Optimal bounds for computing α-gapped repeats

Maxime Crochemore, Roman Kolpakov, Gregory Kucherov

Version of Record: https://www.sciencedirect.com/science/article/pii/S0890540119300513
Manuscript_e67fe1d5ce12486f95dd605b75a8a565

Optimal bounds for computing α-gapped repeats

Maxime Crochemore[a,b], Roman Kolpakov[c,d], Gregory Kucherov[b]

[a] *King's College London, London WC2R 2LS, UK*
[b] *LIGM/CNRS, Université Paris-Est, 77454 Marne-la-Vallée, France*
[c] *Lomonosov Moscow State University, Leninskie Gory, Moscow, 119992 Russia*
[d] *Dorodnicyn Computing Centre, FRC CSC RAS, Vavilov st. 40, Moscow, 119333 Russia*

Abstract

Following (Kolpakov et al., 2013; Gawrychowski and Manea, 2015), we continue the study of *α-gapped repeats* in strings, defined as factors of the form uvu with $|uv| = |u| + |v| \leq \alpha|u|$. Our main result is the $O(\alpha n)$ bound on the number of *maximal* α-gapped repeats in a string of length n, previously proved to be $O(\alpha^2 n)$ in (Kolpakov et al., 2013). For a closely related notion of maximal δ-subrepetition (maximal factors of exponent between $1+\delta$ and 2), our result implies the $O(n/\delta)$ bound on their number, which improves the bound of (Kolpakov et al., 2010) by a $\log n$ factor.

We also prove an algorithmic time bound $O(\alpha n + S)$ (S size of the output) for computing all maximal α-gapped repeats. Our solution, inspired by (Gawrychowski and Manea, 2015), is different from the recently published proof by (Tanimura et al., 2015) of the same bound. Together with our bound on S, this implies an $O(\alpha n)$-time algorithm for computing all maximal α-gapped repeats.

Keywords: combinatorics on words, algorithms on strings, combinatorial algorithms, time complexity, repeats, gapped repeats, subrepetitions

1. Introduction

Notation and basic definitions. Let $w = w[1]w[2]\dots w[n] = w[1\,..\,n]$ be an arbitrary word. The length n of w is denoted by $|w|$. For any $1 \leq i \leq j \leq n$,

Email addresses: Maxime.Crochemore@kcl.ac.uk (Maxime Crochemore), foroman@mail.ru (Roman Kolpakov), Gregory.Kucherov@univ-mlv.fr (Gregory Kucherov)

the word $w[i] \cdots w[j]$ is called a *factor* of w and denoted by $w[i\,..\,j]$. Note that notation $w[i\,..\,j]$ denotes two entities: a word and its occurrence starting at position i in w. To underline the second meaning, we will sometimes use the term *segment*. Speaking about the equality between factors can also be ambiguous, as it may mean that the factors are identical words or the same segment (i.e. the same occurrence). If two factors u and v are identical words, we call them *equal* and denote this by $u = v$. To express that u and v are the same segment, we use the notation $u \equiv v$. For any $i = 1 \ldots n$, the factor $w[1\,..\,i]$ (resp. $w[i\,..\,n]$) is a *prefix* (resp. *suffix*) of w. By *positions* on w we mean indices $1, 2, \ldots, n$ of letters in w. For any segment $v \equiv w[i\,..\,j]$ of w, positions i and j are called respectively *start position* and *end position* of v and denoted by $beg(v)$ and $end(v)$ respectively. Let u, v be two segments of w. Segment u *is contained* in v iff $beg(v) \leq beg(u)$ and $end(u) \leq end(v)$. Letter $w[i]$ *is contained* in v iff $beg(v) \leq i \leq end(v)$.

A positive integer p is called a *period* of w if $w[i] = w[i + p]$ for each $i = 1, \ldots, n - p$. Clearly, a word can have several periods. We denote by $\mathrm{per}(w)$ the *smallest period* of w and define the *exponent* of w as $\exp(w) = |w|/\mathrm{per}(w)$. A word is called *periodic* if its exponent is at least 2. Occurrences of periodic words are called *repetitions*.

Repetitions, squares, runs. Patterns in strings formed by repeated factors are of primary importance in word combinatorics [1] as well as in various applications such as string matching algorithms [2, 3], molecular biology [4], or text compression [5]. The simplest and best known example of such patterns is a factor of the form uu, where u is a nonempty word. Such repetitions are called *squares*. Squares have been extensively studied. While the number of all square occurrences can be quadratic (consider word $\mathtt{a}^n$), it is known that the number of *primitively-rooted* squares is $O(n \log n)$ [3], where a square uu is primitively-rooted if the exponent of u is not an integer greater than 1. An optimal $O(n \log n)$-time algorithm for finding all primitively-rooted squares was proposed in [6] (see also [7, 8, 9]).

Repetitions can be seen as a natural generalization of squares. A repetition in a given word is called *maximal* if it cannot be extended by at least one letter to the left nor to the right without changing (increasing) its minimum period. More precisely, a repetition $r \equiv w[i\,..\,j]$ in w is called *maximal* if it satisfies the following conditions:

1. $w[i - 1] \neq w[i - 1 + \mathrm{per}(r)]$ if $i > 1$,
2. $w[j + 1 - \mathrm{per}(r)] \neq w[j + 1]$ if $j < n$.

For example, word `cababaaa` has two maximal repetitions: `ababa` and `aaa`. Maximal repetitions are usually called *runs* in the literature. Since any repetition is contained in some run, the set of all runs can be considered as a compact encoding of all repetitions in the word. This set has many useful applications, see, e.g., [10]. For any word w, we denote by $\mathcal{R}(w)$ the number of maximal repetitions in w and by $\mathcal{E}(w)$ the sum of exponents of all maximal repetitions in w. The following statements are proved in [11].

Theorem 1. $\max_{|w|=n} \mathcal{E}(w) = O(n)$.

Corollary 1. $\max_{|w|=n} \mathcal{R}(w) = O(n)$.

A series of articles (e.g., [12, 13, 14, 15, 16, 17, 18]) focused on more precise upper bounds on $\mathcal{E}(w)$ and $\mathcal{R}(w)$ trying to obtain the best possible constant factor behind the O-notation. A breakthrough in this direction was recently made in [19] (preliminary version in [20]) where the so-called "runs conjecture" $\mathcal{R}(w[1..n]) < n$ was proved. To the best of our knowledge, the currently best upper bound $\mathcal{R}(w[1..n]) \leq \frac{22}{23}n$ on $\mathcal{R}(w)$ for binary words w is shown in [21].

On the algorithmic side, an $O(n)$-time algorithm for finding all runs in a word of length n was proposed in [11] for the case of constant-size alphabet. Another $O(n)$-time algorithm, based on a different approach, has been proposed in [19]. The $O(n)$ time bound holds for (polynomially-bounded) integer alphabets as well, see, e.g., [19]. However, for the case of unbounded-size alphabets where characters can only be tested for equality, the lower bound $\Omega(n \log n)$ on computing all runs has been known for a long time [22]. It is an interesting open question (raised over 20 years ago in [23]) whether the $O(n)$ bound holds for an unbounded linearly-ordered alphabet. The best known bound for this problem is currently $O(n \cdot \alpha(n))$ [24] (here $\alpha(\cdot)$ is the inverse of the Ackermann function), improving on [25, 26].

Gapped repeats and subrepetitions. Another natural generalization of squares and runs are factors of the form uvu where u and v are nonempty words. We call such factors *gapped repeats*. For a gapped repeat uvu, the left (resp. right) occurrence of u is called its *left arm* (resp. *right arm*), and v is called its *gap*. The *period* of this gapped repeat is $|u| + |v|$. For a gapped repeat π, we denote the length of its arms by $c(\pi)$ and its period by $p(\pi)$. Note that $p(\pi)$ is not necessarily equal to $\mathrm{per}(\pi)$ but $\mathrm{per}(\pi) \leq p(\pi)$. Note also that gapped repeats with distinct periods can have the same start and end

positions in the word, i.e. they can occur in the same segment in the word. For example, in string `cabacaabaa`, segment `abacaaba` corresponds to two gapped repeats having arms `a` and `aba` and periods 7 and 5 respectively. Gapped repeats having different periods but forming the same segment are considered distinct. This means that in order to specify a gapped repeat it is generally not sufficient to specify its segment. Let u' and u'' be equal factors such that $end(u') + 1 < beg(u'')$. Then we denote by (u', u'') the gapped repeat with left arm u' and right arm u''. Let $l = |u'| = |u''|$. Note that, since $u' = u''$, then for any i, j such that $1 \leq i \leq j \leq l$ we have $u'[i\,..\,j] = u''[i\,..\,j]$. We will call factors $u'[i\,..\,j]$ and $u''[i\,..\,j]$ *corresponding factors* for the repeat (u', u'').

For any real $\alpha > 1$, a gapped repeat π is called *α-gapped* if $p(\pi) \leq \alpha c(\pi)$. The maximality of gapped repeats is defined similarly to that of repetitions: a gapped repeat $(w[i'\,..\,j'], w[i''\,..\,j''])$ in w is called *maximal* if it satisfies both of the following conditions:

1. $w[i' - 1] \neq w[i'' - 1]$ if $i' > 1$,
2. $w[j' + 1] \neq w[j'' + 1]$ if $j'' < n$.

In other words, a gapped repeat π is maximal if its arms cannot be extended to the left nor to the right by one letter without breaking its period $p(\pi)$. As observed in [27], any α-gapped repeat is contained either in a (unique) maximal α-gapped repeat with the same period, or in a (unique) maximal repetition with a period that is a divisor of the repeat's period. For example, in the above string `cabacaabaa`, the gapped repeat (`ab`)`aca`(`ab`) is contained in the maximal repeat (`aba`)`ca`(`aba`) with the same period 5. In string `cabaaabaaa`, the gapped repeat (`ab`)`aa`(`ab`) of period 4 is contained in the maximal repetition `abaaabaaa` of period 4. Since all maximal repetitions can be computed efficiently in $O(n)$ time (see above), the problem of computing all α-gapped repeats in a word can be reduced to the problem of finding all maximal α-gapped repeats.

