Circular Pattern Matching with k Mismatches

Panagiotis Charalampopoulos[1], Tomasz Kociumaka[2,3], Solon P. Pissis[4], Jakub Radoszewski[3], Wojciech Rytter[3], Juliusz Straszyński[3], Tomasz Waleń[3], and Wiktor Zuba[3](✉)

[1] Department of Informatics, King's College London, London, UK
panagiotis.charalampopoulos@kcl.ac.uk
[2] Department of Computer Science, Bar-Ilan University, Ramat Gan, Israel
[3] Institute of Informatics, University of Warsaw, Warsaw, Poland
{kociumaka,jrad,rytter,jks,walen,w.zuba}@mimuw.edu.pl
[4] CWI, Amsterdam, The Netherlands
solon.pissis@cwi.nl

Abstract. The k-mismatch problem consists in computing the Hamming distance between a pattern P of length m and every length-m substring of a text T of length n, if this distance is no more than k. In many real-world applications, any cyclic shift of P is a relevant pattern, and thus one is interested in computing the minimal distance of every length-m substring of T and any cyclic shift of P. This is the circular pattern matching with k mismatches (k-CPM) problem. A multitude of papers have been devoted to solving this problem but, to the best of our knowledge, only average-case upper bounds are known. In this paper, we present the first non-trivial worst-case upper bounds for the k-CPM problem. Specifically, we show an $\mathcal{O}(nk)$-time algorithm and an $\mathcal{O}(n+\frac{n}{m}\,k^5)$-time algorithm. The latter algorithm applies in an extended way a technique that was very recently developed for the k-mismatch problem [Bringmann et al., SODA 2019].

1 Introduction

Pattern matching is a fundamental problem in computer science [15]. It consists in finding all substrings of a text T of length n that match a pattern P of length

P. Charalampopoulos—Supported by a Studentship from the Faculty of Natural and Mathematical Sciences at King's College London and an A. G. Leventis Foundation Educational Grant.
T. Kociumaka—Supported by ISF grants no. 824/17 and 1278/16 and by an ERC grant MPM under the EU's Horizon 2020 Research and Innovation Programme (grant no. 683064).
J. Radoszewski and J. Straszyński—Supported by the "Algorithms for text processing with errors and uncertainties" project carried out within the HOMING program of the Foundation for Polish Science co-financed by the European Union under the European Regional Development Fund.

m. In many real-world applications, a measure of similarity is usually introduced allowing for *approximate* matches between the given pattern and substrings of the text. The most widely-used similarity measure is the Hamming distance between the pattern and all length-m substrings of the text.

Computing the Hamming distance between P and all length-m substrings of T has been investigated for the past 30 years. The first efficient solution requiring $\mathcal{O}(n\sqrt{m \log m})$ time was independently developed by Abrahamson [1] and Kosaraju [30] in 1987. The k-mismatch version of the problem asks for finding only the substrings of T that are close to P, specifically, at Hamming distance at most k. The first efficient solution to this problem running in $\mathcal{O}(nk)$ time was developed in 1986 by Landau and Vishkin [31]. It took almost 15 years for a breakthrough result by Amir et al. improving this to $\mathcal{O}(n\sqrt{k \log k})$ [2]. More recently, there has been a resurgence of interest in the k-mismatch problem. Clifford et al. gave an $\mathcal{O}((n/m)(k^2 \log k) + n\text{polylog}n)$-time algorithm [13], which was subsequently improved further by Gawrychowski and Uznański to $\mathcal{O}((n/m)(m + k\sqrt{m})\text{polylog}n)$ [21]. In [21], the authors have also provided evidence that any further progress in this problem is rather unlikely.

The k-mismatch problem has also been considered on compressed representations of the text [10,11,19,37], in the parallel model [18], and in the streaming model [13,14,35]. Furthermore, it has been considered in non-standard stringology models, such as the parameterized model [23] and the order-preserving model [20].

In many real-world applications, such as in bioinformatics [4,7,22,25] or in image processing [3,32–34], any cyclic shift (rotation) of P is a relevant pattern, and thus one is interested in computing the minimal distance of every length-m substring of T and any cyclic shift of P, if this distance is no more than k. This is the circular pattern matching with k mismatches (k-CPM) problem. A multitude of papers [5,6,8,9,17,24] have thus been devoted to solving the k-CPM problem but, to the best of our knowledge, only average-case upper bounds are known; i.e. in these works the assumption is that text T is uniformly random. The main result states that, after preprocessing pattern P, the average-case optimal search time of $\mathcal{O}(n\frac{k+\log m}{m})$ [12] can be achieved for certain values of the error ratio k/m (see [9,17] for more details on the preprocessing costs).

In this paper, we draw our motivation from (i) the importance of the k-CPM problem in real-world applications and (ii) the fact that no (non-trivial) worst-case upper bounds are known. Trivial here refers to running the fastest-known algorithm for the k-mismatch problem [21] separately for each of the m rotations of P. This yields an $\mathcal{O}(n(m+k\sqrt{m})\text{polylog}n)$-time algorithm for the k-CPM problem. This is clearly unsatisfactory: it is a simple exercise to design an $\mathcal{O}(nm)$-time or an $\mathcal{O}(nk^2)$-time algorithm. In an effort to tackle this unpleasant situation, we present two much more efficient algorithms: a simple $\mathcal{O}(nk)$-time algorithm and an $\mathcal{O}(n + \frac{n}{m}k^5)$-time algorithm. Our second algorithm applies in an extended way a technique that was developed very recently for k-mismatch pattern matching in grammar compressed strings by Bringmann et al. [11].

