Cycle Detection and Correction

AMIHOOD AMIR, Bar-Ilan University and Johns Hopkins University
ESTRELLA EISENBERG, Bar-Ilan University
AVIVIT LEVY, Shenkar College and CRI, University of Haifa
ELY PORAT, Bar-Ilan University
NATALIE SHAPIRA, Bar-Ilan University

13

Assume that a natural cyclic phenomenon has been measured, but the data is corrupted by errors. The type of corruption is application-dependent and may be caused by measurements errors, or natural features of the phenomenon. We assume that an appropriate metric exists, which measures the amount of corruption experienced. This article studies the problem of recovering the correct cycle from data corrupted by various error models, formally defined as the *period recovery problem*. Specifically, we define a metric property which we call *pseudolocality* and study the period recovery problem under pseudolocal metrics. Examples of pseudolocal metrics are the Hamming distance, the swap distance, and the interchange (or Cayley) distance. We show that for pseudolocal metrics, periodicity is a powerful property allowing detecting the original cycle and correcting the data, under suitable conditions. Some surprising features of our algorithm are that we can efficiently identify the period in the corrupted data, up to a number of possibilities logarithmic in the length of the data string, even for metrics whose calculation is $\mathcal{NP}$*-hard*. For the Hamming metric, we can reconstruct the corrupted data in near-linear time even for unbounded alphabets. This result is achieved using the property of separation in the self-convolution vector and Reed-Solomon codes. Finally, we employ our techniques beyond the scope of pseudo-local metrics and give a recovery algorithm for the non-pseudolocal Levenshtein edit metric.

Categories and Subject Descriptors: C.4 [**Performance of Systems**]: Fault Tolerance; E.5 [**Files**]: *Backup / recovery*; F.2.2 [**Nonnumerical Algorithms and Problems**]: Pattern Matching; G.2.1 [**Combinatorics**]: Combinatorial Algorithms

General Terms: Design, Algorithms, Theory

Additional Key Words and Phrases: Approximate periodicity, strings algorithms, string metrics

ACM Reference Format:
Amir, A., Eisenberg, E., Levy, A., Porat, E., and Shapira, N. 2012. Cycle detection and correction. ACM Trans. Algor. 9, 1, Article 13 (December 2012), 20 pages.
DOI = 10.1145/2390176.2390189 http://doi.acm.org/10.1145/2390176.2390189

A. Amir was partially supported by NSF grant CCR-09-04581 and ISF grant 347/09.
E. Eisenberg was supported by a Bar-Ilan University President Fellowship. This article is part of E. Eisenberg's Ph.D thesis. A. Levy was partly supported by ISF grant 347/09. E. Porat was supported by ISF, BSF, and Google award.
Authors' addresses: A. Amir, Department of Computer Science, Bar-Ilan University, Ramat-Gan 52900, Israel; email: amir@cs.biu.ac.il, and Department of Computer Science, Johns Hopkins University, Baltimore, MD 21218; E. Eisenberg, E. Porat, and N. Shapira, Department of Computer Science, Bar-Ilan University, Ramat-Gan 52900, Israel; email: {porately, davidan1}@cs.biu.ac.il; A. Levy, Department of Software Engineering, Shenkar College, 12 Anna Frank, Ramat-Gan, Israel; email: avivitlevy@shenkar.ac.il and CRI, University of Haifa, Mount Carmel, Haifa 31905, Israel.

DOI 10.1145/2390176.2390189 http://doi.acm.org/10.1145/2390176.2390189

1. INTRODUCTION

Cyclic phenomena are ubiquitous in nature, from Astronomy, Geology, Earth Science, Oceanography, and Meteorology, to Biological Systems, the Genome, Economics, and more. Part of the scientific process is understanding and explaining these phenomena. The first step is identifying these cyclic occurrences.

Assume, then, that an instrument is making measurements at fixed intervals. When the stream of measurements is analyzed, it is necessary to decide whether these measurements represent a cyclic phenomenon. The "cleanest" version of this 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, exact data periodicity has been amply researched over the years [Lothaire 1983]. Linear-time algorithms for exploring the periodic nature of data represented as strings were suggested (e.g., Crochemore [1981]). Multidimensional periodicity [Amir and Benson 1998; Galil and Park 1996; Régnier and Rostami 1993] and periodicity in parameterized strings [Apostolico and Giancarlo 2008] was also explored. In addition, periodicity has played a role in efficient parallel string algorithms [Galil 1984; Amir et al. 1993; 1998; Cole et al. 1993].

Many phenomena in the real world have a particular type of event that repeats periodically during a certain period of time. Examples of highly periodic events include biological processes, road traffic peaks, load peaks on web servers, monitoring events in computer networks and many others. Finding periodicity in real-world data often leads to useful insights, because it sheds light on the structure of the data, and gives a basis to predicting future events. Moreover, in some applications periodic patterns can point out a problem: In a computer network, for example, repeating error messages can indicate a misconfiguration, or even a security intrusion such as a port scan [Ma and Hellerstein 2001].

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. Nevertheless, it is still valuable to detect and utilize the underlying cycle. Assume, then, that, in reality, there is an underlying periodic string, which had been corrupted. Our task is to discover the original uncorrupted string.

This seems like an impossible task. To our knowledge, reconstruction or correction of data is generally not possible from raw natural data. The field of Error Correcting Codes is based on the premise that the original data is not the transmitted data. Rather, it is converted to another type of data with features that allow correction under the appropriate assumptions. Without this translation to another model, errors on the raw data may render it totally uncorrectable. A simple example is the following: consider the string *aaaaa aaaaa aaaaa aaaab*. This may be the string *aaaaa aaaaa aaaaa aaaaa* with one error at the end (an a was replaced by a b), or the string *aaaaa aaaab aaaaa aaaab* with one error at the 10th symbol (a b was replaced by an a). How can one tell which error it was?

In this article, we show that, surprisingly, data periodicity acts as a feature to aid the data correction under various error models. The simplest natural error model is substitution errors. It generally models the case where the errors may be transient errors due to transmission noise or equipment insensitivity.

Of course, too many errors can completely change the data, making it impossible to identify the original data and reconstruct the original cycle. On the other hand, it is intuitive that few errors should still preserve the periodic nature of the original string. The scientific process assumes a great amount of confidence in the measurements of natural phenomena, otherwise most advances in the natural sciences are meaningless.

Thus, it is natural to assume that the measurements we receive are, by and large, accurate. Therefore, it is reasonable to assume that we are presented with data that is faithful to the original without too many corruptions. Formally, the problem is defined as follows:

The Period Recovery Problem. Let S be n-long string with period P. Given S', which is S possibly corrupted by at most k errors under a metric d, return P.

The term "recovering" is, in a sense, approximating the original period, because it may be impossible to distinguish the original period from other false candidates. The "approximation" of the original period means identifying a small set of candidates that is guaranteed to include the original period. We are able to provide such a set of size $O(\log n)$.

We quantify the number of errors k that still guarantees the possibility of such a recovery. It turns out that this number is dependent on the size of the original cycle. For example, if the cycle is of size 2, up to 12% substitution errors *in any distribution* can be tolerated to enable almost linear time recovery. The number of errors that still allow correction of the data is quantified as a function of the period length *in the worst case*.

We take a further step and consider more general error models. Such metrics may model specific types of natural corruptions that are application dependent. Examples of such are *reversals* [Berman and Hannenhalli 1996], *transpositions* [Bafna and Pevzner 1998], *block-interchanges* [Christie 1996], *interchanges* [Cayley 1849; Amir et al. 2006], or *swaps* [Amir et al. 2000].We study the problem of recovering the cyclic phenomenon for a general set of metrics that satisfy a condition, which we call *pseudolocality*. Intuitively, pseudolocality means that a single corruption has, in some sense, a local effect on the number of mismatches. The set of pseudolocal metrics is quite extensive and includes such well-studied metrics as the *Hamming Distance*, *Swap Distance*, and *Interchange (*or *Cayley) Distance*.

1.1. Results

We give a bound on the number of errors that still allow detection of $O(\log n)$ candidates for the original period, where n is the length of the measured raw data under any pseudolocal metric. The original underlying cycle is guaranteed to be one of these candidates. To our surprise, this recovery can be done efficiently even for metrics whose computation is $\mathcal{NP}$-hard. We also give faster (near linear time) recovery algorithm for the Hamming distance. Finally, we show that our techniques can be employed even beyond the scope of pseudolocal metrics and give a recovery algorithm for the edit distance, which is not a pseudolocal metric. Let S be n-long string with period P of length p. We prove the following.

—Let d be a c-pseudo-local-metric and let $f_d(n)$ be the complexity of computing the distance between two n-long strings under the metric d. Let $\varepsilon > 0$ be a constant. Then, if S is corrupted by at most $\frac{n}{(2c+\varepsilon)\cdot p}$ errors under the metric d, then, a set of $\log_{1+\frac{\varepsilon}{c}} n$ candidates which includes P can be constructed in time $O(n \log n + f_d(n) \cdot n \log n)$ (Theorem 3.4).

