Faster Approximate Pattern Matching: A Unified Approach

Panagiotis Charalampopoulos
Department of Informatics
King's College London, UK
and
Institute of Informatics
University of Warsaw, Poland
panagiotis.charalampopoulos@kcl.ac.uk

Tomasz Kociumaka
Department of Computer Science
Bar-Ilan University
Ramat Gan, Israel
kociumaka@mimuw.edu.pl

Philip Wellnitz
Max Planck Institute for Informatics
Saarland Informatics Campus
Saarbrücken, Germany
wellnitz@mpi-inf.mpg.de

2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS) | 978-1-7281-9621-3/20/$31.00 ©2020 IEEE | DOI: 10.1109/FOCS46700.2020.00095

***Abstract*—In the approximate pattern matching problem, given a text T, a pattern P, and a threshold k, the task is to find (the starting positions of) all substrings of T that are at distance at most k from P. We consider the two most fundamental string metrics: Under the *Hamming distance*, we search for substrings of T that have at most k *mismatches* with P, while under the *edit distance*, we search for substrings of T that can be transformed to P with at most k *edits*.**

Exact occurrences of P in T have a very simple structure: If we assume for simplicity that $|P| < |T| \leq {}^{3}\!/_{2}\,|P|$ and that P occurs both as a prefix and as a suffix of T, then both P and T are periodic with a common period. However, an analogous characterization for occurrences with up to k mismatches was proved only recently by Bringmann et al. [SODA'19]: Either there are $\mathcal{O}(k^2)$ k-mismatch occurrences of P in T, or both P and T are at Hamming distance $\mathcal{O}(k)$ from strings with a common string period of length $\mathcal{O}(m/k)$. We tighten this characterization by showing that there are $\mathcal{O}(k)$ k-mismatch occurrences in the non-periodic case, and we lift it to the edit distance setting, where we tightly bound the number of k-edit occurrences by $\mathcal{O}(k^2)$ in the non-periodic case. Our proofs are constructive and let us obtain a unified framework for approximate pattern matching for both considered distances. In particular, we provide meta-algorithms that only rely on a small set of primitive operations. We showcase the generality of our meta-algorithms with results for the *fully compressed setting*, the *dynamic setting*, and the *standard setting*.

***Keywords*-approximate pattern matching, grammar compression, dynamic strings, Hamming distance, edit distance**

I. Introduction

The *pattern matching* problem, asking to search for occurrences of a given pattern P in a given text T, is perhaps the most fundamental problem on strings. However, in most applications, finding all *exact* occurrences of a pattern is not enough: Think of human spelling mistakes or DNA sequencing errors, for example. In this work, we focus on *approximate* pattern matching, where we are interested in finding substrings of the text that are "similar" to the pattern. While various similarity measures are imaginable, we focus on the two most commonly encountered metrics in this context: the *Hamming distance* and the *edit distance*.

A full version of this paper is available at arxiv.org/abs/2004.08350. Proofs of the claims marked with ♠ are presented only in the full version.

Hamming Distance: Recall that the Hamming distance of two (equal-length) strings is the number of positions where the strings differ. Now, given a text T of length n, a pattern P of length m, and an integer threshold $k > 0$, we want to compute the *k-mismatch occurrences* of P in T, that is, all length-m substrings of T that are at Hamming distance at most k from P. This *pattern matching with mismatches* problem has been extensively studied. In the late 1980s, Abrahamson [2] and Kosaraju [26] independently proposed an FFT-based $\mathcal{O}(n\sqrt{m\log m})$-time algorithm for computing the Hamming distance of P and all the length-m fragments of T. While their algorithms can be used to solve the pattern matching with mismatches problem, the first algorithm to benefit from the threshold k was given by Landau and Vishkin [27] and slightly improved by Galil and Giancarlo [15]: Based on so-called "kangaroo jumping", they obtained an $\mathcal{O}(nk)$-time algorithm, which is faster than $\mathcal{O}(n\sqrt{m\log m})$ even for moderately large k. Amir et al. [4] developed two algorithms with running time $\mathcal{O}(n\sqrt{k\log k})$ and $\tilde{\mathcal{O}}(n + k^3 n/m)$, respectively; the latter algorithm was then improved upon by Clifford et al. [11], who presented an $\tilde{\mathcal{O}}(n + k^2 n/m)$-time solution. Subsequently, Gawrychowski and Uznański [17] provided a smooth trade-off between the running times $\tilde{\mathcal{O}}(n\sqrt{k})$ and $\tilde{\mathcal{O}}(n + k^2 n/m)$ by designing an $\tilde{\mathcal{O}}(n + kn/\sqrt{m})$-time algorithm. Very recently, Chan et al. [10] removed most of the $\operatorname{polylog} n$ factors in the latter solution at the cost of (Monte-Carlo) randomization. Furthermore, Gawrychowski and Uznański [17] showed that a significantly faster "combinatorial" algorithm would have (unexpected) consequences for the complexity of Boolean matrix multiplication. Pattern matching with mismatches on strings is thus well understood in the standard setting. Nevertheless, in the settings where the strings are not given explicitly, a similar understanding is yet to be obtained. One of the main contributions of this work is to improve the upper bounds for two such settings, obtaining algorithms with running times analogous to the algorithm of [11].

Edit Distance: Recall that for two strings S and T, the edit distance (also known as Levenshtein distance) is the minimum number of edits required to transform S into T.

2575-8454/20/$31.00 ©2020 IEEE
DOI 10.1109/FOCS46700.2020.00095

Here, an edit is an insertion, a substitution, or a deletion of a single character. In the *pattern matching with edits* problem, we are given a text T, a pattern P, and an integer threshold $k > 0$, and the task is to find all starting positions of the *k-edit* (or *k-error*) *occurrences* of P in T. Formally, we are to find all positions i in T such that the edit distance between $T[i\,..\,j]$ and P is at most k for some position j. Again, a classic algorithm by Landau and Vishkin [28] runs in $\mathcal{O}(nk)$ time. Subsequent research [37], [13] resulted in an $\mathcal{O}(n + k^4 n/m)$-time algorithm (which is faster for $k \leq \sqrt[3]{m}$). From a lower-bound perspective, we can benefit from the discovery that the classic quadratic-time algorithm for computing the edit distance of two strings is essentially optimal: Backurs and Indyk [5] recently proved that a significantly faster algorithm would yield a major breakthrough for the satisfiability problem. For pattern matching with edits, this means that there is no hope for an algorithm that is significantly faster than $\mathcal{O}(n+k^2 n/m)$; however, apart from that "trivial" lower bound and the 20-year-old conjecture of Cole and Hariharan [13] that an $\mathcal{O}(n + k^3 n/m)$-time algorithm *should be possible*, nothing is known that would close this gap. While we do not manage to tighten this gap, we do believe that the structural insights we obtain may be useful for doing so. What we do manage, however, is to significantly improve the running time of the known algorithms in two settings where T and P are not given explicitly, thereby obtaining running times that can be seen as analogous to the running time of Cole and Hariharan's algorithm [13].

Grammar Compression: One of the settings that we consider in this paper is the *fully compressed* setting, when both the text T and the pattern P are given as straight-line programs. Compressing the text and the pattern is, in general, a natural thing to do—think of huge natural-language texts or genomic databases, which are easily compressible. While one approach to solve pattern matching in the fully compressed setting is to first decompress the strings and then run an algorithm for the standard setting, this voids most benefits of compression in the first place. Hence, there has been a long line of research with the goal of designing text algorithms directly operating on compressed strings. Naturally, such algorithms highly depend on the chosen compression method. In this work, we consider *grammar compression*, where a string T is represented using a context-free grammar that generates exactly $\{T\}$; such a grammar is also called a *straight-line program* (SLP).

Straight-line programs are popular due to mathematical elegance and equivalence [35], [25], [24] (up to logarithmic factors and moderate constants) to widely-used dictionary compression schemes, including the LZ77 parsing [43] and the run-length-encoded Burrows–Wheeler transform [9]. Many more schemes, such as byte-pair encoding [39], Re-Pair [29], Sequitur [32], and further members of the Lempel–Ziv family [44], [42], to name but a few, can be expressed as straight-line programs. We refer an interested reader to [36], [34], [30], [38] to learn more about grammar compression.

Working directly with a compressed representation of a text in general, intuitively at least, seems to be hard—in fact, Abboud et al. [1] showed that, for some problems, decompress-and-solve is the best we can hope for, under some reasonable assumptions from fine-grained complexity theory. Nevertheless, Jeż [22] managed to prove that exact pattern matching can be solved on grammar-compressed strings in near-linear time: Given an SLP of size n representing a string T and an SLP of size m representing a string P, we can find all exact occurrences of P in T in $\mathcal{O}((n+m)\log|P|)$ time. For fully compressed *approximate* pattern matching, no such near-linear time algorithm is known, though. While the $\tilde{\mathcal{O}}((n+|P|)k^4)$-time algorithm by Bringmann et al. [8] for pattern matching with mismatches comes close, it works in an easier setting where only the text is compressed. We fill this void by providing the first algorithm for fully compressed pattern matching with mismatches that runs in near-linear time. Denote by $\mathrm{Occ}_k^H(P,T)$ the set of (starting positions of) k-mismatch occurrences of P in T; then, our result reads as follows.

Theorem I.1 (♠). *Let $\mathcal{G}_T$ denote an SLP of size n generating a text T, let $\mathcal{G}_P$ denote an SLP of size m generating a pattern P, let k denote a threshold, and set $N := |T| + |P|$.*

Then, we can compute $|\mathrm{Occ}_k^H(P,T)|$ in time $\mathcal{O}(m\log N + n\,k^2\log^3 N)$. The elements of $\mathrm{Occ}_k^H(P,T)$ can be reported within $\mathcal{O}(|\mathrm{Occ}_k^H(P,T)|)$ extra time.

