, where p is the period of W. In this case, $c = p$ and $W = p^i$.*

Example: Consider the following periodic strings:

- The string $W = abaabaa$ is periodic with period $p = aba$. It is coverable by $c = abaa = pp'$, where $|c| > |W|/2$. Note that $W = p^2p'$, where $p' = a$ is a prefix of p.
- The string $W = abaababaababa$ is periodic with period $p = abaab$. It is coverable by the string $pp' = abaababa$, however, its shortest cover is $c = aba$, where $|c| < |W|/2$ and $|c| < |p|$. Note that c does not cover p.
- The string $W = abababa$ is periodic with period $p = ab$. It is coverable by $c = pp' = aba$, where $|c| < |W|/2$, and $|c| > |p|$.
- The string $W = ababaababa$ is periodic with period $p = ababa$. It is coverable by its period p since $W = p^2$, however, its shortest cover is $c = aba$, where $|c| < |W|/2$ and $|c| < |p|$. Note that c covers p. We can make a longer string with this period $W_1 = ababaababaaba$ which is still periodic with p and covered by c. However, if we take $W_2 = ababaababaa$ which is again still periodic with p but no longer coverable by aba (which remains its seed). The shortest cover of W_2 is $c_2 = ababaa$.

The next properties require an additional notation:

▶ **Notation 16.** *Let w be a string. Denote by w^j the string w with the j-th symbol substituted by some other symbol, i.e., $|w^{(j)}| = |w|$, for all $i = 1, \ldots, |w|$, $i \neq j$, $w_i = w_i^{(j)}$, and $w_j \neq w_j^{(j)}$. In this case we also write $w =_j w^{(j)}$.*

▶ **Theorem 17.** *Let S be a length n periodic string with period P. Then for any $j \in \{1, \ldots, n\}$, $S^{(j)}$ is not periodic.*

Theorem 17 can be easily proven using a lemma proved in [2]. A similar result for covers is also known [3]:

▶ **Theorem 18.** [Amir et al. [3]] For every string W of length n and index $j \in \{1, \ldots, n\}$, at most one of the strings W, $W^{(j)}$ is coverable.

We will also need the following auxiliary lemma from [3].

▶ **Lemma 19.** [Amir et al. [3]] Let w be a string and j be an index. Then w is not a seed of $w^{(j)}$.

▶ **Corollary 20.** [Amir et al. [3]] Let U and V be two coverable strings of the same length such that $U \neq V$. Then, $H(U, V) > 1$.

We also make use of the following lemma.

▶ **Lemma 21.** *Let W be a coverable string and let j be an index. Then no prefix (suffix) of $W^{(j)}$ of length at most $|W|/2$ is a seed of $W^{(j)}$.*

3 Quasi-Periodicity Persistence Under Mismatch Errors

In this section we analyze the extent to which quasi-periodicity may persist under mismatch errors, and characterize the structure of strings, the number and positions of mismatch errors, where such a persistence of quasi-periodicity is assured. First note that Corollary 20 means that quasi-periodicity is *not* persistent under a single mismatch error, no matter its position. Our analysis effort is, therefore, devoted to study the case where more mismatch errors occur. In such situations the number of errors as well as their exact positions determine wether the quasi-periodicity persists or not, as the following example shows.

Example: Consider the quasi-periodic (specifically, also periodic) string: $S = abbbbabbbb$. Two mismatch errors in positions 2 and 3 give the primitive string $S' = aaabbabbbb$. However, two mismatch errors in positions 2 and 6 give the (quasi-)periodic string $S'' = aabbbaabbb$.

We, therefore, need to refer to the structure of a quasi-periodic string, in order to analyze persistence of quasi-periodicity. Since the structure of the special case of periodic strings is known, giving sufficient conditions for periodicity persistence is easy, as the following observation states.

▶ **Observation 22.** *Let S be a periodic string with period length p. Let $q = a \cdot p$, $a \in \mathbb{N}$, such that $\lfloor \frac{n}{q} \rfloor \geq 2$, and let S' be the string obtained from S by $k = \lfloor \frac{n}{q} \rfloor$ errors of substitution to a character $\sigma \in \Sigma$ where the difference between each subsequent error positions is q, then S' is periodic.*

We also want to give sufficient conditions for quasi-periodicity persistence, which is much more complicated to understand. Our first goal is to give a formal characterization of a structure of quasi-periodic strings. We begin by characterizing the structure of any string that may serve as a cover of another string. We call this string of variables *the free variable scheme* of the cover. We first give a definition of this term.

▶ **Definition 23.** Let Γ be an alphabet, called the free variables alphabet. A string in Γ^* is a *free variable scheme*. A free variable scheme α over alphabet Σ defines a subset S_α of Σ^*, where s is in S_α if there is a function $\Phi : \Sigma \to \Gamma$ such that $\Phi(s) = \alpha$.

