Can We Recover the Cover?

Amihood Amir[1], Avivit Levy[2], Moshe Lewenstein[3], Ronit Lubin[4], and Benny Porat[5]

1 **Bar-Ilan University, Ramat Gan, Israel; and Johns Hopkins University, Baltimore, MD, USA**
amir@cs.biu.ac.il
2 **Shenkar College, Ramat Gan, Israel**
avivitlevy@shenkar.ac.il
3 **Bar-Ilan University, Ramat Gan, Israel**
moshe.lewenstein@gmail.com
4 **Bar-Ilan University, Ramat Gan, Israel**
ronit.moldovan@gmail.com
5 **Bar-Ilan University, Ramat Gan, Israel**
bennyporat@gmail.com

Abstract

Data analysis typically involves *error recovery* and *detection of regularities* as two different key tasks. In this paper we show that there are data types for which these two tasks can be powerfully combined. A common notion of regularity in strings is that of *a cover*. Data describing measures of a natural coverable phenomenon may be corrupted by errors caused by the measurement process, or by the inexact features of the phenomenon itself. Due to this reason, different variants of approximate covers have been introduced, some of which are $\mathcal{NP}$-hard to compute. In this paper we assume that the Hamming distance metric measures the amount of corruption experienced, and study the problem of recovering the correct cover from data corrupted by mismatch errors, formally defined as the *cover recovery problem (CRP)*. We show that for the Hamming distance metric, coverability is a powerful property allowing *detecting* the original cover and *correcting* the data, under suitable conditions.

We also study a relaxation of another problem, which is called the *approximate cover problem (ACP)*. Since the ACP is proved to be $\mathcal{NP}$-hard [5], we study a relaxation, which we call *the candidate-relaxation of the ACP*, and show it has a polynomial time complexity. As a result, we get that the ACP also has a polynomial time complexity in many practical situations. An important application of our ACP relaxation study is also a polynomial time algorithm for the cover recovery problem (CRP).

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems, G.2.1 Combinatorics, G.4 Mathematical Software, I.5.2 Design Methodology

Keywords and phrases periodicity, quasi-periodicity, cover, approximate cover, data recovery

Digital Object Identifier 10.4230/LIPIcs.CPM.2017.25

1 Introduction

Data analysis typically involves error recovery and detection of regularities as two different key tasks. In this paper we show that there are data types for which these two tasks can be powerfully combined. A classical tool for handling data recovery is through the use of error correcting codes. Error correcting codes are an invaluable method of adding redundancy to data so that the initial data can be recovered even after the introduction of a bounded number of errors. Errors in raw natural data with no prior knowledge of its structure are

usually considered beyond the feasible scope of recovery. Nonetheless, it was recently [4] shown, that data regularity, even if its structure is unknown a-priori, can serve as an aid to error recovery.

Regularities in strings arise in various areas of science, including coding and automata theory, formal language theory, combinatorics, molecular biology and many others. 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 [28]).

1.1 Regularities and Data Recovery

Recently, it was shown [4] that periodicity can serve as an aid to error recovery. It was proven that if no more than $\frac{n}{(1+\epsilon)p}$ mismatch errors are introduced to a periodic string of length n having period of length p then, even if p is not known a-priori, it is possible to recover $O(\log n)$ possible candidates, one of which is *guaranteed* to be the original period. This surprising result was further reinforced by discovering that a similar result holds not just for mismatch error corruptions bounded by the Hamming distance, but for any errors bounded by a pseudo local metric (e.g. the swap or interchange metrics). An interesting additional result was that even under some non-pseudo local metrics, such as the edit distance, periodicity can still allow recovery of $O(\log n)$ candidate periods [4, 2]. However, these candidate periods are distinguished in that none are cyclic rotations of each other. In other words, if we take one representative of all candidates that are cyclic rotations of each other, we end up with the small number of candidates. It was unknown whether there are other regularities in natural phenomena that allow recovery of the original string. Identifying such a type of regularity is the first topic of this paper.

In particular, for many phenomena, it is desirable to broaden the definition of periodicity and study wider classes of repetitive patterns in strings. One common such notion is that of a *cover*, 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 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. We study error correction feasibility for coverable phenomena.

1.2 Related Work

We review related regularity types and other approaches to handle errors in regularities. Quasi-periodicity was introduced by Ehrenfeucht in 1990 (according to [7]). 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], [29, 30], [27], [11]). An overview on the first decade of the research on covers can be found in the surveys [7, 20, 32].

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. Due to this reason, different variants

of quasi-periodicity have been introduced. These include *seeds* [19], *maximal quasi-periodic substring* [8], the notion of *k-covers* [21], *λ-cover* [33], *enhanced covers* [16], *partial cover* [23]. Since the notion of a seed is necessary to our study and presentation of results, 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 first and last occurrence of the seed C in T may 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* [34]. Some of the related problems are $\mathcal{NP}$-hard (see e.g., [6, 12, 23]).

In applications such as molecular biology and computer-assisted music analysis, finding exact repetitions and covers is not always sufficient. 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 [25, 26] 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, 18, 17]. Indeed, all these definitions along with other ones were proposed and studied (see [3, 24, 31]). Yet another possibility is that all repeats must be equal, but we allow a fixed total number of mismatches. The possibility presented in [3] 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. Therefore, a (smallest) repeat generating a string with the minimum total number of mismatches with the input string is sought. Extension of this definition approach to approximate covers is another topic of this paper.