For pattern matching with edits, near-linear time algorithms are not known even in the case that the pattern is given explicitly. Currently, the best pattern matching algorithms on an SLP-compressed text run in time $\mathcal{O}(n|P|\log|P|)$ [41] and $\mathcal{O}(n(\min\{|P|k, k^4 + |P|\} + \log|T|))$ [6]. Moreover, an $\tilde{\mathcal{O}}(n\sqrt{|P|}k^3)$-time solution [16] is known for (weaker) LZW compression [42]. Again, we obtain a near-linear time algorithm for fully compressed pattern matching with edits. Denote by $\mathrm{Occ}_k^E(P,T)$ the set of all starting positions of k-error occurrences of P in T; then, our result reads as follows.

Theorem I.2 (♠). *Let $\mathcal{G}_T$ denote an SLP of size n generating a string T, let $\mathcal{G}_P$ denote an SLP of size m generating a string P, let k denote a threshold, and set $N := |T| + |P|$.*

Then, we can compute $|\mathrm{Occ}_k^E(P,T)|$ in time $\mathcal{O}(m\log N + n\,k^4\log^3 N)$. The elements of $\mathrm{Occ}_k^E(P,T)$ can be reported within $\mathcal{O}(|\mathrm{Occ}_k^E(P,T)|)$ extra time.

Note that our algorithms also improve the state of the art when the pattern is given in an uncompressed form; this is because any string P admits a trivial SLP of size $\mathcal{O}(|P|)$.

Dynamic Strings: While compression handles large static data sets, a different approach is available if the data changes frequently. Several works on pattern matching in dynamic strings considered the indexing problem, assuming that the text is maintained subject to updates and the pattern

is given explicitly at query time; we refer the interested reader to [19], [14], [37], [3], [33] and references therein.

Recently, Clifford et al. [12] considered the problem of maintaining a data structure for a text T and a pattern P, both of which undergo character substitutions, in order to be able to efficiently compute the Hamming distance between P and any given fragment of T. Among other results, for the case where $|T| \leq 2|P|$ and constant alphabet size, they presented a data structure with $\mathcal{O}(\sqrt{m \log m})$ time per operation, and they proved that, conditional on the Online Boolean Matrix-Vector Multiplication (OMv) Conjecture [20], one cannot simultaneously achieve $\mathcal{O}(m^{1/2-\varepsilon})$ for the query and update time for any constant $\varepsilon > 0$.

We consider the following more general setting: We maintain an initially empty collection of strings $\mathcal{X}$ that can be modified via the following "update" operations:

- $\texttt{makestring}(U)$: Insert a string U to $\mathcal{X}$.
- $\texttt{concat}(U, V)$: Insert UV to $\mathcal{X}$, for $U, V \in \mathcal{X}$.
- $\texttt{split}(U, i)$: Insert $U[0\,..\,i)$ and $U[i\,..\,|U|)$ in $\mathcal{X}$, for $U \in \mathcal{X}$ and $i \in [1\,..\,|U|)$.

In our setting, the strings in $\mathcal{X}$ are *persistent*, meaning that `concat` and `split` do not destroy their arguments.

The main goal in this model (and for the dynamic setting in general) is to provide algorithms that are faster than recomputing the answer from scratch after every update. Specifically for dynamic strings, already the task of testing equality of strings in $\mathcal{X}$ is challenging. After a long line of research [40], [31], [3], Gawrychowski et al. [18] proved the following: There is a data structure that supports equality queries in $\mathcal{O}(1)$ time, while each of the update operations takes $\mathcal{O}(\log N)$ time, where N is an upper bound on the total length of all strings in $\mathcal{X}$.[1] In fact, as shown in [18], one can also support $\mathcal{O}(1)$-time queries for the longest common prefix of two strings in $\mathcal{X}$ with no increase in the update times. The data structure of [18] is Las-Vegas randomized: the answers are correct, but the update times are guaranteed only with high probability. Randomization can be avoided at the cost of extra logarithmic factors in the running times; see [31], [33]. The same is true for our results.

We extend the data structure of Gawrychowski et al. [18] with approximate pattern matching queries:

Theorem I.3 (♠). *A collection $\mathcal{X}$ of non-empty persistent strings of total length N can be maintained subject to $\texttt{makestring}(U)$, $\texttt{concat}(U, V)$, and $\texttt{split}(U, i)$ operations requiring $\mathcal{O}(\log N + |U|)$, $\mathcal{O}(\log N)$, and $\mathcal{O}(\log N)$ time, respectively, so that given two strings $P, T \in \mathcal{X}$ with $|P| = m$ and $|T| = n$ and an integer threshold $k > 0$, we can compute the set $\mathrm{Occ}_k^H(P, T)$ in time $\mathcal{O}(n/m \cdot k^2 \log^2 N)$ and the set $\mathrm{Occ}_k^E(P, T)$ in time $\mathcal{O}(n/m \cdot k^4 \log^2 N)$.*[2]

In the Hamming distance case, for $k < \sqrt{m}/\log N$, the data structure of Theorem I.3 is faster than recomputing the occurrences from scratch after each update: Recall that, in the standard setting, the fastest known algorithm for pattern matching with mismatches costs $\tilde{\mathcal{O}}(n + kn/\sqrt{m}) = \tilde{\mathcal{O}}(n + k\sqrt{m} \cdot n/m)$ time; in particular, the additive $\tilde{\mathcal{O}}(n)$ term dominates the time complexity for the considered parameter range. Observe further that, for any k, Theorem I.3 is not slower (ignoring $\mathrm{polylog}$ factors) than running the $\tilde{\mathcal{O}}(n + k^2 n/m)$-time algorithm by Clifford et al. after every update.

In the edit distance case, for $k < (m/\log^2 N)^{1/3}$, the data structure of Theorem I.3 is faster than running the $\mathcal{O}(nk)$-time Landau–Vishkin algorithm for the standard setting. Note further that, for any k, Theorem I.3 is not slower (ignoring $\mathrm{polylog}\, N$ factors) than running after every update the $\mathcal{O}(n + k^4 n/m)$-time Cole–Hariharan algorithm, whose bottleneck for $k < m^{1/4}$ is the additive $\mathcal{O}(n)$ term.

Structure of Pattern Matching with Mismatches: As in [8], we obtain our algorithms by exploiting new structural insights for approximate pattern matching. Our contribution in this area is two-fold: We strengthen the structural result of [8] for pattern matching with mismatches, and we prove a similar result for pattern matching with edits.

Before we describe our new structural insights, let us recall the structure of exact pattern matching. Let P denote a pattern of length m and let T denote a text of length $n \leq 3/2\, m$. Assume that T is trimmed so that P occurs both at the beginning and at the end of T, that is $P = T[0\,..\,m) = T[n-m\,..\,n)$. By the length constraints, we have $P[n-m\,..\,m) = P[0\,..\,2m-n)$. Repeating this argument for the overlapping parts of P, we obtain the following well-known characterization (where X^∞ denotes the concatenation of infinitely many copies of a string X):

Fact I.4 (folklore [7])**.** *Let P denote a pattern of length m and let T denote a text of length $n \leq 3/2\, m$. If $T[0\,..\,m) = T[n-m\,..\,m) = P$, then there is a string Q such that $P = Q^\infty[0\,..\,m)$ and $T = Q^\infty[0\,..\,n)$, that is, both the text and the pattern are periodic with a common string period Q, and the starting positions of all exact occurrences of P in T form an arithmetic progression with difference $|Q|$.*

Hence, it is justified to say that the structure of exact pattern matching is fully understood. Surprisingly, a similar characterization for approximate pattern matching was missing for a long time. Only recently, Bringmann et al. [8] proved a similar result for pattern matching with mismatches (we write $\delta_H(S, T)$ for the Hamming distance of S and T):

Theorem I.5 ([8, Theorem 1.2], simplified)**.** *Given a pattern P of length m, a text T of length $n \leq 3/2\, m$, and a threshold $k \leq m$, at least one of the following holds:*

- *The number of k-mismatch occurrences of P in T is bounded by $\mathcal{O}(k^2)$.*
- *There is a primitive string Q of length $\mathcal{O}(m/k)$ such that $\delta_H(P, Q^\infty[0\,..\,m)) \leq 6k$.*

[1] Strictly speaking, $\texttt{makestring}(U)$ costs $\mathcal{O}(|U| + \log N)$ time.

[2] All running time bounds hold with high probability (i.e., $1 - N^{\Omega(1)}$).

Motivated by the absence of examples proving the tightness of their result, Bringmann et al. [8] conjectured that the bound on the number of occurrences in Theorem I.5 can be improved to $\mathcal{O}(k)$. We resolve their conjecture positively by proving the following stronger variant of Theorem I.5.

Theorem I.6 (Compare Theorem I.5)**.** *Given a pattern P of length m, a text T of length $n \leq {}^3\!/_2\, m$, and a threshold $k \leq m$, at least one of the following holds.*

- *The number of k-mismatch occurrences of P in T is bounded by $\mathcal{O}(k)$.*
- *There is a primitive string Q of length $\mathcal{O}(m/k)$ that satisfies $\delta_H(P, Q^\infty[0\,..\,m)) < 2k$.*

Examples from [8], illustrated in Figures 1 and 2, prove the asymptotic tightness of Theorem I.6.

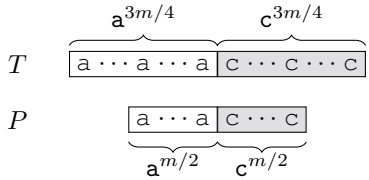

Figure 1. Consider a text $T := \mathtt{a}^{3m/4}\mathtt{c}^{3m/4}$ and a pattern $P := \mathtt{a}^{m/2}\mathtt{c}^{m/2}$, neither of which is approximately periodic. Then, shifting the exact occurrence of P in T by up to k positions in either direction still yields a k-mismatch occurrence. Hence, we need $\Omega(k)$ distinct k-mismatch occurrences to derive approximate periodicity of P.