As a corollary, we get that if S is corrupted by at most $\frac{n}{(4+\varepsilon)\cdot p}$ swap errors, then, a set of $\log_{1+\frac{\varepsilon}{2}} n$ candidates that includes P can be constructed in time $O(n^2 \log^4 n)$ (Corollary 3.6).

—Let d be a c-pseudolocal metric that admits a polynomial-time γ-approximation algorithm with complexity $f_d^{app}(n)$, where $\gamma > 1$ is a constant. Let $\varepsilon > (\gamma - 1) \cdot 2c$ be a constant. Then, if S is corrupted by at most $\frac{n}{(2c+\varepsilon)\cdot p}$ errors under the metric

d, then, a set of $\log_{1+\frac{\varepsilon'}{c}} n$ candidates which includes P can be constructed in time $O(n \log n + f_d^{app}(n) \cdot n \log n)$, where $\varepsilon' = \frac{\varepsilon}{\gamma} - \frac{2c(1-\gamma)}{\gamma} > 0$ is a constant. (Theorem 3.7) As a corollary we get that for every $\varepsilon > \frac{8}{9}$, if S is corrupted by at most $\frac{n}{(4+\varepsilon)\cdot p}$ interchange errors, then, a set of $\log_{1+\frac{\varepsilon'}{2}} n$ candidates which includes P can be constructed in time $O(n^2 \log^2 n)$, where $\varepsilon' = \frac{3\varepsilon}{2} - \frac{4}{3} > 0$ is a constant (Corollary 3.8).

Note that the period can be approximated in polynomial time, even though computing the interchange distance is $\mathcal{NP}$-hard [Amir et al. 2007].

—If S is corrupted by less than $\frac{n}{4p}$ substitution errors then a set of $O(\log n)$ candidates that includes P can be constructed in time $O(n \log n)$ for bounded alphabets (Theorem 4.9), and $O(n \log^2 n)$ for unbounded alphabets (Theorem 4.10).

—Let $\varepsilon > 0$ be a constant. Then, if S is corrupted by at most $\frac{n}{(4+\varepsilon)\cdot p}$ Levenshtein edit operations, then, a set of $\log_{1+\frac{\varepsilon}{2}} n$ candidates such that their cyclic rotations include P, can be constructed in time $O(n^3 \log n)$, (Theorem 5.3).

1.2. Techniques

New concepts were defined, studied, and used in this article in order to achieve the results. The first important concept is the pseudolocality property that enabled a unified study of cycle recovery under a rich set of metrics that includes well-known metrics as the *Hamming Distance*, *Swap Distance*, and *Interchange Distance*. Properties of pseudolocal metrics were proved to support a nontrivial use of the Minimum Augmented Suffix Tree (MAST) data structure in order to achieve our general results.

Another useful concept defined in this article is *separation* in the *self-convolution vector* used to aid in faster discovery of the original period under the Hamming distance. Separation supports the the extension of the results to unbounded alphabets using Reed-Solomon codes. It is somewhat surprising that a tool of Error Correction is brought to bear on achieving correction in raw natural data. We stress that in Error-Correcting-Codes the predesigned structure of the code is known to the decoder. In our situation, however, we attempt to find a cyclic phenomenon where nothing is known a-priori, not even if it really exists.

The main contributions of this article are:

(1) A first formalization and study of the period recovery problem.
(2) Identifying the pseudolocal property in metrics, that enables initiation of a broad unified study of period recovery in many well-studied metrics.
(3) Demonstration of efficient recovery even for metrics whose computation is $\mathcal{NP}$-hard.
(4) Employment of the techniques beyond the scope of pseudolocal metrics, by studying the period recovery problem under Levenshtein edit distance.
(5) An unusual use of Error Correcting Codes for the correction of *raw* natural data. It demonstrates that Error Correcting Codes can be used even in analysis of data that is not predesigned by some chosen structure to meet some requirements. Instead, the properties of such codes are exploited to reveal the unknown possibly corrupted structure of the data, if such a structure exists.

1.3. Organization of This Article

The rest of the article is organized as follows. In Section 2, we give the basic definitions and tools. In Section 3, we study the problem of detecting the original period of a given possibly corrupted string under any pseudolocal metric, and prove Theorem 3.4, Corollary 3.6, Theorem 3.7, and Corollary 3.8. In Section 4, we give faster period recovery algorithms for a string corrupted under the Hamming metric, and prove

Theorems 4.9 and 4.10. Finally, in Section 5, we consider the edit distance and prove Theorem 5.3.

2. PRELIMINARIES

In this section, we give basic definitions of periodicity and approximate periodicity, string metrics and pseudolocal metrics. We also give some basic lemmas.

Definition 2.1. Let S be a string of length n. S is called *periodic* if $S = P^i\text{pref}(P)$, where $i \in \mathbb{N},\ i \geq 2,\ P$ is a substring of S such that $|P| \leq n/2$, P^i is the concatenation of P to itself i times, and $\text{pref}(P)$ is a prefix of P. The smallest such substring P is called the *period* of S. If S is not periodic it is called *aperiodic*.[1]

Remark. Throughout this article, we use p to denote a period length and P the period string, that is, $p = |P|$.

Definition 2.2. Let S be n-long string over alphabet Σ. Let d be a metric defined on strings. S is called *periodic with k errors* if there exists a string P over Σ, $p \in \mathbb{N}$, $p \leq n/2$, such that

$$d(P^{\lfloor n/p \rfloor}\text{pref}(P), S) = k,$$

where $\text{pref}(P)$ is a prefix of P of length $n \bmod p$. The string P is called *a k-error period of S* or *approximate period of S with k errors*.

Consider a set Σ and let x and y be two n-long strings over Σ. We wish to formally define the process of converting x to y through a sequence of operations. An *operator* ψ is a function $\psi : \Sigma^n \to \Sigma^{n'}$, with the intuitive meaning being that ψ converts n-long string x to n'-long string y with a cost associated to ψ. That cost is the distance between x and y. Formally,

Definition 2.3 (*string metric*). Let $s = (\psi_1, \psi_2, \ldots, \psi_k)$ be a sequence of operators, and let $\psi_s = \psi_1 \circ \psi_2 \circ \cdots \circ \psi_k$ be the composition of the ψ_j's. We say that s *converts x into y* if $y = \psi_s(x)$.

Let Ψ be a set of rearrangement operators, we say that Ψ *can convert x to y*, if there exists a sequence s of operators from Ψ that converts x to y. Given a set Ψ of operators, we associate a non-negative *cost* with each sequence from Ψ, *cost* : $\Psi^* \to R^+$. We call the pair $(\Psi, cost)$ an *edit system*. Given two strings $x, y \in \Sigma^*$ and an edit system $\mathcal{R} = (\Psi, cost)$, we define the distance from x to y under $\mathcal{R}$ to be:

$$d_{\mathcal{R}}(x, y) = \min\{cost(s) | s \text{ from } \mathcal{R} \text{ converts } x \text{ to } y\}$$

If there is no sequence that converts x to y, then the distance is ∞.

It is easy to verify that $d_{\mathcal{R}}(x, y)$ is a metric. Definition 2.4 gives examples of string metrics.

Definition 2.4.

(1) *Hamming Distance.* $\Psi = \{\rho^n_{i,\sigma} | i, n \in \mathbb{N},\ \ i \leq n,\ \ \sigma \in \Sigma\}$, where $\rho^n_{i,\sigma}(\alpha)$ substitutes the ith element of n-tuple α by symbol σ.
We denote the Hamming distance by H.

(2) *Edit Distance.* In addition to the substitution operators of the Hamming distance, Ψ also has *insertion* and *deletion* operators. The insertion operators are: $\{\iota^n_{i,\sigma} | i, n \in \mathbb{N}, i \leq n,\ \ \sigma \in \Sigma\}$, where $\iota^n_{i,\sigma}(\alpha)$ adds the symbol σ following the ith element of n-tuple α, creating an $n+1$-tuple α'.

[1] Denote by $|S|$ the length of a string S.

The deletion operators are $\{\delta_i^n | i, n \in \mathbb{N}, i \leq n\}$, where $\delta_i^n(\alpha)$ deletes the symbol at location i of n-tuple α, creating an $n-1$-tuple α'.

(3) *Swap Distance.* $\Psi = \{\zeta_i^n | i, n \in \mathbb{N}, i < n\}$, where $\zeta_i^n(\alpha)$ swaps the ith and $i+1$st elements of n-tuple α, creating an n-tuple α'. A *valid* sequence of operators in the Swap metric has the additional condition that if ζ_i^n and ζ_j^m are operators in a sequence, then $i \neq j,\ i \neq j+1,\ i \neq j-1$, and $n = m$.

(4) *Interchange Distance.* $\Psi = \{\pi_{i,j}^n | i, n \in \mathbb{N},\ \ i \leq j \leq n\}$, where $\pi_{i,j}^n(\alpha)$ interchanges the ith and jth elements of n-tuple α, creating an n-tuple α'.