1.3 Our Results

In this paper we show that *coverability* is also a tool that allows error correction. We formally define the *Cover Recovery Problem (CRP)* and characterize the feasibility of its solution. In particular, we show:

▶ **Theorem 3.** *Let S be a string coverable by a cover C of length c, and let $\varepsilon > 0$. Assume that at most $\frac{n}{(2+\varepsilon)c}$ mismatch errors were introduced to S resulting in a string S'. Then there exist $O(\log n)$ possible primitive substrings of S', one of which is guaranteed to be C or a seed of C.*

In addition, extending the approach of [3] to the notion of covers, [5] define the *approximate cover problem (ACP)*, in which we are given a text that is a sequence of some cover repetitions with possible mismatch errors. Since the ACP is proved to be $\mathcal{NP}$-hard [5], we study a relaxation of this problem. In our relaxation, which we call *the candidate relaxation of the ACP*, a candidate cover is also given, and we seek to align it with the given text (this alignment is called *a tiling*) such that the number of mismatches is minimized. This scenario is quite realistic in the case where a cover is sought for a string where the errors are distributed in a manner that at least one occurrence of the cover appears in the string without errors. We examine this relaxation and show it has polynomial time complexity. As a result, we get that the ACP also has polynomial time complexity in many practical situations. This ACP relaxation study enables also an efficient algorithm for recovering the candidate covers in CRP.

Paper Contributions. The main contributions of this paper are:

- Proving that recovery of raw data from errors is possible not only for periodic phenomena but also for the less rigid coverable phenomena.
- Demonstrating that efficient recovery is feasible even when the underlying problem of computing an approximate cover is $\mathcal{NP}$-hard. This is in line with the previous result of [4] that show efficient recovery for the interchange metric, which is $\mathcal{NP}$-hard to compute.
- Formalizing the candidate relaxation of the ACP and showing it is polynomial time computable. This study served both to give a solution to the CRP and to suggest an efficient solution for the ACP in many practical situations.

The paper is organized as follows. In Section 2, we give formal definitions and basic lemmas. In Section 3, we study the cover recovery problem (CRP) and characterize the extent to which the cover of the unknown uncorrupted original string can be recovered given the possibly corrupted by mismatch errors input string. In Section 4, we study the candidate relaxation of the ACP with its application to the ACP itself and, more importantly, to the CRP. We conclude with some open problems in Section 5.

2 Preliminaries

In this section we give the needed formal definitions and basic lemmas.

▶ **Definition 4** (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 have been introduced to 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 5** (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 1.** *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 every different valid tiling of the m-length string C generates a different n-length string $S(C)$. A unique version can be obtained if a valid tiling L is also given.

▶ **Notation 2.** *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 $S_L(C')$ (running over a suffix of the last copy of C' if the difference between i and the previous index in L is less than m).*

▶ **Definition 6.** 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 7.** Let T be an n-long string 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 8** (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 9.** 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.

The goal of the following definition and lemmas is Lemma 15, which is a crucial tool for the efficiency of the candidate relaxation algorithm.

▶ **Definition 10** (String Mask)**.** Given a string C of length m, the mask M of C is a boolean array of length m, such that $M[i]=1$ if and only if the suffix $C[i..m]$ is equal to the prefix $C[1..m-i+1]$.

▶ **Lemma 11.** *Let C be a string of length m and let M be its mask. Let i,j be indices such that $1\leq i<j\leq m$ and $M[i]=M[j]=1$, then the substring $C[i..m]$ has a period of length $j-i$.*

▶ **Lemma 12.** *Let C be a primitive string of length m and let M be its mask. Let i be the smallest index such that $1<i\leq m$ and $M[i]=1$, then $i>\lfloor\frac{m}{2}\rfloor+1$.*

▶ **Lemma 13.** *Let C be a string of length m and let M be its mask. Let i,j be indices such that $M[i]=M[j]=1$, $j-i=g>0$. Let k be the minimal index such that $j<k\leq m$ and $M[k]=1$. Then, $k=j+g$ or $k\geq j+\lfloor\frac{g}{2}\rfloor$.*

▶ **Lemma 14.** *Let C be a string of length m and let M be its mask. Let i,j,k,ℓ be indices such that $i<j$, $k<\ell$, $M[i]=M[j]=M[k]=M[\ell]=1$ and $j-i=\ell-k$ then $C[i..j-1]=C[k..\ell-1]$.*

▶ **Lemma 15.** *Let C be a primitive string of length m and let M be its mask. Let I_M be the sorted list of indices i such that $1 \leq i \leq m$ and $M[i] = 1$. Let $S_C = \{C[i_k..i_{k+1} - 1] \mid i_k, i_{k+1}$ are adjacent indices in $I_M\} \cup \{C[i_{last}..m] \mid i_{last} = \max_{i_k \in I_M} i_k\}$ be a set of substrings of C. Then, $|S_C| = O(\log m)$.*

3 Characterization of the Cover Recovery Problem Approximation

In this section we study the Cover Recovery Problem (CRP) and characterize the extent to which the cover of the original unknown uncorrupted original string can be recovered given the possibly corrupted by mismatch errors input string. The term *approximation* here refer to the ability to give a relatively small size set of candidates that *includes* the *exact* cover of the original string or a seed of it. We begin with a formal definition of the CRP problem.

▶ **Definition 16** (The Cover Recovery Problem)**.**
INPUT: An $\varepsilon > 0$ and a string S' of length n over alphabet Σ, which is a string S covered by the primitive cover C possibly corrupted by at most $\frac{n}{(2+\varepsilon)c}$ mismatch errors, where c is the length of C.
OUTPUT: A small size set O of candidate strings such that $C \in O$.