Our Approach. We first consider a simple version of the problem (called Anchor-Match) in which we are given a position in T (an *anchor*) which belongs to potential k-mismatch circular occurrences of P. A simple $\mathcal{O}(k)$ time algorithm is given (after linear-time preprocessing) to compute all relevant occurrences. By considering separately each position in T as an anchor we obtain an $\mathcal{O}(nk)$-time algorithm. The concept of an anchor is extended to the so called *matching-pairs*: when we know a pair of positions, one in P and the other in T, that are aligned. Then comes the idea of a *sample* P', which is a fragment of P of length $\Theta(m/k)$ which supposedly exactly matches a corresponding fragment in T. We choose $\mathcal{O}(k)$ samples and work for each of them and for windows of T of size $2m$. As it is typical in many versions of pattern matching, our solution is split into the periodic and non-periodic cases. If P' is non-periodic the sample occurs only $\mathcal{O}(k)$ times in a window and each occurrence gives a matching-pair (and consequently two possible anchors). Then we perform Anchor-Match for each such anchor. The hard part is the case when P' is periodic. Here we compute all exact occurrences of P' and obtain $\mathcal{O}(k)$ groups of occurrences, each one being an arithmetic progression. Now each group is processed using the approach "few matches or almost periodicity" of Bringmann et al. [11]. In the latter case periodicity is approximate, allowing up to k mismatches.

2 Preliminaries

Let $S = S[0]S[1]\cdots S[n-1]$ be a *string* of length $|S| = n$ over an integer alphabet Σ. The elements of Σ are called *letters*. For two positions i and j on S, we denote by $S[i\,..\,j] = S[i]\cdots S[j]$ the *fragment* of S that starts at position i and ends at position j (it equals the empty string ε if $j < i$). A *prefix* of S is a fragment that starts at position 0, i.e. of the form $S[0\,..\,j]$, and a *suffix* is a fragment that ends at position $n-1$, i.e. of the form $S[i\,..\,n-1]$. For an integer k, we define the kth *power* of S, denoted by S^k, as the string obtained from concatenating k copies of S. S^∞ denotes the string obtained by concatenating infinitely many copies. If S and S' are two strings of the same length, then by $S =_k S'$ we denote the fact that S and S' have at most k mismatches, that is, that the Hamming distance between S and S' does not exceed k.

We say that a string S has period q if $S[i] = S[i+q]$ for all $i = 0, \ldots, |S|-q-1$. String S is periodic if it has a period q such that $2q \le |S|$. We denote the smallest period of S by $\mathsf{per}(S)$.

For a string S, by $\mathsf{rot}_x(S)$ for $0 \le x < |S|$, we denote the string that is obtained from S by moving the prefix of S of length x to its suffix. We call the string $\mathsf{rot}_x(S)$ (or its representation x) a *rotation* of S. More formally, we have

$$\mathsf{rot}_x(S) = VU, \text{ where } S = UV \text{ and } |U| = x.$$

2.1 Anatomy of Circular Occurrences

In what follows, we denote by m the length of the pattern P and by n the length of the text T. We say that P has a *k-mismatch circular occurrence* (in short *k-occurrence*) in T at position p if $T[p\,..\,p+m-1] =_k \mathsf{rot}_x(P)$ for some rotation x.

In this case, the position x in the pattern is called the *split point* of the pattern and $p + (m - x) \bmod m$ [1] is called the *anchor* in the text (see Fig. 1).

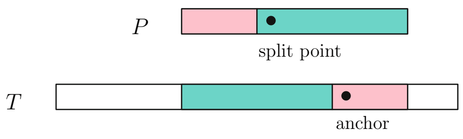

Fig. 1. The anchor and the split point for a k-occurrence of P in T.

In other words, if $P = UV$ and its rotation VU occurs in T, then the first position of V in P is the split point of this occurrence, and the first position of U in T is the anchor of this occurrence.

For an integer z, let us denote $\mathbf{W}_z = [z \mathinner{.\,.} z + m - 1]$ (*window* of size m). For a k-occurrence at position p with rotation x, we introduce a set of pairs of positions in the fragment of the text and the corresponding positions from the original (unrotated) pattern:

$$M(p, x) = \{(i, (i - p + x) \bmod m) \, : \, i \in \mathbf{W}_p\}.$$

The pairs $(i, j) \in M(p, x)$ are called *matching pairs* of an occurrence p with rotation x. In particular, $(p + ((m - x) \bmod m), 0) \in M(p, x)$. An example is provided in Fig. 2.

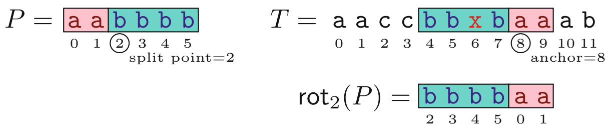

Fig. 2. A 1-occurrence of P = aabbbb in text T = aaccbbxbaaab at position $p = 4$ with rotation $x = 2$; $M(4, 2) = \{(4, 2), (5, 3), (6, 4), (7, 5), (8, 0), (9, 1)\}$.

2.2 Internal Queries in a Text

Let T be a string of length n called text. The length of the longest common prefix (suffix) of strings U and V is denoted by $\mathsf{lcp}(U, V)$ ($\mathsf{lcs}(U, V)$). There is a well-known efficient data structure answering such queries over suffixes (prefixes) of a given text in $\mathcal{O}(1)$ time after $\mathcal{O}(n)$-time preprocessing. It consists of the suffix array and a data structure for range minimum queries; see [15]. Using the kangaroo method [18,31], longest common prefix (suffix) queries can handle mismatches; after an $\mathcal{O}(n)$-time preprocessing of the text, longest common prefix (suffix) queries with up to k mismatches can be answered in $\mathcal{O}(k)$ time.

[1] The modulo operation is used to handle the trivial rotation with $x = 0$.

An Internal Pattern Matching (IPM) query, for two given fragments F and G of the text, such that $|G| \le 2|F|$, computes the set of all occurrences of F in G. If there are more than two occurrences, they form an arithmetic sequence with difference $\mathsf{per}(F)$. For a text of length n, a data structure for IPM queries can be constructed in $\mathcal{O}(n)$ time and answers queries in $\mathcal{O}(1)$ time (see [29] and [26, Theorem 1.1.4]). It can be used to compute all occurrences of a given fragment F of length p in T, expressed as a union of $\mathcal{O}(n/p)$ pairwise disjoint arithmetic sequences with difference $\mathsf{per}(F)$, in $\mathcal{O}(n/p)$ time.

3 An $\mathcal{O}(nk)$-time Algorithm

We first introduce an auxiliary problem in which one wants to compute all k-occurrences of P in T with a given anchor a.

Anchor-Match Problem

Input: Text T of length n, pattern P of length m, positive integer k, and position a.

Output: Find all k-occurrences p of P in T with anchor a.