Several variants of the problem of computing gapped repeats have been studied earlier. In [28], it was shown that all maximal gapped repeats with a gap length belonging to a specified interval can be found in time $O(n \log n + S)$, where n is the word length and S is output size. In [29], an algorithm was proposed for finding all gapped repeats with a fixed gap length d running in time $O(n \log d + S)$. In [27], it was proved that the number of maximal α-gapped repeats in a word of length n is bounded by $O(\alpha^2 n)$ and all maximal α-gapped repeats can be found in $O(\alpha^2 n)$ time for

the case of integer alphabet. A new approach to computing gapped repeats was proposed in [30, 31] and more recently in [32]. In particular, in [30] it was shown that the longest α-gapped repeat in a word of length n over an integer alphabet can be found in $O(\alpha n)$ time. In a recent paper [33], an algorithm was proposed for finding all maximal α-gapped repeats in $O(\alpha n + S)$ time where S is the output size, for a constant-size alphabet. The algorithm uses an approach previously introduced in [34].

Recall that repetitions are segments of exponent at least 2. Another way to approach gapped repeats is to consider segments with exponent smaller than 2, but (strictly) greater than 1. Clearly, such a segment corresponds to a gapped repeat $\pi = uvu$ with $\mathrm{per}(\pi) = p(\pi) = |u| + |v|$. We call such factors (segments) *subrepetitions*. More precisely, for any δ, $0 < \delta < 1$, by a δ-subrepetition we mean a factor z that satisfies $1 + \delta \leq \exp(z) < 2$. Again, the notion of maximality straightforwardly applies to subrepetitions as well: maximal subrepetitions are defined exactly in the same way as maximal repetitions. The relationship between maximal subrepetitions and maximal gapped repeats was clarified in [27]. Directly from the relationship, a maximal subrepetition π in a string w corresponds to a maximal gapped repeat satisfying $p(\pi) = \mathrm{per}(\pi)$. For example, in word `aababababcababac`, the maximal subrepetition `abababcababa` = $\mathtt{abababc}^{12/7}$ of period 7 corresponds to the maximal gapped repeat (`ababa`)`bc`(`ababa`) of period 7. Furthermore, a maximal δ-subrepetition corresponds to a maximal $\frac{1}{\delta}$-gapped repeat. However, there may be more maximal $\frac{1}{\delta}$-gapped repeats than maximal δ-subrepetitions. One reason for that is that one maximal subrepetition may correspond to several maximal gapped repeats with different periods. Thus, in the above example, maximal subrepetition `abababcababa` contains three maximal gapped repeats: (`ababa`)`bc`(`ababa`), (`aba`)`babcab`(`aba`), and (`a`)`bababcabab`(`a`). Another reason is that a maximal gapped repeat may occur inside a repetition and not a subrepetition: e.g. word `aababababb` contains a maximal repetition `ababab` that corresponds to a maximal gapped repeat (`ab`)`ab`(`ab`).

Some combinatorial results on the number of maximal subrepetitions in a string were obtained in [35]. In particular, it was proved that the number of maximal δ-subrepetitions in a word of length n is bounded by $O(\frac{n}{\delta} \log n)$. In [27], an $O(n/\delta^2)$ bound on the number of maximal δ-subrepetitions in a word of length n was obtained. Moreover, in [27], two algorithms were proposed for finding all maximal δ-subrepetitions in the word running respectively in $O(\frac{n \log\log n}{\delta^2})$ time and in $O(n \log n + \frac{n}{\delta^2} \log \frac{1}{\delta})$ expected time,

over the integer alphabet. In [34], it is shown that all subrepetitions with the largest exponent (over all subrepetitions) in an overlap-free string can be found in $O(n)$ time for a constant-size alphabet.

Our results. Throughout the paper we assume a constant-size alphabet. In this work we improve the results of [27] on maximal gapped repeats: we prove an $O(\alpha n)$ bound on the number of maximal α-gapped repeats in a word of length n (Section 3). We show that this bound is asymptotically tight. The same bound has recently been proved independently in [32] using a different method based on techniques from [27]. From our bound, we additionally derive an $O(n/\delta)$ bound on the number of maximal δ-subrepetitions occurring in a word, which improves the bound of [35] by a $\log n$ factor. Then, based on the algorithm of [30], we obtain an asymptotically optimal $O(\alpha n)$ time bound for computing all maximal α-gapped repeats in a word (Section 4). Note that this bound also follows from [33] that presents an $O(\alpha n + S)$ algorithm for computing all maximal α-gapped repeats. Here we present an alternative algorithm with the same bound that we obtained independently of [33, 32].

2. Preliminaries

In this section we state a few propositions that are used later in the paper. The following fact is well-known (see, e.g., [36, Proposition 2]).

Proposition 1. *Any period p of a word w for which $|w| \geq 2p$ is divisible by* $\mathrm{per}(w)$*, the smallest period of w.*

Let Δ be some natural number. A period p of some word w is called a Δ-*period* if p is divisible by Δ. The minimum Δ-period of w, if it exists, is denoted by $p_\Delta(w)$. The word w is called Δ-periodic if $|w| \geq 2p_\Delta(w)$. It is clear that any Δ-periodic word is also periodic. Proposition 1 can be generalized in the following way.

Proposition 2. *Any Δ-period p of a word w for which $|w| \geq 2p$ is divisible by $p_\Delta(w)$.*

Proof. By Proposition 1, period p is divisible by $\mathrm{per}(w)$, so p is divisible by $\mathrm{lcm}(\mathrm{per}(w), \Delta)$. On the other hand, $\mathrm{lcm}(\mathrm{per}(w), \Delta)$ is a Δ-period of w. Thus, $p_\Delta(w) = \mathrm{lcm}(\mathrm{per}(w), \Delta)$, and p is divisible by $p_\Delta(w)$ as stated. □

Consider an arbitrary word $w = w[1\,..\,n]$ of length n. Note that any repetition r in w can be extended to a unique maximal repetition r' with the same minimum period. We call r' the *extension* of r.

Let r be a repetition in the word w. We call any factor of w of length $\mathrm{per}(r)$ that is contained in r a *cyclic root* of r. For cyclic roots we have the following property, proved e.g. in [27, Proposition 2].

Proposition 3. *Two cyclic root u', u'' of a repetition r are equal if and only if $beg(u') \equiv beg(u'') \pmod{\mathrm{per}(r)}$.*

3. Number of maximal repeats and subrepetitions

In this section, we obtain an improved upper bound on the number of maximal gapped repeats and subrepetitions in a string w. Following the general approach of [27], we split all maximal gapped repeats into three categories according to the periodicity properties of repeats: periodic, semiperiodic and ordinary repeats. Bounds for periodic and semiperiodic repeats are directly borrowed from [27], while for ordinary repeats, we obtain a better bound. Since the number of α-gapped repeats can be bounded by the number of $\lceil\alpha\rceil$-gapped repeats, we assume without loss of generality that $\alpha = k$ for a natural number k.

We describe now the three types of maximal gapped repeats.

Periodic repeats. We say that a maximal gapped repeat uvu is *periodic* if its arms u are periodic strings (i.e. of exponent at least 2). The set of all periodic maximal α-gapped repeats in w is denoted by $\mathcal{PP}_\alpha$. The following bound on the size of $\mathcal{PP}_\alpha$ was obtained in [27, Corollary 6].

Lemma 1. *$|\mathcal{PP}_k| = O(kn)$ for any integer $k > 1$.*

Semiperiodic repeats. A maximal gapped repeat uvu is called *prefix* (respectively, *suffix*) *semiperiodic* if the arms u of this repeat are not periodic, but have a periodic prefix (respectively, suffix) whose length is at least half the arm length. For example, the repeat *abababaabacccababaaba* with arms *abababaaba* is prefix semiperiodic since its arms have the periodic prefix *ababa*. A maximal gapped repeat is *semiperiodic* if it is either prefix semiperiodic or suffix semiperiodic. The set of all semiperiodic α-gapped maximal repeats is denoted by $\mathcal{SP}_\alpha$. In [27, Corollary 8], the following bound was obtained on semiperiodic maximal α-gapped repeats.

Lemma 2 ([27]). *$|\mathcal{SP}_k| = O(kn)$ for any integer $k > 1$.*

Ordinary repeats. Maximal gapped repeats which are neither periodic nor semiperiodic are called *ordinary*. The set of all ordinary maximal α-gapped repeats in the word w is denoted by $\mathcal{OP}_\alpha$. In the rest of this section, we prove that the cardinality of $\mathcal{OP}_\alpha$ is $O(\alpha n)$.

To estimate the number of ordinary maximal k-gapped repeats, we use the following idea from [36]. We represent a maximal repeat $\pi \equiv (u', u'')$ in $\mathcal{OP}_k$ by a triple (i, j, c) where $i = beg(u')$, $j = beg(u'')$ and $c = c(\pi) = |u'| = |u''|$. We view such triples as *points* in a three-dimensional space. Obviously, π is uniquely defined by values i, j and c, therefore two different repeats from $\mathcal{OP}_k$ can not be represented by the same point.

For any two points (i', j', c'), (i'', j'', c'') we say that point (i', j', c') *covers* point (i'', j'', c'') if $i' \le i'' \le i' + c'/6$, $j' \le j'' \le j' + c'/6$, $c' \ge c'' \ge \frac{2c'}{3}$. A point is *covered* by a repeat π if it is covered by the point representing π. By $V[\pi]$ we denote the set of all points covered by a repeat π. Then we show that no point can be covered by two different repeats from $\mathcal{OP}_k$.