First, we show the bounds on the number of errors that still guarantees a small-size set O of candidates. We then prove a bound on the size of this set O. In Section 4 we then conclude how this set can be identified, and thus the original uncorrupted string can be *approximately* recovered. Some more formal definitions and lemmas are needed. We start with the definitions of *alignment* and *neighbourhood* that we use to prove the bound on the number of errors that still enable a recovery.

▶ Remark. Throughout this section we use c to denote a cover length and C the cover string, i.e., $c = |C|$.

▶ **Definition 17.** Let $S = S[1], \ldots, S[s]$ and $T = T[1], \ldots, T[t]$ be strings, and let $1 \leq i \leq |T|$. The *alignment of S with T in location i* is the comparison of $S[j]$ and $T[i + j - 1], \forall j = 1, \ldots, \min(s, t - i + 1)$. In other words, we place S above T such that the first location of S is aligned with the i-th location of T.

▶ **Definition 18.** Let $C = C[1], \ldots, C[c]$ be a primitive cover, and let $1 \leq i \leq c$. We call i a *neighbouring index* of C if $\forall j,\ j = 1, \ldots, c - i$, we have $C[i + j] = C[j]$. For any neighbouring index i, denote by $C \circ_i C$ the string composed of the prefix of length i of C concatenated by C. We call $C \circ_i C$ *the neighbourhood of C at index i*. In particular, if $i = c$ then $C \circ_i C$ is C^2, the concatenation of C with itself.

If we are interested in a neighbourhood of C where the location is not important, we will denote it by $C \circ C$.

Lemma 19 is the basic building block in our error bound proof.

▶ **Lemma 19.** *Let C be a primitive cover and $C \circ_i C$ be a neighbourhood of C at location i. Then for every $j \neq i$, $1 < j \leq c$, the alignment of C with $C \circ_i C$ in location j has at least one mismatch.*

Proof. Because C is a primitive cover, then $i > c/2$, by Lemma 12. If $1 < j \leq c/2$ then an exact alignment leads to non-primitivity of C, contradiction. However, if there is an exact alignment for $c/2 < j \neq i$, then $|j - i| < c/2$ and thus we again have a contradiction to the primitivity of C. Therefore, there must be at least one mismatch in an alignment at any index $j \neq i$. ◀

We make use of following lemma for proving the upper bound on the number of candidates in our output set.

▶ **Lemma 20.** *Let S and C be two primitive strings such that C is a seed of S. Then there is at most one string S' with the following properties:*
1. *S' is covered by C.*
2. *S is a substring of S'*
3. *S' is the shortest string with properties 1 and 2 above.*

Proof. Assume there are two such strings, S' and S''. Since they are both shortest possible superstrings of S (i.e., strings containing S as a substring), then S matches each of them in their first occurrence of C. If $S' \neq S''$ then there must be at least one index i in S where C starts in S' but not in S''. However, then by Lemma 19 there must be at least one mismatch in the alignment of at least one of them with S, contradiction to the fact that S is a substring of both of them. ◀

▶ **Lemma 21.** *Let $n \in \mathbb{N}$ and let S_1, S_2 be two n-long coverable strings with C_1 and C_2 the covers of S_1 and S_2 respectively, where $c_1 \geq c_2$ and C_2 is not a seed of C_1. Then*

$$H(S_1, S_2) \geq \frac{n}{c_1}\,.$$

We are now ready to prove our approximation bound for the CRP. Lemma 22 is needed for proving our characterization theorem.

▶ **Lemma 22.** *Let $\varepsilon > 0$ be a constant, S an n-long string, and C_1, C_2 are c_1 and c_2-length approximate seeds of S with at most $\frac{n}{(2+\varepsilon)\cdot c_1}$, $\frac{n}{(2+\varepsilon)\cdot c_2}$ errors respectively (w.l.o.g. assume that $c_1 \geq c_2$), where C_2 is not a seed of C_1. Then,*

$$c_1 \geq (1+\varepsilon)\cdot c_2$$

Proof. Let S_1 be the n-long string such that C_1 is its seed and S_2 be the n-long string such that C_2 is its seed. We are given that $H(S_1, S) \leq \frac{n}{(2+\varepsilon)\cdot c_1}$ and $H(S_2, S) \leq \frac{n}{(2+\varepsilon)\cdot c_2}$. Therefore,

$$\frac{n}{(2+\varepsilon)\cdot c_1} + \frac{n}{(2+\varepsilon)\cdot c_2} \geq H(S_1, S) + H(S_2, S)\,.$$

By triangle inequality we have,

$$H(S_1, S) + H(S_2, S) \geq H(S_1, S_2)\,.$$

By Lemma 21,

$$H(S_1, S_2) \geq \frac{n}{c_1}\,.$$

Therefore,

$$\frac{n}{(2+\varepsilon)\cdot c_1} + \frac{n}{(2+\varepsilon)\cdot c_2} \geq \frac{n}{c_1}$$

from which we get,

$$c_2 + c_1 \geq (2+\varepsilon)c_2$$

or,

$$c_1 \geq (1+\varepsilon)c_2.$$ ◀

We conclude with our characterization theorem, which is a more accurate version of Theorem 3.