Lemma 1. *After $\mathcal{O}(n)$-time preprocessing, the answer to* Anchor-Match *problem, represented as a union of $\mathcal{O}(k)$ intervals, can be computed in $\mathcal{O}(k)$ time.*

Proof. In the preprocessing we prepare a data structure for lcp and lcs queries in $P\#T$, for a special symbol # that does not occur in P and T.

The processing of each query is split into $k+1$ phases. In the jth phase, we compute the interval $[l_j \mathinner{.\,.} r_j]$ such that for every $p \in [l_j \mathinner{.\,.} r_j]$ there exists a k-occurrence p in T that has an anchor at a and the number of mismatches between $T[p \mathinner{.\,.} a-1]$ and the suffix of P of equal length is exactly j.

Let us consider the conditions for interval $[l_j \mathinner{.\,.} r_j]$ (see also Fig. 3):

C1 $[l_j \mathinner{.\,.} r_j] \subseteq [a-m+1 \mathinner{.\,.} a)$ since occurrences must contain anchor a,

C2 $[l_j \mathinner{.\,.} r_j] \subseteq [a-1-s_j \mathinner{.\,.} a-1-s_{j-1})$, where s_i is the length of the longest common suffix of $T[0 \mathinner{.\,.} a-1]$ and P with exactly i mismatches, since we need exactly j mismatches in $T[p \mathinner{.\,.} a-1]$,

C3 $[l_j \mathinner{.\,.} r_j] \subseteq [a-m \mathinner{.\,.} a+p_{k-j}-m)$, where p_{k-j} is the length of the longest common prefix of $T[a \mathinner{.\,.} n-1]$ and P with at most $k-j$ mismatches, since we cannot exceed k mismatches in total.

Using the kangaroo method [18,31], the values s_j, p_j for all $0 \le j \le k$ can be computed in $\mathcal{O}(k)$ time in total. Then the interval $[l_j \mathinner{.\,.} r_j]$ is a simple intersection of the above conditions, which can be computed in $\mathcal{O}(1)$ time. □

Proposition 2. *k-CPM can be solved in $\mathcal{O}(nk)$ time and $\mathcal{O}(n)$ space.*

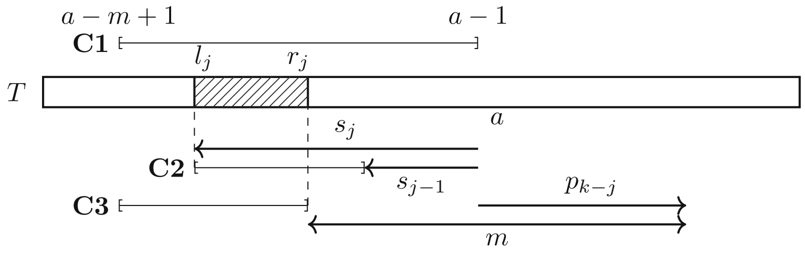

Fig. 3. An illustration of the setting in Lemma 1.

Proof. We invoke the algorithm of Lemma 1 for all $a \in [0\,..\,n-1]$ and obtain $\mathcal{O}(nk)$ intervals of k-occurrences of P in T. Instead of storing all the intervals, we count how many intervals start and end at each position of the text. We can then compute the union of the intervals by processing these counts left to right. $\square$

4 An $\mathcal{O}(n + \frac{n}{m}k^5)$-time Algorithm

In this section, we assume that $m \le n \le 2m$ and aim at an $\mathcal{O}(n + k^5)$-time algorithm.

A *(deterministic) sample* is a short segment P' of the pattern P. An occurrence in the text without any mismatch is called *exact*. We introduce a problem of Sample-Matching that consists in finding all k-occurrences of P in T such that P' matches exactly a fragment of length $|P'|$ in T.

We split the pattern P into $k+2$ fragments of length $\left\lfloor \frac{m}{k+2} \right\rfloor$ or $\left\lceil \frac{m}{k+2} \right\rceil$ each. One of those fragments will occur exactly in the text (up to k fragments may occur with a mismatch and at most one fragment will contain the split point). Let us henceforth fix a sample P' as one of these fragments, let p' be its starting position in P, and let $m' = |P'|$.

We assume that the split point x in P is to the right of P', i.e., that $x \ge p' + m'$. The opposite case—that $x < p'$—can be handled analogously.

4.1 Matching Non-periodic Samples

Let us assume that P' is non-periodic. We introduce a problem in which, intuitively, we compute all k-occurrences of P in T which align $T[i]$ with $P[j]$.

Pair-Match Problem

Input: Text T of length n, pattern P of length m, positive integer k, and two integers $i \in [0\,..\,n-1]$ and $j \in [0\,..\,m-1]$.

Output: The set $A(i,j)$ of all positions in T where we have a k-mismatch occurrence of $\mathsf{rot}_x(P)$ for some x such that (i,j) is a matching pair.

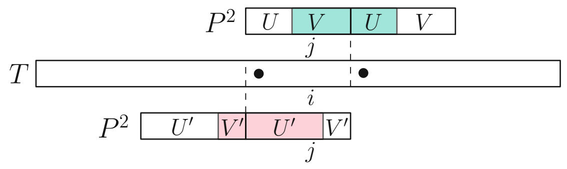

Fig. 4. The two possible anchors for the matching pair of positions (i, j) are shown as bullet points. A possible k-occurrence of P in T corresponding to the left (resp. right) anchor is shown below T (above T, resp.).