▶ **Lemma 24.** *The* free variable scheme *representing any primitive cover is of the form $\alpha\beta\alpha$ where: β is a non empty string of distinct variables that do not occur in α, α is defined recursively, as follows:*

1. *α is empty, or*
2. *α is of the form $\alpha'\beta'\alpha'$, where β' is a, possibly empty, string of variables that do not occur in α', and α' is recursively defined similarly as α.*

Proof. There are two cases to consider:

1. If there are no overlaps of the cover in the string, then the cover is actually a period (having only complete appearances in the string it covers). In this case, it has the form $\alpha\beta\alpha$, where α is empty and β is a non empty string of distinct variables that do not occur in α.
2. If there is at least one overlap of the cover in the string it covers, then the cover cannot be a period. Such an overlap forces the structure of the cover to begin and end with the same string, represented by the sequence of variables α. It is, therefore, of the form $\alpha\beta\alpha$ where β is a string of distinct variables that do not occur in α. The fact that the cover is primitive, means that β is non empty. Otherwise, it is periodic, therefore, by Lemma 15

it is coverable, contradiction to its primitivity. Now, if all the overlaps of the cover are of the same length as α, then the recursion ends and we get $\alpha = \beta'$ and α' is empty. Otherwise, any overlap of different length forces a recursive structure as follows. Assume without loss of generality that the longer overlap is of length as α, then the existence of a smaller overlap means that α begins and ends with the same string, represented by the sequence of variables α'. These occurrences of α' in α may be separated by another string, represented by the sequence of variables β'. ◀

Example: The recursive structure of the following free variable scheme

$$WWYWWZWWYWWABCWWYWWZWWYWW$$

is:
level 1: $\alpha = WWYWWZWWYWW$ and $\beta = ABC$
level 2: $\alpha = WWYWW$ and $\beta = Z$
level 3: $\alpha = WW$ and $\beta = Y$
level 4: $\alpha = W$ and β is the empty string
level 5: α is the empty string and $\beta = W$.

Lemma 24 serves us in two directions. On the one hand, it is used to characterize the structure of any primitive cover, and on the other hand, it serves to characterize the structure of any quasi-periodic string with more than two occurrences of the cover. Note that, by Lemma 15, any quasi-periodic string with only two occurrences of the cover is periodic, for which Observation 22 applies. We, therefore, refer only to quasi-periodic strings having more than two occurrences of their cover. We call such strings *non-degenerate* quasi-periodic. The second direction is applied as formalized by Observation 25.

▶ **Observation 25.** *The free variable scheme applies to any non-degenerate quasi-periodic string S by defining C as follows. Take the longest prefix of S with length q_0 smaller than $|S|/2$, which equals the suffix of S with the same length. Define α to have length q_0. The length of β will be $|S| - 2 \cdot q_0$. Define $\alpha_0 = \alpha$, $\beta_0 = \beta$. The definition of C continues recursively as long as S_{α_i} has a prefix of length at most $|S_{\alpha_i}|/2$ which equals its suffix of the same length. Let $S_{\alpha_{i+1}}$ be the prefix, and define: $\alpha_i = \alpha_{i+1}\beta_{i+1}\alpha_{i+1}$. Otherwise, the recursion stops with $\alpha_i = \beta_{i+1}$. For all i, β_i is defined to be a sequence of distinct variables that do not occur elsewhere.*

Example: Consider the string $S = abaababaabaababaababa$, which is quasi-periodic with cover aba. Applying Observation 25 to S gives:
level 1: $S_{\alpha_0} = abaababa$, $|\alpha_0| = 8$ and $|\beta_0| = 5$
level 2: $S_{\alpha_1} = aba$, $|\alpha_1| = 3$ and $|\beta_1| = 2$
level 3: $S_{\alpha_2} = a$, $|\alpha_2| = 1$ and $|\beta_2| = 1$.
The free variable scheme we get is: $C = Y_1Y_2Y_1Y_3Y_4Y_1Y_2Y_1Y_5Y_6Y_7Y_8Y_9Y_1Y_2Y_1Y_3Y_4Y_1Y_2Y_1$. The assignment: $Y_1 = a$, $Y_2 = b$, $Y_3 = a$, $Y_4 = b$, $Y_5 = a$, $Y_6 = b$, $Y_7 = a$, $Y_8 = a$, $Y_9 = b$, to the variables of C gives the string S. Note that C has a structure of a primitive cover, however, the above assignment results in a non-primitive string.

Observation 25 enables us to refer to any non-degenerate quasi-periodic string as an assignment to its free variable scheme. Our analysis takes into account the positions of the mismatch errors inserted to this string by referring to them as changing the assignment of a set of variables in its free variable scheme. In order to continue with our analysis, we need the following notation.

▶ **Notation 26.** *Let C be a given free variable scheme. Denote by $\mathcal{A}_C$ the string created by an assignment $\mathcal{A}$ of alphabet symbols to the variables of C. Denote by $\mathcal{A}_C(X_{i_1}, ..., X_{i_j})$ the string created by assigning the variables $X_{i_1}, ...X_{i_j}$ of C an alphabet symbol that is different from the one assigned by $\mathcal{A}$, and all other variables get assigned the same alphabet symbol as in the assignment $\mathcal{A}$.*

We begin by showing that, assuming we already have an assignment $\mathcal{A}$ giving a quasi-periodic string $\mathcal{A}_C$, then changing the assignment of **any** variable that appears once in the α or β parts of the free variable scheme C results in a primitive string.

