Theoretical Computer Science ••• (••••) •••–•••

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

# Period recovery of strings over the Hamming and edit distances ☆

Amihood Amir [a,b], Mika Amit [c,*], Gad M. Landau [c,d], Dina Sokol [e]

[a] Department of Mathematics and Computer Science, Bar-Ilan University, Ramat Gan, Israel
[b] Department of Computer Science, Johns Hopkins University, Baltimore, MD, USA
[c] Department of Computer Science, University of Haifa, Mount Carmel, Haifa, Israel
[d] Department of Computer Science and Engineering, NYU Polytechnic School of Engineering, New York University, NY, USA
[e] Department of Computer and Information Science, Brooklyn College of the City University of New York, Brooklyn, NY, USA

ARTICLE INFO

*Article history:*
Received 12 June 2016
Received in revised form 21 October 2017
Accepted 27 October 2017
Available online xxxx

*Keywords:*
Period recovery
Approximate periodicity
Hamming distance
Edit distance

ABSTRACT

A string $T$ of length $m$ is periodic in $P$ of length $p$ if $P$ is a substring of $T$ and $T[i] = T[i+p]$ for all $0 \le i \le m-p-1$ and $m \ge 2p$. The shortest such prefix, $P$, is called *the period* of $T$ (i.e., $P = T[0..p-1]$). In this paper we investigate the period recovery problem. Given a string $S$ of length $n$, find the primitive period(s) $P$ such that the distance between $S$ and a string $T$ that is periodic in $P$ is below a threshold $\tau$. We consider the period recovery problem over both the Hamming distance and the edit distance. For the Hamming distance case, we present an $O(n \log n)$-time algorithm, where $\tau$ is given as $\lfloor \frac{n}{(2+\epsilon)p} \rfloor$, for $\epsilon > 0$. For the edit distance case, $\tau = \lfloor \frac{n}{(3.75+\epsilon)p} \rfloor$ and $\epsilon > 0$, we provide an $O(n^{4/3})$-time algorithm.

© 2017 Elsevier B.V. All rights reserved.

## 1. Introduction

The prevalence and importance of cyclic phenomena in nature is apparent in diverse areas, including astronomy, geology, earth science, oceanography, meteorology, biological systems, genomics, economics, and more. Assume that an instrument is taking measurements at fixed intervals. When the stream of measurements is analyzed, the question of whether the measurements represent a cycle is raised. In stringology terms, the question is whether the string of measurements is *periodic*.

Periodicity is one of the most important properties of a string and plays a key role in data analysis. As such, it has been extensively studied over the years [38], and linear-time algorithms for exploring the periodic nature of a string were presented (e.g. [13,33,25]). However, realistic data may contain errors. Such errors may be caused by the process of gathering the data which might be prone to transient errors. Moreover, errors can also be an inherent part of the data because the periodic nature of the data represented by the string may be inexact. Thus, it is necessary to cope with periods that have errors.

☆ A preliminary version appeared in the proceedings of LATIN 2016.

* Corresponding author.
*E-mail addresses:* amir@cs.biu.ac.il (A. Amir), mika.amit2@gmail.com (M. Amit), landau@cs.haifa.ac.il (G.M. Landau), sokol@sci.brooklyn.cuny.edu (D. Sokol).

https://doi.org/10.1016/j.tcs.2017.10.026
0304-3975/© 2017 Elsevier B.V. All rights reserved.

In this paper, we present algorithms for the *period recovery problem*, defined in [1]. Informally, the problem is to recover a set of periods that are likely to be the underlying period of the corrupted periodic string, $S$. Given a sequence $S$, assume that $S$ was originally a periodic string, which has been corrupted. Our task is to discover the original uncorrupted string, or more specifically, the exact period of the uncorrupted periodic string. Of course, too many errors can completely change the data, making it impossible to identify the original data and reconstruct the original cycle. However, it is intuitive that few errors should still preserve the periodic nature of the original string.

### 1.1. Related work

Aside from [1] which inspired this work, the most closely related problem to the one discussed in the current paper is the approximate periodicity of Sim et al. (see Problem 2 in [43]). Given a string $S$, a substring $P$ of $S$ is a *t-approximate period* of $S$ if there exists a partition of $S$ into disjoint blocks of substrings $P_1 \dots P_r$ such that the distance between $P$ and every $P_i$ $(1 \leq i < r)$ is less than or equal to $t$, $P_r$ has distance at most $t$ with some prefix of $P$, and $|S| \geq 2|P|$. The algorithm presented in [43] finds the substring $P$ of the input string $S$ that is the period of $S$ with the minimum distance. Let $|S| = n$. When the relative Hamming distance is used, the complexity of the algorithm is $O(n^3)$, and with the (weighted or relative) edit distance it is $O(n^4)$. The key difference between the definition of approximate periodicity of [43] and the current paper is that $t$ is a threshold on the distance between *each* occurrence $P_i$ and $P$, while here we use a more global definition. Our threshold is on the number of errors over all copies, allowing the errors to occur anywhere in the input string.