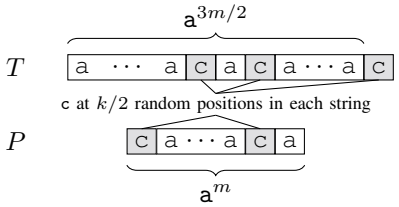

Figure 2. Consider a text T and a pattern P obtained from $\mathtt{a}^{3m/2}$ and $\mathtt{a}^m$, respectively, by substituting $\mathtt{a}$ to $\mathtt{c}$ at $k/2$ random positions. Then, all length-m fragments of T are k-mismatch occurrences of P, but, with high probability, neither T nor P is perfectly periodic. Hence, we need a relaxed periodicity notion allowing for $\Omega(k)$ mismatches.

As in the exact pattern matching case, we can also characterize the (approximately) periodic case in more detail.

Theorem I.7 (Compare [8, Claim 3.1])**.** *Let P denote a pattern of length m, let T denote a text of length $n \leq {}^3\!/_2\, m$, and let $0 \leq k \leq m$ denote a threshold. Suppose that both $T[0\,..\,m)$ and $T[n-m\,..\,n)$ are k-mismatch occurrences of P. If there is a positive integer $d \geq 2k$ and a primitive string Q with $|Q| \leq m/8d$ and $\delta_H(P, Q^\infty[0\,..\,m)) \leq d$, then each of following holds:*

(a) The string T satisfies $\delta_H(T, Q^\infty[0\,..\,n)) \leq 3d$.
(b) Every k-mismatch occurrence of P in T starts at a position that is a multiple of $|Q|$.
(c) The set $\mathrm{Occ}_k^H(P,T)$ can be decomposed into $\mathcal{O}(d^2)$ arithmetic progressions with difference $|Q|$.

Recall that Theorem I.5, originating from [8], includes a weaker version of Theorem I.7. We also note that part (c) of the new characterization is asymptotically tight, as justified by modifying the example of Figure 2: Let P be obtained from $\mathtt{a}^m$ by placing $\mathtt{c}$ at $(k+1)/2$ random positions, and let T be obtained from $\mathtt{a}^{3m/2}$ by placing $\mathtt{c}$ at $(k+1)/2$ random positions within the middle third of $\mathtt{a}^{3m/2}$. Then, each k-mismatch occurrence must align at least one $\mathtt{c}$ from P with one $\mathtt{c}$ from T and, conversely, each such alignment results in a k-mismatch occurrence. Hence, the number of k-mismatch occurrences is $\Theta(k^2)$. Furthermore, for every q, with high probability, $\mathrm{Occ}_k^H(P,T)$ can only be decomposed into $\Theta(k^2)$ progressions with difference q.

Structure of Pattern Matching with Edits: Having understood the structure of pattern matching with mismatches, we turn to the more complicated situation for pattern matching with edits. First, observe that the examples of Figures 1 and 2 are still valid: Any k-mismatch occurrence is also a k-error occurrence. However, as the edit distance allows insertions and deletions of characters, we can construct an example where neither P nor T is approximately periodic, yet the number of k-error occurrences is $\Omega(k^2)$; see Figure 3.

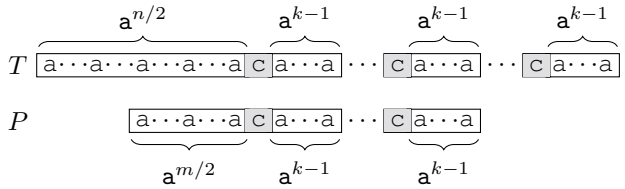

Figure 3. Consider a text $T := \mathtt{a}^{n/2}(\mathtt{c}\cdot\mathtt{a}^{k-1})^{n/2k}$ and a pattern $P := \mathtt{a}^{m/2}(\mathtt{c}\cdot\mathtt{a}^{k-1})^{m/2k}$ for $n := m + 2k^2$. Now, for every $i \in [-k\,..\,k]$, an $|i|$-mismatch occurrence of P starts at position $n/2 - m/2 + i \cdot k$ in T. The remaining budget on the number of errors can be spent on shifting the starting positions, so for every $j \in [|i| - k\,..\,k - |i|]$, there is a k-error occurrence starting at position $n/2 - m/2 + i \cdot k + j$ in T. Overall, the number of k-error occurrences of P in T is $\Omega(k^2)$, but neither P nor T is approximately periodic.

In the example of Figure 3, there are still only $\mathcal{O}(k)$ regions of size $\mathcal{O}(k)$ each where k-error occurrences start. In fact, we can show that this is the worst that can happen (we write $\delta_E(S,T)$ for the edit distance of S and T):

Theorem I.8 (♠). *Given a pattern P of length m, a text T of length $n \leq {}^3\!/_2\, m$, and a threshold $k \leq m$, at least one of the following holds:*

- *The starting positions of all k-error occurrences of P in T lie in $\mathcal{O}(k)$ intervals of length $\mathcal{O}(k)$ each.*
- *There is a primitive string Q of length $\mathcal{O}(m/k)$ and integers i, j such that $\delta_E(P, Q^\infty[i\,..\,j]) < 2k$.*

Again, we treat the (approximately) periodic case separately, thereby obtaining a result similar to Theorem I.7:

Theorem I.9 (♠). *Let P denote a pattern of length m, let T denote a text of length n, and let $0 \leq k \leq m$ denote an integer threshold such that $n < {}^3\!/_2\, m + k$. Suppose that the k-error occurrences of P in T include a prefix of T and a suffix of T. If there are positive integers $d \geq 2k$, i, j, and a primitive string Q with $|Q| \leq m/8d$ and $\delta_E(P, Q^\infty[i\,..\,j]) \leq d$, then each of following holds:*

(a) The string T satisfies $\delta_E(T, Q^\infty[i'\,..\,j']) \leq 3d$ for some integers i' and j'.

(b) For every $p \in \mathrm{Occ}_k^E(P, T)$, we have $p \bmod |Q| \leq 3d$ or $p \bmod |Q| \geq |Q| - 3d$.

(c) The set $\mathrm{Occ}_k^E(P, T)$ can be decomposed into $\mathcal{O}(d^3)$ arithmetic progressions with difference $|Q|$.

Technical Overview

Gaining Structural Insights: To highlight the novelty of our approach, let us first outline the proof Theorem I.5 by Bringmann et al. [8]. Consider a pattern P of length m and a text T of length $n \leq {}^3\!/_2\, m$. Split the pattern into $\Theta(k)$ blocks of length $\Theta(m/k)$ each and process each such block P_i as follows: Compute the shortest string period Q_i of P_i and align P with a substring of Q_i^∞, starting from $P_i = Q_i^\infty[0\,..\,|P_i|)$ and extending to both directions, allowing mismatches. If there are $\mathcal{O}(k)$ mismatches for any block P_i, then P is approximately periodic; otherwise, there are many mismatches for every block P_i. In particular, in every k-mismatch occurrence where a block P_i is matched exactly, all but at most k of these mismatches between P and Q_i^∞ must be aligned to the corresponding mismatches between T and Q_i^∞. Observing that, in any k-mismatch occurrence, at least k of the blocks must be matched exactly, this yields an $\mathcal{O}(k^2)$ bound on the number of k-mismatch occurrences of P in T.

The main shortcoming of this approach is the initial treatment of the pattern: Since the pattern P is independently aligned with Q_i^∞ for every block P_i, the same position in P may be accounted for as a mismatch for multiple blocks P_i. In particular, this happens if several adjacent blocks share the same period. This leads to an overcounting of the k-mismatch occurrences that is hard to control.

What we do instead is a more careful analysis of the pattern. Instead of creating all blocks P_i at once, we process P from left to right, as described below. Suppose that $P[j\,..\,m)$ is the unprocessed suffix of P. We first consider the length-$m/8k$ prefix P' of $P[j\,..\,m)$ and compute its shortest string period Q. If $|Q|$ exceeds a certain constant fraction of $|P'|$, we set P' aside as a *break* and continue processing $P[j + |P'|\,..\,m)$. Now, if P' is the $2k$-th break that we set aside, our process stops, and we continue to work only with the breaks. If P' does not form a break, we try extending P' to a prefix R of $P[j\,..\,m)$ that satisfies $\delta_H(R, Q^\infty[0\,..\,|R|)) = \Theta(k \cdot |R|/m)$. If such a prefix R exists, we set it aside as a *repetitive region* and continue processing $P[j+|R|\,..\,m)$. Now, if all the repetitive regions collected so far have a total length of at least $3/8 \cdot m$, we stop our process and continue to work only with the repetitive regions computed so far. A repetitive region R does not exist only if $P[j\,..\,m)$ has too few mismatches with Q^∞. In this case, we try extending $P[j\,..\,m)$ to a suffix R' of P that satisfies $\delta_H(R', \overline{Q}^\infty[0\,..\,|R'|)) = \Theta(k \cdot |R'|/m)$; where $\overline{Q}$ is a suitable rotation of Q. If we fail again, we report that P is approximately periodic; otherwise we continue to work with the single repetitive region R' generated. For this, we note that $|R'| \geq 3/8 \cdot m$ because all breaks and repetitive regions found beforehand have a total length of at most $5/8 \cdot m$.

Overall, for every pattern P, we obtain either $2k$ disjoint breaks, or disjoint repetitive regions of total length at least $3/8 \cdot m$, or a string with period $\mathcal{O}(m/k)$ at Hamming distance $\mathcal{O}(k)$ from P; consider Lemma III.6 for a formal proof.

If the analysis results in breaks, we observe that at least k breaks need to be matched exactly in each k-mismatch occurrence of P in T. As both the length and the shortest period of each break are $\Theta(n/k)$, there are at most $\mathcal{O}(k)$ exact matches of each break in the text. Now, a simple marking argument shows that the number of k-mismatch occurrences of P in T is $\mathcal{O}(k)$; consider Lemma III.8 for a formal proof of this case.

If the analysis results in repetitive regions, for each region R_i, we consider its k_i-mismatch occurrences in T with $k_i := \Theta(k \cdot |R_i|/m)$. Intuitively, this distributes the available budget of k mismatches among the repetitive regions according to their lengths. Next, we try extending each k_i-mismatch occurrence of each R_i to an approximate occurrence of P, and we assign $|R_i|$ marks to this extension. Using insights gained in the periodic case, we bound the total number of marks by $\mathcal{O}(k \cdot \sum_i |R_i|)$. Independently, we show that each k-mismatch occurrence of P has at least $\sum_i |R_i| - m/4$ marks. Using $\sum_i |R_i| \geq 3/8 \cdot m$, we finally obtain a bound of $\mathcal{O}(k)$ on the number of k-mismatch occurrences of P in T; consider Lemma III.11 for the formal proof.