▶ **Lemma 27.** *Let C be a given free variable scheme, and let Y be a variable that appears only once in the α or β parts of C. Assume that $\mathcal{A}_C$ is non-primitive, then $\mathcal{A}_C(Y)$ is primitive.*

We can now proceed with checking the case of variables that appear more than once in α. Changing the assignment of such a variable may in some cases result in a primitive string, but in other cases result in another non-primitive string, as the next example shows.

Example: Consider $C = XYVXYZXYVXY$, where $\alpha = XYVXY$ and $\beta = Z$. Note that both X and Y appear twice in α. Now the string $\mathcal{A}_C = aaaaaaaaaaa$ is non-primitive, however, both $\mathcal{A}_C(X) = baabaabaaba$ and $\mathcal{A}_C(Y) = abaabaabaab$ are also non-primitive with covers $baaba$ and $abaab$, respectively. Nonetheless, changing the assignment of both X and Y also yields a non-primitive string $\mathcal{A}_C(X, Y) = bbabbabbabb$ with a cover $bbabb$.

Lemma 30 characterizes the case of a variable Y for which $\mathcal{A}_C(Y)$ is also non-primitive. We need Definition 28 and Lemma 29 for the statement and proof of Lemma 30.

▶ **Definition 28** (Superimposed Tiling)**.** Let L_1 and L_2 be valid tilings over length n. Then, L_1 is said to be *superimposed on* L_2, if for every index r in L_2, let s be the greatest index in L_1 satisfying $s \leq r$, then $r + m_2 \leq s + m_1$, where m_1, m_2 are the lengths of the covers in L_1 and L_2, respectively ($m = n - L_{last}$, where L_{last} is the last index in a given tiling L).

▶ **Lemma 29.** *Let $\mathcal{A}_C = \mathcal{A}_\alpha S \mathcal{A}_\alpha$ and $\mathcal{A}_C(Y) = \mathcal{A}_\alpha(Y) S \mathcal{A}_\alpha(Y)$, where $\mathcal{A}_\alpha$ is a sequence of at least 2 appearances of W and $\mathcal{A}_\alpha(Y)$ is sequence of the same number of appearances of $W^{(j)}$, then only one of the strings $\mathcal{A}_C$, $\mathcal{A}_C(Y)$ can be non-primitive and the other must be primitive.*

▶ **Lemma 30** (Quasi-Periodicity Persistence Necessary Condition)**.** *If there exists a variable Y in C such that both $\mathcal{A}_C$ and $\mathcal{A}_C(Y)$ are non-primitive, then:*

1. *Y appears 2^i times in C, where $i \geq 2$.*
2. *If Y appears 2^i times in C for some $i \geq 2$, then $C = \alpha_i \beta_i \alpha_i$, and for each $1 < \ell \leq i$, $\alpha_\ell = \alpha_{\ell-1}\beta_{\ell-1}\alpha_{\ell-1}$, where Y appears once in α_1.*
3. *There exists an ℓ, $1 \leq \ell < i$ such that β_ℓ is non-empty.*
4. *Let L_c and $L_{c'}$ be the tiling of the cover c in $\mathcal{A}_C$ and c' in $\mathcal{A}_C(Y)$, respectively. Then, if $|c| \leq |c'|$ then $L_{c'}$ is superimposed on L_c, else, L_c is superimposed on $L_{c'}$.*

Proof. First, by Lemma 27, Y must appear at least twice in α (and therefore, exactly twice as that in C). Thus, by the recursive structure of C (Lemma 24), Y appears a power of 2 times in α and, therefore, a power of two times in C. Also, by Lemma 24, this recursive structure is of the form $C = \alpha_i \beta_i \alpha_i$, and for each $1 < \ell \leq i$, $\alpha_\ell = \alpha_{\ell-1}\beta_{\ell-1}\alpha_{\ell-1}$, where Y appears once in α_1. Denote by W the string $\mathcal{A}_{\alpha_1}$ and by S_ℓ the string $\mathcal{A}_{\beta_\ell}$, for all ℓ. Note that since Y only appears once in α_1 and doesn't appear elsewhere, we have that

$\mathcal{A}_{\beta_\ell}(Y) = \mathcal{A}_{\beta_\ell} = S_\ell$, for all ℓ, and $\mathcal{A}_{\alpha_1}(Y) = W^{(j)}$ for the index j that indicates the position of Y in α_1.

We now prove that there exists an ℓ, $1 \leq \ell < i$ such that β_ℓ is non-empty. Assume to the contrary that only S_i is nonempty (because β_i must be nonempty by the definition of C), then we have that $\mathcal{A}_C = \mathcal{A}_{\alpha_i} S_i \mathcal{A}_{\alpha_i}$ and $\mathcal{A}_C(Y) = \mathcal{A}_{\alpha_i}(Y) S_i \mathcal{A}_{\alpha_i}(Y)$, where $\mathcal{A}_{\alpha_i}$ is a sequence of 2^{i-1} appearances of W and $\mathcal{A}_{\alpha_i}(Y)$ is a sequence of 2^{i-1} appearances of $W^{(j)}$. Note that since $2^{i-1} \geq 2$, then by Lemma 29, only one of the strings $\mathcal{A}_C$, $\mathcal{A}_C(Y)$ can be non-primitive and the other must be primitive, contradiction. Therefore, there exists an ℓ, $1 \leq \ell < i$ such that β_ℓ is non-empty.