Definition 2.5 (*pseudolocal metric*). Let d be a string metric. d is called a *pseudolocal metric* if there exists a constant $c \geq 1$ such that, for every two strings S_1, S_2, if $d(S_1, S_2) = 1$ then $1 \leq H(S_1, S_2) \leq c$.

A metric that is pseudolocal with constant c is called a *c-pseudolocal metric*.

Note that pseudolocality allows the resulted number of mismatches to be unboundedly far from each other (as may happen in an interchange) and therefore, a pseudolocal metric is not necessarily also local in the intuitive sense. Lemma 2.6 shows some interesting pseudolocal metrics and follows immediately from Definitions 2.4 and 2.5.

LEMMA 2.6. *The following metrics are c-pseudolocal metrics:*

(1) *Hamming distance (with $c = 1$).*
(2) *Swap distance (with $c = 2$).*
(3) *Interchange distance (with $c = 2$).*

On the other hand, the Edit distance is not a pseudolocal metric, because a single deletion or insertion may cause an unbounded number of mismatches. Lemma 2.9 states a basic property of pseudolocal metrics that is exploited in this article. Lemma 2.8 is needed for proving Lemma 2.9 and Lemma 4.4.

Definition 2.7. Let $S = S[0], \ldots, S[s]$ and $T = T[0], \ldots, T[t]$ be strings, and let $0 \leq i \leq |T|$. The *alignment of S with T in location i* is the comparison of $S[j]$ and $T[i+j], \forall j = 0, \ldots, \min(s, t-i)$. In other words, we place S above T such that the first location of S is aligned with the ith location of T.[2]

LEMMA 2.8. *Let P be a period, then, for every i, $0 < i \leq p-1$, the alignment of P with P^2 in location i has at least two mismatches.*

PROOF. By the definition of a period, there must be at least one mismatch in this alignment; otherwise, there is a smaller period. We now show that there are at least two mismatches in this alignment. We first prove the claim for the case where the first mismatch is in location i of P^2, and then show the claim holds in the general case. For simplicity of exposition, consider the infinite strings P^∞ aligned with P^{2^∞} at location i. We are assuming that $P[0] \neq P^2[i]$ and also that there are no other mismatches. If $P^2[i] \neq P^2[2i]$, we get an additional mismatch, since $P[i]$ is aligned with $P^2[2i]$, and $P[i] = P^2[i]$. Therefore, we must have that $P^2[i] = P^2[2i]$.

Because of the alignment in location i of P^∞ and P^{2^∞}, we conclude that for every integer $x,\ P^{2^\infty}[x \cdot i] = P[i]$. Also, by the definition of periodicity all $P^{2^\infty}[y \cdot p]$ for every integer y must be equal. In particular, they are also all equal to $P^{2^\infty}[i \cdot p]$ (since one may choose $y = i$). That means that they are also all equal to $P^{2^\infty}[x \cdot i]$ for all x (because of the case $x = p$). We get that $P^{2^\infty}[x \cdot i] = P^{2^\infty}[y \cdot p]$ for all x and y, specifically for $x = 1$ and $y = 0$. However, $P^{2^\infty}[0] = P^2[0] = P[0] \neq P^2[i] = P^{2^\infty}[i]$, contradiction.

[2]The notion of alignment is called *shift* in Cormen et al. [2001].

We now show that the proof holds for the general case where in the alignment with location i of P^2 the mismatch is at location $j > 0$ of P. Note that we can cut the first j symbols from both P^2 and P and paste them at the end. Then, we get exactly the case of mismatch at the first location of P, for which the claim is proven. Clearly, neither the alignment nor its number of mismatches is changed by this cut and paste. The lemma then follows. □

LEMMA 2.9. *Let d be a c-pseudolocal metric. Then, for every $n \in \mathbb{N}$ and n-long periodic strings S_1, S_2 with P_1 and P_2 the periods of S_1 and S_2 respectively, $p_1 \geq p_2$ and $P_1 \neq P_2$ (i.e., there exists at least one mismatch between P_1 and P_2), it holds that*

$$d(S_1, S_2) \geq \frac{n}{c \cdot p_1}$$

PROOF. Let $n \in \mathbb{N}$. Let S_1, S_2 be two n-long periodic strings with P_1 and P_2 the periods of S_1 and S_2 respectively, $p_1 \geq p_2$ and $P_1 \neq P_2$. By c-pseudolocality definition, it is enough to prove that $H(S_1, S_2) \geq \frac{n}{p_1}$, from which we immediately get that $d(S_1, S_2) \geq \frac{n}{c \cdot p_1}$.

Denote the infinite string composed of concatenations of P_2 by P_2^∞. If P_1 is not a substring of P_2^∞ then the claim is proven, since every occurrence of P_1 in S_1 will have at least one mismatch with the corresponding position in S_2. Therefore, assume that P_1 is a substring of P_2^∞. Without loss of generality, we can assume that P_1 is equal to a prefix of P_2^∞. P_1^2 is not a substring of P_2^∞, otherwise, by the periodicity lemma, both S_1 and S_2 have a period smaller than P_1 and P_2, in contradiction to the fact that P_1 and P_2 are the periods of S_1 and S_2, respectively. Therefore, for every match of P_1 in S_2, there is a mismatch of the subsequent concatenation of P_1. We claim that the subsequent concatenation has, in fact 2 mismatches. This is true because P_2 is, in fact, a prefix of P_1. Therefore, any alignment of P_1 with any index of P_2 other than the first, causes two mismatches, by Lemma 2.8. Therefore, every two occurrences of P_1 contribute at least two mismatches. Thus, $H(S_1, S_2) \geq \frac{n}{p_1}$. □

3. APPROXIMATING THE ORIGINAL PERIOD UNDER PSEUDOLOCAL METRICS

The term *approximation* here refers to the ability to give a relatively small size set of candidates that includes the exact original period of the string. We first, show the bounds on the number of errors that still guarantees a small-size set of candidates. We then show how this set can be identified, and thus the original uncorrupted string can be approximately recovered. This section is organized as follows. We begin with a characterization of a bound on the number of candidate periods that admit a given bound on the number of errors under any pseudolocal metric. We then describe and analyze a general period recovery algorithm that works for any pseudolocal metric. We conclude with corollaries for specific pseudolocal metrics: Hamming, swap, and interchange distances.

3.1. General Approximation Characterization

LEMMA 3.1. *Let d be a c-pseudolocal metric. Let $\varepsilon > 0$ be a constant, S an n-long string and P_1, P_2, $P_1 \neq P_2$, approximate periods of S with at most $\frac{n}{(2c+\varepsilon)\cdot p_1}$, $\frac{n}{(2c+\varepsilon)\cdot p_2}$ errors respectively (without loss of generality assume that $p_1 \geq p_2$). Then,*

$$p_1 \geq \left(1 + \frac{\varepsilon}{c}\right) \cdot p_2.$$

PROOF. Let S_1 be the n-long string with period P_1 and S_2 be the n-long string with period P_2. Then, $d(S_1, S) \leq \frac{n}{(2c+\varepsilon)\cdot p_1}$ and $d(S_2, S) \leq \frac{n}{(2c+\varepsilon)\cdot p_2}$. Therefore,

$$\frac{n}{(2c+\varepsilon)\cdot p_1} + \frac{n}{(2c+\varepsilon)\cdot p_2} \geq d(S_1, S) + d(S_2, S)$$

By triangle inequality we have,

$$d(S_1, S) + d(S_2, S) \geq d(S_1, S_2)$$

By pseudolocality of d and Lemma 2.9,

$$d(S_1, S_2) \geq \frac{n}{c \cdot p_1}$$

Therefore,

$$\frac{n}{(2c+\varepsilon)\cdot p_1} + \frac{n}{(2c+\varepsilon)\cdot p_2} \geq \frac{n}{c \cdot p_1}$$

from which we get,

$$p_2 + p_1 \geq \frac{(2c+\varepsilon)}{c} \cdot p_2$$

or,

$$p_1 \geq \left(1 + \frac{\varepsilon}{c}\right) \cdot p_2. \quad \square$$

COROLLARY 3.2. *Let d be a c-pseudolocal-metric. Let S be a n-long string. Then, there are at most $\log_{1+\frac{\varepsilon}{c}} n$ different approximate periods P of S with at most $\frac{n}{(2c+\varepsilon)\cdot p}$ errors.*

PROOF. First, note that there cannot be two such different approximate periods of the same length p, because then, by Lemma 3.1, we get $p \geq (1+\frac{\varepsilon}{c})\cdot p$, contradiction. Thus, such different approximate periods must have different length. Now, let $1 \leq l_1 < l_2 < \cdots < l_{t-1} < l_t \leq \frac{n}{2}$ be the different lengths of approximate periods of S. By Lemma 3.1,

$$1 \leq l_1 < \left(1 + \frac{\varepsilon}{c}\right) \cdot l_2 < \left(1 + \frac{\varepsilon}{c}\right)^2 \cdot l_3 < \cdots < \left(1 + \frac{\varepsilon}{c}\right)^{t-2} \cdot l_{t-1} < l_t \leq \frac{n}{2}$$

Therefore, $t \leq \log_{1+\frac{\varepsilon}{c}} n$. $\square$

3.2. The General Period Recovery Algorithm

We now describe an algorithm for recovering the original period up to $O(\log n)$ candidates under any pseudolocal metric defining the string corruption process. The algorithm has two stages: candidates extraction stage and validation stage. The algorithm starts by extracting from the string S a set of candidates for the original period. Due to pseudolocality of the metric and the number of errors assumed, there are copies of the original period that are left uncorrupted (see proof of Lemma 3.3). A trivial $O(n^2)$-size set would, therefore, be the set of all substrings of S.