In total, this proves Theorem I.6. For the characterization of the periodic case (Theorem I.7), we use a similar reasoning as [8]. As in the theorem, assume that P has k-mismatch occurrences both as a prefix and as a suffix of T. Further, fix a threshold $d \geq 2k$ and a primitive string Q such that $\delta_H(P, Q^\infty[0\,..\,m)) \leq d$. First, we show that every k-mismatch occurrence of P in T starts at a multiple of $|Q|$. In particular, $|Q|$ divides $n - m$ and, using this observation, we bound $\delta_H(T, Q^\infty[0\,..\,n))$. Finally, to decompose $\mathrm{Occ}_k^H(P, T)$ into $\mathcal{O}(k^2)$ arithmetic progressions, we analyze the sequence of Hamming distances between P and the length-m fragments of T starting at the multiples of $|Q|$: we observe that the number of changes in this sequence is bounded by $\mathcal{O}(d^2)$, which then yields the claim.

For pattern matching with edits, surprisingly few modifications in our arguments are necessary. In fact, the analysis of the pattern stays essentially the same. The main difference

in the subsequent arguments is that we need to account for shifts of up to $\mathcal{O}(k)$ positions; this causes the increase in the bound on the number of occurrences. Unfortunately, for the periodic case of pattern matching with edits, the situation is messier. The key difficulty that we overcome is that an alignment corresponding to a specific edit distance may not be unique. In particular, due to insertions and deletions, combining (the arguments for) two disjoint substrings is not as easy as in the Hamming distance case. We solve these issues by enclosing individual errors between a string and its approximate period with so-called *locked fragments*, which admit a unique canonic alignment. (A similar idea was used by Cole and Hariharan [13].) Combining this with a more involved marking scheme, we then obtain Theorem I.9.

A Unified Approach to Approximate Pattern Matching: The proofs of our new structural insights are already essentially algorithmic. To obtain algorithms for all the considered settings at once, we proceed in two steps. In the first step, we devise meta-algorithms that only rely on a core set of abstract operations; in the second step, we implement these operations in various settings. Specifically, we introduce the PILLAR model—A novel abstract interface to handle strings represented in a setting-specific manner. For two strings S and T, the following operations are supported:

- $\texttt{Extract}(S, \ell, r)$: Retrieve a string $S[\ell\,..\,r]$.
- $\texttt{LCP}(S, T)$: Compute the length of the longest common prefix of S and T.
- $\texttt{LCP}^R(S, T)$: Compute the length of the longest common suffix of S and T.
- $\texttt{IPM}(S, T)$: Assuming that $|T| \leq 2|S|$, compute the starting positions of all exact occurrences of S in T.
- $\texttt{Access}(S, i)$: Retrieve the character $S[i]$.
- $\texttt{Length}(S)$: Compute the length $|S|$ of the string S.

Using the PILLAR-model operations, the meta-algorithms for both pattern matching with mismatches and with errors follow the same overall structure:

- First, we implement the analysis of the pattern. Here, the key difficulty is to detect repetitive regions. Our algorithm finds the *shortest* repetitive region: Starting from the prefix P' of the unprocessed suffix $P[j\,..\,m)$, we enumerate the mismatches (or errors) between $P[j\,..\,m)$ and Q^∞. We stop when the number of mismatches (or errors) within the constructed region R exceeds $\Theta(k/m{\cdot}|R|)$. Intuitively, this is correct because the number of mismatches increases at most as fast as the length of $|R|$. We treat the special case when we reach the end of the pattern symmetrically.
  Note that computing the next mismatch between two strings is a prime application of the LCP operation. For finding a next edit, we adapt the Landau–Vishkin algorithm [28], which builds on LCP operations as well.
- Next, we deal with the periodic case. This turns out to be the main difficulty. For the Hamming distance, implementing the proof of Theorem I.7 is rather straightforward. However, for the edit distance case, the more complicated proof of Theorem I.9 gets complemented with even more sophisticated algorithms. Hence, we do not discuss them in this outline.
- Finding the occurrences in the presence of $2k$ breaks is easy: We first use IPM operations to find exact occurrences of the breaks in the text and then perform a straightforward marking step; for the Hamming distance, we lose an $\mathcal{O}(\log\log k)$ factor for sorting marks.
- Finding the occurrences in the presence of repetitive regions is implemented similarly; the key difference is that we use our algorithm for the periodic case to find the approximate occurrences of the repetitive regions.

Overall, this approach then yields the main technical results of this work (stated below for strings of arbitrary lengths):

Theorem I.10 (♠). *Given a pattern P of length m, a text T of length n, and a positive integer $k \leq m$, we can compute (a representation of) the set $\mathrm{Occ}_k^H(P, T)$ using $\mathcal{O}(n/m \cdot k^2 \log\log k)$ time plus $\mathcal{O}(n/m \cdot k^2)$ PILLAR operations.*

For pattern matching with edits, the number of PILLAR-model operations matches the time cost of non-PILLAR-model operations; hence the simplified theorem statement.

Theorem I.11 (♠). *Given a pattern P of length m, a text T of length n, and a positive integer $k \leq m$, we can compute (a representation of) the set $\mathrm{Occ}_k^E(P, T)$ using $\mathcal{O}(n/m \cdot k^4)$ time in the PILLAR model.*

Finally, we show how to implement the PILLAR model in the settings that we consider:

- As a toy example, we start with the standard setting. Here, implementing the PILLAR-model operations boils down to collecting known tools on strings.
- For the fully compressed setting, we heavily rely on the recompression technique by Jeż [22], [23] (especially for internal pattern matching queries), as well as on other works on straight-line programs [6], [21].
- Finally, for the dynamic setting, we use the data structure by Gawrychowski et al. [18] (for LCP and $\texttt{LCP}^R$ operations). Further, we reuse some tools from the fully compressed setting—the data structure of [18] actually works with (a form of) straight-line programs.

As the primitive operations of the PILLAR model are rather simple, we believe that they can be implemented in further settings not considered here.

II. Preliminaries

Sets: We write $[n]$ to denote the set $\{1, \ldots, n\}$. Further, we write $[i\,..\,j]$ to denote $\{i, \ldots, j\}$ and $[i\,..\,j)$ to denote $\{i, \ldots, j{-}1\}$. The set $\{a{+}j{\cdot}d \mid j \in [0\,..\,\ell)\}$ is an *arithmetic progression* with starting value a, difference d, and length ℓ.

Strings: We write $T = T[0]\,T[1]\cdots T[n-1]$ to denote a *string* of length $|T| = n$ over an alphabet Σ. The elements of Σ are called *characters*.

For two positions $i \le j$ in T, we write $T[i\,..\,j+1) := T[i\,..\,j] := T[i]\cdots T[j]$ for the *fragment* of T that starts at position i and ends at position j. A *prefix* of a string T is a fragment that starts at position 0; a *suffix* of a string T is a fragment that ends at position $|T|-1$.

A string P of length m with $0 < m \le n$ is a *substring* of T if there is a fragment $T[i\,..\,i+m)$ matching P. In this case, we say that there is an *exact occurrence* of P at position i in T, or, more simply, that P *exactly occurs in* T.

For two strings U and V, we write UV to denote their concatenation. We also write U^k to denote the concatenation of k copies of the string U. Further, U^∞ denotes an infinite string obtained by concatenating infinitely many copies of U. A string T is called *primitive* if it cannot be expressed as $T = U^k$ for a string U and an integer $k > 1$.

A positive integer p is called a *period* of a string T if $T[i] = T[i+p]$ for all $i \in [0\,..\,|T|-p)$. We refer to the smallest period as *the period* $\mathrm{per}(T)$ of the string. The string $T[0\,..\,\mathrm{per}(T))$ is called the *string period* of T. We call a string *periodic* if its period is at most half of its length.

For a string T, we define the following *rotation* operations. The operation $\mathrm{rot}(\cdot)$ takes as input a string, and moves its last character to the front. We write $\mathrm{rot}^{-1}(\cdot)$ to denote the corresponding inverse operation. Note that a primitive string T does not match any of its non-trivial rotations.

Hamming Distance and Pattern Matching with Mismatches: For two strings S and T of the same length n, we define the set of *mismatches* between S and T as $\mathrm{Mis}(S,T) := \{i \in [0\,..\,n) \mid S[i] \neq T[i]\}$; the size of $\mathrm{Mis}(S,T)$ is the *Hamming distance* $\delta_H(S,T)$ between S and T. As we are often concerned with the Hamming distance of a string S and a prefix of T^∞ for a string T, we write $\mathrm{Mis}(S,T^*) := \mathrm{Mis}(S,T^\infty[0\,..\,|S|))$ and $\delta_H(S,T^*) := |\mathrm{Mis}(S,T^*)|$.

For a *pattern* string P, a *text* string T, and a *threshold* $k \in [0\,..\,|P|]$, a fragment $T[i\,..\,i+|P|)$ of T is a *k-mismatch occurrence* of P in T if $\delta_H(P,T[i\,..\,i+|P|)) \le k$. We write $\mathrm{Occ}_k^H(P,T)$ to denote the set of starting positions of all k-mismatch occurrences of P in T.

Lastly, we define *pattern matching with mismatches*.

Problem II.1 (Pattern matching with mismatches)**.** *Given a pattern P, a text T, and a threshold k, compute* $\mathrm{Occ}_k^H(P,T)$.

III. Improved Structural Insights into Pattern Matching with Mismatches

In this section, we improve the result of [8] and show the following asymptotically tight structural characterization of the k-mismatch occurrences of a pattern P in a text T.