▶ **Theorem 23.** *Let S be an n-long string. Then, there are at most $\log_{1+\varepsilon} n + 1$ different c-length approximate covers C of S with at most $\frac{n}{(2+\varepsilon)\cdot c}$ errors such that none is a seed of another.*

Proof. First, note that there cannot be two such different c-length approximate covers unless one is a seed of the other, because then, by Lemma 22, we get $c \geq (1+\varepsilon)c$, contradiction. Thus, such different c-length approximate covers must have different length. Now, let $1 \leq l_1 < l_2 < \ldots < l_{t-1} < l_t \leq n$ be the different lengths of c-length approximate covers of S. By Lemma 22,

$$(1+\varepsilon)^{t-1} \leq (1+\varepsilon)^{t-1} \cdot l_1 < (1+\varepsilon)^{t-2} \cdot l_2 < \ldots < (1+\varepsilon)^2 \cdot l_{t-2} < (1+\varepsilon) \cdot l_{t-1} < l_t \leq n$$

Therefore, $t - 1 \leq \log_{1+\varepsilon} n$. ◀

▶ **Example 24.** We now show an example where a string has many substrings that all cover the given string with two errors. However, all these substrings have a single shortest 2-error seed. Consider the string $S = aaaaaaaaa(baaaa)^k baaaaaaaaa$. Then, all the following primitive strings cover S with two errors: *aaaabaaaa*, *aaaabaaa*, *aaaabaa*, *aaaaba*, *aaabaaaa*, *aabaaaa*, *abaaaa*. They all have either *abaaaa* or *aaaaba* as a seed. Note that there are 2 such shortest 2-error covers, however, each is a seed of the other.

4 The Candidate Relaxation of the ACP

In this section we study the following relaxation of the approximate cover problem:

▶ **Definition 25** (The Candidate Relaxation of the ACP)**.**
INPUT: String T of length n over alphabet Σ, and a candidate cover C of length m over alphabet Σ.
OUTPUT: $\min_{S(C)\in\Sigma^n} H(S(C), T)$, i.e., the minimum number of errors in any valid tiling of C in T.

▶ Remark. If $k = \min_{S(C)\in\Sigma^n} H(S(C), T)$, we use the term k-error cover for the given C.

Note that, since a candidate cover must be primitive, we may assume that this is indeed the case. A linear-time verification is possible using the algorithm of [9]. We describe a dynamic programming algorithm for this problem, which uses the well-known Knuth-Morris-Pratt [22] and Abrahamson-Kosaraju [1] algorithms. Our algorithm consists of a preparation phase, and a dynamic programming phase. We denote by m^* the number of set bits in the mask M of the given candidate C.

4.1 The preparation phase

The preparation phase is composed of the following three stages:

1. **Computing the mask of C.** This computation can be performed efficiently using the KMP algorithm. We compute the "failure automaton" for C. Denote the states of the automaton by $s_0, s_1, s_2, \ldots, s_m$. We consider the final state s_m of the automaton, and follow the sequence of fail links that start from it. Assume that this sequence is s_m, s_{i_1}, s_{i_2}, etc. The first link in the sequence means that C_1, the longest proper prefix of C that is equal to the corresponding suffix, is of length i_1. The second link means that C_2, the

longest proper prefix of C_1 that is equal to the corresponding suffix of C_1, is of length i_2. However, C_2 is also the second longest prefix of C that is equal to the corresponding suffix of C. By continuing in this process, we obtain the sequence $C_1, C_2, \ldots$ of *all* prefixes of C that are equal to the corresponding suffixes. Hence, the corresponding sequence of lengths $i_1, i_2, \ldots$ gives the (decreasing) sequence of indices $j_\ell = m - i_\ell + 1$, for which $M[j_\ell] = 1$, where M is the mask of C.

2. **Dividing C into disjoint substrings.** We divide C into substrings according to the indices i for which $M[i] = 1$. Specifically, if the (increasing) sequence of indices i for which $M[i] = 1$ is $i_1, i_2, \ldots i_{m^*}$ where $1 = i_1 < i_2 < \ldots < i_{m^*}$, then the substrings we consider are all substrings of C of the form $s_j = C[i_j..i_{j+1} - 1]$, for $1 \leq j \leq m^* - 1$, along with the suffix $s_{m^*} = C[i_{m^*}..m]$.
3. **Computing the Hamming distance from substrings of T to the strings s_j.** For each string s_j, $1 \leq j \leq m^*$, we compute its Hamming distance to all substrings of T simultaneously using the Abrahamson-Kosaraju algorithm. Since for many values of j, s_j is equal to s_{j-1} (actually, by Lemma 15, the sequence $s_1, s_2, \ldots, s_{m^*}$ contains only $O(\log m)$ distinct elements), we first check whether $s_j = s_{j-1}$ and apply the Abrahamson-Kosaraju algorithm only in the rare cases of inequality. The array of Hamming distances returned by the Abrahamson-Kosaraju algorithm is denoted below by $Hamming(s_j, T)$.

4.2 The dynamic programming phase

When the preparation phase is done, we are ready to compute the minimal k such that C is a k-error cover of T. This computation is performed in a dynamic fashion. Namely, we go over all suffixes of T in an increasing order, and for each suffix $T[i..n]$, we compute the minimal $k(T[i..n])$ such that C is a $k(T[i..n])$-cover of $T[i..n]$, utilizing the computations performed for the previous suffixes. The values $k(T[i..n])$ are stored in an array MIN, where $MIN[i] = k(T[i..n])$. In the beginning of the algorithm, all values of MIN are initialized to ∞. The output of the algorithm is $MIN[1]$.