Lemma 3. *Two different repeats from $\mathcal{OP}_k$ cannot cover the same point.*

Proof. By contradiction let $\pi_1 \equiv (u'_1, u''_1)$, $\pi_2 \equiv (u'_2, u''_2)$ be two different repeats from $\mathcal{OP}_k$ covering the same point (i, j, c). Denote $c_1 = c(\pi_1)$, $c_2 = c(\pi_2)$, $p_1 = \mathrm{per}(\pi_1)$, $p_2 = \mathrm{per}(\pi_2)$. Note that $beg(u''_1) - beg(u'_1) = p_1$ and $beg(u''_2) - beg(u'_2) = p_2$. Without loss of generality, we assume $c_1 \ge c_2$. From $c_1 \ge c_2$, $c_1 \ge c \ge \frac{2c_1}{3}$, and $c_2 \ge c \ge \frac{2c_2}{3}$ we have $c_1 \ge c_2 \ge \frac{2c_1}{3}$, i.e. $c_2 \le c_1 \le \frac{3c_2}{2}$. Note that $w[i]$ is contained in both u'_1, u'_2, i.e. these left arms overlap. Therefore, if $p_1 = p_2$, then the left arms u'_1, u'_2 must coincide due to the maximality of these repeats, and then the repeats π_1 and π_2 must coincide as well. Thus, $p_1 \ne p_2$. Denote $\Delta = |p_1 - p_2| > 0$. From $beg(u'_1) \le i \le beg(u'_1) + c_1/6$ and $beg(u''_1) \le j \le beg(u''_1) + c_1/6$ we have

$$(j - i) - c_1/6 \le p_1 \le (j - i) + c_1/6.$$

Analogously, we have

$$(j - i) - c_2/6 \le p_2 \le (j - i) + c_2/6.$$

Thus $\Delta \le (c_1 + c_2)/6$, which, together with the inequality $c_1 \le \frac{3c_2}{2}$, implies $\Delta \le \frac{5c_2}{12}$.

First consider the case when one of the arms u'_1, u'_2 is contained in the other, i.e. u'_2 is contained in u'_1 since $c_1 \ge c_2$. In this case, u''_1 contains some

factor $\widehat{u}_2''$ corresponding to the factor u_2' in u_1'. Thus $beg(u_2'') - beg(u_2') = p_2$, $beg(\widehat{u}_2'') - beg(u_2') = p_1$ and $u_2'' = \widehat{u}_2'' = u_2'$, so that

$$|beg(u_2'') - beg(\widehat{u}_2'')| = \Delta,$$

and then Δ is a period of u_2'' for which $\Delta \leq \frac{5}{12}c_2 = \frac{5}{12}|u_2''|$. Thus, u_2'' is periodic, which contradicts that π_2 is not periodic.

Now consider the case when u_1', u_2' are not contained in one another. Denote by z' the overlap between u_1' and u_2'. Let z' be a suffix of u_l' and a prefix of u_r' where $l, r = 1, 2$, $l \neq r$. Then u_l'' contains a suffix z'' corresponding to the suffix z' in u_l', and u_r'' contains a prefix $\widehat{z}''$ corresponding to the prefix z' in u_r'. Then $z'' = \widehat{z}'' = z'$. Thus $beg(z'') - beg(z') = p_l$ and $beg(\widehat{z}'') - beg(z') = p_r$ and $z'' = \widehat{z}'' = z'$, so that

$$|beg(z'') - beg(\widehat{z}'')| = |p_l - p_r| = \Delta,$$

therefore Δ is a period of z'' and $\widehat{z}''$, and so Δ is a period of z'. Note that in this case

$$beg(u_l') < beg(u_r') \leq i \leq beg(u_l') + c_l/6,$$

therefore $0 < beg(u_r') - beg(u_l') \leq c_l/6$. Thus

$$|z'| = c_l - (beg(u_r') - beg(u_l')) \geq \frac{5}{6}c_l \geq \frac{5}{6}c_2.$$

From $\Delta \leq \frac{5}{12}c_2$ and $c_2 \leq \frac{6}{5}|z'|$ we obtain $\Delta \leq |z'|/2$. Thus, z' is a periodic suffix of u_l' such that $|z'| \geq \frac{5}{6}|u_l'|$, i.e. π_l is either suffix semiperiodic or periodic which contradicts $\pi_l \in \mathcal{OP}_k$ and ends the proof. □

Denote by $\mathcal{Q}_k$ the set of all points (i, j, c) for which $1 \leq i, j, c \leq n$ and $i < j \leq i + (\frac{3}{2}k + \frac{1}{4})c$.

Lemma 4. *Any point covered by a repeat from $\mathcal{OP}_k$ belongs to $\mathcal{Q}_k$.*

Proof. Let a point (i, j, c) be covered by some repeat $\pi \equiv (u', u'')$ from $\mathcal{OP}_k$. Denote $c' = c(\pi)$. Note that $w[i]$ and $w[j]$ are contained respectively in u' and u'' and $n > c' \geq c \geq \frac{2c'}{3} > 0$, so inequalities $1 \leq i, j, c \leq n$ and $i < j$ follow. Note also that

$$j \leq beg(u'') + c'/6 = beg(u') + \mathrm{per}(\pi) + c'/6 \leq i + kc' + c'/6,$$

therefore, taking into account $c' \leq \frac{3c}{2}$, we have $j \leq i + (\frac{3}{2}k + \frac{1}{4})c$. This shows that (i, j, c) belongs to $\mathcal{Q}_k$ as stated. □

From Lemmas 3 and 4, we obtain

Lemma 5. $|\mathcal{OP}_k| = O(nk)$.

PROOF. Assign to each point (i, j, c) the weight $\rho(i, j, c) = 1/c^3$. For any finite set A of points, we define

$$\rho(A) = \sum_{(i,j,c)\in A} \rho(i, j, c) = \sum_{(i,j,c)\in A} \frac{1}{c^3}.$$

Let π be an arbitrary repeat from $\mathcal{OP}_k$ represented by the point (i', j', c'). Then

$$\begin{aligned} \rho(V[\pi]) &= \sum_{i'\le i\le i'+c'/6} \sum_{j'\le j\le j'+c'/6} \sum_{2c'/3\le c\le c'} \frac{1}{c^3} \\ &> \frac{c'^2}{36} \sum_{2c'/3\le c\le c'} \frac{1}{c^3}. \end{aligned}$$

Using an estimation of sums by integrals (as in [27]), one can deduce that $\sum_{2c'/3\le c\le c'} \frac{1}{c^3} \ge \frac{5}{32}\frac{1}{c'^2}$ for any c'. Thus, for any π from $\mathcal{OP}_k$

$$\rho(V[\pi]) > \frac{1}{36}\frac{5}{32} = \Omega(1).$$

Therefore,

$$\sum_{\pi\in\mathcal{OP}_k} \rho(V[\pi]) = \Omega(|\mathcal{OP}_k|). \tag{1}$$

Note also that

$$\begin{aligned} \rho(\mathcal{Q}_k) &\le \sum_{i=1}^{n} \sum_{i<j\le i+(\frac{3}{2}k+\frac{1}{4})c} \sum_{c=1}^{n} \frac{1}{c^3} \\ &< n(\frac{3}{2}k+\frac{1}{4})c \sum_{c=1}^{n} \frac{1}{c^3} < 2nk \sum_{c=1}^{n} \frac{1}{c^2} < 2nk \sum_{c=1}^{\infty} \frac{1}{c^2} = \frac{nk\pi^2}{3}. \end{aligned}$$

Thus,

$$\rho(\mathcal{Q}_k) = O(nk). \tag{2}$$

By Lemma 4, any point covered by a repeat from $\mathcal{OP}_k$ belongs to $\mathcal{Q}_k$. On the other hand, by Lemma 3, no point of $\mathcal{Q}_k$ can be covered by two repeats from $\mathcal{OP}_k$. Therefore,

$$\sum_{\pi \in \mathcal{OP}_k} \rho(V[\pi]) \leq \rho(\mathcal{Q}_k).$$

Thus, using (1) and (2), we conclude that $|\mathcal{OP}_k| = O(nk)$ as expected. □

Eventually, putting together Lemma 1, Lemma 2, and Lemma 5, we obtain that, for any integer $k \geq 2$, the number of maximal k-gapped repeats in w is $O(nk)$. As mentioned above, the bound straightforwardly generalizes to the case of real $\alpha > 1$ since the number of α-gapped repeats can be bounded by the number of $\lceil \alpha \rceil$-gapped repeats. We conclude with the following statement.

Theorem 2. *For any $\alpha > 1$, the number of maximal α-gapped repeats in w is $O(\alpha n)$.*

Note that the bound of Theorem 2 is asymptotically tight under the natural restriction $\alpha \leq n$. To see this, consider words $w_k = (0110)^k$. It is easy to check that for a large enough α and $k = \Omega(\alpha)$, w_k contains $\Theta(\alpha|w_k|)$ maximal α-gapped repeats whose arms are single-letter words. For example, for $\alpha = 3+4\ell$ where $\ell = 0, 1, 2, \ldots$, word w_k contains $\frac{1}{2}(\ell+1)(2k-\ell)$ maximal α-gapped repeats $01(1001)^i10$ with arms 0, for $i \leq \ell$.

We now apply Theorem 2 to obtain an upper bound on the number of maximal δ-subrepetitions. The following proposition [27, Proposition 3] follows from the fact that each maximal δ-subrepetition defines at least one maximal $1/\delta$-gapped repeat (cf. Introduction).

Proposition 4 ([27]). *For $0 < \delta < 1$, the number of maximal δ-subrepetitions in a string is no more than the number of maximal $1/\delta$-gapped repeats.*

Theorem 2 combined with Proposition 4 immediately implies the upper bound for maximal δ-subrepetitions stated in the following theorem. The bound improves on the bound of [35] by a $\log n$ factor.

Theorem 3. *For $0 < \delta < 1$, the number of maximal δ-subrepetitions in w is $O(n/\delta)$.*

The $O(n/\delta)$ bound on the number of maximal δ-subrepetitions is asymptotically tight, at least on an unbounded alphabet: word $\mathtt{ab}_1\mathtt{ab}_2 \ldots \mathtt{ab}_k$ of length $n = 2k$ contains $\Omega(n/\delta)$ maximal δ-subrepetitions for $\delta \leq 1/2$.

4. Computing all maximal α-gapped repeats

In this section, we present an $O(\alpha n + S)$ algorithm for computing all maximal α-gapped repeats in a word w of length n. The same bound has been independently proved in [33], here we present a different solution. Together with the $O(\alpha n)$ bound of Theorem 2, this yields an $O(\alpha n)$-time algorithm.

To get the result, α-gapped repeats are split into two classes analyzed in the following sections.