Theorem III.1. *Given a pattern P of length m, a text T of length n, and a threshold $k \in [1\,..\,m]$, at least one of the following holds:*

- *The number of k-mismatch occurrences of P in T is bounded by $|\mathrm{Occ}_k^H(P,T)| \le 576 \cdot n/m \cdot k$.*
- *There is a primitive string Q of length $|Q| \le m/128k$ that satisfies $\delta_H(P,Q^*) < 2k$.*

A. Characterization of the Periodic Case

In order to prove Theorem III.1, we first discuss the (approximately) periodic case, that is, the case when we have $\delta_H(P,Q^*) < 2k$. In particular, we prove the following theorem, which is a stronger variant of [8, Claim 3.1].

Theorem III.2 (Compare [8, Claim 3.1])**.** *Let P denote a pattern of length m, let T denote a text of length $n \le {}^3\!/_2\, m$, and let $k \in [0\,..\,m]$ denote a threshold. Suppose that both $T[0\,..\,m)$ and $T[n-m\,..\,n)$ are k-mismatch occurrences of P (that is, $\{0, n-m\} \subseteq \mathrm{Occ}_k^H(P,T)$). If there is a positive integer $d \ge 2k$ and a primitive string Q with $|Q| \le m/8d$ and $\delta_H(P,Q^*) \le d$, then each of following holds:*

(a) Every position in $\mathrm{Occ}_k^H(P,T)$ is a multiple of $|Q|$.
(b) The string T satisfies $\delta_H(T,Q^*) \le 3d$.
(c) The set $\mathrm{Occ}_k^H(P,T)$ can be decomposed into $3d(d+1)$ arithmetic progressions with difference $|Q|$.
(d) If $\delta_H(P,Q^*) = d$, then $|\mathrm{Occ}_k^H(P,T)| \le 6d$.

Before proving Theorem III.2, we characterize the values $\delta_H(T[j|Q|\,..\,j|Q|+m), P)$ under the extra assumption that $\delta_H(T,Q^*)$ is small as well; this assumption is dropped in Theorem III.2.

Lemma III.3. *Let P denote a pattern of length m and let T denote a text of length $n \le {}^3\!/_2\, m$. Further, let Q denote a string of length q and set $d := \delta_H(P,Q^*)$ and $d' := \delta_H(T,Q^*)$. Then, the sequence of values $h_j := \delta_H(T[jq\,..\,jq+m), P)$ for $0 \le j \le (n-m)/q$ contains at most $d'(2d+1)$ entries h_j with $h_j \neq h_{j+1}$ and, unless $d = 0$, at most $2d'$ entries h_j with $h_j \le d/2$.*

Proof: For every $\tau \in \mathrm{Mis}(T,Q^*)$ and $\pi \in \mathrm{Mis}(P,Q^*)$, let us put $(2 - \delta_H(P[\pi],T[\tau]))$ marks at position $\tau - \pi$ in T, if it exists. For each $0 \le j \le (n-m)/q$, let $\mu_j(\tau,\pi)$ denote the number of marks placed at position jq due to the mismatches τ in T and π in P, that is, $\mu_j(\tau,\pi) = 2 - \delta_H(P[\pi],T[\tau])$ if $\pi \in \mathrm{Mis}(P,Q^*)$ and $\tau = jq + \pi \in \mathrm{Mis}(T,Q^*)$, and $\mu_j(\tau,\pi) = 0$ otherwise. Further, define $\mu_j := \sum_{\tau,\pi} \mu_j(\tau,\pi)$ as the total number of marks at position jq.

Next, for every $0 \le j \le (n-m)/q$, we relate the Hamming distance $h_j := \delta_H(T[jq\,..\,jq+m), P)$ to the number of marks μ_j at position jq and the Hamming distances $\delta_H(T[jq\,..\,jq+m), Q^*)$ and $\delta_H(P,Q^*)$; consult Figure 4 for an illustration.

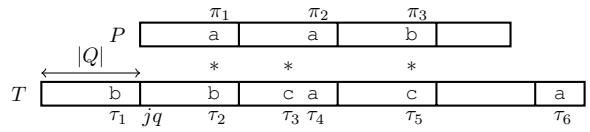

Figure 4. In both strings, all blocks apart from the last one in T are of length q. For each string $X \in \{P, T\}$ we show the characters at positions in $\mathrm{Mis}(X, Q^*)$, only. At position jq in T, we place $\mu_j(\tau_2, \pi_1) + \mu_j(\tau_4, \pi_2) + \mu_j(\tau_5, \pi_3) = 1 + 2 + 1 = 4$ marks. We have $\delta_H(P, Q^*) = |\{\pi_1, \pi_2, \pi_3\}| = 3$ and $\delta_H(T[jq\,..\,jq+m), Q^*) = |\{\tau_2, \tau_3, \tau_4, \tau_5\}| = 4$. Using Claim III.4, we obtain that $h_j = 3$; the three corresponding mismatches are indicated by asterisks.

Claim III.4. *For each $0 \le j \le (n-m)/q$, we have $h_j = \delta_H(P, Q^*) + \delta_H(T[jq\,..\,jq+m), Q^*) - \mu_j$.*

Proof: We show the following equivalent statement:

$$|\mathrm{Mis}(T[jq\,..\,jq+m), P)| = |\mathrm{Mis}(P, Q^*)| + |\mathrm{Mis}(T[jq\,..\,jq+m), Q^*)| - \sum_{\tau,\pi} \mu_j(\tau, \pi). \quad (1)$$

By construction, $\mu_j(\tau, \pi) = 0$ whenever $\tau \neq \pi + jq$. Hence, we can prove (1) by showing that for every position $\pi \in [0\,..\,m)$ in P and every position $\tau := jq + \pi$ in T, the following equation holds:

$$\delta_H(T[\tau], P[\pi]) = \delta_H(P[\pi], Q^\infty[\pi]) + \delta_H(T[\tau], Q^\infty[\tau]) - \mu_j(\tau, \pi).$$

By a case distinction on whether $\pi \in \mathrm{Mis}(P, Q^*)$ and whether $\tau \in \mathrm{Mis}(T, Q^*)$, one can see that this is indeed the case. Combining the equations obtained for every pair of positions π and τ, we derive (1). ∎

In particular, Claim III.4 yields

$$h_{j+1} - h_j = |\mathrm{Mis}(T, Q^*) \cap [jq + m\,..\,(j+1)q + m)| - |\mathrm{Mis}(T, Q^*) \cap [jq\,..\,(j+1)q)| - \mu_{j+1} + \mu_j.$$

Hence, in order for h_{j+1} not to equal h_j, at least one of the four terms on the right hand side of the equation above must be non-zero. Let us analyze when this is possible. To that end, we first observe that the set $\mathrm{Mis}(T, Q^*) \cap [jq + m\,..\,(j+1)q + m)$ contains only elements $\tau \in \mathrm{Mis}(T, Q^*)$ with $\tau \ge m$, and that the set $\mathrm{Mis}(T, Q^*) \cap [jq\,..\,(j+1)q)$ only contains elements $\tau \in \mathrm{Mis}(T, Q^*)$ with $\tau < n - m$. Using $n \le {}^3\!/_2\, m$, we observe that h_{j+1} can be different from h_j due to the first or second term at most d' times. Further, each non-zero value in one of the terms μ_{j+1} and μ_j can be attributed to a marked position (jq or $(j+1)q$, respectively). The total number of marked positions is dd', so h_{j+1} can be different from h_j due one of the terms μ_{j+1} or μ_j at most $2dd'$ times. In total, we conclude that the number of entries h_j with $h_j \neq h_{j+1}$ is at most $d'(2d+1)$.

Next, observe that $\mu_j \le 2|\mathrm{Mis}(T, Q^*) \cap [jq\,..\,jq+m)| = 2\delta_H(T[jq\,..\,jq+m), Q^*)$, and therefore

$$h_j = \delta_H(P, Q^*) + \delta_H(T[jq\,..\,jq+m), Q^*) - \mu_j \ge d - \mu_j/2.$$

Consequently, $h_j \le d/2$ yields $\mu_j \ge d$, that is, that there are at least d marks at position jq. Given that the total number of marks is at most $2dd'$, the number of entries h_j with $h_j \le d/2$ is at most $2d'$, assuming that $d > 0$. ∎

Now, we drop the assumption that $\delta_H(T, Q^*)$ is small, and prove Theorem III.2.

Proof of Theorem III.2: Consider any position $\ell \in \mathrm{Occ}_k^H(P, T)$. By the definition of a k-mismatch occurrence, we have $\delta_H(T[\ell\,..\,\ell+m), P) \le k \le d/2$. Combining this inequality with $\delta_H(P, Q^*) \le d$ via the triangle inequality yields $\delta_H(T[\ell\,..\,\ell+m), Q^*) \le {}^3\!/_2\, d$. Note that, similarly, for the position $0 \in \mathrm{Occ}_k^H(P, T)$, we obtain $\delta_H(T[0\,..\,m), Q^*) \le {}^3\!/_2\, d$, which lets us compare the overlapping parts of Q^∞. Replacing strings by superstrings and applying the triangle inequality yields

$$\begin{aligned}
&\delta_H(Q^\infty[\ell\,..\,m), Q^\infty[0\,..\,m-\ell)) \\
&\quad \le \delta_H(T[\ell\,..\,m), Q^\infty[\ell\,..\,m)) + \delta_H(T[\ell\,..\,m), Q^\infty[0\,..\,m-\ell)) \\
&\quad \le \delta_H(T[0\,..\,m), Q^\infty[0\,..\,m)) + \delta_H(T[\ell\,..\,\ell+m), Q^\infty[0\,..\,m)) \\
&\quad = \delta_H(T[0\,..\,m), Q^*) + \delta_H(T[\ell\,..\,\ell+m), Q^*) \le 3d.
\end{aligned}$$

Towards a proof by contradiction, suppose that ℓ is not an integer multiple of $|Q|$. As Q is primitive, we have

$$\begin{aligned}
3d &\ge \delta_H(Q^\infty[\ell\,..\,m), Q^\infty[0\,..\,m-\ell)) \\
&\ge \lfloor (m-\ell)/|Q| \rfloor \ge \lfloor (m/2)/(m/8d) \rfloor = 4d,
\end{aligned}$$

where the second bound follows from $\ell \le m/2$ and $|Q| \le m/8d$. This contradiction yields Claim (a).

