On the Cyclic Regularities of Strings

Oluwole Ajala[1(✉)], Miznah Alshammary[1(✉)], Mai Alzamel[1(✉)], Jia Gao[1(✉)], Costas Iliopoulos[1(✉)], Jakub Radoszewski[2(✉)], Wojciech Rytter[2(✉)], and Bruce Watson[3(✉)]

[1] Faculty of Natural and Mathematical Sciences, King's College London, London, UK
{oluwole.ajala,miznah.alshammary,mai.alzamel,jia.gao, c.ilioupoulos}@kcl.ac.uk

[2] Faculty of Mathematics, Informatics and Mechanics, University of Warsaw, Warsaw, Poland
{jrad,rytter}@mimuw.edu.pl

[3] Faculty of Informatics Science, Stellenbosch University, Stellenbosch, South Africa
bwwatson@sun.ac.za

Abstract. Regularities in strings are often related to periods and covers, which have extensively been studied, and algorithms for their efficient computation have broad application. In this paper we concentrate on computing cyclic regularities of strings, in particular, we propose several efficient algorithms for computing: (i) cyclic periodicity; (ii) all cyclic periodicity; (iii) maximal local cyclic periodicity; (iv) cyclic covers.

Keywords: Cyclic regularities · Periods · Covers

1 Introduction and Related Work

A fundamental concept of repeating patterns or *regularities* is that of periods (also known as powers). A period of order k is defined by a concatenation of k identical blocks of symbols. The study of periods can be traced to as far back as the early 1900s with the work of [9], who researched a set of strings that do not contain any substrings that are periods. Periods in diverse forms gained prominence, when they became key structures in computational biology, where they are associated with various regulatory mechanisms and play an important role in genomic fingerprinting [6].

In regularities in strings, one of the most general notions is related to period or power, for instance, given a string x of length n, a period k of a string x is a sub string of x, if it can be decomposed into equal-length blocks of symbols, such that $x = u^k u'$, where u' is a prefix of u. However, for simplicity we will discard u' and only consider u^k.

So far, regularities in strings related with periods and powers, which have been extensively studied, [3–5,8] and algorithms for their efficient computation

Published by Springer Nature Switzerland AG 2019
J. MacIntyre et al. (Eds.): AIAI 2019 Workshops, IFIP AICT 560, pp. 219–224, 2019.
https://doi.org/10.1007/978-3-030-19909-8_19

have broad applications. In this paper, we study cyclic factors of strings. The motivation of cyclic factors comes from viruses. The viruses are circular strings, for example Escherichia coli (E.coli) has 154 bases and it is circular [12] (Fig. 1). Formally, the viruses break up at any point of the circle, for example, that can appear in the DNA sequence as $x_\delta \dots x_n x_1 \dots x_{\delta-1}$ breaking up at position δ (Fig. 2). Now, we propose several efficient algorithms for computing: (i) cyclic periodicity; (ii) all cyclic periodicity; (iii) maximal local cyclic periodicity; (iv) cyclic covers.

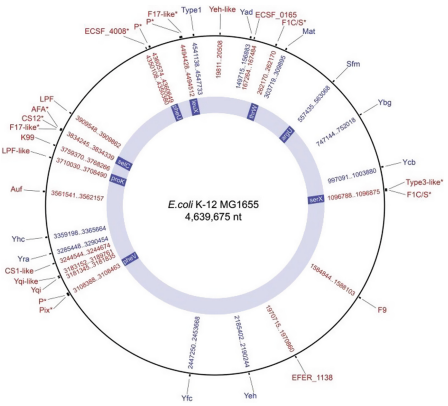

Fig. 1. The E. coli K-12 MG1655 chromosome (outer black ring) was used as a reference map to visualise the locus position of 30 chromosome-borne CU fimbrial types. Types highlighted in blue are present in E. coli K-12 MG1655, types in red are absent in this strain. Fimbrial types associated with PAIs are indicated by an asterisk. A number of PAI associated fimbrial gene clusters occupy different locus positions relative to the MG1655 genome. tRNA sites that flank CU-containing PAIs are indicated on the inner blue ring [12]. (Color figure online)

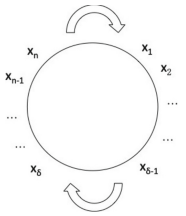

Fig. 2. Circular pattern.