4.1. Computing PR-repeats

Some maximal α-gapped repeats can be located within maximal repetitions (runs). For example, word `cababababababaa` contains maximal gapped repeats (`a`)`bababababab`(`a`), (`aba`)`babab`(`aba`) and (`ababa`)`b`(`ababa`) within the run `abababababa` = $(\mathtt{ab})^{11/2}$. In this section, we describe the structure of such repeats, and in particular the structure of repeats that are periodic (see Section 3), like the repeat (`ababa`)`b`(`ababa`) above. We show how those maximal α-gapped repeats can be extracted from the runs in the input string. Repeats that are located within runs but are not periodic are found separately, together with other repeats (periodic or not) that are not located within runs. This latter part is described in the next section.

Let $\pi \equiv (u', u'')$ be a periodic gapped repeat. If the extensions of u' and u'' are the same repetition r then we say that r *generates* π and we call π a *PR-repeat* (that stands for *Periodic Run-generated*). Other gapped repeats are just called *non-PR repeats*. Later on we use the following sufficient condition for PR-repeats.

Proposition 5. *Let $\pi \equiv (u', u'')$ be a maximal gapped repeat whose arms u' and u'' contain a pair of corresponding factors having the same extension r. Then π is generated by r.*

Proof. Observe that to prove the proposition, it is enough to show that both arms u' and u'' are contained in r, i.e. $beg(r) \leq beg(u')$ and $end(r) \geq end(u'')$. By contradiction, let $beg(r) > beg(u')$. Then both letters $w[beg(r) - 1]$ and $w[beg(r) - 1 + \mathrm{per}(r)]$ are contained in u'. Let these letters be respectively j-th and $(j + \mathrm{per}(r))$-th letters of u'. Then we have $u''[j] = u'[j] \neq$

$u'[j + \mathrm{per}(r)] = u''[j + \mathrm{per}(r)]$, i.e. $u''[j] \neq u''[j + \mathrm{per}(r)]$, which is a contradiction to the fact that both letters $u''[j]$ and $u''[j + \mathrm{per}(r)]$ are contained in r. The inequality $end(r) \geq end(u'')$ is proved similarly. □

All maximal PR-repeats can be easily computed according to the following lemma.

Lemma 6. *A maximal gapped periodic repeat $\pi \equiv (u', u'')$ is generated by a maximal repetition r if and only if $p(\pi)$ is divisible by $\mathrm{per}(r)$ and*

$$|r|/2 < p(\pi) \leq |r| - 2\,\mathrm{per}(r),$$
$$u' \equiv w[beg(r) \,..\, end(r) - p(\pi)],$$
$$u'' \equiv w[beg(r) + \mathrm{per}(r) \,..\, end(r)].$$

Proof. Let π be generated by r. Consider prefixes of u' and u'' of length $\mathrm{per}(r)$. These prefixes are equal cyclic roots of r, and by Proposition 3 the difference $beg(u'') - beg(u') = p(\pi)$ is divisible by $\mathrm{per}(r)$. Inequalities $|r|/2 < p(\pi) \leq |r| - 2\mathrm{per}(r)$ follow immediately from the definition of a repeat generated by a repetition. To prove the last two conditions of the lemma, it is sufficient to prove $beg(u') = beg(r)$ and $end(u'') = end(r)$. Let $beg(u') \neq beg(r)$, i.e. $beg(u') > beg(r)$. Then both letters $w[beg(u') - 1]$ and $w[beg(u'') - 1]$ are contained in r. Thus, since the difference $(beg(u'') - 1) - (beg(u') - 1) = p(\pi)$ is divisible by $\mathrm{per}(r)$, we have $w[beg(u') - 1] = w[beg(u'') - 1]$ which contradicts the maximality of π. The relation $end(u'') = end(r)$ is proved analogously. Thus, all the conditions of the lemma are proved. On the other hand, if π satisfies all the conditions of the lemma then π is obviously generated by r. □

Corollary 2. *A maximal repetition r generates no more than $\exp(r)/2$ maximal PR-repeats, and all these repeats can be computed from r in $O(\exp(r))$ time.*

To find all maximal α-gapped PR-repeats in a string w, we first compute all maximal repetitions in w in $O(n)$ time (see Introduction). Then, for each maximal repetition r, we output all maximal α-gapped repeats generated by r. Using Corollary 2, this can be done in $O(\exp(r))$ time. Thus the total time of processing all maximal repetitions is $O(\mathcal{E}(w))$. Since $\mathcal{E}(w) = O(n)$ by Theorem 1, all maximal α-gapped PR-repeats in w can be computed in $O(n)$ time.

4.2. Computing non-PR-repeats

We now turn to the most laborious part of the algorithm: computation of maximal α-gapped non-PR-repeats. Recall that non-PR-repeats are those which are either non-periodic, or periodic but not located within a single run that is the extension of their arms. For example, gapped repeat (ababa)aba(ababa) is periodic but is a non-PR-repeat because its periodic arms *ababa* with minimum period 2 are not contained in a single run with the same minimum period. Our goal is to show that all maximal α-gapped non-PR-repeats can be found in $O(\alpha n)$ time. Observe that there exists a straightforward algorithm for computing all maximal α-gapped repeats in $O(n^2)$ time that proceeds as follows: for each period $p < n$, find all maximal α-gapped repeats with period p in $O(n)$ time by consecutively comparing symbols $w[i]$ and $w[i+p]$ for $i = 1, 2, \ldots, n-p$.

From the results of [28], it follows that all maximal α-gapped repeats can be found in time $O(n \log n + S)$. This, together with Theorem 2, leads to an $O(\alpha n)$-time algorithm for the case $\alpha \geq \log n$. Therefore, we only have to consider the case $\alpha < \log n$.

(i) Preliminaries

Assume that $\alpha < \log n$. For this case, we proceed with a modification of the algorithm of [30]. We compute all maximal α-gapped non-PR-repeats π in w for which $c(\pi) \geq \log n$. To do so, we divide w into *blocks* of $\Delta = (\log n)/4$ consecutive symbols of w. Without loss of generality, we assume that $n = 2^k\Delta$, i.e. w contains exactly 2^k blocks. A word x of length $2^l\Delta$ where $0 \leq l \leq k-1$ is called a *basic factor* of w if $x = w[i\Delta+1\,..\,(i+2^l)\Delta]$ for some i. Such an occurrence $w[i\Delta+1\,..\,(i+2^l)\Delta]$ of x starting at a block frontier will be called *aligned*. A basic factor x of length $2^l\Delta$, where $1 \leq l \leq k-1$, is called *superbasic* if $x = w[q2^l\Delta+1\,..\,(q+1)2^l\Delta]$ for some q. Similar to basic factors, an occurrence of a superbasic factor x starting at the frontier between two consecutive tuples of 2^l blocks (i.e. at a position $(i2^l\Delta+1)$ for some l) is also called *aligned*. Aligned and unaligned occurrence of basic and superbasic factors are illustrated in Figure 1. Note that w contains $O(n)$ aligned occurrences of basic factors and $O(\frac{n}{\log n})$ aligned occurrences of superbasic factors.

Let $z \equiv w[q2^l\Delta+1\,..\,(q+1)2^l\Delta]$ be an aligned occurrence of a superbasic factor of length $2^l\Delta$ in w. For $\tau = 0, 1, \ldots \Delta-1$, an occurrence $w[q2^l\Delta+1+\tau\,..\,(q2^l+2^{l-1})\Delta+\tau]$ of a basic factor of length $2^{l-1}\Delta$ is called τ-*associated* (or simply *associated*) with z. Note that any basic factor occurrence τ-associated

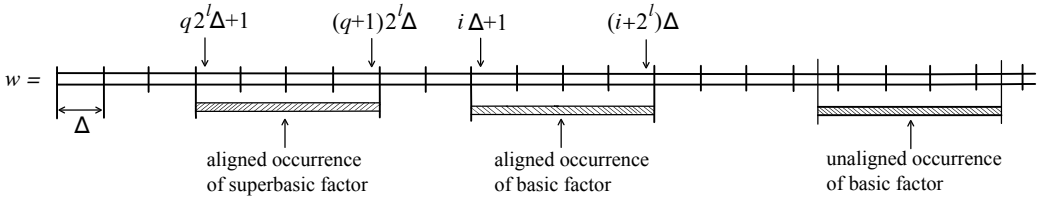

Figure 1: Basic and superbasic factors

with z is entirely contained in z and is uniquely defined by the word z and τ. Thus, z has no more than Δ associated occurrences of basic factors.

To continue, we need one more definition: for $1 \leq i, j \leq n$, denote by $\mathrm{LCP}(i, j)$ the length of the longest common prefix of $w[i\,..\,n]$ and $w[j\,..\,n]$, and by $\mathrm{LCS}(i, j)$ the length of the longest common suffix of $w[1\,..\,i]$ and $w[1\,..\,j]$.

Let $\pi \equiv (u', u'')$ be a maximal gapped repeat in w for which $c(\pi) \geq \log n = 4\Delta$. Note that in this case, the left arm u' contains at least one aligned occurrence of superbasic factor. Consider aligned occurrences of superbasic factors of maximal length contained in u'. Note that u' can contain either one or two adjacent such occurrences. Let z be the leftmost of them. Remark that in this case, we have the following restrictions imposed on u':

$$\begin{array}{c} beg(z) - |z| < beg(u') \leq beg(z), \\ end(z) \leq end(u') < end(z) + 2|z|. \end{array} \tag{3}$$

Thus, $c(\pi) < 4|z|$. Consider factor z'' in u'' corresponding to z in u'. Note that z'' can be non-aligned. Consider in z'' the leftmost aligned basic factor y'' of length $|z''|/2$. Observe that $beg(z'') \leq beg(y'') < beg(z'') + \Delta$ and y'' is entirely contained in z''. Let y' be the factor of z corresponding to factor y'' in z''. It is easily seen that y' is an occurrence of a basic factor associated with z, and π is uniquely defined by z, y' and y''. Thus, any maximal gapped repeat π for which $c(\pi) \geq \log n$ is uniquely defined by a triple (z, y', y''), where z is an aligned occurrence of some superbasic factor, y' is an occurrence of some basic factor associated with z, and y'' is an aligned occurrence of the same basic factor. From now on, we say in such case that π *is defined* by the triple (z, y', y'').