Since both $\mathcal{A}_C$ and $\mathcal{A}_C(Y)$ are non-primitive, they have shortest covers c and c', respectively. Assume without loss of generality that $|c| \leq |c'|$. It remains to show that $L_{c'}$ is superimposed on L_c. Assume to the contrary that this is not the case, and let r be the first index in L_c, such that for the greatest index s in $L_{c'}$ satisfying $s \leq r$, we have $r + |c| > s + |c'|$. First, note that $s \neq r$, otherwise, we necessarily have: $r + |c| \leq s + |c'|$, since $|c| \leq |c'|$, which contradicts the assumption. Therefore, $s < r$.

Let $\hat{c}$ be the $|c'|$-length prefix of $\mathcal{A}_C$. Since c covers $\mathcal{A}_C$ then c is both a prefix and a suffix of G_C and a prefix of $\hat{c}$. Also, since c' covers $\mathcal{A}_C(Y)$, then c' is both a prefix and a suffix of $\mathcal{A}_C(Y)$ and, therefore, $\hat{c}$ is both a prefix and a suffix of $\mathcal{A}_C$. Thus, c is also a suffix of $\hat{c}$. We get that the last complete occurrence of c, before r is at index $s + |c'| - |c|$. Now, any occurrence of c' in $\mathcal{A}_C(Y)$ before index r (including r) contradicts the maximality of s. Also, any occurrence of c' in index greater than r and at most $s + |c'|$ contradicts the choice of r as the first index to contradict the assumption due to the occurrence of c in index $s + |c'| - |c| < r$. Therefore, the next occurrence of c' in $\mathcal{A}_C(Y)$ is at index $s + |c'| + 1$. Thus, we have in $\mathcal{A}_C$ two consecutive occurrences of $\hat{c}$, where c is a suffix of the first occurrence (due to the occurrence of c in index $s + |c'| - |c|$) and a prefix of the second occurrence (due to the fact that c is a prefix of $\hat{c}$), and there is an occurrence of c overlapping to suffix of the first and the prefix of the second (due to the occurrence in in index r). However, this means that c is periodic, which contradicts the minimality of c by Lemma 15. This concludes the proof of the lemma. ◀

The following observations and lemma describe basic properties of the superimposition relation between tilings and the equation systems that tilings impose. Lemma 34 follows.

▶ **Observation 31.** *Any tiling L of the string created by an assignment $\mathcal{A}$ on the m-length free variable scheme C imposes an equation system E on the variables of C. Moreover, if L_1 is superimposed on L_2, then $E_1 \subseteq E_2$, where E_1 and E_2 are the equation systems imposed by L_1 and L_2, respectively.*

▶ **Observation 32.** *Assume that the strings $\mathcal{A}_C(Y_1)$ and $\mathcal{A}_C(Y_2)$ are both non-primitive, and let L_1 and L_2 be the tilings of the strings $\mathcal{A}_C(Y_1)$ and $\mathcal{A}_C(Y_2)$, respectively. Let E_1 and E_2 be the equation systems imposed by L_1 and L_2, respectively. Then, $\mathcal{A}_C(Y_1, Y_2)$ imposes the equation system $E = E_1 \cap E_2$ (which may be empty).*

We need the following lemma for the proof of Lemma 34 below.

▶ **Lemma 33.** *Assume that the strings $\mathcal{A}_C$, $\mathcal{A}_C(Y_1)$ and $\mathcal{A}_C(Y_2)$ are non-primitive, and let L, L_1 and L_2 be the tilings of the strings $\mathcal{A}_C$, $\mathcal{A}_C(Y_1)$ and $\mathcal{A}_C(Y_2)$, respectively. Assume that both L_1 and L_2 are superimposed on L. Then, L_1 is superimposed on L_2 or L_2 is superimposed on L_1.*

▶ **Lemma 34.** *If there exist variables $Y_1, \dots Y_k$, $k \geq 1$, in C such that $\mathcal{A}_C$, $\mathcal{A}_C(Y_i)$, for every i, $1 \leq i \leq k$, are non-primitive, then $\mathcal{A}_C(Y_1, \dots, Y_k)$ is also non-primitive. Moreover, let L and L' be the tilings of the covers in the strings $\mathcal{A}_C$ and $\mathcal{A}_C(Y_1, \dots, Y_k)$, respectively, then L is superimposed on L' or viceversa.*

Proof. The proof is by induction on k. The case $k = 1$ follows trivially. Let $Y_1, \dots Y_{k+1}$, $k \geq 1$, be variables in C such that $\mathcal{A}_C$, $\mathcal{A}_C(Y_i)$, for every i, $1 \leq i \leq k+1$, are non-primitive. By induction hypothesis, $\mathcal{A}_C(Y_1, \dots, Y_k)$ is also non-primitive. Moreover, let L and L' be the tilings of the covers c, c' in the strings $\mathcal{A}_C$ and $\mathcal{A}_C(Y_1, \dots, Y_k)$, respectively, then L is superimposed on L' or viceversa. Also, let L'' be the tiling of the cover c'' of the string $\mathcal{A}_C(Y_{k+1})$. By Lemma 30, we have that if $|c| \leq |c''|$ then L'' is superimposed on L, else, L is superimposed on L''.