Lemma 3. *After $\mathcal{O}(n)$-time preprocessing, the* PAIR-MATCH *problem can be solved in $\mathcal{O}(k)$ time, where the output is represented as a union of $\mathcal{O}(k)$ intervals.*

Proof. The PAIR-MATCH problem can be essentially reduced to the ANCHOR-MATCH problem, since for a given matching pair of characters in P and T, there are at most two ways of choosing the anchor depending on the relation between j and a split point: these are $i-j$ (if $i-j \geq 0$) and $i+|P|-j$ (if $i+|P|-j < |T|$); see Fig. 4. We then have to take the intersection of the answer with $[i-m+1\,..\,i]$ to ensure that the k-occurrence contains position i. □

Lemma 4. *After $\mathcal{O}(n)$-time preprocessing, the* SAMPLE-MATCHING *problem for a non-periodic sample can be solved in $\mathcal{O}(k^2)$ time and outputs a union of $\mathcal{O}(k^2)$ intervals of occurrences.*

Proof. If P' is non-periodic, then it has $\mathcal{O}(k)$ occurrences in T, which can be computed in $\mathcal{O}(k)$ time after an $\mathcal{O}(n)$-time preprocessing using IPM queries [26, 29] in $P\#T$. Let j be the starting position of P' in P and i be a starting position of an occurrence of P' in T. For each of the $\mathcal{O}(k)$ such pairs (i, j), the computation reduces to the PAIR-MATCH problem for i and j. The statement follows by Lemma 3. □

4.2 Simple Geometry of Arithmetic Sequences of Intervals

Before we proceed with showing how to efficiently handle periodic samples, we present algorithms that will be used in the proofs for handling regular sets of intervals. For an interval I and integer r, let $I \oplus r = \{\, i+r : i \in I \,\}$. We define

$$\mathsf{Chain}_q(I, a) = I \cup (I \oplus q) \cup (I \oplus 2q) \cup \cdots \cup (I \oplus aq).$$

This set is further called an *interval chain*. Note that it can be represented in $\mathcal{O}(1)$ space (with four integers: a, q, and the endpoints of I).

For a given value of q, let us fit the integers from $[1\,..\,n]$ into the cells of a grid of width q so that the first row consists of numbers 1 through q, the second of numbers $q+1$ to $2q$, etc. Let us call this grid $\mathcal{G}_q$. A chain Chain_q can be conveniently represented in the grid $\mathcal{G}_q$ using the following lemma; it was stated in [28] and its proof can be found in the full version of that paper [27].

Lemma 5 ([27,28]). *The set* $\mathsf{Chain}_q(I,a)$ *is a union of* $\mathcal{O}(1)$ *orthogonal rectangles in* $\mathcal{G}_q$. *The coordinates of the rectangles can be computed in* $\mathcal{O}(1)$ *time.*

Lemma 6 can be used to compute a union of interval chains; its proof is deferred to the full version of this paper.

Lemma 6. *Assume that we are given* m *interval chains whose elements are subsets of* $[0\mathinner{.\,.} n]$. *The union of these chains, expressed as a subset of* $[0\mathinner{.\,.} n]$, *can be computed in* $\mathcal{O}(n+m)$ *time.*

We will also use the following auxiliary lemma.

Lemma 7. *Let* X *and* Z *be intervals and* q *be a positive integer. Then the set* $Z' := \{z \in Z \,:\, \exists_{x\in X}\, z \equiv x \pmod q\}$, *represented as a disjoint sum of at most three interval chains with difference* q, *can be computed in* $\mathcal{O}(1)$ *time.*

Proof. If $|X| \geq q$, then $Z' = Z$ is an interval and thus an interval chain. If $|X| < q$, then Z' can be divided into disjoint intervals of length smaller than or equal to $|X|$. The intervals from the second until the penultimate one (if any such exist), have length $|X|$. Hence, they can be represented as a single chain, as the first element of each such interval is equal mod q to the first element of X. The two remaining intervals can be treated as chains as well. □

4.3 Matching Periodic Samples

Let us assume that P' is periodic, i.e., it has a period q with $2q \leq |P'|$. A fragment of a string S containing an inclusion-maximal arithmetic sequence of occurrences of P' in a string S with difference q is called here a P'-run. If P' matches a fragment in the text, then the match belongs to a P'-run. For example, the underlined substring of $S = \mathtt{bb}\underline{\mathtt{abababab}}\mathtt{aa}$ is a P'-run for $P' = \mathtt{abab}$.

Lemma 8. *If a string* P' *is periodic, the number of* P'*-runs in the text is* $\mathcal{O}(k)$ *and they can all be computed in* $\mathcal{O}(k)$ *time after* $\mathcal{O}(n)$*-time preprocessing.*

Proof. We construct the data structure for IPM queries on $P\#T$. This allows us to compute the set of all occurrences of P' in T as a collection of $\mathcal{O}(k)$ arithmetic sequences with difference $\mathsf{per}(P')$. We then check for every two consecutive sequences if they can be joined together. This takes $\mathcal{O}(k)$ time and results in $\mathcal{O}(k)$ P'-runs. □

For two equal-length strings S and S', we denote the set of their *mismatches* by

$$\mathsf{Mis}(S,S') = \{i = 0,\dots,|S|-1 \,:\, S[i] \neq S'[i]\}.$$

Let $Q = S[i\mathinner{.\,.} j]$. We say that position a in S is a *misperiod* with respect to the fragment $S[i\mathinner{.\,.} j]$ if $S[a] \neq S[b]$ where b is the unique position such that $b \in [i\mathinner{.\,.} j]$ and $|Q| \mid b-a$. We define the set $\mathsf{LeftMisper}_k(S,i,j)$ as the set of k maximal misperiods that are smaller than i and $\mathsf{RightMisper}_k(S,i,j)$ as the set

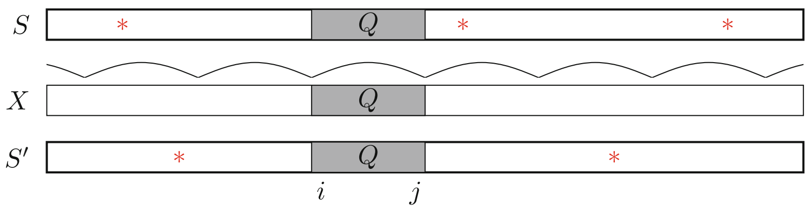

Fig. 5. Let S, S', and X be equal-length strings such that X is a factor of Q^∞ and $S[i\,.\,.\,j] = S'[i\,.\,.\,j] = X[i\,.\,.\,j] = Q$. The asterisks in S denote the positions in $\mathsf{Mis}(S,X)$, or equivalently, the misperiods with respect to $S[i\,.\,.\,j]$. Similarly for S'. One can observe that $\mathsf{Mis}(S,X) \cap \mathsf{Mis}(S',X) = \emptyset$ and that $\mathsf{Mis}(S,X) \cup \mathsf{Mis}(S',X) = \mathsf{Mis}(S,S')$.

of k minimal misperiods that are greater than j. Each of the sets can have less than k elements if the corresponding misperiods do not exist. We further define

$$\mathsf{Misper}_k(S,i,j) = \mathsf{LeftMisper}_k(S,i,j) \cup \mathsf{RightMisper}_k(S,i,j)$$

and $\mathsf{Misper}(S,i,j) = \bigcup_{k=0}^{\infty} \mathsf{Misper}_k(S,i,j)$.