Observe that $\pi \equiv (u', u'')$ can be retrieved from (z, y', y'') using LCP and

LCS functions as follows:

$$\begin{array}{c} beg(u') = beg(y') - \mathrm{LCS}(beg(y') - 1, beg(y'') - 1), \\ end(u') = end(y') + \mathrm{LCP}(end(y') + 1, end(y'') + 1), \\ beg(u'') = beg(y'') - \mathrm{LCS}(beg(y') - 1, beg(y'') - 1), \\ end(u'') = end(y'') + \mathrm{LCP}(end(y') + 1, end(y'') + 1). \end{array} \tag{4}$$

Assume additionally that π is an α-gapped repeat for $\alpha > 1$. Then, on the one hand, taking into account inequalities (3) and $c(\pi) < 4|z|$, we have

$$\begin{array}{rcl} end(y'') & \leq & end(u'') = end(u') + \mathrm{per}(\pi) < end(z) + 2|z| + \alpha c(\pi) \\ & < & end(z) + 2|z| + 4\alpha|z| < end(z) + 6\alpha|z| = end(z) + 12\alpha|y''|. \end{array}$$

On the other hand, $beg(y'') \geq beg(u'') > end(u') \geq \mathrm{end}(z)$. Thus, for any triple (z, y', y'') defining a maximal α-gapped repeat in w, the occurrence y'' is contained in the segment $w[end(z)+1\,..\,end(z)+12\alpha|y''|]$ of length $12\alpha|y''|$ to the right of z. We denote this segment by $\mathcal{I}(z)$.

(ii) High-level description of the computation of non-PR-repeats

The computation of non-PR-repeats is described in the rest of this section. Before going into detail, we give a high-level description of this part.

The main idea of the algorithm is to consider all triples (z, y', y'') which can define maximal α-gapped non-PR-repeats and, for each such triple, to check if it actually defines such a repeat, which is then computed and output. All the triples (z, y', y'') are considered in a natural way: for each aligned occurrence z of a superbasic factor and each occurrence y' of a basic factor associated with z, we consider all aligned occurrences y'' of the same basic factor in segment $\mathcal{I}(z)$. The main difficulty in the algorithm is to enumerate all aligned occurrences y'' efficiently. In order to overcome this difficulty, we use precomputed linked lists *alignocc* contained all aligned occurrences of basic factors in the left-to-right order. In order to compute the lists *alignocc*, we first compute all basic and superbasic factors by naming occurrences of the same factor by a unique name, using the suffix tree of the whole word. Then, again with the suffix tree, we identify all occurrences of basic factors associated with aligned occurrences of superbasic factors. Finally, for each basic factor y we insert all aligned occurrences of y to the list $alignocc(y)$. We now proceed to a detailed description.

(iii) Naming basic factors on a suffix tree and computing their associated occurrences

We now describe how this computation is implemented. First we construct the suffix tree of the input string w. Suffix tree is a classical data structure of size $O(n)$ that can be built in $O(n)$ time for a constant-size alphabet (see e.g. [4, 9]). A suffix tree can be preprocessed in $O(n)$-time to support retrieving $\mathrm{LCP}(i,j)$ for any i,j in constant time (see e.g. [4]). Similarly, we precompute w to support $\mathrm{LCS}(i,j)$ for any i,j in constant time. Then we compute all basic factors of w including superbasic factors. This computation is performed by naming all the basic factors, i.e. assigning to each aligned occurrence of a basic factor a name of this factor. The most convenient way to name basic factors is to assign to a basic factor y of length 2^l a pair (l,i), where i is the start position of the leftmost aligned occurrence of y in w. Note that since we have only n/Δ distinct start positions i, the size of the two-dimensional array required for working with these pairs is $O(n)$. To perform the required computation, we first mark in the suffix tree each node labeled by a basic factor by the name of this factor (in the case when this node is implicit we make it explicit). To this end, for each node v of the suffix tree we compute the value $\mathit{minleaf}(v)$ which is the smallest leaf number divisible by Δ in the subtree rooted in v if such a number exists. This can be easily done in $O(n)$ time during a bottom-up traversal of the tree. Then, each suffix tree edge (u,v) for which the string depth of u is less than 2^l, the string depth of v is not less than 2^l, and $\mathit{minleaf}(v)$ is defined is treated in the following way: if the string depth of v is 2^l, node v is marked by name $(l,\mathit{minleaf}(v))$, otherwise a new node of string depth 2^l is created within edge (u,v) and marked by name $(l,\mathit{minleaf}(v))$. We call the obtained tree *marked suffix tree*. Since we have $O(n)$ distinct basic factors, the marked suffix tree contains no more than $O(n)$ additionally inserted nodes. Thus, this tree has $O(n)$ size and is built in $O(n)$ time.

To assign to each aligned occurrence $w[i\,..\,i+2^l-1]$ of a basic factor the name of this factor, we perform a depth-first top-down traversal of the marked suffix tree. During the traversal we maintain an auxiliary array $\mathit{basancestor}$: at the first visit of a node marked by a name (l,m) we set $\mathit{basancestor}[l]$ to m, and at the second visit of this node we reset $\mathit{basancestor}[l]$ to undefined. While during the traversal we get to a leaf i divisible by Δ, for each $l=0,1,\ldots,k-1$ we identify $w[i\,..\,i+2^l-1]$ as an occurrence of the basic factor named by $(l,\mathit{basancestor}[l])$. Note that this traversal is performed in $O(n)$

time.

Then, we compute all occurrences of basic factors associated with aligned occurrences of superbasic factors. This is done again during a depth-first top-down traversal of the marked suffix tree. During the traversal, we maintain the same auxiliary array *basancestor*. Assume that during the traversal we get to a leaf labelled by a position $q2^p\Delta + 1 + \tau$, where q is odd and $0 \leq \tau < \Delta$. Then for each $l = 0, 1, \ldots, p-1$ for which $basancestor[l]$ is defined, we identify $w[q2^p\Delta + 1 + \tau \,..\, (q2^p + 2^l)\Delta + \tau]$ as an occurrence of the basic factor named $(l, basancestor[l])$, which is τ-associated with the superbasic factor occurrence $w[q2^p\Delta + 1 \,..\, (q2^p + 2^{l+1})\Delta]$. Observe that this traversal is performed in $O(n)$ time as well.

(iv) Computing lists of aligned occurrences of basic factors

Let y be a Δ-periodic basic factor (cf Section 2). Note that y is also periodic, and then any occurrence of y in w is a repetition. By Proposition 1, the period $\mathrm{per}(y)$ is a divisor of $p_\Delta(y)$. Given the value $p_\Delta(y)$, we can compute in constant time the extension r of any occurrence y' of a Δ-periodic basic factor y as follows:

$$\begin{aligned} beg(r) &= beg(y') - \mathrm{LCS}(beg(y') - 1, beg(y') + p_\Delta(y) - 1), \\ end(r) &= end(y') + \mathrm{LCP}(beg(y') + 1, beg(y') - p_\Delta(y) + 1). \end{aligned}$$

Using Proposition 2, it is easy to show that any set of all aligned occurrences of y having the same extension is a sequence of occurrences, where the difference between start positions of any two consecutive occurrences is equal to $p_\Delta(y)$, i.e. the start positions of all these occurrences form a finite arithmetic progression with common difference $p_\Delta(y)$. We will call such a set a *series of occurrences*. The following fact can be easily proved.

Proposition 6. *Let y', y'' be two consecutive aligned occurrences of a basic factor y in w. Then $|beg(y') - beg(y'')| \leq |y|/2$ if and only if y is Δ-periodic, y' and y'' are contained in the same series of occurrences, and, moreover, $|beg(y') - beg(y'')| = p_\Delta(y)$.*

At the next step of the algorithm, in order to effectively select appropriate occurrences y'' in the checked triples (z, y', y''), for each basic factor y we construct a linked list $alignocc(y)$ of all aligned occurrences of y in the left-to-right order in w. If y is not Δ-periodic, each item of $alignocc(y)$ consists of only one aligned occurrence of y defined, for example, by its start position

(we will call such items *ordinary*). If y is Δ-periodic, each item of $alignocc(y)$ contains a series of aligned occurrences of y. If a series of aligned occurrences of y consists of only one occurrence, we will consider the item of $alignocc(y)$ for this series as ordinary, otherwise, if a series of aligned occurrences of y consists of at least two occurrences, the item of $alignocc(y)$ for this series will be defined, for example, by start positions of leftmost and rightmost occurrences in the series and the value $p_\Delta(y)$ (such item will be called an *s-item*). The following fact follows from Proposition 6.

Proposition 7. *Let y', y'' be two consecutive aligned occurrences of a basic factor y in w. Then $|beg(y') - beg(y'')| \leq |y|/2$ if and only if y' and y'' are contained in the same s-item of $alignocc(y)$ and, moreover, $|beg(y') - beg(y'')| = p_\Delta(y)$.*

Proposition 7 implies that if two aligned occurrences y', y'' of a basic factor y are contained in distinct items of $alignocc(y)$ then $|beg(y') - beg(y'')| > |y|/2$. Therefore, we have the following consequence of the proposition.

Corollary 3. *Let y be a basic factor of w. Then for any segment v in w, the list $alignocc(y)$ contains $O(|v|/|y|)$ items having at least one occurrence of y contained in v.*

To construct the lists *alignocc*, we insert for each $i = 1, 2, \ldots, n$ and each $l = 0, 1, \ldots, k-1$ consecutively the occurrence $y' \equiv w[i \,..\, i + 2^l - 1]$ of some basic factor y to the appropriate list $alignocc(y)$ as follows. Consider the last item in the current list $alignocc(y)$. Let it be an ordinary item consisting of an occurrence y'' of y starting at position j. Denote $\delta = i - j$. Consider the following two cases for δ. Case 1: $\delta > |y|/2$. Then, by Proposition 7, y'' and y' are contained in distinct items of $alignocc(y)$, and in this case we insert y' to $alignocc(y)$ as a new ordinary item. Case 2: $\delta \leq |y|/2$. In this case, by Proposition 7, y'' and y' are the first two occurrences of the same series of occurrences of y and, moreover, $\delta = p_\Delta(y)$. Let r be the extension of the occurrences of this series. It is easy to see that

$$end(r) = end(y') + \mathrm{LCP}(end(y'') + 1, end(y') + 1),$$

i.e. $end(r)$ can be computed in constant time. From the values $\mathrm{beg}(y'')$, $end(r)$ and $p_\Delta(y)$, we can compute in constant time the start position of the last occurrence of y in the considered series of occurrences and thereby

identify completely this series. Thus, in this case we replace the last item of $\mathit{alignocc}(y)$ by the identified series of occurrences of y. Now let the last item in $\mathit{alignocc}(y)$ be a series of occurrences. Then, if y' is not contained in this series, we insert y' to $\mathit{alignocc}(y)$ as a new ordinary item. Thus, each occurrence of a basic factor in w is processed in constant time, and the total time for construction of lists *alignocc* is $O(n)$.