2 Preliminaries

A *string* x of length $|x| = n$ over an alphabet Σ of size σ can be denoted as $x[1\,..\,n] = x[1]x[2]\ldots x[i]\ldots x[n]$, where $1 \leq i \leq n$ and the *i-th letter* of x is denoted by $x[i] \in \Sigma$. The empty string ϵ is the string of length 0. The string x^R is the *reverse* of string x. And $x[i\,..\,j]$, $1 \leq i \leq j \leq n$, denotes the contiguous substring (or factor) of letters, such as $x[i]x[i+1]x[i+2]\ldots x[j]$. A substring $x[i\,..\,j]$ is a suffix of x, if $j = n$ and a substring $x[i\,..\,j]$ is a prefix of x, if $i = 1$. Given a cyclic factor u of length k, $1 \leq k \leq n$, we denote by $c(u)$, for instance, $u = ababc$, $c(u)$ is one of the following rotations: *ababc*, *babca*, *abcab*, *bcaba*, *cabab*. Moreover, we say $u = u_1u_2\ldots u_n$, $c_\delta(u) = u_\delta\ldots u_nu_1\ldots u_{\delta-1}$.

In this paper, suffix trees are used extensively as computation tools. For a general introduction to suffix trees, see [2,7,10,11].

K-Cyclic Period
Input: Given a string x of length n, and an integer $k < n$, compute k-cyclic-period ℓ of x, where $x = u_1u_2\ldots u_\ell$, $u_i = c(u_j)$, $|u_i| = |u_j| = k$, $\forall\ i, j$, $1 \leq i \leq \ell$, $1 \leq j \leq \ell$, and $k \times \ell = n$.
Output: k-cyclic-period ℓ of x

Example 1. Consider a string $x = aaabaabaabaabaaa =: u_1u_2u_3u_4$, where $u_1 = aaab$, $u_2 = aaba$, $u_3 = abaa$, $u_4 = baaa$ and $k = 4$, $\ell = 4$. Therefore x has a period of length ℓ.

Definition 1. *A cyclic periodic array A of a string x of length n is defined to be as follows: $A[i] := \ell$, $1 \leq i \leq n$, if and only if $x[1\,..\,i]$ has cyclic periodicity ℓ by a string u and there no u', with $|u'| \leq |u|$ that is a cyclic period of $x[1\,..\,i]$.*

Example 2. Consider a string $x = aababa$ of length 6, a cyclic periodic array A as follows:

$x[1] = a \implies A[1] := 1$ $\quad$ $x[1\,..\,4] = aaba \implies A[4] := 1$
$x[1\,..\,2] = a\ a \implies A[2] := 2$ $\quad$ $x[1\,..\,5] = aabab \implies A[5] := 1$
$x[1\,..\,3] = aab \implies A[3] := 1$ $\quad$ $x[1\,..\,6] = aab\ aba \implies A[6] := 2$

Definition 2. *We define maximal local k-cyclic periodicity of a string x, if a substring y is cyclic periodic and y is not a substring of another cyclic periodic strings.*

Example 3. Consider a string $x = aaaababaaab$, $\Sigma = \{a, b\}$, and a substring $y = aababababa$ is 3-cyclic periodic and substring $y\alpha = aababababaaa$, $\alpha \in \Sigma$, is not cyclic periodic, and substring $\beta y = aaababababa$, $\beta \in \Sigma$ is not cyclic periodic. Therefore, the substring $y = aababababa$ is maximal local 3-cyclic periodic in string $x = aaaababababaaa$.