We, however, do better than that. Our algorithm extracts a set of only $O(n \log n)$ candidates. These candidates are then verified to find those that comply with the given error bound. By Corollary 3.2, we know that only $O(\log n)$ candidates can survive the validation stage.

The algorithm extraction stage uses the *MAST* (Minimal Augmented Suffix Tree) data structure presented by Apostolico and Preparata [1996]. This data structure is a suffix-tree with additional nodes and information in each node. The additional information in a node is the number of nonoverlapping occurrences of the string represented

PERIOD RECOVERY ALGORITHM
Input: A string S of length n, a constant $\varepsilon > 0$.
Extraction Stage:
Construct the Minimal Augmented Suffix Tree $MAST(S)$
for each s substring of S of length $\leq \frac{c}{\varepsilon}$
    $C \leftarrow C \cup s$
Perform EDFS on $MAST(S)$, where:
    for each node v in $MAST(S)$
        $C \leftarrow C \cup S(v)$
    for each edge e in $MAST(S)$
        $C \leftarrow C \cup s_e$
Validation Stage:
for each candidate string $P \in C$ do
    Construct the n-long string S_P with period P
    if $d(S_P, S) > \frac{n}{(2c+\varepsilon)\cdot p}$ then
        $C \leftarrow C \setminus P$
Output:
return C

Fig. 1. The period recovering algorithm for a c-pseudolocal metric d.

by the path from the root to that node. The number of nodes in the *MAST* is proven in Apostolico and Preparata [1996] to be $O(n \log n)$, and this is also its space bound. The *MAST* construction time is proven to be $O(n \log n)$ in Brodal et al. [2002]. As we show in Lemma 3.3, from the *MAST* nodes, a set of candidates that includes all possible approximate periods can be extracted. To do that, the algorithm performs an *Extended Depth First Search* on *MAST(S)*, denoted EDFS, which also keeps track of the string representing the path from the root to the current node. We use the following notations for the EDFS on *MAST*(S). $S(v)$ is the string associated with the path from the root of *MAST*(S) to node v and $occ(v)$ is the number of nonoverlapping occurrences of $S(v)$ in S stored in each node v in *MAST*(S). It is important to note that the strings associated with paths in *MAST*(S) are only implicitly kept, as references to the corresponding locations in S (as it is in a Suffix Tree). This enables the *MAST* data structure to consume only near linear space. For each edge in *MAST*(S) we denote by s_e the string associated with the path from the root of *MAST*(S) to e ending with the first character of the substring represented by e. We use the notation $occ(s_e)$ for the number of nonoverlapping occurrences of the substring s_e. This number is equal to $occ(v)$ for the node v in which e ends. A detailed description of the algorithm is given in Figure 1.

3.3. Analysis of the Algorithm

LEMMA 3.3. *Let d be a c-pseudolocal metric. Let $\varepsilon > 0$ be a constant, S n-long string with P an approximate period of S with at most $\frac{n}{(2c+\varepsilon)\cdot p}$ errors. Then, if $p > \frac{c}{\varepsilon}$, there exists a node v in MAST(S) such that $S(v) = P$ or $S(v)$ is the prefix of P of length $p - 1$.*

PROOF. Let P be an approximate period of S with at most $\frac{n}{(2c+\varepsilon)\cdot p}$ errors, and let S_P be the n-long string with period P. We first show that P has an exact appearance in S, and therefore, appears (implicitly) on some path of *MAST*(S). Note that, $d(S_P, S) \leq \frac{n}{(2c+\varepsilon)\cdot p}$. Thus, by c-pseudolocality of d, $H(S_P, S) \leq c \cdot \frac{n}{(2c+\varepsilon)\cdot p} = \frac{n}{(2+\frac{\varepsilon}{c})\cdot p}$. This, means that at least half of the appearances of P in S_P are exact occurrences of P in S. By the definition of a period, it has at least two appearances, and therefore, there is at least one exact occurrence of P in S, so it appears on some path of *MAST*(S).

Assume to the contrary that the corresponding path of P in $MAST(S)$ does not end with a node. Thus, P ends within an edge of $MAST(S)$. Consider the string P' which is the prefix of P of length $p' = p - 1$. If P' ends in a $MAST(S)$ node, the proof is complete, we, therefore, assume that P' also ends within the same $MAST(S)$ edge. Now, since P and P' are on an edge leading to the same $MAST(S)$ node, they both have the same nonoverlapping occurrences in S. Moreover, P and P' are the same string up to one character and all the non-overlapping occurrences of P are also non-overlapping occurrences of P'. Let $S_{P'}$ be the n-long string with period P'. We, thus, have, $d(S_{P'}, S) < d(S_P, S)$. In addition, P is an approximate period with at most $\frac{n}{(2c+\varepsilon)\cdot p}$ errors. Therefore,

$$d(S_{P'}, S) < d(S_P, S) \leq \frac{n}{(2c+\varepsilon)\cdot p} < \frac{n}{(2c+\varepsilon)\cdot p'}$$

However, this means that there are two approximate periods that only differ by one letter, in contradiction to Lemma 3.1, unless $p \leq \frac{c}{\varepsilon}$. □

Theorem 3.4 follows.

THEOREM 3.4. *Let d be a c-pseudolocal-metric and let $f_d(n)$ be the complexity of computing the distance between two n-long strings under the metric d. Let S be n-long string with period P. Let $\varepsilon > 0$ be a constant. Then, if S is corrupted by at most $\frac{n}{(2c+\varepsilon)\cdot p}$ errors under the metric d, then, a set of $\log_{1+\frac{\varepsilon}{c}} n$ candidates that includes P can be constructed in time $O(n \log n + f_d(n) \cdot n \log n)$.*

PROOF. Let S' be the given corruption of S. It is enough to show that the period recovering algorithm described in Figure 1 on input S' has the requested property and its time is bounded by $O(n \log n + f_d(n) \cdot n \log n)$. We first show that every approximate period P' of S' with at most $\frac{n}{(2c+\varepsilon)\cdot p'}$ errors is put in C in the extraction stage of the algorithm (and thus P is put in this set). By Lemma 3.3, P' falls into one of the following three cases:

(1) $p' \leq \frac{c}{\varepsilon}$ In this case, P' is extracted in lines 2–3 of the algorithm.
(2) $P' = S(v)$ for some node v in $MAST(S')$ In this case, P' is extracted in lines 5–6 of the algorithm.
(3) $P' = s_e$ for some edge e in $MAST(S')$ In this case P' is extracted in lines 7–8 of the algorithm.

Since, P' is an approximate period of S' with at most $\frac{n}{(2c+\varepsilon)\cdot p'}$ errors, it also survives the check in lines 9–12 of the algorithm's validation stage. Therefore, the set of candidates returned by the algorithm includes P. By Corollary 3.2, we have that the size of the set returned by the algorithm is $\log_{1+\frac{\varepsilon}{c}} n$, as required.

We now prove the time bound of the algorithm. Brodal et al. [2002] assure that line 1 can be done in time $O(n \log n)$. Since $\frac{c}{\varepsilon}$ is constant, lines 2–3 can be done in $O(n)$ time. The space of the $MAST(S')$ constructed in line 1 is $O(n \log n)$ and its nodes number is also $O(n \log n)$ [Apostolico and Preparata 1996]. Therefore, the EDFS performed in lines 4–8 takes time linear in the $MAST(S')$ size, that is, $O(n \log n)$ time. Also, the size of the extracted candidates set before the validation is $O(n \log n)$, as explained above. Only for these $O(n \log n)$ candidates the validation stage is done. The theorem then follows. □

From Theorem 3.4, we get a simple cycle recovery under the Hamming distance (Corollary 3.5), recovery under the swap distance (Corollary 3.6) and recovery under

any pseudolocal metric that admits efficient constant approximation (Theorem 3.7), specifically, under the interchange distance (Corollary 3.8).