Another related problem on which much work has been accomplished is finding all *runs* in a given string. Runs are substrings that contain two or more consecutive copies of a pattern and are maximally extended while preserving the shortest period. Kolpakov and Kucherov [33] (see also Chapter 8 in [14]) were the first to prove that a string of length $n$ can contain $O(n)$ runs, and based on this, they presented a linear-time algorithm for locating all runs in a string. Using methods based upon Lyndon words, Bannai et al. [6] demonstrated that the number of runs in a string of length $n$ is actually at most $n - 1$.

In practice, it is usually necessary to allow some degree of mutation in the runs. The problem of finding all *approximate runs* in a string, also called *approximate tandem repeats* (ATR), has been widely researched. There is a myriad of algorithms to search for tandem repeats, and this stems from the fact that "fuzziness" can be introduced into a run in many different ways. In [34] two definitions are explored for which efficient algorithms are presented. Both definitions count the mismatches between consecutive copies of the repeat. A similar definition is used in [23], where *evolutive tandem repeats* are defined over the Hamming distance. In [2], $k$ mismatches are allowed across the entire run, allowing a more global definition of mutations, albeit only in the form of mismatches.

In reality, it is important to allow gaps, in terms of insertions and deletions, in the parts of a tandem repeat. Along these lines, TRfinder, developed by Benson [9], defines tandem repeats based on a probabilistic model. Their algorithm uses a collection of statistical criteria in combination with k-tuple exact matches to detect statistically significant tandem repeats. In [44,45] gaps are accounted for by allowing a sum of $k$ edit distance errors between consecutive copies of the repeat.

The current work differs from the problem of finding ATRs in a string, in that an algorithm that searches for ATRs does not know in advance which substrings are ATRs. The key challenge is to locate the substrings of the input string that are approximately periodic. In the Period Recovery problem, we are given a string, possibly found by an ATR algorithm, that is assumed to be approximately periodic. The question is: what was the original period? It would be useful to be able to answer this question on the output of any of the above-mentioned ATR algorithms. In particular, given a substring that was found to be an ATR, what indeed is the possible set of periods? The fact that the algorithm in this paper does not rely on similarities between neighboring copies, which many of the ATR finding algorithms use, makes it even more useful as a post-processor to an ATR finding algorithm since it validates the ATR from the perspective of a different definition.

*Quasiperiodicity* is a generalization of periodicity in which the occurrences of the period may overlap. A string $S$ is said to have a *cover* or *quasiperiod* $x$ if $S$ can be constructed by concatenations as well as overlaps of $x$. The shortest cover, as well as all covers of a string, can be found in linear time [3,4,37]. The cover problem has been extended to the case where the input string is indeterminate [7,27], and the $k$-cover problem, which finds a *set* of length $k$ covers [12], has been extended to approximate $k$-cover for the Hamming and edit distances [24]. The challenge of finding an *approximate cover* is similar to that of an approximate period – if the cover is assumed to be a substring of the input string, then the problem is much easier than allowing *any* string over the given alphabet as the cover. In a sense the problem of approximate cover is more difficult than approximate periodicity since overlap is more complex than partitioning. The polynomial-time results for approximate cover and for the cover of indeterminate strings are only for special cases. In [16], the general cover problem for indeterminate strings is proven to be NP-complete, and the minimum $k$-cover problem is proven NP-hard in [12]. Similarly, the general problem of locating the approximate period over weighted edit distance has been shown to be NP-complete (see problem 3 in [42,43]).

In the current paper we follow the trend of previous research and we require that the period be a substring of the input string and satisfy a distance threshold. These assumptions, while not being very limiting, yield extremely efficient recovery of the period.

The rest of this section is organized as follows. We start with definitions and notations needed for formally defining the problems. Then, in Section 1.3, the problems are defined. Finally, in Section 1.4 we present a summary of our results.

Please cite this article in press as: A. Amir et al., Period recovery of strings over the Hamming and edit distances, Theoret. Comput. Sci. (2017), https://doi.org/10.1016/j.tcs.2017.10.026

Download English Version:

https://daneshyari.com/en/article/6875659

Download Persian Version:

https://daneshyari.com/article/6875659

Daneshyari.com