Definition 3. *We say that a string x of length n is cyclic-coverable by a string u of length k', if and only if, for every position i of x, the following condition holds $x[\beta\,..\,\gamma] = c(u)$, $1 \leq \beta \leq i \leq \gamma \leq n$.*

Example 4. Consider a string $x = aababaa = u_1u_2$, $u_1 = aaba$, $u_2 = abaa$, $k' = 4$, $\gamma = 2$, is cyclic coverable by a string u, for every position i of x, $x[1\,..\,4] = x[4\,..\,7] = c(u)$.

Definition 4. *Compute all cyclic covers of a given string x, that is for all possible length cyclic covers.*

Example 5. Consider a string $x = ababbaba$, then ab, $abab$, $ababb$, $ababbab$ are cyclic covers of x.

3 Computing k-cyclic Periodicity

Theorem 1. *Given a string x of length n and an integer k, $1 \leq k \leq n$, test whether it is k-cyclic periodic; this can be determined in $\mathcal{O}(n/k)$ time and $\mathcal{O}(n)$ space.*

Proof. We construct the suffix tree of x (see [7,10,11]). We let $u = x[1 \ldots k]$, then let ℓ_m denote the depth of the lowest common ancestor of $x[1\,..\,n]$ (see [1]), and $x[i_m\,..\,n]$. We compute the LCA ℓ_m of $x[1\,..\,n]$ and $x[i_m\,..\,n]$ for $i_m = 2k, 3k, \ldots,$ and $\ell_k = n - k$, if $\ell_m = 1$ for some m, then x is not k-cyclic periodic string. Now consider $C_m^{rignt} = (u_{\ell_m+1} \ldots u_k)^R$, compute the $\ell_m^{'}$ the LCA of u^R and C_m^{right}. If $\ell_m^{'} \geq \ell_m$ for all m, then x is k-cyclic periodic. □

4 Computing All Cyclic Periodicities

Theorem 2. *Given a string x of length n, test whether it is k-cyclic periodicity for all $1 \leq k \leq n$, this can be determined in $\mathcal{O}(n \log n)$ time and $\mathcal{O}(n)$ space.*

Proof. We apply the algorithm of Theorem 1 for $k = 1, 2, \ldots, n$ and we test all cyclic periods of length k. The construction of the suffix tree of string x and x^R is done once costing $\mathcal{O}(n)$. The total cost is

$$\mathcal{O}(n) + \mathcal{O}(\sum_{k=1}^{n} n/k) = \mathcal{O}(n \log n)$$

□

Lemma 1. *Compute the cyclic period of x.*

Proof. The smallest cyclic-period of x is the cyclic-period of x. □

5 Computing Maximal Local k-cyclic Periodicity

Theorem 3. *We can compute all k-cyclic periodicity of x in $\mathcal{O}(n \log n)$ time.*

Proof. We apply the algorithm for $k = 1, 2, \ldots, n$ and in this case, extend it to cyclic periods of length $k+1$, where $|y| = m$ is cyclic periodic and $y\alpha = m+1$ is not cyclic periodic. Next, we perform this algorithm on string x^R as $\mathcal{T}(x^R)$, where $|y| = m$, again is cyclic periodic and $\beta y = m+1$ is not cyclic periodic.

The construction of the suffix tree of string x is done once. The total cost is

$$\mathcal{O}(\sum_{k=1}^{n} n/k) = \mathcal{O}(n \log n)$$

□

Lemma 2. *Compute maximal local k-cyclic periodicity of x.*

Proof. We compute and merge the arrays for $y\alpha$ and βy of x. That is the maximal local k-cyclic periodicity of x. □

6 Computing k'-cyclic Coverability

Theorem 4. *Given a string x of length n and an integer k', $1 \leq k' \leq n$, test whether it is k'-cyclic coverable, this can be determined in $\mathcal{O}(n)$ time and $\mathcal{O}(n)$ space.*

Proof. We compute the suffix tree of string x as $\mathcal{T}(x)$, and also we compute the suffix tree of string x^R as $\mathcal{T}(x^R)$.