As a cover must be a suffix of the covered string, we have $MIN[i] = \infty$ for all $i > n-m+1$, meaning that there does not exist a string of length $n - i + 1$ that can be covered by C. For the same reason, $MIN[n - m + 1] = H(C, T[n - m + 1..n])$, as there is a unique way to cover a string of length m by C. Since any two overlapping occurrences of C in a tiling that covers the suffix $T[i..n]$ must differ by a value j such that $M[j + 1] = 1$, and since $|s_1| = \min(\{j : 1 < j \leq m, M[j + 1] = 1\})$, it is impossible to cover a string of length $m + j$, $1 \leq j < |s_1|$, by copies of C. Thus, $MIN[i] = \infty$ for all $n - m - |s_1| + 1 < i < n - m + 1$. The following steps are performed for all $i \leq n - m - |s_1|$, in a decreasing order.

For each such i, we go over all possible strings of length $n - i + 1$, $S_{L_i}(C)$ that cover $T[i..n]$ by C with k-errors (resulted from different tiling L_i for which its first index is aligned with index i in the text). As each such tiling must start with a copy of C, and as the second occurrence of C in this tiling must differ from the initial one either by m or by a value j such that $M[j + 1] = 1$, we can compute the minimal number of error in any such tiled strings $S_{L_{ij}}(C)$ (for which the first occurrence of C is aligned with index i in T and the second occurrence of C is index j) as $Error(S_{L_{ij}}(C)) = H(C[1..j], T[i..i + j - 1]) + MIN[i + j]$ (note that by the structure of the algorithm, $MIN[i + j]$ is already known at this stage.) The value $MIN[i]$ is given by:

$$MIN[i] = \min_{j \in \{j : M[j+1]=1\} \cup \{m\}} Error(S_{L_{ij}}(C)).$$

Naively, we can go over all m^* possible values of j, compute $Error(S_{L_{ij}}(C))$ for each of them, and find out the minimum. For the sake of efficiency, we compute these values incrementally,

by advancing the starting point of the second occurrence of C in the covering by $|s_j|$ every time. Formally, this is performed as follows.

We define a counter j that corresponds to the initial shift of the second occurrence of C in the tiling relative to the position i in T. j is initialized to 0. Then, for $\ell = 1, 2, \ldots, m^*$, we advance j by $|s_\ell|$ and check whether $H(C[1..j], T[i..i+j-1]) + MIN[i+j]$ for $j = \sum_{r=1}^{\ell} |s_r|$ is lower than the previously best value of $Error$. If the answer is positive, the temporary value of $MIN[i]$ is replaced by $H(C[1..j], T[i..i+j-1]) + MIN[i+j]$.

In order to compute the values $H(C[1..j], T[i..i+j-1])$ efficiently, we observe that for $j = \sum_{r=1}^{\ell} |s_r|$, we have

$$\begin{aligned} H(C[1..j], T[i..i+j-1]) &= H(s_1, T[i..i+|s_1|-1]) \\ &+ H(s_2, T[i+|s_1|..i+|s_1|+|s_2|-1]) + \ldots \\ &+ H(s_\ell, T[i+\sum_{r=1}^{\ell-1}|s_r|..i+\sum_{r=1}^{\ell-1}|s_r|]) \end{aligned}$$

Hence, we compute $H(C[1..j], T[i..i+j-1])$ incrementally by keeping a counter err, initializing it to 0, and advancing it by $H(s_\ell, T[i+\sum_{r=1}^{\ell-1}|s_r|..i+\sum_{r=1}^{\ell-1}|s_r|])$ when j is advanced by $|s_\ell|$. Finally, in order to skip unnecessary operations, for each ℓ we check whether $i+j+|s_\ell| \leq n-m+1$, as otherwise, an occurrence of C clearly cannot start at position $i+j$.

After going over $\ell = 1, 2, \ldots, m^*$, we fix the last temporary value $MIN[i]$ to be its final value, and proceed to $i-1$. As mentioned before, $MIN[1]$ is the output of the algorithm. A pseudo-code of the algorithm is presented in Figure 1.

The correctness of the Candidate Relaxation Dynamic Programming algorithm is given in Lemma 26. The complexity of the algorithm is given in Lemma 27.

▶ **Lemma 26.** *Let T be a length-n string and let C be a length-m cover. Let MIN be the final array obtained by the dynamic programming algorithm described above with input T and C. Then for any $1 \leq i \leq n$, $MIN[i]$ is equal to the minimal k such that C is a k-error cover of $T[i..n]$.*

Proof. The proof is by an inverse induction on i. The induction basis is the cases $i > n-m-|s_1|+1$, for which $MIN[i]$ was calculated explicitly above and is easily seen to be equal to their final value computed by the algorithm.

Assume that the claim holds for all $i > i_0$, and consider the case $i = i_0$. Let $S_{L_{i_0}}(C)$ be the tiled string of $T[i_0..n]$ by copies of C starting from index i_0, for which the minimal number of errors $k(T[i_0..n])$ is attained. The tiling $S_{L_{i_0}}(C)$ must start with a copy of C, and the second occurrence of C in $S_{L_{i_0}}(C)$ must differ from the initial one either by m or by a value j such that $M[j+1] = 1$. As the total error of $S_{L_{i_0}}(C)$ is $k(T[i_0..n])$, we have

$$k(T[i_0..n]) \geq H(C[1..j], T[i..i_0+j-1]) + k(T[i_0+j..n]).$$

On the other hand, by the structure of our algorithm, its outputs satisfy

$$MIN[i] \leq H(C[1..j], T[i..i_0+j-1]) + MIN[i_0+j] = H(C[1..j], T[i..i_0+j-1]) + k(T[i_0+j..n]),$$

where the equality holds by the induction assumption. Hence, $MIN[i_0] \leq k(T[i_0..n])$. Finally, since $MIN[i]$ is obtained in the algorithm by computing the error of a concrete cover (that can be traced inductively), it is clear that $MIN[i] \geq k(T[i_0..n])$. This completes the proof. ◀