The following lemma captures the main combinatorial property behind the new technique of Bringmann et al. [11]. Its proof is deferred to the full version of this paper; the intuition is shown in Fig. 5.

Lemma 9. *Assume that $S =_k S'$ and that $S[i\,.\,.\,j] = S'[i\,.\,.\,j]$. Let*

$$I = \mathsf{Misper}_{k+1}(S,i,j) \text{ and } I' = \mathsf{Misper}_{k+1}(S',i,j).$$

If $I \cap I' = \emptyset$, then $\mathsf{Mis}(S,S') = I \cup I'$, $I = \mathsf{Misper}(S,i,j)$, and $I' = \mathsf{Misper}(S',i,j)$.

A string S is *k-periodic w.r.t. an occurrence i of Q* if $|\mathsf{Misper}(S,i,i+|Q|-1)| \le k$. In particular, in the conclusion of the above lemma S and S' are $|I|$-periodic and $|I'|$-periodic, respectively, w.r.t. $Q = S[i\,.\,.\,j] = S'[i\,.\,.\,j]$. This notion forms the basis of the following auxiliary problem in which we search for k-occurrences in which the rotation of the pattern and the fragment of the text are k-periodic for the same period Q.

Let U and V be two strings and J and J' be sets containing positions in U and V, respectively. We say that length-m fragments $U[p\,.\,.\,p+m-1]$ and $V[x\,.\,.\,x+m-1]$ are *(J,J')-disjoint* if the sets $(\mathbf{W}_p \cap J) \oplus (-p)$ and $(\mathbf{W}_x \cap J') \oplus (-x)$ are disjoint. For example, if $J = \{2,4,11,15,16,17\}$, $J' = \{5,6,15,18,19\}$, and $m = 12$, then $U[3\,.\,.\,14]$ and $V[6\,.\,.\,17]$ are (J,J')-disjoint for:

$$\begin{array}{rr} U = & \mathsf{ab}\bullet\ \boxed{\mathsf{a}\bullet\mathsf{b}\ \mathsf{abc}\ \mathsf{ab}\bullet\ \mathsf{abc}}\ \bullet\bullet\bullet \\ V = \ \mathsf{abc}\ & \mathsf{ab}\bullet\ \boxed{\bullet\mathsf{bc}\ \mathsf{abc}\ \mathsf{abc}\ \bullet\mathsf{bc}}\ \bullet\bullet\mathsf{c} \end{array}$$

Periodic-Periodic-Match Problem

Input: A string U which is $2k$-periodic w.r.t. to an exact occurrence i of a length-q string Q and a string V which is $2k$-periodic w.r.t. to an exact occurrence i' of the same string Q such that $m \le |U|, |V| \le 2m$ and

$$J = \mathsf{Misper}(U, i, i+q-1), \quad J' = \mathsf{Misper}(V, i', i'+q-1).$$

(The strings U and V are not stored explicitly.)

Output: The set of positions p in U for which there exists a (J, J')-disjoint k-occurrence $U[p\mathinner{.\,.}p+m-1]$ of $V[x\mathinner{.\,.}x+m-1]$ for x such that

$$i - p \equiv i' - x \pmod{q}.$$

Intuitively, the condition on the output of the problem corresponds to the fact that the k-mismatch periodicity is aligned. We defer the solution to this problem to Lemma 12. Let us now show how it can be used to solve Sample-Matching for a periodic sample.

Let us define

$$\text{Pairs-Match}(T, I, P, J) = \bigcup_{i \in I, j \in J} \text{Pair-Match}(T, i, P, j).$$

Let A be a set of positions in a string S and m be a positive integer. We then denote $A \bmod m = \{a \bmod m : a \in A\}$ and by $\mathsf{frag}_A(S)$ we denote the fragment $S[\min A\mathinner{.\,.}\max A]$. We provide a pseudocode of an algorithm that computes all k-occurrences of P such that P' matches a fragment of a given P'-run below.

```
Data: A periodic fragment $P'$ of pattern $P$, a $P'$-run $R$ in
text $T$, $q = \mathsf{per}(P')$, and $k$.
Result: A compact representation of $k$-occurrences of $P$ in $T$
including all $k$-occurrences where $P'$ in $P$ matches a fragment of $R$ in $T$.
Let $R = T[s\mathinner{.\,.}s+|R|-1]$;
$J := \mathsf{Misper}_{k+1}(T, s, s+q-1)$; { $\mathcal{O}(k)$ time }
$J' := \mathsf{Misper}_{k+1}(P^2, m+p', m+p'+q-1)$; { $\mathcal{O}(k)$ time }
$U := \mathsf{frag}_J(T)$; $V := \mathsf{frag}_{J'}(P^2)$;
$Y :=$ Periodic-Periodic-Match$(U, V)$; { $\mathcal{O}(k^2)$ time }
$Y := Y \oplus \min(J)$;
$J' := J' \bmod m$;
$X :=$ Pairs-Match$(T, J, P, J')$; { $\mathcal{O}(k^3)$ time }
return $X \cup Y$;
```

Algorithm 1. Run-Sample-Matching

Lemma 10. *After $\mathcal{O}(n)$-time preprocessing, algorithm Run-Sample-Matching works in $\mathcal{O}(k^3)$ time and returns a compact representation that consists of $\mathcal{O}(k^3)$ interval chains.*

Proof. See the pseudocode. The sets J and J' can be computed in $\mathcal{O}(k)$ time:

Claim. If S is a string of length n, then the sets $\mathsf{RightMisper}_k(S,i,j)$ and $\mathsf{LeftMisper}_k(S,i,j)$ can be computed in $\mathcal{O}(k)$ time after $\mathcal{O}(n)$-time preprocessing.