Furthermore, in order to optimize the selection of appropriate occurrences y'' in the checked triples (z, y', y''), for each pair (z, y') where z is an aligned occurrence of a superbasic factor and y' is an occurrence of some basic factor y associated with z, we compute a pointer $\mathit{firstocc}(z, y')$ to the first item in $\mathit{alignocc}(y)$ containing at least one occurrence of y to the right of z. For these purposes, we use auxiliary lists $\mathit{factends}(i)$ defined for each position i in w. Lists $\mathit{factends}(i)$ consist of pairs (z, y') and are constructed at the stage computation of occurrences associated with aligned occurrences of superbasic factors: each time we find a new occurrence y' associated with an aligned occurrence z of a superbasic factor, we insert the pair (z, y') into the list $\mathit{factends}(\mathrm{end}(z)+1)$. After construction of lists *alignocc*, we compute consecutively for each $i = 1, 2, \ldots, n$ pointers $\mathit{firstocc}(z, y')$ for all pairs (z, y') from the list $\mathit{factends}(i)$. During the computation, we save in each list $\mathit{alignocc}(y)$ the last item pointed to before (this item is denoted by $\mathit{lastpnt}(y)$). To compute $\mathit{firstocc}(z, y')$, we go through the list $\mathit{alignocc}(y)$ from $\mathit{lastpnt}(y)$ (or from the beginning of $\mathit{alignocc}(y)$ if $\mathit{lastpnt}(y)$ does not exist) until we find the first item containing at least one occurrence of y to the right of the position i. The found item is pointed to by $\mathit{firstocc}(z, y')$ and becomes a new item $\mathit{lastpnt}(y)$. Since the total size of lists *alignocc* and *factends* is $O(n)$, the total time for computing $\mathit{firstocc}(z, y')$ is also $O(n)$.

(v) Main step: computing large repeats

At the main stage of the algorithm, in order to process each pair (z, y'), note that all occurrences y'' contained in $\mathcal{I}(z)$ such that the triple (z, y', y'') defines a maximal α-gapped repeat are located in the fragment of $\mathit{alignocc}(y)$ consisting of all items having at least one occurrence of y contained in $\mathcal{I}(z)$. We call this fragment a *checked fragment*. Thus, we consider all items of the checked fragment by going through this fragment from the first item which can be found in constant time by the value $\mathit{firstocc}(z, y')$. For each considered item, we check triples (z, y', y'') for all occurrences y'' from this item as follows.

Let the considered item be an ordinary item consisting of only one occur-

rence y''. Recall that gapped repeat (u', u'') defined by the triple (z, y', y'') can be computed in constant time by formulas (4). Thus, if (u', u'') is an α-gapped repeat satisfying conditions (3), we output it.

Now let the item considered in the checked fragment be a s-item. This implies that the basic factor y is Δ-periodic, i.e y is Δ-periodic. Moreover, from the s-item we can derive the value $p_\Delta(y)$. Therefore we can compute in constant time extensions r' and r'' of occurrences y' and y'' respectively. Denote by ρ the series of occurrences contained in the s-item. Recall that our goal is to compute effectively all α-gapped repeats defined by triples (z, y', y'') for which $y'' \in \rho$. Observe that if r' and r'' are the same repetition, then by Proposition 5 all such repeats are PR-repeats, therefore we can assume that r' and r'' are distinct repetitions. Note that $\mathrm{per}(r') = \mathrm{per}(r'') = \mathrm{per}(y)$. Let (u', u'') be an α-gapped repeat defined by a triple (z, y', y'') where $y'' \in \rho$. First, consider the case when u' is not contained in r', i.e. either $beg(u') < beg(r')$ or $end(u') > end(r')$.

Proposition 8. *If $beg(u') < beg(r')$, then $beg(r') - beg(u') = beg(r'') - beg(u'')$.*

Proof. Define $\gamma' = beg(r') - beg(u')$, $\gamma'' = beg(r'') - beg(u'')$. Assume by contradiction $\gamma' > \gamma''$. Since repetition r' is maximal we have

$$u'[\gamma'] \equiv w[beg(r') - 1] \neq w[beg(r') - 1 + \mathrm{per}(r')] \equiv u'[\gamma' + \mathrm{per}(r')].$$

Moreover, from $\gamma' > \gamma''$ we have that both letters $u''[\gamma']$ and $u''[\gamma' + \mathrm{per}(r')]$ are contained in the repetition r'', and $u''[\gamma'] = u''[\gamma' + \mathrm{per}(r')]$ since $\mathrm{per}(r') = \mathrm{per}(r'')$. Thus, from $u'[\gamma'] = u''[\gamma']$ we obtain a contradiction $u'[\gamma' + \mathrm{per}(y)] \neq u''[\gamma' + \mathrm{per}(y)]$. Similarly, we obtain a contradiction $u'[\gamma'' + \mathrm{per}(y)] \neq u''[\gamma'' + \mathrm{per}(y)]$ if it is assumed $\gamma' < \gamma''$. □

The following proposition can be proved similarly.

Proposition 9. *If $end(u') > end(r')$, then $end(u') - end(r') = end(u'') - end(r'')$.*

Define

$$\begin{aligned} s_{left} &= beg(y') + (beg(r'') - beg(r')), \\ s_{right} &= beg(y') + (end(r'') - end(r')). \end{aligned}$$

From Propositions 8 and 9, we derive the following fact.

Corollary 4. *If* $beg(u') < beg(r')$ *then* $beg(y'') = s_{left}$. *If* $end(u') > end(r')$ *then* $beg(y'') = s_{right}$.

Thus, for computing α-gapped repeats (u', u'') for which u' is not contained in r', it is enough to consider in ρ only occurrences y''_{left} and y''_{right} with start positions s_{left} and s_{right} respectively, provided that these occurrences exist. We check the occurrences y''_{left} and y''_{right} in the same way as we did for occurrence y'' in the case of ordinary item. Then, it remains to check all occurrences from ρ except for possible occurrences y''_{left} and y''_{right}. Denote by $\rho' = \rho \setminus \{y''_{left}, y''_{right}\}$ the set of all such occurrences. Assume that $|r'| \leq |r''|$, i.e. $s_{left} \leq s_{right}$ (the case $|r'| > |r''|$ is similar). In order to check all occurrences from ρ', we consider the following subsets of ρ' separately: subset ρ'_1 of all occurrences y'' such that $beg(y'') < s_{left}$, subset ρ'_2 of all occurrences y'' for which $s_{left} < beg(y'') < s_{right}$, and subset ρ'_3 of all occurrences y'' for which $s_{right} < beg(y'')$. Note that start positions of all occurrences in each of these subsets form a finite arithmetic progression with common difference $p_\Delta(y)$. Thus, we unambiguously denote all occurrences in each of the subsets ρ'_i, $i = 1, 2, 3$, by $y''_0, y''_1, \ldots, y''_k$ where y''_0 is the leftmost occurrence in the subset ρ'_i and $beg(y''_j) = beg(y''_0) + jp_\Delta(y)$ for $j = 1, \ldots, k$. Note that values $beg(y''_0)$ and k for each subset ρ'_i can be computed in constant time.

First, consider an occurrence y''_j from ρ'_1. Let $\pi \equiv (u', u'')$ be the repeat defined by triple (z, y', y''_j). Note that

$$\mathrm{per}(\pi) = beg(y''_j) - beg(y') = q + jp_\Delta(y), \tag{5}$$

where $q = beg(y''_0) - beg(y')$. Taking into account that y' and y''_j are contained in maximal repetitions r' and r'' respectively, it is easy to verify that

$$\mathrm{LCS}(beg(y') - 1, beg(y''_j) - 1) = beg(y''_j) - beg(r''),$$
$$\mathrm{LCP}(end(y') + 1, end(y''_j) + 1) = end(r') - end(y').$$

Therefore, $beg(u') = beg(r'') - \mathrm{per}(\pi) = q' - jp_\Delta(y)$, where $q' = beg(r'') - q$, and $end(u') = end(r')$. It follows that

$$c(\pi) = |u'| = end(u') - beg(u') + 1 = q'' + jp_\Delta(y),$$

where $q'' = end(r') + 1 - q'$. Recall that for any α-gapped repeat π, we have $c(\pi) < \mathrm{per}(\pi) \leq \alpha c(\pi)$. Thus, π is an α-gapped repeat if and only if

$$q'' < q \leq \alpha q'' + (\alpha - 1)jp_\Delta(y). \tag{6}$$

Moreover, u' has to satisfy conditions (3). Thus, the triple (z, y', y''_j) defines an α-gapped repeat if and only if conditions (6) and (3) are verified for j. Note that all these conditions are linear inequalities on j, and then can be resolved in constant time. Thus, we output all α-gapped repeats defined by triples (z, y', y'') for which $y'' \in \rho'_1$ in time $O(1 + S)$, where S is the size of the output.

Consider now an occurrence y''_j from ρ'_2. Let $\pi \equiv (u', u'')$ be the repeat defined by the triple (z, y', y''_j). Note that in this case, $\mathrm{per}(\pi)$ also satisfies relation (5). Analogously to the previous case of set ρ'_1, we obtain that $beg(u') = beg(r')$ and $end(u') = end(r')$, and then $c(\pi) = |r'|$. Therefore, π is an α-gapped repeat if and only if

$$|r'| < q + jp_\Delta(y) \leq \alpha|r'|. \tag{7}$$

Thus, in this case, we output all α-gapped repeats defined by triples (z, y', y''_j) for which j satisfies conditions (7) and (3). Since all these conditions can be resolved for j in constant time, all these repeats can be output in time $O(1 + S)$ where S is the output size.