THE CANDIDATE RELAXATION DYNAMIC PROGRAMMING ALGORITHM
Input: A string T of length n, and a candidate cover C of length m
find the mask M of C using the KMP algorithm
$start \leftarrow 1$
for $i \leftarrow 2$ to m do
    if $M[i] = 1$ then
        $s \leftarrow s \cup C[start..i-1]$
        $start \leftarrow i$
$s \leftarrow s \cup C[start..m]$
for each substring s_i do
    if $|s_i| = |s_{i-1}|$ then
        $Hamming(s_i, T) \leftarrow Hamming(s_{i-1}, T)$
    else
        $Hamming(s_i, T) \leftarrow Abrahamson - Kosaraju(s_i, T)$
for $i \leftarrow 1$ to n do
    $MIN[i] \leftarrow \infty$
$MIN[n-m+1] \leftarrow H(C, T[n-m+1..n])$
for $i \leftarrow n-m+1-|s_1|$ to 1 by -1 do
    $j \leftarrow 0$
    $err \leftarrow 0$
    for each substring s_ℓ do
        if $j + |s_\ell| \leq n-m$ then
            $err \leftarrow err + Hamming(s_\ell, T[i+j])$
            if $MIN[i] > err + MIN[i+j+|s_\ell|]$ then
                $MIN[i] \leftarrow err + MIN[i+j+|s_\ell|]$
            $j \leftarrow j + |s_\ell|$
Output:
$MIN[1]$

Figure 1 The dynamic programming algorithm for the candidate relaxation of the ACP.

▶ **Lemma 27.** *Let T be a text of length n and C a candidate cover of length m. Then, the time complexity of the Candidate Relaxation Dynamic Programming algorithm on T and C is $O(n \cdot m^* + n\sqrt{m \log m})$, where m^* is the number of set bits in the mask M of C.*

Proof. First, we analyze the preparation phase of the algorithm. As explained above in the description of the algorithm, computing the mask M of C can be done by running the KMP algorithm for C, which requires $O(m)$ operations. Dividing C into disjoint substrings given the mask M of C can clearly be done in $O(m)$ operations. Computing the Hamming distance from substrings of T to the strings s_j can be performed by applying the Abrahamson-Kosaraju algorithm once for each of the substrings s_j. As by Lemma 15, the number of distinct substrings s_j is $O(\log m)$, the Abrahamson-Kosaraju algorithm is applied only $O(\log m)$ times, while for the other values of j (whose total number is bounded from above by m) we perform only a simple "copy" operation. The complexity of each application of the Abrahamson-Kosaraju algorithm is $O(n\sqrt{m \log m})$, and hence, the total complexity of this step is $O(\log m \cdot n\sqrt{m \log m})$.

A refinement of the analysis of this computation shows that the complexity is actually $O(n\sqrt{m \log m})$. Note that the Abrahamson-Kosaraju algorithm is applied for distinct strings of the form s_j. Consider the lengths of these strings. By Lemma 15, if we denote $|s_k| = g_k$ and let h_k denote the distance from the end of s_k to the end of C, we have that whenever

$s_{k+1} \neq s_k$, either $g_{k+1} \leq g_k/2$ or $h_{k+1} \leq 3h_k/4$. Moreover, as the latter condition arises only in the case $h_k \leq 2g_k$ (see the proof of Lemma 15), it follows that the sequence of lengths $g_1 > g_2 > \ldots$ of strings on which the Abrahamson-Kosaraju algorithm is applied satisfies $g_{k+4} < g_k/4$. Since $g_1 \leq m$, the total complexity of this step is at most $O(n\sqrt{m \log m})$.

We now analyze the dynamic programming phase. The main loop of the dynamic programming is performed for all $1 \leq i \leq n - m - |s_1|$, i.e., $O(n)$ times. For each i, we go over the m^* strings s_j, and for each of them, we perform a few simple operations (i.e., table lookups and comparisons). Hence, the time complexity of this phase is $O(n \cdot m^*)$.

Therefore, the total time complexity of the algorithm is $O(n \cdot m^* + n\sqrt{m \log m})$. ◀

This completes the proof of Theorem 28.

▶ **Theorem 28.** *Given a text T of length n a candidate cover C of length m over alphabet Σ. Then, the candidate relaxation of the approximate cover problem of T can be solved in $O(n \cdot m^* + n\sqrt{m \log m})$ time, where m^* is the number of set bits in the mask M of C.*

Theorem 28 has the following useful applications to the ACP (Corollary 29) and CRP (Corollary 30).

▶ **Corollary 29.** *Let T be a text of length n over alphabet Σ. Denote by $\gamma(T)$ the maximum of $m^* + \sqrt{m \log m}$ over all primitive substrings C of T with length $m < n$, where m^* is the number of set bits in the mask M of C. Assume that the error distribution guarantees that at least one occurrence of an approximate cover of the text is without errors. Then, the approximate cover problem of T can be solved in $O(n^3 \cdot \gamma(T))$ time.*

Proof. The condition implies that C is a substring of T. Take each of the $O(n^2)$ primitive substrings of T of length less than n as a candidate cover in the algorithm and run the dynamic programming algorithm of Figure 1. The corollary then follows from Theorem 28. ◀

▶ **Corollary 30.** *Let S be a n-long string and $\varepsilon > 0$. Denote by $\gamma(S)$ the maximum of $m^* + \sqrt{m \log m}$ over all primitive substrings C of S with length $m < n$, where m^* is the number of set bits in the mask M of C. Then, a set of at most $\log_{1+\varepsilon} n$ different m-length approximate covers C of S such that none is a seed of another, each with at most $\frac{n}{(2+\varepsilon) \cdot m}$ errors, can be constructed in $O(n^3 \cdot \gamma(S))$ time.*