Proof. For $\mathsf{RightMisper}_k(S,i,j)$, we use the kangaroo method [18,31] to compute the longest common prefix with at most k mismatches of $S[j+1\,..\,n-1]$ and U^∞ for $U = S[i\,..\,j]$. The value $\mathsf{lcp}(X^\infty, Y)$ for a substring X and a suffix Y of a string S, occurring at positions a and b, respectively, can be computed in constant time as follows. If $\mathsf{lcp}(S[a\,..\,n-1], S[b\,..\,n-1]) < |X|$ then we are done. Otherwise the answer is given by $|X| + \mathsf{lcp}(S[b\,..\,n-1], S[b+|X|\,..\,n-1])$. The computations for $\mathsf{LeftMisper}_k(S,i,j)$ are symmetric. $\square$

The $\mathcal{O}(k^3)$ and $\mathcal{O}(k^2)$ time complexities of computing X and Y follow from Lemmas 3 and 12, respectively (after $\mathcal{O}(n)$-time preprocessing). The sets X and Y consist of $\mathcal{O}(k^3)$ intervals and $\mathcal{O}(k^2)$ interval chains. The claim follows. $\square$

The correctness of the algorithm follows from Lemma 9. A detailed proof of the following lemma is deferred to the full version of this paper.

Lemma 11. *Assume $n \le 2m$. Let P' be a periodic sample in P with smallest period q and R be a P'-run in T. Let X and Y be defined as in the pseudocode of Run-Sample-Matching. Then $X \cup Y$ is a set of k-occurrences of P in T which is a superset of the solution to* SAMPLE-MATCH *for P' in R.*

4.4 Solution to Periodic-Periodic-Match Problem

Lemma 12. *We can compute in $\mathcal{O}(k^2)$ time a set of k-occurrences of P in T represented as $\mathcal{O}(k^2)$ interval chains that is a superset of the solution to the* PERIODIC-PERIODIC-MATCH *problem.*

Proof. We reduce our problem to the following abstract problem (see also Fig. 6).

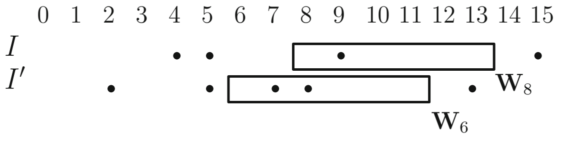

Fig. 6. An instance of the ABSTRACT PROBLEM with $m = 6$, $k = 3$, $q = 3$, $\delta = 2$, $I = \{4,5,9,15\}$ and $I' = \{2,5,7,8,13\}$. $8 \in A$, since for 6, we have that $|\mathbf{W}_8 \cap I| + |\mathbf{W}_6 \cap I'| \le 3$, $8 \equiv 2+6 \pmod 3$, $\mathbf{W}_8 \subseteq (4,15)$ and $\mathbf{W}_6 \subseteq (2,13)$.

Abstract Problem

Input: Positive integers m, k, q, δ and two sets I and I' such that $2 \leq |I|, |I'| \leq 2k+4$.

Output: The set A of integers z for which there exists z' such that:

1. $|\mathbf{W}_z \cap I| + |\mathbf{W}_{z'} \cap I'| \leq k$
2. $z \equiv \delta + z' \pmod{q}$
3. $\mathbf{W}_z \subseteq (\min I, \max I)$, $\mathbf{W}_{z'} \subseteq (\min I', \max I')$.

Claim. Periodic-Periodic-Match can be reduced in $\mathcal{O}(k)$ time to the Abstract Problem so that if z belongs to the solution to the Abstract Problem then $p = z$ is a solution to Periodic-Periodic-Match, which potentially may not satisfy the third condition of the problem.

Proof. Let the parameters m, k and q remain unchanged. We set $I = J \cup \{-1, |U|\}$, $I' = J' \cup \{-1, |V|\}$, and $\delta = i - i'$. □

Claim. Abstract Problem can be solved in $\mathcal{O}(k^2)$ time with the output represented as a collection of $\mathcal{O}(k^2)$ interval chains.

Proof. Let us denote $Z = (\min I, \max I - m + 1)$, $Z' = (\min I', \max I' - m + 1)$. We partition the set Z into intervals such that for all z in an interval, the set $\mathbf{W}_z \cap I$ is the same. For this, we use a sliding window approach. We generate events corresponding to x and $x - m + 1$ for all $x \in I$ and sort them. When z crosses an event, the set $\mathbf{W}_z \cap I$ changes. Thus we obtain a partition of Z into intervals $Z_1, \ldots, Z_{n_1}$ for $n_1 \leq 4k$. We obtain a similar partition of Z' into intervals $Z'_1, \ldots, Z'_{n_2}$ for $n_2 \leq 4k$.

Let us now fix Z_j and $Z'_{j'}$ (see also Fig. 7). First we check if condition 1 is satisfied for $z \in Z_j$ and $z' \in Z'_{j'}$. If so, we compute the set $X = \{(\delta + z') \bmod q : z' \in Z'_{j'}\}$. It is a single circular interval and can be computed in constant time.

The sought result is $\{z \in Z_j : z \bmod q \in X\}$. By Lemma 7, this set can be represented as a union of three chains and, as such, can be computed in $\mathcal{O}(1)$ time. The conclusion follows. □

This completes the proof of the lemma. □

In the solution we do not check if the sets $(\mathbf{W}_p \cap J) \oplus (-p)$ and $(\mathbf{W}_x \cap J') \oplus (-x)$ are disjoint. However, a k'-occurrence is found for some $k' < k$ otherwise.

4.5 Main Result

The following proposition summarizes the results from the previous subsections.

Proposition 13. *If $m \leq n \leq 2m$, k-CPM can be solved in $\mathcal{O}(n + k^5)$ time.*

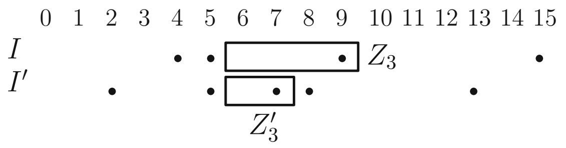

Fig. 7. The same instance of the ABSTRACT PROBLEM as in Fig. 6. For $Z_3 = \{6, 7, 8, 9\}$ and $Z'_3 = \{6, 7\}$ we get $X = \{0, 2\}$ and hence the sought result is $\{6, 8, 9\}$.