3.4. Simple Period Recovery under the Hamming Distance

COROLLARY 3.5. *Let S be n-long string with period P. Let $\varepsilon > 0$ be a constant. Then, if S is corrupted by at most $\frac{n}{(2+\varepsilon)\cdot p}$ substitution errors, then, a set of $\log_{1+\varepsilon} n$ candidates which includes P can be constructed in time $O(n^2 \log n)$.*

PROOF. By Lemma 2.6, the Hamming metric is 1-pseudolocal. Therefore, by Theorem 3.4, if S is corrupted by at most $\frac{n}{(4+\varepsilon)\cdot p}$ substitution errors, then, a set of $\log_{1+\varepsilon} n$ candidates that includes P can be constructed in time $O(n \log n + f_{Hamming}(n) \cdot n \log n)$, where $f_{Hamming}(n)$ is the complexity of computing the Hamming distance between two n-long strings. Since, $f_{Hamming}(n)$ is simply $O(n)$, the corollary follows. □

3.5. Period Recovery under the Swap Distance

COROLLARY 3.6. *Let S be n-long string with period P. Let $\varepsilon > 0$ be a constant. Then, if S is corrupted by at most $\frac{n}{(4+\varepsilon)\cdot p}$ swap errors, then, a set of $\log_{1+\frac{\varepsilon}{2}} n$ candidates that includes P can be constructed in time $O(n^2 \log^4 n)$.*

PROOF. By Lemma 2.6, the swap metric is 2-pseudo-local. Therefore, by Theorem 3.4, if S is corrupted by at most $\frac{n}{(4+\varepsilon)\cdot p}$ swap errors, then, a set of $\log_{1+\frac{\varepsilon}{2}} n$ candidates which includes P can be constructed in time $O(n \log n + f_{swap}(n) \cdot n \log n)$, where $f_{swap}(n)$ is the complexity of computing the swap distance between two n-long strings. In [Amir et al. 2003] it is shown that $f_{swap}(n)$ is $O(n \log^3 n)$. The corollary follows. □

3.6. Period Recovery under the Interchange Distance

THEOREM 3.7. *Let S be n-long string with period P. Let d be a c-pseudo-local metric that admits a polynomial time γ-approximation algorithm with complexity $f_d^{app}(n)$, where $\gamma > 1$ is a constant. Let $\varepsilon > (\gamma - 1) \cdot 2c$ be a constant. Then, if S is corrupted by at most $\frac{n}{(2c+\varepsilon)\cdot p}$ errors under the metric d, then, a set of $\log_{1+\frac{\varepsilon'}{c}} n$ candidates which includes P can be constructed in time $O(n \log n + f_d^{app}(n) \cdot n \log n)$, where $\varepsilon' = \frac{\varepsilon}{\gamma} - \frac{2c(1-\gamma)}{\gamma} > 0$ is a constant.*

PROOF. By Theorem 3.4, if S is corrupted by at most $\frac{n}{(2c+\varepsilon)\cdot p}$ errors under the metric d, then, a set of $\log_{1+\frac{\varepsilon}{c}} n$ candidates that includes P can be constructed in time $O(n \log n + f_{int}(n) \cdot n \log n)$, where $f_d(n)$ is the complexity of computing the distance between two n-long strings under the metric d. However, computing the distance may not be done efficiently. Nevertheless, the efficient $O(f_d^{app}(n))$ γ-approximation can be used instead of $f_d(n)$. Let S' be the given corruption of S. The algorithm in Figure 1 is adapted with only line 11 replaced by:

$$\textit{if } d_{app}(S_P, S') > \gamma \cdot \frac{n}{(2c+\varepsilon)\cdot p},$$

where d_{app} is the distance computed by the algorithm f_d^{app}. The time complexity of this modified algorithm is clearly $O(n \log n + f_d^{app}(n) \cdot n \log n)$. We now show that the returned set of candidates is $\log_{1+\frac{\varepsilon'}{c}} n$. Note that if S_P is an approximate period with at most $\frac{n}{(2c+\varepsilon)\cdot p}$ errors under the metric d, then,

$$d(S_P, S') \le d_{app}(S_P, S') \le \gamma \cdot d(S_P, S') \le \gamma \cdot \frac{n}{(2c+\varepsilon)\cdot p} = \frac{n}{(2c+\varepsilon')\cdot p}$$

where $\varepsilon' = \frac{\varepsilon}{\gamma} + \frac{2c(1-\gamma)}{\gamma} > 0$ is a constant. Therefore, as in Lemma 3.1 and Corollary 3.2, we get the claimed bound on the returned candidates set. The theorem follows. □

COROLLARY 3.8. *Let S n-long string with period P. Let $\varepsilon > \frac{8}{9}$ be a constant. Then, if S is corrupted by at most $\frac{n}{(4+\varepsilon)\cdot p}$ interchange errors, then, a set of $\log_{1+\frac{\varepsilon'}{2}} n$ candidates that includes P can be constructed in time $O(n^2 \log^2 n)$, where $\varepsilon' = \frac{3\varepsilon}{2} - \frac{4}{3} > 0$ is a constant.*

PROOF. By Lemma 2.6, the interchange metric is 2-pseudo-local. Therefore, by Theorem 3.4, if S is corrupted by at most $\frac{n}{(4+\varepsilon)\cdot p}$ interchange errors, then, a set of $\log_{1+\frac{\varepsilon}{2}} n$ candidates that includes P can be constructed in time $O(n \log n + f_{int}(n) \cdot n \log n)$, where $f_{int}(n)$ is the complexity of computing the interchange distance between two n-long strings. Indeed, in Amir et al. [2007] it is shown that the interchange distance problem is $\mathcal{NP}$-hard. Nevertheless, the $O(n \log n)$ time 1.5-approximation showed in Amir et al. [2007] can be used instead of $f_{int}(n)$. By Theorem 3.7, the corollary follows. □

4. FASTER PERIOD RECOVERY UNDER THE HAMMING DISTANCE

By Corollary 3.5, the algorithm in Figure 1 gives a $O(n^2 \log n)$ time bound for the Hamming metric. In this section, we use stronger tools to achieve cycle recovery under the Hamming metric in near-linear time. The section is organized as follows. We need some metric-specific definitions and tools, so we begin with definitions and properties of the self-convolution vector and the separation property. We then describe and analyze the faster period recovery algorithm for the Hamming distance. We also give a lower bound on the number of candidates the algorithm produces in the worst case. Finally, we show how to extend the results for unbounded alphabets.

Remark. Throughout this section, the terms, *errors* or *corruption*, refer to substitution errors.

Definition 4.1. Let S be n-long string over alphabet Σ, and let $p \in \mathbb{N}$, $p \leq n/2$. For every i, $0 \leq i \leq p-1$, *the ith frequent element with regard to p*, denoted F_i^p, is the symbol $\sigma \in \Sigma$ which appears the largest number of times in locations $i + c \cdot p$ in S, where $c \in [0..\lfloor n/p \rfloor]$, and[3] $i + c \cdot p \leq n$.

Remark. For simplicity of exposition, positive integers of the form $i+c\cdot p, c \in [0..\lfloor n/p \rfloor]$ for some $p \leq n/2$ are referred to as locations in S without mentioning explicitly the condition $i + c \cdot p \leq n$. It is always assumed that only valid positions in S are referred to.

4.1. The Self-Convolution Vector

The main tool we exploit in this aticle to give faster recovery for the Hamming metric is the *self-convolution vector* defined in Definition 4.2.

Definition 4.2. Let S be n-long string over alphabet Σ, and let $\bar{S}$ be the string S concatenated with n \$'s (where \$ $\notin \Sigma$). The *self-convolution vector of S*, v, is defined for every i, $0 \leq i \leq n/2 - 1$,

$$v[i] = \sum_{j=0}^{n-1} f(\bar{S}[i+j], S[j]),$$

where $f(\bar{S}[i+j], S[j]) = \begin{cases} 1, & \text{if } \bar{S}[i+j] \neq \text{S[j] and } \bar{S}[i+j] \neq \$; \\ 0, & \text{otherwise.} \end{cases}$.

[3]For simplicity of exposition, positive integers of the form $i+c\cdot p, c \in [0..\lfloor n/p \rfloor]$ for some $p \leq n/2$ are referred to as locations in S without mentioning explicitly the condition $i + c \cdot p \leq n$. It is always assumed that only valid positions in S are referred to.

Using standard FFT techniques gives Lemma 4.3.

LEMMA 4.3 [FISCHER AND PATERSON 1974]. *The self-convolution vector of a length n string S over alphabet Σ can be computed in time $O(|\Sigma| n \log n)$.*

4.2. The Structure of the Self-Convolution Vector

The self-convolution vector of a periodic string (without corruptions) is a highly structured vector. Lemma 4.4 describes this structure, which is used here for proving Theorem 4.9.

LEMMA 4.4. *Let S be n-long string with period P, then the self-convolution vector of S is:*

(1) *0, in every location i, $i \equiv 0 \bmod p$, and*
(2) *at least $\lfloor \frac{n}{p} \rfloor$, in all other locations.*

PROOF. There are two cases to consider.

Case $i \equiv 0 \bmod p$. By the definition of periodicity for every j, $0 \leq j \leq n-1$, $\bar{S}[i+j] = S[j]$. Thus, $f(\bar{S}[i+j], S[j]) = 0$, for every j and therefore, $v[i] = 0$.