Proof. Use the same algorithm as in the proof of Corollary 29 but retain as candidates in the output set only m-length approximate covers C of S, for which the candidate relaxation algorithm finds at most $\frac{n}{(2+\varepsilon) \cdot m}$ errors. From this set retain only candidates that do not have a shorter or same length candidates as seeds. ◀

5 Open Problems

In this paper we initiated the study of the CRP as well as a new relaxation of the ACP. Some interesting questions and open problems are:

- Since the ACP is proved to be $\mathcal{NP}$-hard, it is interesting to find other polynomial time relaxations of the ACP, besides the candidate relaxation studied in this paper. Such a study will broaden our understanding as well as suggest practical solutions.
- In this paper we considered the Hamming distance as a metric in the definition of approximate cover. Other string metrics can be considered as well. It is interesting to see if and how the complexity of the problem changes with the use of other string metrics.

References

1. Karl R. Abrahamson. Generalized string matching. *SIAM J. Comput.*, 16(6):1039–1051, 1987. doi:10.1137/0216067.
2. Amihood Amir, Mika Amit, Gad M. Landau, and Dina Sokol. Period recovery over the Hamming and edit distances. In Evangelos Kranakis, Gonzalo Navarro, and Edgar Chávez, editors, *Proceedings of the 12th Latin American Symposium on Theoretical Informatics (LATIN 2016)*, volume 9644 of *LNCS*, pages 55–67. Springer, 2016. doi:10.1007/978-3-662-49529-2_5.
3. Amihood Amir, Estrella Eisenberg, and Avivit Levy. Approximate periodicity. In Otfried Cheong, Kyung-Yong Chwa, and Kunsoo Park, editors, *Proceedings of the 21st International Symposium on Algorithms and Computation (ISAAC 2010)*, volume 6506 of *LNCS*, pages 25–36. Springer, 2010. doi:10.1007/978-3-642-17517-6_5.
4. Amihood Amir, Estrella Eisenberg, Avivit Levy, Ely Porat, and Natalie Shapira. Cycle detection and correction. *ACM Trans. Algorithms*, 9(1):13:1–13:20, 2012. doi:10.1145/2390176.2390189.
5. Amihood Amir, Avivit Levy, Ronit Lubin, and Ely Porat. Approximate cover of strings. In Juha Kärkkäinen, Jakub Radoszewski, and Wojciech Rytter, editors, *Proceedings of the 28th Annual Symposium on Combinatorial Pattern Matching (CPM 2017)*, volume 78 of *LIPIcs*, pages 26:1–26:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.CPM.2017.26.
6. Pavlos Antoniou, Maxime Crochemore, Costas S. Iliopoulos, Inuka Jayasekera, and Gad M. Landau. Conservative string covering of indeterminate strings. In Jan Holub and Jan Zdárek, editors, *Proceedings of the Prague Stringology Conference (PSC 2008)*, pages 108–115. Czech Technical University in Prague, 2008. URL: http://www.stringology.org/event/2008/p10.html.
7. Alberto Apostolico and Dany Breslauer. Of periods, quasiperiods, repetitions and covers. In Jan Mycielski, Grzegorz Rozenberg, and Arto Salomaa, editors, *Structures in Logic and Computer Science: A Selection of Essays in Honor of Andrzej Ehrenfeucht*, volume 1261 of *LNCS*, pages 236–248. Springer, 1997. doi:10.1007/3-540-63246-8_14.
8. Alberto Apostolico and Andrzej Ehrenfeucht. Efficient detection of quasiperiodicities in strings. *Theor. Comput. Sci.*, 119(2):247–265, 1993. doi:10.1016/0304-3975(93)90159-Q.
9. Alberto Apostolico, Martin Farach, and Costas S. Iliopoulos. Optimal superprimitivity testing for strings. *Inf. Process. Lett.*, 39(1):17–20, 1991. doi:10.1016/0020-0190(91)90056-N.
10. Dany Breslauer. An on-line string superprimitivity test. *Inf. Process. Lett.*, 44(6):345–347, 1992. doi:10.1016/0020-0190(92)90111-8.
11. Dany Breslauer. Testing string superprimitivity in parallel. *Inf. Process. Lett.*, 49(5):235–241, 1994. doi:10.1016/0020-0190(94)90060-4.
12. Manolis Christodoulakis, Costas S. Iliopoulos, Kunsoo Park, and Jeong Seop Sim. Approximate seeds of strings. *J. Autom. Lang. Comb.*, 10(5/6):609–626, 2005.
13. Tim Crawford, Costas S. Iliopoulos, and Rajeev Raman. String-matching techniques for musical similarity and melodic recognition. In Walter B. Hewlett and Eleanor S. Field, editors, *Melodic Similarity: Concepts, Procedures, and Applications*, volume 11 of *Computing in Musicology*, pages 73–100. MIT Press, Cambridge, Massachusetts, 1998.
14. Maxime Crochemore, Costas S. Iliopoulos, Solon P. Pissis, and German Tischler. Cover array string reconstruction. In Amihood Amir and Laxmi Parida, editors, *Proceedings of the 21st Annual Symposium on Combinatorial Pattern Matching (CPM 2010)*, volume 6129 of *LNCS*, pages 251–259. Springer, 2010. doi:10.1007/978-3-642-13509-5_23.