Let E_c, $E_{c'}$ and $E_{c''}$ be the equation systems imposed, according to Observation 31, by the tilings L, L' and L'', respectively. There are four cases to consider:

1. If L' is superimposed on L and L'' is superimposed on L, then by Lemma 33, either L' is superimposed on L'' or viceversa. Therefore, either $E_{c'} \subseteq E_{c''} \subseteq E_c$ or $E_{c''} \subseteq E_{c'} \subseteq E_c$. Thus, by Observation 32, $\mathcal{A}_C(Y_1, \dots, Y_{k+1})$ imposes the equation system $E = E_{c'} \cap E_{c''} = E_{c'}$ or $E = E_{c'} \cap E_{c''} = E_{c''}$. Consequently, $\mathcal{A}_C(Y_1, \dots, Y_{k+1})$ is non-primitive. Moreover, we have that $\hat{L}$ is superimposed on L, where $\hat{L}$ is the tiling defined by E.
2. If L' is superimposed on L and L is superimposed on L'', then $E_{c'} \subseteq E_c$ and $E_c \subseteq E_{c''}$. Therefore, $E_{c'} \subseteq E_{c''}$. Thus, by Observation 32, $\mathcal{A}_C(Y_1, \dots, Y_{k+1})$ imposes the equation system $E = E_{c'} \cap E_{c''} = E_{c'}$. Consequently, $\mathcal{A}_C(Y_1, \dots, Y_{k+1})$ is non-primitive. Moreover, we have that $\hat{L}$ is superimposed on L, where $\hat{L}$ is the tiling defined by E.
3. If L is superimposed on L' and L'' is superimposed on L, then $E_c \subseteq E_{c'}$ and $E_{c''} \subseteq E_c$. Therefore, $E_{c''} \subseteq E_{c'}$. Thus, by Observation 32, $\mathcal{A}_C(Y_1, \dots, Y_{k+1})$ imposes the equation system $E = E_{c'} \cap E_{c''} = E_{c''}$. Consequently, $\mathcal{A}_C(Y_1, \dots, Y_{k+1})$ is non-primitive. Moreover, we have that $\hat{L}$ is superimposed on L, where $\hat{L}$ is the tiling defined by E.
4. If L is superimposed on L' and L is superimposed on L'', then $E_c \subseteq E_{c'}$ and $E_c \subseteq E_{c''}$. Thus, by Observation 32, $\mathcal{A}_C(Y_1, \dots, Y_{k+1})$ imposes the equation system $E = E_{c'} \cap E_{c''} \supseteq E_c$. Consequently, $\mathcal{A}_C(Y_1, \dots, Y_{k+1})$ is non-primitive. Moreover, we have that L is superimposed on $\hat{L}$, where $\hat{L}$ is the tiling defined by E.

This concludes the proof of the lemma. ◀

Theorem 35 follows.

▶ **Theorem 35.** [**Quasi-Periodicity Persistence Theorem**]
Let S be a non-degenerate quasi-periodic string. Let C be the free variable scheme of S, $\mathcal{A}$ be the assignment on the variables of C such that $\mathcal{A}_C = S$, and

$$V = \{Y \in C |\ \ Y \ \ appears \ \ k(Y) = 2^i \ \ times, \ \ i \geq 2, \ \ and \ \ \mathcal{A}_C(Y) \ \ is \ \ quasi-periodic\}.$$

Let $V' \subseteq V$ and let S' be the string obtained from S by $k = \sum_{Y \in V'} k(Y)$ substitution errors in positions where the variables $Y \in V'$ appear in C by an assignment $\mathcal{A}_C(V')$, then S' is quasi-periodic.

4 Application: Closing the Complexity Gap in ACP Relaxations Study

In this section we apply the analysis of quasi-periodicity persistence described in Section 3, to study the full-tiling relaxation of the approximate cover problem, in which we are given a retained tiling of the cover before the errors has occurred together with the input string

itself. This relaxation was first introduced and studied in [5] and its complexity remained open. Proving the correctness of this algorithm was beyond the reach of current research of quasi-periodicity in the presence of mismatch errors prior to this paper. The results of Section 3 enable proving its correctness, thus showing that the full-tiling relaxation of the ACP is polynomial time computable, which closes the gap in understanding the complexity hierarchy of ACP relaxations, presented in [5]. In order to formally define the relaxation we need Definition 36.

▶ **Definition 36.** Let T be an n-length string over alphabet Σ and let L be a valid tiling of T. Let $m = n + 1 - L_{last}$, where L_{last} is the last index in the tiling L. Then, an *L-approximate cover of T* is a primitive string C such that for every string C' of length m over Σ, $H(S_L(C'), T) \geq H(S_L(C), T)$, where H is the Hamming distance of the given strings.
$\min_{C \in \Sigma^m} H(S_L(C), T)$ is the number of errors of an L approximate cover of T.