Finally, consider an occurrence y''_j from ρ'_3. Let $\pi \equiv (u', u'')$ be the repeat defined by triple (z, y', y''_j). In this case, $\mathrm{per}(\pi)$ also satisfies relation (5). Analogously to the case of set ρ'_1, we obtain that $beg(u') = beg(r')$ and $end(u') = end(r') - \mathrm{per}(\pi) = \widehat{q}' - jp_\Delta(y)$, where $\widehat{q}' = end(r'') - q$, and then

$$c(\pi) = end(u') - beg(u') + 1 = \widehat{q}'' - jp_\Delta(y),$$

where $\widehat{q}'' = \widehat{q}' - beg(r') + 1$. Therefore, π is an α-gapped repeat if and only if

$$\widehat{q}'' - jp_\Delta(y) < q + jp_\Delta(y) \leq \alpha(\widehat{q}'' - jp_\Delta(y)). \tag{8}$$

Thus, in this case, we output all α-gapped repeats defined by triples (z, y', y''_j) for which j satisfies conditions (8) and (3). Like in the previous cases, this can be done in time $O(1 + S)$, where S is the output size.

Putting together all the three considered cases, we conclude that all α-gapped repeats defined by triples (z, y', y'') for which $y'' \in \rho$ can be computed in time $O(1 + S)$ where S is the output size. Thus, in $O(1 + S)$ time we can process each item of the checked fragment. Therefore, since by Corollary 3 the checked fragment has $O(\alpha)$ items, the total time for processing pair (z, y') is $O(\alpha + S)$ where S is the total number of α-gapped repeats defined by triples (z, y', y''). Since each occurrence z has no more than Δ associated

occurrences y', the total number of processed pairs (z, y') is $O(n)$. Thus the time complexity of the main stage of the algorithm is $O(\alpha n + S)$, where S is the size of the output. Taking into account that $S = O(\alpha n)$ by Theorem 2, we conclude that the time complexity of the main stage is $O(\alpha n)$. Thus, all maximal α-gapped non-PR-repeats π in w for which $c(\pi) \geq \log n$ can be computed in $O(\alpha n)$ time. Algorithm 1 summarizes the computation of all large maximal α-gapped non-PR-repeats.

Algorithm 1 Computing of all large maximal α-gapped non-PR-repeats

```
1: construct the suffix trees of the string w and the reverse string w^R
2: preprocess the suffix trees of w and w^R for computing LCP(i, j) and
   LCS(i, j) in constant time
3: compute and name basic and superbasic factors
4: compute all occurrences of basic factors associated with superbasic fac-
   tors
5: compute lists alignocc(y) of aligned occurrences of basic factors y
6: for l = 1, ..., log(n/Δ) − 1 do
7:    for each aligned occurrence z ≡ w[q2^l Δ + 1..(q + 1)2^l Δ] of superbasic
      factor do
8:       for each occurrence y' of a basic factor y associated with z do
9:          compute all aligned occurrences y'' of the basic factor y which are
            contained in I(z) by using list alignocc(y)
10:         for each computed occurrence y' do
11:            check if triple (z, y', y'') defines a maximal α-gapped non-PR-
               repeat and, if so, compute and output this repeat
12:         end for
13:      end for
14:   end for
15: end for
```

(vi) Computing small repeats

To compute all remaining maximal α-gapped non-PR-repeats in w, i.e. maximal α-gapped non-PR-repeats π such that $c(\pi) < \log n$, note that the length of any such repeat π is not greater than

$$(1 + \alpha)c(\pi) < (1 + \log n) \log n < 2 \log^2 n.$$

Thus, setting $\Delta' = \lfloor 2\log^2 n \rfloor$, any such repeat is contained in at least one of the segments $\mathcal{I}'_i \equiv w[i\Delta' + 1\,..\,(i+2)\Delta']$ for $0 \le i < n/\Delta'$. Therefore, all the remaining α-gapped repeats can be found by searching segments $\mathcal{I}'_i$ separately. The procedure of searching for repeats in $\mathcal{I}'_i$ is similar to the algorithm described above. If $\alpha \ge \log\log n$, searching for repeats in $\mathcal{I}'_i$ can be done by the algorithm proposed in [28]. The $O(|\mathcal{I}'_i|\log|\mathcal{I}'_i| + S)$ time complexity implied by this algorithm, where by Theorem 2 the output size S is $O(\alpha|\mathcal{I}'_i|)$, can be bounded here by $O(\alpha\Delta')$. Thus, the total time complexity for searching all segments $\mathcal{I}'_i$ is $O(\alpha n)$. In the case of $\alpha < \log\log n$, we search each segment $\mathcal{I}'_i$ for all remaining maximal α-gapped non-PR-repeats π in w for which $c(\pi) \ge \log|\mathcal{I}'_i|$ in time $O(\alpha\Delta')$, in the same way as we described above for the word w. The total time for searching all segments $\mathcal{I}'_i$ is $O(\alpha n)$. Then, it remains to compute all maximal α-gapped non-PR-repeats π in w for which $c(\pi) < \log|\mathcal{I}'_i| \le 3\log\log n$. Observe that the length of any such repeat is not greater than

$$(1+\alpha)3\log\log n < (1+\log\log n)3\log\log n \le 6\log^2\log n.$$

Thus, setting $\Delta'' = \lfloor 6\log^2\log n \rfloor$, any such repeat is contained in at least one of the segments $\mathcal{I}''_i \equiv w[i\Delta''+1\,..\,(i+2)\Delta'']$ for $0 \le i < n/\Delta''$. Note that these segments are words of length $2\Delta''$ over an alphabet of size σ, therefore the total number of distinct segments $\mathcal{I}''_i$ is not greater than $\sigma^{2\Delta''} \le \sigma^{12\log^2\log n}$. In each of the distinct segments $\mathcal{I}''_i$, all maximal α-gapped repeats can be found by the trivial algorithm described above in $O(\Delta''^2) = O(\log^4\log n)$ time. Thus, maximal α-gapped repeats in all distinct segments $\mathcal{I}''_i$ can be found in $O(\sigma^{12\log^2\log n}\log^4\log n) = o(n)$ time. We conclude that all remaining maximal α-gapped repeats in w can be found in $O(n+S)$ time where S is the total number of maximal α-gapped repeats contained in all segments $\mathcal{I}''_i$. According to Theorem 2, this number can be bounded by $O(\alpha n)$, and the time for finding all the remaining maximal α-gapped repeats can be bounded by $O(\alpha n)$ as well. This leads to the final result.

Theorem 4. *For a fixed $\alpha > 1$, all maximal α-gapped repeats in a word of length n over a constant alphabet can be found in $O(\alpha n)$ time.*

Note finally that since, as mentioned earlier, a word can contain $\Theta(\alpha n)$ maximal α-gapped repeats, the $O(\alpha n)$ time bound stated in Theorem 4 is asymptotically optimal.

5. Conclusion

Besides gapped repeats we can also consider gapped palindromes which are factors of the form uvu^R where u and v are nonempty words and u^R is the reversal of u [37]. A gapped palindrome uvu^R in a word w is called *maximal* if $w[end(u)+1] \neq w[beg(u^R)-1]$ and $w[beg(u)-1] \neq w[end(u^R)+1]$ for $beg(u) > 1$ and $end(u^R) < |w|$. A maximal gapped palindrome uvu^R is α-gapped if $|u|+|v| \leq \alpha|u|$ [30]. It can be shown in a way similar to the results of this paper that for $\alpha > 1$ the number of maximal α-gapped palindromes in a word of length n is bounded by $O(\alpha n)$ and for the case of constant alphabet, all these palindromes can be found in $O(\alpha n)$ time[1].

In this paper we consider maximal α-gapped repeats with $\alpha > 1$. However this notion can be formally generalized to the case $\alpha \leq 1$. In particular, maximal 1-gapped repeats are maximal repeats whose arms are adjacent or overlapping. It is easy to see that such repeats form runs whose minimum periods are divisors of the periods of these repeats. Moreover, each run in a word is formed by at least one maximal 1-gapped repeat, therefore the number of runs in a word is not greater than the number of maximal 1-gapped repeats. More precisely, each run r is formed by $\lfloor \exp(r)/2 \rfloor$ distinct maximal 1-gapped repeats. Thus, if a word contains runs with exponent not less than 4 then the number of maximal 1-gapped repeats is strictly greater than the number of runs. However, using an easy modification of the proof of "runs conjecture" from [19], it can be also proved that the number of maximal 1-gapped repeats in a word is strictly less than the length of the word.

Denoting by $\mathcal{R}(n)$ (respectively, $\mathcal{R}_1(n)$) the maximal possible number of runs (respectively, maximal possible number of maximal 1-gapped repeats) in words of length n, based on the fact that known words with a relatively large number of runs have no runs with big exponents, it seems that $\mathcal{R}(n) = \mathcal{R}_1(n)$.

Conjecture 1. *The maximal number of runs in words of length n is equal to the maximal number of 1-gapped repeats.*

We can also consider the case of $\alpha < 1$ for repeats with overlapping arms, in particular, the case of maximal $1/k$-gapped repeats where k is integer greater than 1. It is easy to see that such repeats form runs with exponents

[1]Note that in [30], the number of maximal α-gapped palindromes was conjectured to be $O(\alpha^2 n)$.

not less than $k+1$. It is known from [19, Theorem 11] that the number of such runs in a word of length n is less than n/k, and it seems to be possible to modify the proof of this fact for proving that the number of maximal $1/k$-gapped repeats in the word is also less than $n/k = \alpha n$. These observations together with results of computer experiments for the case of $\alpha > 1$ leads to the following conjecture.

Conjecture 2. *For any $\alpha > 0$, the maximal number of α-gapped repeats in a word of length n is less than αn.*

This generalization of the "runs conjecture" constitutes an interesting open problem. Another interesting open question is whether the obtained $O(n/\delta)$ bound on the number of maximal δ-subrepetitions is asymptotically tight for the case of constant-size alphabet.