In order to prove Claim (b), we observe that $n - m \in \mathrm{Occ}_k^H(P, T)$ is a multiple of $|Q|$. Consequently,

$$\delta_H(T, Q^*) = \delta_H(T[0\,..\,n-m), Q^*) + \delta_H(T[n-m\,..\,n), Q^*) \le \delta_H(T[0\,..\,m), Q^*) + {}^3\!/_2\, d \le 3d,$$

which concludes the proof of Claim (b).

For a proof of Claims (c) and (d), we apply Lemma III.3. Due to Claim (a), each position in $\mathrm{Occ}_k^H(P, T)$ corresponds to an entry h_j with $h_j \le k$. In particular, each block of consecutive entries $h_j, \ldots, h_{j'}$ not exceeding k yields an arithmetic progression (with difference $|Q|$) in $\mathrm{Occ}_k^H(P, T)$. The number of entries h_j with $h_j \le k < h_{j+1}$ or $h_j > k \ge h_{j+1}$ is in total at most $3d(2d+1)$, so the number of arithmetic progressions is at most $1 + 1/2 \cdot 3d(2d+1) \le 3d(d+1)$, which proves Claim (c).

For Claim (d), we observe that if $d = \delta_H(P, Q^*)$, then each position in $\mathrm{Occ}_k^H(P, T)$ corresponds to an entry h_j with $h_j \le k \le d/2$; thus $|\mathrm{Occ}_k^H(P, T)| \le 2 \cdot 3d \le 6d$. ∎

Corollary III.5. *Let P denote a pattern of length m, let T denote a text of length n, and let $k \in [0\,..\,m]$ denote a threshold. If there is a positive integer $d \ge 2k$ and a primitive string Q with $|Q| \le m/8d$ and $\delta_H(P, Q^*) \le d$, then the set $\mathrm{Occ}_k^H(P, T)$ can be decomposed into $6{\cdot}n/m{\cdot}d(d{+}1)$ arithmetic progressions with difference $|Q|$. Moreover, if $\delta_H(P, Q^*) = d$, then $|\mathrm{Occ}_k^H(P, T)| \le 12 \cdot n/m \cdot d$.*

Proof: Partition the string T into $\lfloor 2n/m \rfloor$ blocks $T_0, \ldots, T_{\lfloor 2n/m \rfloor - 1}$ of length less than $3/2\, m$ each, where the ith block starts at position $\lfloor i \cdot m/2 \rfloor$; formally, we set $T_i := T[\lfloor i \cdot m/2 \rfloor \,..\, \min\{n, \lfloor (i+3) \cdot m/2 \rfloor - 1\})$. If $\mathrm{Occ}_k^H(P, T_i) \neq \emptyset$, we define T_i' to be the shortest fragment of T_i containing all k-mismatch occurrences of P in T_i. As a result, T_i' satisfies the assumptions of Theorem III.2. Hence, $\mathrm{Occ}_k^H(P, T_i')$ can be decomposed into $3d(d+1)$ arithmetic progressions with difference $|Q|$, and $|\mathrm{Occ}(P, T_i')| \leq 6d$ if $\delta_H(P, Q^*) = d$.

We conclude that $\mathrm{Occ}_k^H(P, T_i')$ decomposes into $6 \cdot n/m \cdot d(d+1)$ arithmetic progressions with difference $|Q|$; further, $|\mathrm{Occ}(P, T_i)| \leq 12 \cdot n/m \cdot d$ if $\delta_H(P, Q^*) = d$. ∎

B. The Non-Periodic Case

Having dealt with the (approximately) periodic case, we now turn to the general case. In particular, we show that whenever the string P is sufficiently far from being periodic, the number of k-mismatch occurrences of P in any string T of length $n \leq 3/2\, m$ is $\mathcal{O}(k)$.

Intuitively, we proceed (and thereby prove Theorem III.1) as follows: We first analyze the string P for useful structure that can help in bounding the number of occurrences of P in any string T. If we fail to find any special structure in P, then we conclude that the string P is close to a periodic string with a small period (compared to $|P|$)—a case that we already understand thanks to the previous subsection.

We start by investigating the structure of any string P.

Lemma III.6. *Given a string P of length m and and a threshold $k \in [1\,..\,m]$, at least one of the following holds:*

(a) The string P contains $2k$ disjoint breaks $B_1, \ldots, B_{2k}$ *each having periods $\mathrm{per}(B_i) > m/128k$ and length $|B_i| = \lfloor m/8k \rfloor$.*

(b) The string P contains r disjoint repetitive regions $R_1, \ldots, R_r$ *of total length $\sum_{i=1}^{r} |R_i| \geq 3/8 \cdot m$ such that each region R_i satisfies $|R_i| \geq m/8k$ and has a primitive* approximate period Q_i *with $|Q_i| \leq m/128k$ and $\delta_H(R_i, Q_i^*) = \lceil 8k/m \cdot |R_i| \rceil$.*

(c) The string P has a primitive approximate period Q *with $|Q| \leq m/128k$ and $\delta_H(P, Q^*) < 8k$.*

Proof: We prove the claim constructively, that is, we construct either a set $\mathcal{B}$ of $2k$ breaks, or a set $\mathcal{R}$ of repetitive regions, or, if we fail to construct either, we derive an approximate string period Q of the string P with the desired properties.

We process the string P from left to right as follows: If the fragment P' of the next $\lfloor m/8k \rfloor$ (unprocessed) characters of P has a long period, we have found a new break and continue (or return the found set of $2k$ breaks). Otherwise, if P' has a short string period Q, we try to extend the fragment P' (to the right) into a repetitive region. If we succeed, we have found a new repetitive region and continue (or return the found set of repetitive regions if the total length of all repetitive regions found so far is at least $3/8 \cdot m$). If we fail to construct a new repetitive region, then we conclude that the suffix of P starting with P' has an approximate period Q. We try to construct a repetitive region by extending this suffix to the left, dropping all other repetitive regions computed so far. If we fail again, we declare that Q is an approximate period of the string P. Consider Algorithm 1 for a detailed description.

Algorithm 1: A constructive proof of Lemma III.6.

```
B ← {}; R ← {};
while true do
    Consider the fragment P' = P[j .. j + ⌊m/8k⌋) of the
      next ⌊m/8k⌋ unprocessed characters of P;
    if per(P') > m/128k then
        B ← B ∪ {P'};
        if |B| = 2k then return breaks B;
    else
        Q ← P[j .. j + per(P'));
        Search for a prefix R of P[j .. m) with |R| > |P'|
          and δ_H(R, Q*) = ⌈8k/m · |R|⌉;
        if such R exists then
            R ← R ∪ {(R, Q)};
            if Σ_{(R,Q)∈R} |R| ≥ 3/8 · m then return regions R;
        else
            Search for a suffix R' of P with |R'| ≥ m − j
              and δ_H(R', rot^{|R'|−m+j}(Q)*) = ⌈8k/m · |R'|⌉;
            if such R' exists then
                return repetitive region (R', rot^{|R'|−m+j}(Q))
            else return approximate period rot^j(Q);
```

Note that, by construction, all breaks in the set $\mathcal{B}$ and repetitive regions in the set $\mathcal{R}$ returned by the algorithm are disjoint and satisfy the claimed properties. To prove that the algorithm is also correct when it fails to find a new repetitive region, we start by bounding from above the length of the processed prefix of P.

Claim III.7. *Whenever we consider a new fragment $P[j\,..\,j+\lfloor m/8k \rfloor)$ of the next $\lfloor m/8k \rfloor$ unprocessed characters of P, such a fragment starts at a position $j < 5/8 \cdot m$.*

Proof: Observe that whenever we consider a new fragment $P[j\,..\,j+\lfloor m/8k \rfloor)$, the string $P[0\,..\,j)$ has been partitioned into breaks and repetitive regions. The total length of breaks is less than $2k\lfloor m/8k \rfloor \leq 2/8 \cdot m$, and the total length of repetitive regions is less than $3/8 \cdot m$. Hence, $j < 5/8 \cdot m$, yielding the claim. ∎

Note that Claim III.7 also shows that whenever we consider a new fragment P' of $\lfloor m/8k \rfloor$ characters, there is indeed such a fragment, that is, P' is well-defined.

Now, consider the case when, for a fragment $P' = P[j\,..\,j+\lfloor m/8k \rfloor)$ (that is not a break) and its string period $Q = P[j\,..\,j+\mathrm{per}(P'))$, we fail to obtain a new repetitive region R. In this case, we search for a repetitive region R' of length $|R'| \geq m - j$ that is a suffix of P and has an approximate period $Q' := \mathrm{rot}^{|R'|-m+j}(Q)$. If we indeed find

such a region R', then $|R'| \geq m-j \geq m-5/8\cdot m = 3/8\cdot m$ by Claim III.7, so R' is long enough to be reported on its own. However, if we fail to find such R', we need to show that $\mathrm{rot}^j(Q)$ can be reported as an approximate period of P, that is, $\delta_H(P, \mathrm{rot}^j(Q)^*) < 8k$.

We first derive $\delta_H(P[j\,..\,m), Q^*) < \lceil 8k/m\cdot(m-j)\rceil$. For this, we inductively prove that the values $\Delta_\rho := \lceil 8k/m\cdot\rho\rceil - \delta_H(P[j\,..\,j+\rho), Q^*)$ for $\rho \in [|P'|\,..\,m-j]$ are all at least 1. In the base case of $\rho = |P'|$, we have $\Delta_\rho = 1-0$ because Q is the string period of P'. To carry out an inductive step, suppose that $\Delta_{\rho-1} \geq 1$ for some $\rho \in [|P'|\,..\,m-j]$. Notice that $\Delta_\rho \geq \Delta_{\rho-1} - 1 \geq 0$: The first term in the definition of Δ_ρ has not decreased compared to $\Delta_{\rho-1}$, and the term $\delta_H(P[j\,..\,j+\rho), Q^*)$ may have increased by at most one. Moreover, $\Delta_\rho \neq 0$ because $R = P[j\,..\,j+\rho)$ could not be reported as a repetitive region. Since $\Delta_\rho \in \mathbb{Z}$, we conclude that $\Delta_\rho \geq 1$. This inductive reasoning ultimately shows that $\Delta_{m-j} > 0$, that is, $\delta_H(P[j\,..\,m), Q^*) < \lceil 8k/m\cdot(m-j)\rceil$.