The formal definition of the full-tiling relaxation of the ACP is given below.

▶ **Definition 37** (The Full-Tiling Relaxation of the ACP)**.** *INPUT:* String T of length n over alphabet Σ, and a valid tiling L of T.
OUTPUT: An L-approximate cover C of T.

[5] suggest a polynomial-time algorithm for the full-tiling relaxation of the approximate cover problem in two parts. The algorithm has a mandatory part, called the Histogram Greedy Algorithm. This algorithm does the main work in finding an approximate cover subject to the tiling L. It returns a candidate for the final L approximate cover to be output. This candidate is legal if it is primitive and illegal, otherwise. In the latter case, a second part of the algorithm is needed: the Full-Tiling Primitivity Coercion. In this part, the legality of the candidate is checked, and if needed, the candidate is corrected in order to coerce the primitivity requirement. In order to give a self-contained presentation of our results, we give a description of the Histogram Greedy Algorithm in Subsection 4.1 and the Full-Tiling Primitivity Coercion Algorithm in Subsection 4.2. We then complete its analysis in Subsection 4.3.

4.1 The Histogram Greedy Algorithm

This part of the algorithm performs the following steps given the text T and the valid tiling L:

1. Find m, the length of an approximate cover subject to the tiling L, by computing the difference between $n + 1$, and the last index in the tiling L, L_{last}, which indicates the last occurrence of the cover in T.
2. Compute the m-length mask M of an approximate cover, by initializing M to zeroes, setting $M[1] = 1$, then reading the tiling L from beginning to end and for each $i_k, i_{k+1} \in L$ setting $M[i_{k+1} - i_k] = 1$.
3. Compute the m-long string V_C of variables from an auxiliary alphabet

$$\Sigma_V = \{v_1, v_2, \ldots, v_m\}.$$

First, we initialize the m-long string V_C to $v_1 v_2 \ldots v_m$. Then, we read the mask M from end to beginning, and for every j such that $M[j] = 1$, we update the string V_C by equalizing the substrings $V_C[1..m - j + 1]$ and $V_C[j..m]$. In the equalization process, when we obtain an equation $v_k = v_\ell$ for $k < \ell$, we replace both letters by v_k. The resulting

string V_C represents C in the following sense: for any pair of indices $1 \leq i < j \leq m$, if $V_C[i] = V_C[j]$ then $C[i] = C[j]$. However, it can be that $V_C[i] \neq V_C[j]$, while $C[i] = C[j]$. In other words, V_C carries the information on equalities imposed by the mask M between indices of C.

4. Compute the string of length n, V_T, with variables from the auxiliary alphabet Σ_V, which is a string covered by V_C according to the tiling L of T. V_C is computed using the tiling L and V_C as follows: it begins with a copy of V_C and for each index i in L a new copy of V_C is concatenated starting from index i of V_T (maybe running over a suffix of the last copy of V_C).
5. Compute the histogram $Hist_{V_C,\Sigma}$ using the alignment of T with V_T and counting for each variable $V \in V_C$ and each $\sigma \in \Sigma$, the number of indices i in T, V_T for which $V_T[i] = V$ and $T[i] = \sigma$.
6. Compute an L-approximate cover candidate C greedily according to the histogram $Hist_{V_C,\Sigma}$, as follows: for every index $1 \leq i \leq m$, set $C[i] = \sigma_0$, where $Hist_{V_C,\Sigma}[V_C[i], \sigma_0] = \max_{\sigma \in \Sigma} Hist_{V_C,\Sigma}[V_C[i], \sigma]$, i.e., for each index in C we choose the alphabet symbol that minimizes the number of mismatch errors between $S_L(C)$ and T in the relevant indices according to the tiling L.

The algorithm outputs the m-length string C from its last step and the histogram table $Hist_{V_C,\Sigma}$.

As discussed in [5], the output C of the Histogram Greedy algorithm might not be an L-approximate cover of T, because it might not be primitive, as the following example shows.

Example: Assume that $V_C = XYZWXY$ and $\Sigma = \{a, b\}$ and that the histogram $Hist_{V_C,\Sigma}$ computed by the algorithm is the following:

$V_C \setminus \Sigma$	a	b
X	4	1
Y	2	3
Z	2	1
W	0	3

Then, the Histogram Greedy algorithm chooses: $X = a$, $Y = b$, $Z = a$, $W = b$, and outputs $C = ababab$, which cannot be considered a legal cover since it is not primitive, i.e., C itself can be covered by the shorter string ab. Note, that the input tiling L requires an m-length string as an output. Therefore, the (primitive) 2-length approximate cover ab is precluded as an L-approximate cover. Assuming that the input tiling L is the retained tiling of the cover of the original text before the errors occurred, such a case means that, though ab is a string covering T subject to a partial tiling L with the least number of errors, it does not cover T with L as a full tiling. In this sense, L is an evidence that the original cover is of larger length than ab and that more errors actually happened.