Another interesting question is a generalization of the upper bound to the case of gapped repeats where the interval of possible gap lengths is specified by arbitrary functions on the arm length. Such a generalization was recently obtained in [38].

Acknowledgements. A part of this work was done during a visit of RK to the LIGM Lab of *Université Paris-Est*, supported by a Metchnikov grant of the French Embassy in Russia. RK was partially supported by Russian Foundation for Fundamental Research (Grant 15-07-03102). GK was partially supported by *Labex Bézout* grant of the French government.

6. References

[1] M. Lothaire, Combinatorics on Words, Addison Wesley, 1983.

[2] Z. Galil, J. I. Seiferas, Time-space-optimal string matching, Journal of Computer and System Sciences 26 (3) (1983) 280–294.

[3] M. Crochemore, W. Rytter, Squares, cubes, and time-space efficient string searching, Algorithmica 13 (5) (1995) 405–425.

[4] D. Gusfield, Algorithms on Strings, Trees, and Sequences - Computer Science and Computational Biology, Cambridge University Press, 1997.

[5] J. A. Storer, Data Compression: Methods and Theory, Computer Science Press, 1988.

[6] M. Crochemore, An optimal algorithm for computing the repetitions in a word, Information Processing Letters 12 (5) (1981) 244–250.

[7] A. Apostolico, F. Preparata, Optimal off-line detection of repetitions in a string, Theoretical Computer Science 22 (3) (1983) 297 – 315.

[8] M. Main, R. Lorentz, An $O(n \log n)$ algorithm for finding all repetitions in a string, J. of Algorithms 5 (3) (1984) 422–432.

[9] M. Crochemore, C. Hancart, T. Lecroq, Algorithms on Strings, Cambridge University Press, 2007.

[10] M. Crochemore, C. S. Iliopoulos, M. Kubica, J. Radoszewski, W. Rytter, T. Walen, Extracting powers and periods in a string from its runs structure, in: E. Chávez, S. Lonardi (Eds.), String Processing and Information Retrieval - 17th International Symposium, SPIRE 2010, Los Cabos, Mexico, October 11-13, 2010. Proceedings, Vol. 6393 of Lecture Notes in Computer Science, Springer, 2010, pp. 258–269.

[11] R. Kolpakov, G. Kucherov, On maximal repetitions in words, J. Discrete Algorithms 1 (1) (2000) 159–186.

[12] W. Rytter, The number of runs in a string: Improved analysis of the linear upper bound, in: B. Durand, W. Thomas (Eds.), Proc. 23rd Annual Symposium on Theoretical Aspect of Computer Science, STACS, Vol. 3884 of LNCS, Springer, 2006, pp. 184–195.

[13] W. Rytter, The number of runs in a string, Information and Computation 205 (9) (2007) 1459–1469. doi:10.1016/j.ic.2007.01.007.
URL `http://dx.doi.org/10.1016/j.ic.2007.01.007`

[14] M. Crochemore, L. Ilie, Maximal repetitions in strings, Journal of Computer and System Sciences 74 (2008) 796–807.

[15] S. J. Puglisi, J. Simpson, W. F. Smyth, How many runs can a string contain?, Theor. Comput. Sci. 401 (1-3) (2008) 165–171.

[16] M. Crochemore, L. Ilie, L. Tinta, Towards a solution to the "runs" conjecture, in: P. Ferragina, G. M. Landau (Eds.), Combinatorial Pattern Matching, 19th Annual Symposium, CPM 2008, Pisa, Italy, June 18-20, 2008, Proceedings, Vol. 5029 of Lecture Notes in Computer Science, Springer, 2008, pp. 290–302.

[17] A. Deza, F. Franek, A d-step approach to the maximum number of distinct squares and runs in strings, Discrete Applied Mathematics 163 (3) (2014) 268–274.

[18] M. Crochemore, M. Kubica, J. Radoszewski, W. Rytter, T. Walen, On the maximal sum of exponents of runs in a string, J. Discrete Algorithms 14 (2012) 29–36.

[19] H. Bannai, T. I, S. Inenaga, Y. Nakashima, M. Takeda, K. Tsuruta, A new characterization of maximal repetitions by Lyndon trees, CoRR abs/1406.0263 (2014) 10pp.

[20] H. Bannai, T. I, S. Inenaga, Y. Nakashima, M. Takeda, K. Tsuruta, A new characterization of maximal repetitions by Lyndon trees, in: P. Indyk (Ed.), Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015, SIAM, 2015, pp. 562–571.

[21] J. Fischer, S. Holub, T. I, M. Lewenstein, Beyond the runs theorem, in: C. S. Iliopoulos, S. J. Puglisi, E. Yilmaz (Eds.), String Processing and Information Retrieval - 22nd International Symposium, SPIRE 2015, London, UK, September 1-4, 2015, Proceedings, Vol. 9309 of Lecture Notes in Computer Science, Springer, 2015, pp. 277–286.

[22] M. G. Main, R. J. Lorentz, Linear time recognition of squarefree strings, in: A. Apostolico, Z. Galil (Eds.), Combinatorial Algorithms on Words, Vol. 12 of NATO Advanced Science Institutes, Series F, Springer Verlag, 1985, pp. 271–278.

[23] D. Breslauer, Efficient string algorithmics, Ph.D. thesis, Columbia University (1992).

[24] M. Crochemore, C. S. Iliopoulos, T. Kociumaka, R. Kundu, S. P. Pissis, J. Radoszewski, W. Rytter, T. Walen, Near-optimal computation of runs over general alphabet via non-crossing LCE queries, in: S. Inenaga, K. Sadakane, T. Sakai (Eds.), String Processing and Information Retrieval - 23rd International Symposium, SPIRE 2016, Beppu, Japan, October 18-20, 2016, Proceedings, Vol. 9954 of Lecture Notes in Computer Science, 2016, pp. 22–34.

[25] D. Kosolobov, Computing runs on a general alphabet, Information Processing Letters 116 (3) (2016) 241–244.

[26] P. Gawrychowski, T. Kociumaka, W. Rytter, T. Walen, Faster longest common extension queries in strings over general alphabets, in: R. Grossi, M. Lewenstein (Eds.), 27th Annual Symposium on Combinatorial Pattern Matching, CPM 2016, June 27-29, 2016, Tel Aviv, Israel, Vol. 54 of LIPIcs, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016, pp. 5:1–5:13.

[27] R. Kolpakov, M. Podolskiy, M. Posypkin, N. Khrapov, Searching of gapped repeats and subrepetitions in a word, in: A. S. Kulikov, S. O. Kuznetsov, P. A. Pevzner (Eds.), Combinatorial Pattern Matching - 25th Annual Symposium, CPM 2014, Moscow, Russia, June 16-18, 2014. Proceedings, Vol. 8486 of Lecture Notes in Computer Science, Springer, 2014, pp. 212–221, extended version in arxiv:1309.4055, 2013.

[28] G. S. Brodal, R. B. Lyngsø, C. N. S. Pedersen, J. Stoye, Finding maximal pairs with bounded gap, Journal of Discrete Algorithms 1 (1) (2000) 77–104.

[29] R. M. Kolpakov, G. Kucherov, Finding repeats with fixed gap, in: P. de la Fuente (Ed.), Seventh International Symposium on String Processing and Information Retrieval, SPIRE 2000, A Coruña, Spain, September 27-29, 2000, IEEE Computer Society, 2000, pp. 162–168.

[30] P. Gawrychowski, F. Manea, Longest α-gapped repeat and palindrome, in: A. Kosowski, I. Walukiewicz (Eds.), Fundamentals of Computation Theory - 20th International Symposium, FCT 2015, Gdańsk, Poland, August 17-19, 2015, Proceedings, Vol. 9210 of Lecture Notes in Computer Science, Springer, 2015, pp. 27–40.

[31] M. Dumitran, F. Manea, Longest gapped repeats and palindromes, in: G. F. Italiano, G. Pighizzini, D. Sannella (Eds.), Mathematical Foundations of Computer Science 2015 - 40th International Symposium, MFCS 2015, Milan, Italy, August 24-28, 2015, Proceedings, Part I, Vol. 9234 of Lecture Notes in Computer Science, Springer, 2015, pp. 205–217.

[32] P. Gawrychowski, T. I, S. Inenaga, D. Köppl, F. Manea, Efficiently finding all maximal α-gapped repeats, in: N. Ollinger, H. Vollmer (Eds.),

33rd Symposium on Theoretical Aspects of Computer Science, STACS 2016, February 17-20, 2016, Orléans, France, Vol. 47 of LIPIcs, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016, pp. 39:1–39:14.

[33] Y. Tanimura, Y. Fujishige, T. I, S. Inenaga, H. Bannai, M. Takeda, A faster algorithm for computing maximal α-gapped repeats in a string, in: C. S. Iliopoulos, S. J. Puglisi, E. Yilmaz (Eds.), String Processing and Information Retrieval - 22nd International Symposium, SPIRE 2015, London, UK, September 1-4, 2015, Proceedings, Vol. 9309 of Lecture Notes in Computer Science, Springer, 2015, pp. 124–136.

[34] G. Badkobeh, M. Crochemore, C. Toopsuwan, Computing the maximal-exponent repeats of an overlap-free string in linear time, in: L. Calderón-Benavides, C. N. González-Caro, E. Chávez, N. Ziviani (Eds.), String Processing and Information Retrieval - 19th International Symposium, SPIRE 2012, Cartagena de Indias, Colombia, October 21-25, 2012. Proceedings, Vol. 7608 of Lecture Notes in Computer Science, Springer, 2012, pp. 61–72.

[35] R. Kolpakov, G. Kucherov, P. Ochem, On maximal repetitions of arbitrary exponent, Information Processing Letters 110 (7) (2010) 252–256.

[36] R. Kolpakov, On primary and secondary repetitions in words, Theoretical Computer Science 418 (2012) 71–81.

[37] R. Kolpakov, G. Kucherov, Searching for gapped palindromes, Theoretical Computer Science 410 (51) (2009) 5365–5373.

[38] R. Kolpakov, On the number of gapped repeats with arbitrary gap, CoRR abs/1701.01190.
URL `http://arxiv.org/abs/1701.01190`