Then we check $x[1, k']$ with each one of $x[n-k'+1, n]$, $x[n-k', n-1]$, $x[n-k'-1, n-2] \ldots x[2, k'+1]$, together with the reverse pairs in $T(x^R)$. This way we build a collection of cyclic covers if there is one.

The construction of the suffix tree costs $\mathcal{O}(n)$; checking of equality costs $\mathcal{O}(1)$ and there are n factors. The total time is $\mathcal{O}(n)$. □

7 Computing All Cyclic Coverability

Theorem 5. *Given a string x of length n, test whether it is k'-cyclic coverable for $1 \leq k' \leq n$, this can be determined in $\mathcal{O}(n^2)$ time and $\mathcal{O}(n)$ space.*

Proof. We apply the algorithm for $k' = 1, 2, \ldots, n$ and we compare all cyclic coverable of length k'. The construction of the suffix tree of string x is done once. The total cost is

$$\mathcal{O}(\sum_{k'=1}^{n} n) = \mathcal{O}(n^2)$$

□

Lemma 3. *Compute the cyclic coverability of x.*

Proof. The smallest cyclic coverable of x is the all the cyclic coverable of x. □

8 Conclusions and Open Problems

In this paper, we defined k-cyclic periodicity, we presented several efficient algorithms for computing: (i) cyclic periodicity; (ii) all cyclic periodicity; (iii) maximal local cyclic periodicity; (iv) cyclic covers.

Future work will be focused on computing the cyclic-periodic array, that is the cyclic periodicity of every prefix of string x and computing the cyclic-coverability array, that is testing each prefix of x, for cyclic-coverability. Finally, we will extend the cyclic periodicity to cover the case $u_1u_2u_2 \dots u_ku^1$, where $u_i{=}c(u_j)\ \forall\ i, j$ and u^1 is a substring of some u_i.

References

1. Bender, M.A., Farach-Colton, M.: The LCA problem revisited. In: Gonnet, G.H., Viola, A. (eds.) LATIN 2000. LNCS, vol. 1776, pp. 88–94. Springer, Heidelberg (2000). https://doi.org/10.1007/10719839_9
2. Crochemore, M., Hancart, C., Lecroq, T.: Algorithms on Strings. Cambridge University Press, New York (2007)
3. Defant, C.: Anti-power prefixes of the Thue-Morse word. arXiv preprint arXiv:1607.05825 (2016)
4. Erdős, P., et al.: Anti-ramsey theorems (1973)
5. Fujita, S., Magnant, C., Ozeki, K.: Rainbow generalizations of ramsey theory: a survey. Graphs Comb. **26**(1), 1–30 (2010)
6. Kolpakov, R., Bana, G., Kucherov, G.: mreps: efficient and flexible detection of tandem repeats in DNA. Nucleic Acids Res. **31**(13), 3672–3678 (2003)
7. McCreight, E.M.: A space-economical suffix tree construction algorithm. J. ACM (JACM) **23**(2), 262–272 (1976)
8. Narayanan, S.: Functions on antipower prefix lengths of the Thue-Morse word. arXiv preprint arXiv:1705.06310 (2017)
9. Thue, A.: Uber unendliche zeichenreihen. Norske Vid Selsk. Skr. I Mat-Nat Kl. (Christiana) **7**, 1–22 (1906)
10. Ukkonen, E.: Constructing suffix trees on-line in linear time. In: Proceedings of the IFIP 12th World Computer Congress on Algorithms, Software, Architecture-Information Processing 1992, Volume 1-Volume I, pp. 484–492. North-Holland Publishing Co. (1992)
11. Weiner, P.: Linear pattern matching algorithms. In: 14th Annual Symposium on Switching and Automata Theory, SWAT 1973, pp. 1–11. IEEE (1973)
12. Wurpel, D.J., Beatson, S.A., Totsika, M., Petty, N.K., Schembri, M.A.: Chaperone-usher fimbriae of Escherichia coli. PloS one **8**(1), e52835 (2013)