Case $i \not\equiv 0 \bmod p$. By Lemma 2.8, every instance of the period in the alignment adds at least two mismatches to the vector result at location i. Since by the definition of the self-convolution vector at every location i, there are at least $\lfloor \frac{n}{2p} \rfloor$ instances of the period in the alignment that do not contain a \$, the lemma follows for this case. □

4.3. The Separation Property

The structure of the self-convolution vector in an uncorrupted string S is described by Lemma 4.4. By this structure, the locations that are multiplicities of the period can be easily identified, and therefore also the length of the period can be easily found. The separation property of the self-convolution vector of a given possibly corrupted string S' relaxes the demand for 0 at the 0 mod p locations and at least $\lfloor \frac{n}{p} \rfloor$ elsewhere, but insists on a fixed separation nevertheless.

Definition 4.5. Let S' be n-long string with period P, and let v be the self-convolution vector of S'. We say that v is (α, β)-*separable* if $0 \leq \alpha < \beta$, and v has values:

(1) $< \alpha \frac{n}{p}$, in every location i, $i \equiv 0 \bmod p$, and
(2) $> \beta \frac{n}{p}$, in all other locations.

Separation is a seemingly very strong property. Does it ever occur in reality? Lemma 4.6 shows that if there are less substitution errors than $\frac{n}{4p}$, separation is guaranteed.

LEMMA 4.6. *Let S be n-long string with period P. Let S' be the string S corrupted with at most $\lfloor \delta \frac{n}{p} \rfloor$ substitution errors, where $0 \leq \delta < \frac{1}{4}$, then the self-convolution vector of S' is $(2\delta, 1-2\delta)$-separable.*

PROOF. It is clear that $0 \leq 2\delta, 1-2\delta$. $2\delta < 1-2\delta$ if and only if $4\delta < 1$ if and only if $\delta < \frac{1}{4}$, which is the lemma's condition. We now need to show separation. Consider each of the two cases separately.

Case $i \equiv 0 \bmod p$. By Lemma 4.4, these locations had 0 value before the corruption. By Lemma 2.8, each substitution error may add two mismatches for the self-convolution

vector of such locations. Since the total number of errors is at most $\lfloor 2\delta\frac{n}{p}\rfloor$, the lemma follows for this case.

Case $i \not\equiv 0 \bmod p$. By Lemma 4.4, these locations had value at least $\lfloor\frac{n}{p}\rfloor$ before the corruption. In the worst case, each substitution error "fixes" a position in the string which caused a mismatch for the alignment of this location. By Lemma 2.8, it then removes two mismatches from the self-convolution vector value of this location. We get: $\lfloor\frac{n}{p}\rfloor - 2\cdot\lfloor\delta\frac{n}{p}\rfloor \geq \lfloor(1-2\delta)\frac{n}{p}\rfloor$. □

Example: If there are $\frac{n}{6p}$ substitution errors, then there is a $(\frac{1}{3},\frac{2}{3})$ separation. This case is particularly interesting since the ratio at a separation location is 2.

Remark. Note that, for $p = 1$, there no separation by definition. Since separation is based on distinguishing between the positions that are multiplicities of the period and the other positions, and for $p = 1$ there are no other positions, therefore the concept of separation fails in this case. It is also intuitive that such a periodicity is very sensitive to errors. On the other hand, we argue that such a phenomenon is usually not interesting.

4.4. The Period *H*-Recovery Algorithm

The period recovery algorithm under Hamming metric first gathers information on the corrupted string S' by computing the self-convolution vector v. Then, (α,β)-separation is used in order to reconstruct the original period. By Lemma 4.6, if there are not too many substitution errors, separation is guaranteed. The algorithm sorts all values of the self-convolution vector in ascending order $v_{i_1},\ldots,v_{i_n}$ and considers all locations where $\frac{v_{i_{\ell+1}}}{v_{i_\ell}} \geq \frac{\beta}{\alpha}$. Such an i_ℓ is called a *separating location*, and v_{i_ℓ}, a *separating value*. Clearly there are no more that $O(\log n)$ separating locations. Each one needs to be verified if it indeed defines a period. The algorithm uses the fact that all the multiplicities of the original period have values at most the separating value, in order to find the period length, which is the common difference between the sorted list of multiplicities. Given the period length, the period can be easily reconstructed by a linear scan and the majority criterion. A detailed description of the algorithm is given in Figure 2.

4.5. Analysis of the Period *H*-Recovery Algorithm

We now give the worst-case analysis of Algorithm Period H-Recovery. Lemma 4.7 proves the algorithm's correctness. The complexity guarantee of the algorithm is given by Lemma 4.8. Theorem 4.9 follows.

LEMMA 4.7. *Let S be n-long string with period P, and let S' be S corrupted by substitution errors. Assume that the self-convolution vector of S' is (α,β)-separable. Then, algorithm Period H-Recovery returns a nonempty set $\hat{C}$, $|\hat{C}| = O(\log n)$, such that $P \in \hat{C}$.*

PROOF. Algorithm Period H-Recovery returns the set $\hat{C}$ constructed in lines 17–23. This set has an element P' for each element $p' \in C$, where the set C is constructed in lines 5–16 of the algorithm. Thus, it is enough to show that $|C| = O(\log n)$ and that $p \in C$. The condition in line 7 allows to insert a candidate to C only for values in the array A of the sorted self-convolution values for which the next value is larger than the previous one by at least a factor of $\frac{\beta}{\alpha}$. Since the maximum value of the self-convolution is $O(n)$, there can be only $O(\log_{\frac{\beta}{\alpha}} n)$ such values, and therefore, $|C| = O(\log n)$.

We now show that $p \in C$. By the (α,β)-separation, the values in the self-convolution vector of all the locations that are in the $i \bmod p$ positions of S' are all to the left of a separation value, therefore, for them, the condition in line 7 is true. Also, all these

ALGORITHM PERIOD H-RECOVERY
Input: A string S' of length n
1 $v \leftarrow SelfConvolution(S')$
2 $A \leftarrow Sort(v)$
3 for $i = 0$ to $n/2 - 1$ do
4 $locc_A[i] \leftarrow \{j : A[i]\ is\ v[j]\}$
5 $C \leftarrow \phi$
6 for $i = 0$ to $n/2 - 1$
7 if $\frac{A[i+1]}{A[i]} > \frac{\beta}{\alpha}$ do
8 $M \leftarrow Sort(\{locc_A[j] : 0 \le j \le i\})$
9 if $i = 0$
10 $d_0 \leftarrow M[0] + 1$
11 else for $j = 0$ to $i - 1$
12 $d_j \leftarrow M[j+1] - M[j]$
13 if $d_j = d_{j+1}$ for every j
14 $p' \leftarrow d_0$
15 if $i + 1 = \lfloor \frac{n}{2p'} \rfloor$
16 $C \leftarrow C \cup \{p'\}$
17 $\hat{C} \leftarrow \phi$
18 if $C \neq \phi$
19 for each $p' \in C$
20 for $i = 0$ to $p' - 1$
21 compute $F_i^{p'}$
22 $P' \leftarrow F_0^{p'} \ldots F_{p'-1}^{p'}$
23 $\hat{C} \leftarrow \hat{C} \cup \{P'\}$
Output:
24 return $\hat{C}$

Fig. 2. Algorithm for period recovery under the Hamming metric.

values and only them will appear before the separation value, so we get that $d_0 = p$ and the condition in line 13, is also true. Therefore, p is inserted to C in line 16. □

LEMMA 4.8. *Algorithm Period H-Recovery runs in* $O(|\Sigma|n \log n)$ *steps.*

PROOF. Line 1 of the algorithm is the computation of the self-convolution vector, which by Lemma 4.3 takes $O(|\Sigma|n \log n)$ steps. Line 2 can be done in linear time using a linear sorting method. Lines 3–4 also take linear time. Lines 6–16 take $O(n \log n)$ time since the condition in line 7 is true at most $\log n$ times and the work done in lines 8–16 is $O(n)$. Since, $|C| = O(\log n)$, $|\hat{C}| = |C|$ and computing the period string given the period length can be done in linear time, lines 17–23 also take $O(n \log n)$ time. The overall time is, therefore, $O(|\Sigma|n \log n)$. □

Theorem 4.9 follows.

THEOREM 4.9. *Let S be n-long string with period P. Then, if S is corrupted but preserves a* (α, β) *separation, a set of* $O(\log n)$ *candidates that includes P can be constructed in time* $O(|\Sigma|n \log n)$.

4.6. A $\Omega(\log n)$ Lower Bound on the Candidates Set Size

A shortcoming of the algorithm seems to be that there are cases where a set of $O(\log n)$ possible candidates for the "original" cycle are returned. In the following discussion, we consider the possibility of tightening the analysis of our algorithm.

```
WORST-CASE INPUT CONSTRUCTION
Input: n (the string length) and p (the period length), where p ≤ n/2.
for i = 0 to log p − 1
        j ← 0
        σ ← 5^i
        jump ← 5^i
        while j ≤ p − 1 do
                P[j] ← σ
                j ← j + jump
I ← Null
for i = 1 to n/p
        I ← I concatenated with P
Output: I
```

Fig. 3. A Construction of Worst-Case Input for the Period H-Recovery Algorithm.