In order to impose the requirement of the definition of an L-approximate cover of T to be a primitive string such that all its repetitions to cover T (with minimum number of errors) are marked in the tiling L, we need a primitivity coercion algorithm. This algorithm was suggested by [5], and is described in Subsection 4.2.

4.2 The Full-Tiling Primitivity Coercion Algorithm

This part of the algorithm gets as input the string C returned by the Histogram Greedy algorithm (Subsection 4.1) and performs the following steps:

1. Check the primitivity of C (using the linear-time algorithm of [9]). If C is primitive, return C.
2. Else, find $V_k \in V_C$ such that if the assignment of V_k is changed from the symbol with the largest value in the row of V_k in $Hist_{V_C,\Sigma}$ to the symbol with the second largest value in this row, thus obtaining a new m-length candidate string C', such that the difference $H(S_L(C'), T) - H(S_L(C), T)$ is minimized and where C' is primitive.

Lemma 38 below describes the time complexity of the Full-Tiling Primitivity Coercion algorithm and immediately follows from the linear-time complexity of the algorithm [9] used in the first step and the description of the second step.

▶ **Lemma 38** ([5]). *The time complexity of the Full-Tiling Primitivity Coercion algorithm is* $O(|\Sigma| \cdot m)$.

The following subsection is devoted to proving the correctness of the Full-Tiling Primitivity Coercion algorithm, thus proving that the full-tiling relaxation of the ACP is polynomial-time computable.

4.3 Correctness of the Full-Tiling Primitivity Coercion Algorithm

We begin by noting that the structure of the string of variables created by the Histogram Greedy algorithm has the free variable scheme, as defined by Lemma 24.

If the cover generated by the assignment of the Histogram Greedy algorithm to this scheme is not primitive, then Corollary 20 guarantees that changing the value of any free variable in β results in a primitive cover. The problem is that this may not necessarily be the cover with minimum cost. Checking, for every single free variable, if changing it for the alphabet symbol with the second largest value in the histogram results in a primitive cover, and choosing the cover that generates the smallest Hamming distance, will indeed guarantee that we have a primitive cover with the smallest Hamming distance that results from changing a **single** variable. We need to show that it is impossible to get a primitive cover that generates an even smaller Hamming distance, by choosing the alphabet symbol with the second highest histogram in a **set** of free variables. We prove that this situation can not happen.

Note that if C is the free variable scheme generated be the Histogram Greedy algorithm. Then $\mathcal{A}_C$ can be the string created by the assignment of this algorithm to the variables of C, and $\mathcal{A}_C(X_{i_1}, ..., X_{i_j})$ can be the string created by assigning the variables $X_{i_1}, ...X_{i_j}$ of C an alphabet symbol whose histogram value is *second highest*, and all other variables get assigned an alphabet symbol whose histogram value is the highest.

Theorem 39 follows.

▶ **Theorem 39.** *Given a text T of length n over alphabet Σ and a valid tiling L. Let L_{last} be the last index in L. Then, the full-tiling relaxation of the approximate cover problem of T can be solved in $O(\Sigma \cdot m + n)$ time, where $m = n + 1 - L_{last}$.*

Proof. First, note that an L-approximate cover of T must have length $m = n + 1 - L_{last}$, where L_{last} is the last index in L. If the string C' returned by the Histogram greedy algorithm is primitive, then C' is an L-approximate cover of T since by its construction, it is the m-length primitive string such that its n-length tiled string according to the given tiling L, $S_L(C')$ has the minimum Hamming distance from T. In this case C' is also the string C returned by the Full-Tiling Primitivity Coercion algorithm.

Assume then that C' is not a primitive string and, therefore, the second step of the Full-Tiling Primitivity Coercion algorithm is performed. By Lemma 27, a change in a variable that appear once in the β or α parts of the free variable form of C' results in a primitive m-length string. Lemma 30 characterizes C' in case there exists a variable such that changing its assignment does not yield a primitive string. Note that by Lemma 34 taking a set of such variables and changing their assignment also does not yield a primitive string. Therefore, any set of variables such that changing their assignment yields a primitive string, necessarily contains a variable that changing its assignment only is enough to yield a primitive string. However, it is obvious that changing this variable only gives a primitive string that covers the input string with less mismatch errors.

The second step of the Full-tiling Primitivity Coercion algorithm chooses a character that minimizes the difference $H(S_L(C'), T) - H(S_L(C), T)$. Therefore, the resulting m-length string C is the m-length primitive string such that its n-length tiled string according to the given tiling L, $S_L(C)$ has the minimum Hamming distance from T. ◀

References

1 A. Amir, E. Eisenberg, and A. Levy. Approximate periodicity. In *Proc. ISAAC 2010*, LNCS 6506, pages 25–36. Springer, 2010.

2 A. Amir, E. Eisenberg, A. Levy, E. Porat, and N. Shapira. Cycle detection and correction. *ACM Transactions on Algorithms*, 9(1)(13), 2012.

3 A. Amir, C. S. Iliopoulos, and J. Radoszewski. Two strings at hamming distance 1 cannot be both quasiperiodic. *Information Processing Letters*, 128:54–57, 2017.