Proof. There are $k + 2$ ways to choose a sample P' in the pattern.

If the sample P' is not periodic, we use the algorithm of Lemma 4 for SAMPLE MATCHING in $\mathcal{O}(k^2)$ time (after $\mathcal{O}(n)$-time preprocessing). It returns a representation of k-occurrences as a union of $\mathcal{O}(k^2)$ intervals.

If the sample P' is periodic, we need to find all P'-runs in T. By Lemma 8, there are $\mathcal{O}(k)$ of them and they can all be computed in $\mathcal{O}(k)$ time (after $\mathcal{O}(n)$-time preprocessing). For every such P'-run R, we apply the Run-Sample-Matching algorithm. Its correctness follows from Lemma 11. By Lemma 10, it takes $\mathcal{O}(k^3)$ time and returns $\mathcal{O}(k^3)$ interval chains of k-occurrences of P in T (after $\mathcal{O}(n)$-time preprocessing). Over all P'-runs, this takes $\mathcal{O}(k^4)$ time after the preprocessing.

In total, SAMPLE MATCHING takes $\mathcal{O}(k^4)$ time for a given sample (after preprocessing), $\mathcal{O}(n+k^5)$ time in total, and returns $\mathcal{O}(k^5)$ intervals and interval chains of k-occurrences. Let us note that an interval is a special case of an interval chain. Hence, in the end, we apply Lemma 6 to compute the union of all chains of occurrences in $\mathcal{O}(n + k^5)$ time. ◻

We use the standard trick: splitting the text into $\mathcal{O}(n/m)$ fragments, each of length $2m$ (perhaps apart from the last one), starting at positions equal to $0 \bmod m$. We need to ensure that the data structures for answering lcp, lcs, and other internal queries over each such fragment of the text can be constructed in $\mathcal{O}(m)$ time in the case when our input alphabet Σ is large. As a preprocessing step we hash the letters of the pattern using perfect hashing. For each key, we assign a rank value from $\{1, \ldots, m\}$. This takes $\mathcal{O}(m)$ (expected) time and space [16]. When reading a fragment F of length (at most) $2m$ of the text we look up its letters using the hash table. If a letter is in the hash table we replace it in F by its rank value; otherwise we replace it by rank $m + 1$. We can now construct the data structures in $\mathcal{O}(m)$ time and the whole algorithm is implemented in $\mathcal{O}(m)$ space. If $\Sigma = \{1, \ldots, n^{\mathcal{O}(1)}\}$, the same bounds can be achieved deterministically using [36]. We combine Propositions 2 and 13 to get our final result.

Theorem 14. *Circular Pattern Matching with k Mismatches can be solved in $\mathcal{O}(\min(nk,\ n + \frac{n}{m}\,k^5))$ time and $\mathcal{O}(m)$ space.*

Our algorithms output all positions in the text where some rotation of the pattern occurs with k mismatches. It is not difficult to extend the algorithms to output, for each of these positions, a corresponding rotation of the pattern.

References

1. Abrahamson, K.R.: Generalized string matching. SIAM J. Comput. **16**(6), 1039–1051 (1987). https://doi.org/10.1137/0216067
2. Amir, A., Lewenstein, M., Porat, E.: Faster algorithms for string matching with k mismatches. J. Algorithms **50**(2), 257–275 (2004). https://doi.org/10.1016/S0196-6774(03)00097-X
3. Ayad, L.A.K., Barton, C., Pissis, S.P.: A faster and more accurate heuristic for cyclic edit distance computation. Pattern Recognit. Lett. **88**, 81–87 (2017). https://doi.org/10.1016/j.patrec.2017.01.018
4. Ayad, L.A.K., Pissis, S.P.: MARS: improving multiple circular sequence alignment using refined sequences. BMC Genomics **18**(1), 86 (2017). https://doi.org/10.1186/s12864-016-3477-5
5. Azim, M.A.R., Iliopoulos, C.S., Rahman, M.S., Samiruzzaman, M.: A filter-based approach for approximate circular pattern matching. In: Harrison, R., Li, Y., Măndoiu, I. (eds.) ISBRA 2015. LNCS, vol. 9096, pp. 24–35. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-19048-8_3
6. Azim, M.A.R., Iliopoulos, C.S., Rahman, M.S., Samiruzzaman, M.: A fast and lightweight filter-based algorithm for circular pattern matching. In: Baldi, P., Wang, W. (eds.) 5th ACM Conference on Bioinformatics, Computational Biology, and Health Informatics, BCB 2014, pp. 621–622. ACM (2014). https://doi.org/10.1145/2649387.2660804
7. Barton, C., Iliopoulos, C.S., Kundu, R., Pissis, S.P., Retha, A., Vayani, F.: Accurate and efficient methods to improve multiple circular sequence alignment. In: Bampis, E. (ed.) SEA 2015. LNCS, vol. 9125, pp. 247–258. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-20086-6_19
8. Barton, C., Iliopoulos, C.S., Pissis, S.P.: Fast algorithms for approximate circular string matching. Algorithms Mol. Biol. **9**, 9 (2014). https://doi.org/10.1186/1748-7188-9-9
9. Barton, C., Iliopoulos, C.S., Pissis, S.P.: Average-case optimal approximate circular string matching. In: Dediu, A.-H., Formenti, E., Martín-Vide, C., Truthe, B. (eds.) LATA 2015. LNCS, vol. 8977, pp. 85–96. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-15579-1_6
10. Bille, P., Fagerberg, R., Gørtz, I.L.: Improved approximate string matching and regular expression matching on Ziv-Lempel compressed texts. ACM Trans. Algorithms **6**(1), 3:1–3:14 (2009). https://doi.org/10.1145/1644015.1644018
11. Bringmann, K., Wellnitz, P., Künnemann, M.: Few matches or almost periodicity: faster pattern matching with mismatches in compressed texts. In: Chan, T.M. (ed.) 30th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, pp. 1126–1145. SIAM (2019). https://doi.org/10.1137/1.9781611975482.69
12. Chang, W.I., Marr, T.G.: Approximate string matching and local similarity. In: Crochemore, M., Gusfield, D. (eds.) CPM 1994. LNCS, vol. 807, pp. 259–273. Springer, Heidelberg (1994). https://doi.org/10.1007/3-540-58094-8_23
13. Clifford, R., Fontaine, A., Porat, E., Sach, B., Starikovskaya, T.: The k-mismatch problem revisited. In: Krauthgamer, R. (ed.) 27th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, pp. 2039–2052. SIAM (2016). https://doi.org/10.1137/1.9781611974331.ch142
14. Clifford, R., Kociumaka, T., Porat, E.: The streaming k-mismatch problem. In: Chan, T.M. (ed.) 30th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, pp. 1106–1125. SIAM (2019). https://doi.org/10.1137/1.9781611975482.68