Algorithm Period H-Recovery was implemented and tested on various corrupted input strings. It turned out to be very successful. Our experiments on various randomly corrupted input strings demonstrated that in the presence of random substitution errors, if the number of errors is at most $\lfloor \delta \frac{n}{p} \rfloor$, it is very unlikely to get a set of more than the single correct candidate and maybe one more false candidate. Thus, a natural question is whether the worst-case analysis we give is tight. Is it possible that our reconstruction algorithm is actually better in the worst case than we have proven? We now show that it is indeed tight by showing that a carefully constructed input yields an $\Theta(\log n)$-size set of candidates returned by the algorithm.

4.6.1. Description of a Worst-Case Input String. In order to have a set of $\Theta(\log n)$ candidates returned by the algorithm the input string must be such that the array A, which is the sorted self-convolution vector of the string, has $\Theta(\log n)$ separation locations. The simple for-loop in Figure 3 constructs a period P for such an input string, from which the input I is then constructed by repeating the period P. Algorithm Period H-Recovery returns for this input string a set of $\Theta(\log p)$ candidates. By choosing the period length $p = \Theta(n)$, we get the worst case.

4.7. Unbounded Alphabets

Theorem 4.9 assures that we can give a set of $O(\log n)$ candidates that includes P, for a corrupted n-length string S with period P that preserves a (α, β) separation, in time $O(|\Sigma| n \log n)$. For small alphabets, this result is fine, but for unbounded alphabets, for example, of size $p + n/p$, it is inefficient. Of course, one can immediately lower the time complexity to $O(n\sqrt{p \log n})$ by using Abrahamson's result [Abrahamson 1987]. But we can do even better.

THEOREM 4.10. *Let S be n-long string with period P over alphabet Σ. Then, if S is corrupted but preserves a (α, β) separation, a set of $O(\log n)$ candidates that includes P can be constructed in time $O(n \log^2 n)$.*

PROOF. Without loss of generality, we can assume that our alphabet is $\{1, \ldots, n\}$, since we can sort the elements of S and replace them by elements from $\{1, \ldots, n\}$. The time required is $O(n \log n)$, which we pay for the self-convolution anyway.

We use Reed-Solomon codes [Reed and Solomon 1960] to encode the alphabet $\{1, \ldots, n\}$. Reed-Solomon codes are $[N, k, N-k+1]$ linear codes over F_2, where N is the length of the codeword, k the original length, and $N - k + 1$ is the minimum distance

between any two codewords. The assured distance of this code is used to support the separation property, and its small rate enables a fast algorithm.

In our case, $k = \log n$, since the original alphabet is $\{1, \ldots, n\}$. We will choose a constant b such that $b > \lceil \frac{\beta}{\beta-\alpha} \rceil$, and take $N = b \log n$. $N - k + 1 = b \log n - \log n + 1 > (b-1)\log n$. The new self-convolution vector can be computed via the FFT in time $O(N \log N) = O(n \log n \log(n \log n)) = O(n \log^2 n)$. We analyze the values of the self-convolution vector. Two cases interest us.

Case $i \equiv 0 \bmod (pb \log n)$. Because of the (α, β)-separation of S, we know that these locations have at most $\alpha \frac{n}{p} b \log n$ errors.

Case $i \not\equiv 0 \bmod (pb \log n)$, but $i \equiv 0 \bmod (b \log n)$. By the (α, β)-separation and the fact that the R-S code guarantees a distance of at least $(b-1)\log n$ between any two symbols, the value at these locations is at least $\beta \frac{n}{p}(b-1)\log n$.

However, in order to assure separation, we had to choose the constant b such that $\alpha b < \beta(b-1)$. This means:

$$\alpha < \beta \cdot \frac{b-1}{b}$$

or,

$$\frac{\beta}{\beta - \alpha} < b.$$

But our b satisfies this condition. Thus, there is a separation in the R-S encoded alphabet, and by Lemma 4.7, the original cycle will be one of the set of $O(\log n)$ reconstructed periods. □

5. PERIOD RECOVERY UNDER THE EDIT DISTANCE

All the results described in the previous sections are achieved only for pseudolocal metrics. In general, we do not know how to get period recovery for non-pseudolocal metrics. Nevertheless, in this section, we show how to achieve period recovery under the non pseudolocal edit metric.

As mentioned previously, the edit distance is not pseudolocal. In fact, it is also clear that Corollary 3.2, the main result proving that if we are guaranteed $O(\frac{n}{(2+\varepsilon)p})$ corruptions, then there can not be more than $O(\log n)$ candidates for an approximate period, does not hold. As an example consider a period P of length $n^{1/4}$. The allotted number of corruptions is $\frac{n}{(2+\varepsilon)n^{1/4}} = \frac{n^{3/4}}{(2+\varepsilon)}$. Nevertheless, every one of the $n^{1/4}$ cyclic rotations of P generates a string whose edit distance from $P^{3/4}$ is at most $n^{1/4}$, by an appropriate number of leading deletions. This number of errors is well within the bounds tolerated by Corollary 3.2, yet the number of possible approximate periods is quite large. Fortunately, Lemma 5.1 assures that even for edit distance, there are $O(\log n)$ candidates up to cyclic rotations.

LEMMA 5.1. *Let $\varepsilon > 0$ be a constant. Let S an n-long string and let P_1, P_2, $P_1 \neq P_2$, be approximate periods of S with at most $\frac{n}{(4+\varepsilon)\cdot p_1}$, $\frac{n}{(4+\varepsilon)\cdot p_2}$ edit errors respectively (without loss of generality, assume that $p_1 \geq p_2$), such that P_1 is not a cyclic rotation of P_2. Then,*

$$p_1 \geq \left(1 + \frac{\varepsilon}{2}\right) \cdot p_2.$$

PROOF. Let S_1 be the n-long string with period P_1 and S_2 be the n-long string with period P_2. Then, $d(S_1, S) \leq \frac{n}{(4+\varepsilon)\cdot p_1}$ and $d(S_2, S) \leq \frac{n}{(4+\varepsilon)\cdot p_2}$. Therefore,

$$\frac{n}{(4+\varepsilon)\cdot p_1} + \frac{n}{(4+\varepsilon)\cdot p_2} \geq d(S_1, S) + d(S_2, S)$$

By the triangle inequality, we have,

$$d(S_1, S) + d(S_2, S) \geq d(S_1, S_2)$$

If $d(S_1, S_2) \geq \frac{n}{2p_1}$ holds, we then have,

$$\frac{n}{(4+\varepsilon)\cdot p_1} + \frac{n}{(4+\varepsilon)\cdot p_2} \geq \frac{n}{2p_1}$$

from which we get,

$$p_2 + p_1 \geq \left(2 + \frac{\varepsilon}{2}\right)\cdot p_2$$

or,

$$p_1 \geq \left(1 + \frac{\varepsilon}{2}\right)\cdot p_2.$$

Thus, it remains to show that indeed $d(S_1, S_2) \geq \frac{n}{2p_1}$.

We prove that every two subsequent appearances of P_1 contribute at least one edit operation to $d(S_1, S_2)$. Consider the first two appearances of P_1. We will show that these two appearances contribute at least one edit operation. Note that the substring consisting of two consecutive appearances of P_1 is the minimal string with period P_1 for which Lemma 2.9 holds. By Lemma 2.6 and Lemma 2.9, we know that every two subsequent appearances of P_1 contribute two mismatches. Since there are mismatches, at least one edit operation is necessary to remove them. So the claim is proven for the first two appearances.