A symmetric argument holds for the values $\Delta'_\rho := \lceil 8k/m\cdot\rho\rceil - \delta_H(P[m-\rho\,..\,m), \mathrm{rot}^{\rho-m+j}(Q)^*)$ for $\rho \in [m-j\,..\,m]$ because no repetitive region R' was found as an extension of $P[j\,..\,m)$ to the left. Thus, $\delta_H(P, \mathrm{rot}^j(Q)^*) < 8k$, that is, $\mathrm{rot}^j(Q)$ is an approximate period of P. ∎

In the next steps, we discuss how to exploit the structure obtained by Lemma III.6. First, we discuss the case that a string P contains $2k$ disjoint breaks.

Lemma III.8. *Let P denote a pattern of length m, let T denote a text of length n, and let $k \in [1\,..\,m]$ denote a threshold. Suppose that P contains $2k$ disjoint breaks $B_1, \ldots, B_{2k}$ each satisfying $\mathrm{per}(B_i) \geq m/128k$. Then, $|\mathrm{Occ}_k^H(P,T)| \leq 256\cdot n/m\cdot k$.*

Proof: For every break $B_i = P[b_i\,..\,b_i+|B_i|)$ we mark a position j in T if $j + b_i \in \mathrm{Occ}(B_i, T)$.

Claim III.9. *We place at most $256\cdot n/m\cdot k^2$ marks in total.*

Proof: Fix a break B_i and notice that the positions in $\mathrm{Occ}(B_i,T)$ are at distance at least $\mathrm{per}(B_i)$ from each other. Hence, for the break B_i, we place at most $128\cdot n/m\cdot k$ marks in T. In total, we therefore place at most $2k\cdot 128n/m\cdot k = 256\cdot n/m\cdot k^2$ marks in T. ∎

Next, we show that every k-mismatch occurrence of P in T starts at a position with at least k marks.

Claim III.10. *Each $\ell \in \mathrm{Occ}_k^H(P,T)$ has at least k marks.*

Proof: Fix $\ell \in \mathrm{Occ}_k^H(P,T)$. Out of the $2k$ breaks, at least k breaks are matched exactly, as not matching a break exactly incurs at least one mismatch. If a break B_i is matched exactly, then we have $\ell + b_i \in \mathrm{Occ}(B_i,T)$. Hence, we have placed a mark at position ℓ. Thus, there is a mark at position ℓ for every break B_i matched exactly in the corresponding occurrence of P in T. In total, there are at least k marks at position ℓ in T. ∎

By Claims III.9 and III.10, we have $|\mathrm{Occ}_k^H(P,T)| \leq (256\cdot n/m\cdot k^2)/k = 256\cdot n/m\cdot k$. ∎

Secondly, we discuss how to use repetitive regions in the string P to bound $|\mathrm{Occ}_k^H(P,T)|$.

Lemma III.11. *Let P denote a pattern of length m, let T denote a text of length n, and let $k \in [1\,..\,m]$ denote a threshold. If P contains disjoint repetitive regions $R_1, \ldots, R_r$ of total length at least $\sum_{i=1}^r |R_i| \geq 3/8\cdot m$ such that each region R_i satisfies $|R_i| \geq m/8k$ and has a primitive approximate period Q_i with $|Q_i| \leq m/128k$ and $\delta_H(R_i, Q_i^*) = \lceil 8k/m\cdot|R_i|\rceil$, then $|\mathrm{Occ}_k^H(P,T)| \leq 576\cdot n/m\cdot k$.*

Proof: Set $m_R := \sum_{i=1}^r |R_i|$. For each repetitive region $R_i = P[r_i\,..\,r_i+|R_i|)$, set $k_i := \lfloor 4k/m\cdot|R_i|\rfloor$, and place $|R_i|$ marks at each position j with $j + r_i \in \mathrm{Occ}_{k_i}^H(R_i,T)$.

Claim III.12. *We place at most $192\cdot n/m\cdot k\cdot m_R$ marks in total.*

Proof: We use Corollary III.5 to bound $|\mathrm{Occ}_{k_i}^H(R_i,T)|$. For this, we set $d_i := \delta_H(R_i, Q_i^*)$ and notice that $d_i = \lceil 8k/m\cdot|R_i|\rceil \leq 16\cdot k/m\cdot|R_i|$ since $|R_i| \geq m/8k$. Moreover, $d_i \geq 2k_i$ and $|Q_i| \leq m/128k \leq |R_i|/8d_i$ due to $d_i \leq 16\cdot k/m\cdot|R_i|$. Hence, the assumptions of Corollary III.5 are satisfied. Consequently, $|\mathrm{Occ}_{k_i}^H(R_i,T)| \leq 12\cdot n/|R_i|\cdot d_i \leq 192\cdot n/m\cdot k$; the last inequality holds as $d_i \leq 16\cdot k/m\cdot|R_i|$.

The total number of marks placed due to R_i is therefore bounded by $192\cdot n/m\cdot k\cdot|R_i|$. Across all repetitive regions, this sums up to $192\cdot n/m\cdot k\cdot m_R$, yielding the claim. ∎

Next, we show that every k-mismatch occurrence of P in T, starts at a position with many marks.

Claim III.13. *Each $\ell \in \mathrm{Occ}_k^H(P,T)$ has at least $m_R - m/4$ marks.*

Proof: Let us fix $\ell \in \mathrm{Occ}_k^H(P,T)$ and denote $k'_i := \delta_H(R_i, T[\ell+r_i\,..\,\ell+r_i+|R_i|))$ to be the number of mismatches incurred by repetitive region R_i. Further, let $I := \{i \in [1\,..\,r] \mid k'_i \leq k_i\} = \{i \in [1\,..\,r] \mid k'_i \leq 4k/m\cdot|R_i|\}$ denote the set of indices of all repetitive regions that have k_i-mismatch occurrences at the corresponding positions in T. By construction, for each $i \in I$, we have placed $|R_i|$ marks at position ℓ. Hence, the total number of marks at position ℓ is at least $\sum_{i\in I}|R_i| = m_R - \sum_{i\notin I}|R_i|$. It remains to bound the term $\sum_{i\notin I}|R_i|$. Using the definition of I, we obtain

$$\sum_{i\notin I}|R_i| = \sum_{i\notin I}\tfrac{4mk}{4mk}\cdot|R_i| = \tfrac{m}{4k}\cdot\sum_{i\notin I}(4k/m\cdot|R_i|) < \tfrac{m}{4k}\cdot\sum_{i\notin I}k'_i \leq \tfrac{m}{4k}\cdot\sum_{i=1}^r k'_i \leq \tfrac{m}{4},$$

where the last bound holds as, in total, all repetitive regions incur at most $\sum_{i=1}^r k'_i \leq k$ mismatches (since all repetitive regions are pairwise disjoint). Hence, the number of marks placed at position ℓ is at least $m_R - m/4$. ∎

In total, by Claims III.12 and III.13, the number of k-mismatch occurrences of P in T is at most

$$|\mathrm{Occ}_k^H(P,T)| \leq \tfrac{192\cdot n/m\cdot k\cdot m_R}{m_R - m/4} = \tfrac{192\cdot n/m\cdot k}{1-m/(4m_R)}.$$

As this bound is a decreasing function in m_R, the assumption $m_R \geq 3/8\cdot m$ yields the upper bound

$$|\mathrm{Occ}_k^H(P,T)| \leq \tfrac{192\cdot n/m\cdot k\cdot 3/8\cdot m}{3/8\cdot m - m/4} = 576\cdot n/m\cdot k,$$

completing the proof. ∎

Finally, we consider the case that P is approximately periodic, but not too close to the periodic string in scope.

Lemma III.14. *Let P denote a string of length m, let T denote a string of length n, and let $k \in [1\,..\,m]$ denote threshold. If there is a primitive string Q of length at most $|Q| \leq m/128k$ that satisfies $2k \leq \delta_H(P,Q^*) \leq 8k$, then $|\mathrm{Occ}_k^H(P,T)| \leq 96\cdot n/m\cdot k$.*

Proof: We apply Corollary III.5 with $d = \delta_H(P,Q^*)$. As $2k \leq d \leq 8k$ yields $|Q| \leq m/128k \leq m/8d$, the assumptions of Corollary III.5 are met. Consequently, $|\mathrm{Occ}_k^H(P,T)| \leq 12\cdot n/m\cdot d \leq 96\cdot n/m\cdot k$. ∎

Gathering Lemmas III.6, III.8, III.11, and III.14, we are now ready to prove Theorem III.1.

Proof of Theorem III.1: We apply Lemma III.6 on the string P and proceed depending on the structure found in P.

If the string P contains $2k$ disjoint breaks $B_1,\ldots,B_{2k}$ (in the sense of Lemma III.6), we apply Lemma III.8 and obtain that $|\mathrm{Occ}_k^H(P,T)| \leq 256\cdot n/m\cdot k$.

If the string P contains r disjoint repetitive regions $R_1,\ldots,R_r$ (again, in the sense of Lemma III.6), we apply Lemma III.11 and obtain that $|\mathrm{Occ}_k^H(P,T)| \leq 576\cdot n/m\cdot k$.

Otherwise, Lemma III.6 guarantees that there is a primitive string Q of length at most $|Q| \leq m/128k$ that satisfies $\delta_H(P,Q^*) < 8k$. If $\delta_H(P,Q^*) \geq 2k$, then Lemma III.14 yields $|\mathrm{Occ}_k^H(P,T)| \leq 96\cdot n/m\cdot k$. If, however, $\delta_H(P,Q^*) < 2k$, then we are in the second alternative of the theorem statement. ∎

Acknowledgments

P. Charalampopoulos was partially supported by ERC grant TOTAL under the European Union's Horizon 2020 Research and Innovation Programme (agreement no. 677651).