15. Crochemore, M., Hancart, C., Lecroq, T.: Algorithms on Strings. Cambridge University Press, Cambridge (2007). https://doi.org/10.1017/cbo9780511546853
16. Fredman, M.L., Komlós, J., Szemerédi, E.: Storing a sparse table with $\mathcal{O}(1)$ worst case access time. J. ACM **31**(3), 538–544 (1984). https://doi.org/10.1145/828.1884
17. Fredriksson, K., Navarro, G.: Average-optimal single and multiple approximate string matching. ACM J. Exp. Algorithmics **9**(1.4), 1–47 (2004). https://doi.org/10.1145/1005813.1041513
18. Galil, Z., Giancarlo, R.: Parallel string matching with k mismatches. Theor. Comput. Sci. **51**, 341–348 (1987). https://doi.org/10.1016/0304-3975(87)90042-9
19. Gawrychowski, P., Straszak, D.: Beating $\mathcal{O}(nm)$ in approximate LZW-compressed pattern matching. In: Cai, L., Cheng, S.-W., Lam, T.-W. (eds.) ISAAC 2013. LNCS, vol. 8283, pp. 78–88. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-45030-3_8
20. Gawrychowski, P., Uznański, P.: Order-preserving pattern matching with k mismatches. Theor. Comput. Sci. **638**, 136–144 (2016). https://doi.org/10.1016/j.tcs.2015.08.022
21. Gawrychowski, P., Uznański, P.: Towards unified approximate pattern matching for Hamming and L_1 distance. In: Chatzigiannakis, I., Kaklamanis, C., Marx, D., Sannella, D. (eds.) Automata, Languages, and Programming, ICALP 2018. LIPIcs, vol. 107, pp. 62:1–62:13. Schloss Dagstuhl-Leibniz-Zentrum für Informatik (2018). https://doi.org/10.4230/LIPIcs.ICALP.2018.62
22. Grossi, R., Iliopoulos, C.S., Mercas, R., Pisanti, N., Pissis, S.P., Retha, A., Vayani, F.: Circular sequence comparison: algorithms and applications. Algorithms Mol. Biol. **11**, 12 (2016). https://doi.org/10.1186/s13015-016-0076-6
23. Hazay, C., Lewenstein, M., Sokol, D.: Approximate parameterized matching. ACM Trans. Algorithms **3**(3), 29 (2007). https://doi.org/10.1145/1273340.1273345
24. Hirvola, T., Tarhio, J.: Bit-parallel approximate matching of circular strings with k mismatches. ACM J. Exp. Algorithmics **22**, 1–5 (2017). https://doi.org/10.1145/3129536
25. Iliopoulos, C.S., Pissis, S.P., Rahman, M.S.: Searching and indexing circular patterns. In: Elloumi, M. (ed.) Algorithms for Next-Generation Sequencing Data, pp. 77–90. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-59826-0_3
26. Kociumaka, T.: Efficient data structures for internal queries in texts. Ph.D. thesis, University of Warsaw, October 2018. https://www.mimuw.edu.pl/~kociumaka/files/phd.pdf
27. Kociumaka, T., Radoszewski, J., Rytter, W., Straszyński, J., Waleń, T., Zuba, W.: Efficient representation and counting of antipower factors in words (2018). http://arxiv.org/abs/1812.08101
28. Kociumaka, T., Radoszewski, J., Rytter, W., Straszyński, J., Waleń, T., Zuba, W.: Efficient representation and counting of antipower factors in words. In: Martín-Vide, C., Okhotin, A., Shapira, D. (eds.) LATA 2019. LNCS, vol. 11417, pp. 421–433. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-13435-8_31
29. Kociumaka, T., Radoszewski, J., Rytter, W., Waleń, T.: Internal pattern matching queries in a text and applications. In: Indyk, P. (ed.) 26th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, pp. 532–551. SIAM (2015). https://doi.org/10.1137/1.9781611973730.36
30. Kosaraju, S.: Efficient string matching (1987, manuscript)
31. Landau, G.M., Vishkin, U.: Efficient string matching with k mismatches. Theor. Comput. Sci. **43**, 239–249 (1986). https://doi.org/10.1016/0304-3975(86)90178-7

32. Palazón-González, V., Marzal, A.: On the dynamic time warping of cyclic sequences for shape retrieval. Image Vision Comput. **30**(12), 978–990 (2012). https://doi.org/10.1016/j.imavis.2012.08.012
33. Palazón-González, V., Marzal, A.: Speeding up the cyclic edit distance using LAESA with early abandon. Pattern Recognit. Lett. **62**, 1–7 (2015). https://doi.org/10.1016/j.patrec.2015.04.013
34. Palazón-González, V., Marzal, A., Vilar, J.M.: On hidden Markov models and cyclic strings for shape recognition. Pattern Recognit. **47**(7), 2490–2504 (2014). https://doi.org/10.1016/j.patcog.2014.01.018
35. Porat, B., Porat, E.: Exact and approximate pattern matching in the streaming model. In: 50th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2009, pp. 315–323. IEEE Computer Society (2009). https://doi.org/10.1109/FOCS.2009.11
36. Ružić, M.: Constructing efficient dictionaries in close to sorting time. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008. LNCS, vol. 5125, pp. 84–95. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-70575-8_8
37. Tiskin, A.: Threshold approximate matching in grammar-compressed strings. In: Holub, J., Zdárek, J. (eds.) Prague Stringology Conference 2014, PSC 2014, pp. 124–138. Department of Theoretical Computer Science, Faculty of Information Technology, Czech Technical University in Prague (2014). http://www.stringology.org/event/2014/p12.html