It remains to show the lemma also holds for the suffixes of S_1 and S_2. Consider the suffix of S_1 from the third appearance of P_1. After the edit operation of the preceding two appearances of P_1 is performed, this suffix is aligned with S_2', a suffix of S_2, with a period P_2' which is a cyclic shift of P_2. Observe that the specific edit operation, that is, mismatch, insertion or deletion determines which cyclic shift of P_2 is P_2' (it can also be P_2 itself), however, the general structure of the suffixes alignment remains invariant: the suffix of S_1 has a period P_1 and the suffix of S_2, S_2' has a period P_2' of length p_2. If the edit operation performed in the first two appearances of P_1, caused these suffixes to be identical, it must be that $p_1 = p_2$ and $P_1 = P_2'$. This is because $p_1 > p_2$ implies that there is a smaller period for the suffix of S_1. It, therefore, implies a smaller period for S_1, contradicting the well-known periodicity lemma that precludes such a smaller period than the period P_1. However, if $P_1 = P_2'$ and P_2' is a cyclic shift of P_2, then P_1 is a cyclic shift of P_2, contradiction. Therefore, $P_1 \neq P_2'$ and the same argument used for the first two appearances of P_1 in S_1 can be repeatedly used for the suffixes of S_1 and S_2. The lemma then follows. □

COROLLARY 5.2. *Let S be a n-long string. Then, there are at most $\log_{1+\frac{\varepsilon}{2}} n$ different approximate periods P of S with at most $\frac{n}{(4+\varepsilon)\cdot p}$ edit errors, that are not cyclic rotations of any other approximate period.*

PROOF. Note that there cannot be two such different approximate periods of the same length p, because then, by Lemma 5.1, we get $p \geq (1+\frac{\varepsilon}{2})\cdot p$, contradiction. Thus, such different approximate periods must have different length. Now, let $1 \leq l_1 < l_2 < \cdots <$

$l_{t-1} < l_t \leq \frac{n}{2}$ be the different lengths of approximate periods of S. By Lemma 5.1,

$$1 \leq l_1 < \left(1+\frac{\varepsilon}{2}\right) \cdot l_2 < \left(1+\frac{\varepsilon}{2}\right)^2 \cdot l_3 < \cdots < \left(1+\frac{\varepsilon}{2}\right)^{t-2} \cdot l_{t-1} < l_t \leq \frac{n}{2}.$$

Therefore, $t \leq \log_{1+\frac{\varepsilon}{2}} n$. □

The naive construction of the approximate period candidates is done as follows. For each of the $O(n \log n)$ candidates output by the extraction stage of the Period Recovery Algorithm (see Figure 1), and each of its rotations (for a total of $O(n^2 \log n)$ candidates), compute the edit distance with the input string. The edit distance is computable in time $O(n^2)$ [Levenshtein 1966] by dynamic programming. Thus, the total time is $O(n^4 \log n)$. But we can do better.

THEOREM 5.3. *Let S be n-long string with period P. Let $\varepsilon > 0$ be a constant. Then, if S is corrupted by at most $\frac{n}{(4+\varepsilon)\cdot p}$ Levenshtein edit operations, then, a set of $\log_{1+\frac{\varepsilon}{2}} n$ candidates such that their cyclic rotations include P, can be constructed in time $O(n^3 \log n)$.*

PROOF. Split the $O(n \log n)$ candidates extracted from $MAST(S)$ into two sets, those whose length is at most $\sqrt{n}$, which we will call *small*, and those whose length exceeds $\sqrt{n}$, referred to as *large*.

For every small candidate P, Tiskin [2010] proved that the edit distance of the string of length n generated by P and S can be computed in time $O(np) = O(n^{1.5})$. Since the edit distance of all cyclic rotations of each candidate need to be computed as well, but since there are at most $\sqrt{n}$ such cyclic rotations for small candidates, the computation time is $O(n^2)$ for each small candidate and its rotations. As the output of the extraction stage is $O(n \log n)$ candidates, this brings the total time for handling small candidates to $O(n^3 \log n)$.

For a large candidate P, there are at most $O(\frac{n}{(4+\varepsilon)p}) = O(\frac{\sqrt{n}}{(4+\varepsilon)})$ errors allowed. Using the Landau-Vishkin algorithm [Landau and Vishkin 1989] for computing the edit distance up to k errors, we can therefore compute a candidate in time $O(n^{1.5})$. As in the case of small candidates, allowing for all rotations up to $\sqrt{n}$, since that is the allowed number of errors, and multiplying by the number of candidates, brings the processing time of all large candidates to $O(n^3 \log n)$.

In order to return a set of approximate candidates of size $O(\log n)$, we select from each length the candidate with the least number of errors (which does not necessarily includes P). By Corollary 5.2, we are assured that this set of candidates is within the requested size. Moreover, Lemma 5.1 assures that this set includes P or a cyclic rotation of P. □

REFERENCES

ABRAHAMSON, K. 1987. Generalized string matching. *SIAM J. Comput. 16,* 6, 1039–1051.

AMIR, A., AUMANN, Y., BENSON, G., LEVY, A., LIPSKY, O., PORAT, E., SKIENA, S., AND VISHNE, U. 2006. Pattern matching with address errors: rearrangement distances. In *Proceedings of the 17th ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 1221–1229.

AMIR, A., AUMANN, Y., LANDAU, G., LEWENSTEIN, M., AND LEWENSTEIN, N. 2000. Pattern matching with swaps. *J. Algor. 37*, 247–266.

AMIR, A. AND BENSON, G. 1998. Two-dimensional periodicity and its application. *SIAM J. Comput. 27,* 1, 90–106.

AMIR, A., BENSON, G., AND FARACH, M. 1993. Optimal parallel two dimensional pattern matching. In *Proceedings of the 5th ACM Symposium on Parallel Algorithms and Architectures*, 79–85.

AMIR, A., BENSON, G., AND FARACH, M. 1998. Optimal parallel two dimensional text searching on a crew pram. *Info. Computat. 144,* 1, 1–17.

AMIR, A., COLE, R., HARIHARAN, R., LEWENSTEIN, M., AND PORAT, E. 2003. Overlap matching. *Info. Computat. 181,* 1, 57–74.

AMIR, A., HARTMAN, T., KAPAH, O., LEVY, A., AND PORAT, E. 2007. On the cost of interchange rearrangement in strings. In *Proceedings of the 16th Annual European Symposium on Algorithms (ESA)*, Lature Notes in Computer Science, vol. 4698, Springer, 99–110.

APOSTOLICO, A. AND GIANCARLO, R. 2008. Periodicity and repetitions in parameterized strings. *Disc. Appl. Math. 156,* 9, 1389–1398.

APOSTOLICO, A. AND PREPARATA, F. P. 1996. Data structures and algorithms for the string statistics problem. *Algorithmica 15,* 5, 481–494.

BAFNA, V. AND PEVZNER, P. 1998. Sorting by transpositions. *SIAM J. Disc. Math. 11*, 221–240.

BERMAN, P. AND HANNENHALLI, S. 1996. Fast sorting by reversal. In *Proceedings of the 8th Annual Symposium on Combinatorial Pattern Matching (CPM)*, D. Hirschberg and E. Myers, Eds. Lecture Notes in Computer Science, vol. 1075, Springer, 168–185.

BRODAL, G. S., LYNGSØ, R. B., ÖSTLIN, A., AND PEDERSEN, C. N. S. 2002. Solving the string statistics problem in time $O(nlogn)$. In *Proceedings of the 29th International Colloquium on Automata, Languages, and Programming (ICALP)*. 728–739.

CAYLEY, A. 1849. Note on the theory of permutations. *Philosophical Mag. 34*, 527–529.

CHRISTIE, D. A. 1996. Sorting by block-interchanges. *Info. Proc. Lett. 60*, 165–169.

COLE, R., CROCHEMORE, M., GALIL, Z., GĄSIENIEC, L., HARIHAN, R., MUTHUKRISHNAN, S., PARK, K., AND RYTTER, W. 1993. Optimally fast parallel algorithms for preprocessing and pattern matching in one and two dimensions. In *Proceedings of the 34th Annual IEEE Symposium on Foundations of Computer Science*, 248–258.

CORMEN, T. H., LEISERSON, C. E., RIVEST, R. L., AND STEIN, C. 2001. *Introduction to Algorithms* 2nd Ed. MIT Press and McGraw-Hill.

CROCHEMORE, M. 1981. An optimal algorithm for computing the repetitions in a word. *Info. Proc. Lett. 12,* 5, 244–250.

FISCHER, M. AND PATERSON, M. 1974. String matching and other products. *Complexity of Computation*, SIAM-AMS Proceedings Vol. 7, 113–125.

GALIL, Z. 1984. Optimal parallel algorithms for string matching. In *Proceedings of the 16th ACM Symposium on Theory of Computing 67*, 144–157.

GALIL, Z. AND PARK, K. 1996. Alphabet-independent two-dimensional witness computation. *SIAM J. Comput. 25,* 5, 907–935.

LANDAU, G. M. AND VISHKIN, U. 1989. Fast parallel and serial approximate string matching. *J. Algor. 10,* 2, 157–169.

LEVENSHTEIN, V. I. 1966. Binary codes capable of correcting, deletions, insertions and reversals. *Soviet Phys. Dokl. 10*, 707–710.

LOTHAIRE, M. 1983. *Combinatorics on Words*. Addison-Wesley, Reading, Mass.

MA, S. AND HELLERSTEIN, J. L. 2001. Mining partially periodic event patterns with unknown periods. In *Proceedings of the 17th International Conference on Data Engineering (ICDE)*. IEEE Computer Society, 205–214.

REED, I. AND SOLOMON, G. 1960. Polynomial codes over certain finite fields. *SIAM J. Appl. Math. 8,* 2, 300–304.

RÉGNIER, M. AND ROSTAMI, L. 1993. A unifying look at D-dimensional periodicities and space coverings. In *Proceedings of the 4th Symposium on Combinatorial Pattern Matching 15*, 215–227.

TISKIN, A. 2010. Fast distance multiplication of unit-monge matrices. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 1287–1296.

Received October 2010; revised November 2011; accepted January 2012