T. Kociumaka was supported by ISF grants no. 1278/16 and 1926/19, by a BSF grant no. 2018364, and by an ERC grant MPM under the EU's Horizon 2020 Research and Innovation Programme (agreement no. 683064).

References

[1] A. Abboud, A. Backurs, K. Bringmann, and M. Künnemann, "Fine-grained complexity of analyzing compressed data: Quantifying improvements over decompress-and-solve," in *58th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2017*, C. Umans, Ed. IEEE Computer Society, 2017, pp. 192–203.

[2] K. R. Abrahamson, "Generalized string matching," *SIAM Journal on Computing*, vol. 16, no. 6, pp. 1039–1051, 1987.

[3] S. Alstrup, G. S. Brodal, and T. Rauhe, "Pattern matching in dynamic texts," in *11th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2000*, D. B. Shmoys, Ed. SIAM, 2000, pp. 819–828.

[4] A. Amir, M. Lewenstein, and E. Porat, "Faster algorithms for string matching with k mismatches," *Journal of Algorithms*, vol. 50, no. 2, pp. 257–275, 2004.

[5] A. Backurs and P. Indyk, "Edit distance cannot be computed in strongly subquadratic time (unless SETH is false)," *SIAM Journal on Computing*, vol. 47, no. 3, pp. 1087–1097, 2018.

[6] P. Bille, G. M. Landau, R. Raman, K. Sadakane, S. R. Satti, and O. Weimann, "Random Access to Grammar-Compressed Strings and Trees," *SIAM Journal on Computing*, vol. 44, no. 3, pp. 513–539, 2015.

[7] D. Breslauer and Z. Galil, "Finding all periods and initial palindromes of a string in parallel," *Algorithmica*, vol. 14, no. 4, pp. 355–366, 1995.

[8] K. Bringmann, M. Künnemann, and P. Wellnitz, "Few Matches or Almost Periodicity: Faster Pattern Matching with Mismatches in Compressed Texts," in *30th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019*, T. M. Chan, Ed. SIAM, 2019, pp. 1126–1145.

[9] M. Burrows and D. J. Wheeler, "A block-sorting lossless data compression algorithm," Digital Equipment Corporation, Palo Alto, California, Tech. Rep. 124, 1994.

[10] T. M. Chan, S. Golan, T. Kociumaka, T. Kopelowitz, and E. Porat, "Approximating text-to-pattern hamming distances," in *52nd Annual ACM Symposium on Theory of Computing, STOC 2020*, J. Chuzhoy, Ed. ACM, 2020, pp. 643–656.

[11] R. Clifford, A. Fontaine, E. Porat, B. Sach, and T. Starikovskaya, "The k-mismatch problem revisited," in *27th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016*, R. Krauthgamer, Ed. SIAM, 2016, pp. 2039–2052.

[12] R. Clifford, A. Grønlund, K. G. Larsen, and T. A. Starikovskaya, "Upper and lower bounds for dynamic data structures on strings," in *35th Symposium on Theoretical Aspects of Computer Science, STACS 2018*, R. Niedermeier and B. Vallée, Eds. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2018, pp. 22:1–22:14.

[13] R. Cole and R. Hariharan, "Approximate String Matching: A Simpler Faster Algorithm," *SIAM Journal on Computing*, vol. 31, no. 6, pp. 1761–1782, 2002.

[14] P. Ferragina and R. Grossi, "Optimal on-line search and sublinear time update in string matching," *SIAM Journal on Computing*, vol. 27, no. 3, pp. 713–736, 1998.

[15] Z. Galil and R. Giancarlo, "Improved string matching with k mismatches," *SIGACT News*, vol. 17, no. 4, pp. 52–54, 1986.

[16] P. Gawrychowski and D. Straszak, "Beating $\mathcal{O}(nm)$ in approximate LZW-compressed pattern matching," in *24th International Symposium on Algorithms and Computation, ISAAC 2013*, L. Cai, S. Cheng, and T. W. Lam, Eds. Springer, 2013, pp. 78–88.

[17] P. Gawrychowski and P. Uznański, "Towards unified approximate pattern matching for Hamming and L_1 distance," in *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018*, I. Chatzigiannakis, C. Kaklamanis, D. Marx, and D. Sannella, Eds. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2018, pp. 62:1–62:13.

[18] P. Gawrychowski, A. Karczmarz, T. Kociumaka, J. Łącki, and P. Sankowski, "Optimal dynamic strings," in *29th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018*, A. Czumaj, Ed. SIAM, 2018, pp. 1509–1528.

[19] M. Gu, M. Farach, and R. Beigel, "An efficient algorithm for dynamic text indexing," in *5th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 1994*, D. D. Sleator, Ed. ACM/SIAM, 1994, pp. 697–704.

[20] M. Henzinger, S. Krinninger, D. Nanongkai, and T. Saranurak, "Unifying and strengthening hardness for dynamic problems via the online matrix-vector multiplication conjecture," in *47th Annual ACM on Symposium on Theory of Computing, STOC 2015*, R. Rubinfeld, Ed. ACM, 2015, pp. 21–30.

[21] T. I, "Longest common extensions with recompression," in *28th Annual Symposium on Combinatorial Pattern Matching, CPM 2017*, J. Kärkkäinen, J. Radoszewski, and W. Rytter, Eds. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2017, pp. 18:1–18:15.

[22] A. Jeż, "Faster fully compressed pattern matching by recompression," *ACM Transactions on Algorithms*, vol. 11, no. 3, pp. 20:1–20:43, 2015.

[23] ——, "Recompression: A simple and powerful technique for word equations," *Journal of the ACM*, vol. 63, no. 1, pp. 4:1–4:51, 2016.

[24] D. Kempa and T. Kociumaka, "Resolution of the Burrows–Wheeler transform conjecture," in *61st Annual IEEE Symposium on Foundations of Computer Science, FOCS 2020*, S. Irani, Ed. IEEE Computer Society, 2020.

[25] D. Kempa and N. Prezza, "At the roots of dictionary compression: string attractors," in *50th Annual ACM Symposium on Theory of Computing, STOC 2018*, M. Henzinger, Ed. ACM, 2018, pp. 827–840.

[26] S. Kosaraju, "Efficient string matching," 1987, manuscript.

[27] G. M. Landau and U. Vishkin, "Efficient string matching with k mismatches," *Theoretical Computer Science*, vol. 43, pp. 239–249, 1986.

[28] ——, "Fast parallel and serial approximate string matching," *Journal of Algorithms*, vol. 10, no. 2, pp. 157–169, 1989.

[29] N. Larsson and A. Moffat, "Off-line dictionary-based compression," *Proceedings of the IEEE*, vol. 88, no. 11, pp. 1722–1732, 2000.

[30] M. Lohrey, "Algorithmics on SLP-compressed strings: A survey," *Groups Complexity Cryptology*, vol. 4, no. 2, pp. 241–299, 2012.

[31] K. Mehlhorn, R. Sundar, and C. Uhrig, "Maintaining Dynamic Sequences under Equality Tests in Polylogarithmic Time," *Algorithmica*, vol. 17, no. 2, pp. 183–198, 1997.

[32] C. G. Nevill-Manning and I. H. Witten, "Compression and explanation using hierarchical grammars," *The Computer Journal*, vol. 40, no. 2 and 3, pp. 103–116, 1997.

[33] T. Nishimoto, T. I, S. Inenaga, H. Bannai, and M. Takeda, "Dynamic index and LZ factorization in compressed space," *Discrete Applied Mathematics*, vol. 274, pp. 116–129, 2020.

[34] R. Radicioni and A. Bertoni, "Grammatical compression: compressed equivalence and other problems," *Discrete Mathematics and Theoretical Computer Science*, vol. 12, no. 4, p. 109, 2010.

[35] W. Rytter, "Application of Lempel-Ziv factorization to the approximation of grammar-based compression," *Theoretical Computer Science*, vol. 302, no. 1-3, pp. 211–222, 2003.

[36] ——, "Grammar compression, LZ-encodings, and string algorithms with implicit input," in *31st International Colloquium on Automata, Languages, and Programming, ICALP 2004*, J. Díaz, J. Karhumäki, A. Lepistö, and D. Sannella, Eds. Springer, 2004, pp. 15–27.

[37] S. C. Sahinalp and U. Vishkin, "Efficient approximate and dynamic matching of patterns using a labeling paradigm (extended abstract)," in *37th Annual IEEE Symposium on Foundations of Computer Science, FOCS 1996*, M. Tompa, Ed. IEEE Computer Society, 1996, pp. 320–328.

[38] H. Sakamoto, "Grammar compression: Grammatical inference by compression and its application to real data." in *12th International Conference on Grammatical Inference, ICGI 2014*. JMLR.org, 2014, pp. 3–20.

[39] Y. Shibata, T. Kida, S. Fukamachi, M. Takeda, A. Shinohara, T. Shinohara, and S. Arikawa, "Byte pair encoding: A text compression scheme that accelerates pattern matching," Technical Report DOI-TR-161, Department of Informatics, Kyushu University, Tech. Rep., 1999.

[40] R. Sundar and R. E. Tarjan, "Unique binary-search-tree representations and equality testing of sets and sequences," *SIAM Journal on Computing*, vol. 23, no. 1, pp. 24–44, 1994.

[41] A. Tiskin, "Threshold approximate matching in grammar-compressed strings," in *Prague Stringology Conference, PSC 2014*, J. Holub and J. Žďárek, Eds., 2014, pp. 124–138.

[42] T. Welch, "A technique for high-performance data compression," *Computer*, vol. 17, pp. 8–19, 1984.

[43] J. Ziv and A. Lempel, "A universal algorithm for sequential data compression," *IEEE Transactions on Information Theory*, vol. 23, no. 3, pp. 337–343, 1977.

[44] ——, "Compression of individual sequences via variable-rate coding," *IEEE Transactions on Information Theory*, vol. 24, no. 5, pp. 530–536, 1978.