4 A. Amir, A. Levy, M. Lewenstein, R. Lubin, and B. Porat. Can we recover the cover? In *Proc. CPM*, 2017.

5 A. Amir, A. Levy, R. Lubin, and E. Porat. Approximate cover of strings. In *Proc. CPM*, 2017.

6 P. Antoniou, M. Crochemore, C. S. Iliopoulos, I. Jayasekera, and G. M. Landau. Conservative string covering of indeterminate strings. In *Proc. Stringology*, pages 108–115, 2008.

7 A. Aposolico and A. Ehrenfeucht. Efficient detection of quasiperiodicities in strings. *Theoret. Comput. Sci.*, 119:247–265, 1993.

8 A. Apostolico and D. Breslauer. Of periods, quasiperiods, repetitions and covers. In *Proc. Structures in Logic and Computer Science*, LNCS 1261, pages 236–248, 1997.

9 A. Apostolico, M. Farach, and C. S. Iliopoulos. Optimal superprimitivity testing for strings. *Information Processing Letters*, 39:17–20, 1991.

10 D. Breslauer. An on-line string superprimitivity test. *Information Processing Letters*, 44:345–347, 1992.

11 D. Breslauer. Testing string superprimitivity in parallel. *Information Processing Letters*, 49(5):235–241, 1994.

12 M. Christodoulakis, C. S. Iliopoulos, K. Park, and J. S. Sim. Approximate seeds of strings. *Journal of Automata, Languages and Combinatorics*, 10:609–626, 2005.

13 T. Crawford, C. S. Iliopoulos, and R. Raman. String matching techniques for musical similarity and melodic recognition. *Comput. Musicol.*, 11:73–100, 1998.

14 M. Crochemore, C. S. Iliopoulos, S. P. Pissis, and G. Tischler. Cover array string reconstruction. In *Proc. CPM*, pages 251–259, 2010.

15 M. Crochemore, C. S. Iliopoulos, and H. Yu. Algorithms for computing evolutionary chains in molecular and musical sequences. In *Proc. 9th Austral. Workshop on Combinatorial Algorithms*, pages 172–185, 1998.

16 T. Flouri, C. S. Iliopoulos, T. Kociumaka, S. P. Pissis, S. J. Puglisi, W. F. Smyth, and W. Tyczynski. Enhanced string covering. *Theor. Comput. Sci.*, 506:102–114, 2013.

17 O. Guth and B. Melichar. *Using Finite Automata Approach for Searching Approximate Seeds of Strings*, pages 347–360. Springer, 2010.

18 C. S. Iliopoulos and L. Mouchard. Quasiperiodicity and string covering. *Theor. Comput. Sci.*, 218(1):205–216, 1999.

19 C. S. Iliopoulos and W. F. Smyth. An on-line algorithm of computing a minimum set of k-covers of a string. In *Proc. 9th Australasian Workshop on Combinatorial Algorithms (AWOCA)*, pages 97–106, 1998.

20 C. S. Iliopoulus, D. W. G. Moore, and K. Park. Covering a string. *Algorithmica*, 16(3):288–297, 1996.

21 T. Kociumaka, S. P. Pissis, J. Radoszewski, W. Rytter, and T. Walen. Fast algorithm for partial covers in words. In *Proc. CPM*, pages 177–188, 2013.

22 R. M. Kolpakov and G. Kucherov. Finding approximate repetitions under hamming distance. *Theor. Comput. Sci.*, 303:135–156, 2003.

23 G. M. Landau and J. P. Schmidt. An algorithm for approximate tandem repeats. In *Proc. 4th Symp. Combinatorial Pattern Matching*, LNCS 648, pages 120–133, 1993.

24 G. M. Landau, J. P. Schmidt, and D. Sokol. An algorithm for approximate tandem repeats. *J. of Computational Biology*, 8(1):1–18, 2001.

25 Y. Li and W. F. Smyth. Computing the cover array in linear time. *Algorithmica*, 32(1):95–106, 2002.

26 M. Lothaire. *Combinatorics on words*. Addison-Wesley, 1983.

27 D. Moore and W. F. Smyth. An optimal algorithm to compute all the covers of a string. *Information Processing Letters*, 50(5):239–246, 1994.

28 D. Moore and W. F. Smyth. A correction to: An optimal algorithm to compute all the covers of a string. *Information Processing Letters*, 54:101–103, 1995.

29 B. Melichar O. Guth and M. Balik. *All Approximate Covers and Their Distance using Finite Automata*, pages 21–26. CEUR-WS, 2009.

30 J. S. Sim, C. S. Iliopoulos, K. Park, and W. F. Smyth. Approximate periods of strings. *Theor. Comput. Sci.*, 262:557–568, 2001.

31 W. F. Smyth. Repetitive perhaps, but certainly not boring. *Theor. Comput. Sci.*, 249(2):343–355, 2000.

32 H. Zhang, Q. Guo, and C. S. Iliopoulos. Algorithms for computing the lambda-regularities in strings. *Fundam. Inform.*, 84(1):33–49, 2008.

33 H. Zhang, Q. Guo, and C. S. Iliopoulos. Varieties of regularities in weighted sequences. In *Proc. AAIM*, LNCS 6142, pages 271–280, 2010.