15 Maxime Crochemore, Costas S. Iliopoulos, and Hiafeng Yu. Algorithms for computing evolutionary chains in molecular and musical sequences. In Costas S. Iliopoulos, editor, *Proceedings of the 9th Australian Workshop on Combinatorial Algorithms (AWOCA 1998)*, pages 172–184, France, 1998. URL: https://hal-upec-upem.archives-ouvertes.fr/hal-00619988/file/9807-EC.pdf.

16 Tomás Flouri, Costas S. Iliopoulos, Tomasz Kociumaka, Solon P. Pissis, Simon J. Puglisi, William F. Smyth, and Wojciech Tyczyński. Enhanced string covering. *Theor. Comput. Sci.*, 506:102–114, 2013. doi:10.1016/j.tcs.2013.08.013.

17 Ondřej Guth and Bořivoj Melichar. Using finite automata approach for searching approximate seeds of strings. In Xu Huang, Sio-Iong Ao, and Oscar Castillo, editors, *Intelligent Automation and Computer Engineering*, volume 52 of *Lecture Notes in Electrical Engineering*, pages 347–360. Springer Netherlands, 2010. doi:10.1007/978-90-481-3517-2_27.

18 Ondřej Guth, Bořivoj Melichar, and Miroslav Balík. Searching all approximate covers and their distance using finite automata. In Peter Vojtáš, editor, *Proceedings of the Conference on Theory and Practice of Information Technologies (ITAT 2008)*, volume 414 of *CEUR Workshop Proceedings*, pages 21–26, 2009. URL: http://ceur-ws.org/Vol-414/paper4.pdf.

19 Costas S. Iliopoulos, Dennis W. G. Moore, and Kunsoo Park. Covering a string. *Algorithmica*, 16(3):288–297, 1996. doi:10.1007/BF01955677.

20 Costas S. Iliopoulos and Laurent Mouchard. Quasiperiodicity and string covering. *Theor. Comput. Sci.*, 218(1):205–216, 1999. doi:10.1016/S0304-3975(98)00260-6.

21 Costas S. Iliopoulos and William F. Smyth. An on-line algorithm of computing a minimum set of k-covers of a string. In Costas S. Iliopoulos, editor, *Proceedings of the 9th Australian Workshop on Combinatorial Algorithms (AWOCA 1998)*, pages 97–106, 1998.

22 Donald E. Knuth, James H. Morris Jr., and Vaughan R. Pratt. Fast pattern matching in strings. *SIAM J. Comput.*, 6(2):323–350, 1977. doi:10.1137/0206024.

23 Tomasz Kociumaka, Solon P. Pissis, Jakub Radoszewski, Wojciech Rytter, and Tomasz Waleń. Fast algorithm for partial covers in words. *Algorithmica*, 73(1):217–233, 2015. doi:10.1007/s00453-014-9915-3.

24 Roman M. Kolpakov and Gregory Kucherov. Finding approximate repetitions under Hamming distance. *Theor. Comput. Sci.*, 1(303):135–156, 2003. doi:10.1016/S0304-3975(02)00448-6.

25 Gad M. Landau and Jeanette P. Schmidt. An algorithm for approximate tandem repeats. In Alberto Apostolico, Maxime Crochemore, Zvi Galil, and Udi Manber, editors, *Proceedings of the 4th Annual Symposium on Combinatorial Pattern Matching (CPM 1993)*, volume 684 of *LNCS*, pages 120–133. Springer, 1993. doi:10.1007/BFb0029801.

26 Gad M. Landau, Jeanette P. Schmidt, and Dina Sokol. An algorithm for approximate tandem repeats. *J. Comput. Biol.*, 8(1):1–18, 2001. doi:10.1089/106652701300099038.

27 Yin Li and William F. Smyth. Computing the cover array in linear time. *Algorithmica*, 32(1):95–106, 2002. doi:10.1007/s00453-001-0062-2.

28 M. Lothaire, editor. *Combinatorics on words*. Cambridge Mathematical Library. Cambridge University Press, 1997. doi:10.1017/CBO9780511566097.

29 Dennis Moore and William F. Smyth. An optimal algorithm to compute all the covers of a string. *Inf. Process. Lett.*, 50(5):239–246, 1994. doi:10.1016/0020-0190(94)00045-X.

30 Dennis Moore and William F. Smyth. A correction to "An optimal algorithm to compute all the covers of a string". *Inf. Process. Lett.*, 54(2):101–103, 1995. doi:10.1016/0020-0190(94)00235-Q.

31 Jeong Seop Sim, Costas S. Iliopoulos, Kunsoo Park, and William F. Smyth. Approximate periods of strings. *Theor. Comput. Sci.*, 262(1):557–568, 2001. doi:10.1016/S0304-3975(00)00365-0.

32 William F. Smyth. Repetitive perhaps, but certainly not boring. *Theor. Comput. Sci.*, 249(2):343–355, 2000. doi:10.1016/S0304-3975(00)00067-0.

33 Hui Zhang, Qing Guo, and Costas S. Iliopoulos. Algorithms for computing the lambda-regularities in strings. *Fundam. Inform.*, 84(1):33–49, 2008. URL: http://content.iospress.com/articles/fundamenta-informaticae/fi84-1-04.

34 Hui Zhang, Qing Guo, and Costas S. Iliopoulos. Varieties of regularities in weighted sequences. In Bo Chen, editor, *Proceedings of the 6th International Conference on Algorithmic Aspects in Information and Management (AAIM 2010)*, volume 6124 of *LNCS*, pages 271–280. Springer, 2010. doi:10.1007/978-3-642-14355-7_28.