Discrete Applied Mathematics 163 (2014) 287–297

Contents lists available at ScienceDirect

Discrete Applied Mathematics

journal homepage: www.elsevier.com/locate/dam

Algorithms for computing Abelian periods of words[☆]

Gabriele Fici [a,*], Thierry Lecroq [b], Arnaud Lefebvre [b], Élise Prieur-Gaston [b]

[a] *Dipartimento di Matematica e Informatica, Università di Palermo, Italy*

[b] *Normandie Université, LITIS EA4108, Université de Rouen, 76821 Mont-Saint-Aignan Cedex, France*

ARTICLE INFO

Article history:
Received 13 December 2011
Received in revised form 14 June 2013
Accepted 19 August 2013
Available online 24 September 2013

Keywords:
Abelian period
Abelian repetition
Weak repetition
Design of algorithms
Text algorithms
Combinatorics on words

ABSTRACT

Constantinescu and Ilie [S. Constantinescu, L. Ilie. Fine and Wilf's theorem for abelian periods, Bulletin of the European Association for Theoretical Computer Science 89 (2006) 167–170] introduced the notion of an *Abelian period* of a word. A word of length n over an alphabet of size σ can have $\Theta(n^2)$ distinct Abelian periods. The Brute-Force algorithm computes all the Abelian periods of a word in time $O(n^2 \times \sigma)$ using $O(n \times \sigma)$ space. We present an offline algorithm based on a *select* function having the same worst-case theoretical complexity as the Brute-Force one, but outperforming it in practice. We then present online algorithms that also enable us to compute all the Abelian periods of all the prefixes of w.

1. Introduction

An integer $p > 0$ is a (classical) period of a word w of length n if $w[i] = w[i+p]$ for every $1 \leqslant i \leqslant n-p$. Classical periods have been extensively studied in Combinatorics on Words [13] due to their direct applications in data compression and pattern matching.

The Parikh vector of a word w enumerates the cardinality of each letter of the alphabet in w. For example, given the alphabet $\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$, the Parikh vector of the word $w = \mathtt{aaba}$ is $(3, 1, 0)$. The reader can refer to [3] for a list of applications of Parikh vectors.

An integer p is an *Abelian period* of a word w over a finite alphabet $\Sigma = \{a_1, a_2, \ldots, a_\sigma\}$ if w can be written as $w = u_0u_1 \cdots u_{k-1}u_k$ where for $0 < i < k$ all the u_i's have the same Parikh vector $\mathcal{P}$ such that $\sum_{i=1}^{\sigma} \mathcal{P}[i] = p$ and the Parikh vectors of u_0 and u_k are contained in $\mathcal{P}$ [6]. For example, the word $w = \mathtt{ababbbabb}$ can be written as $w = u_0u_1u_2u_3$, with $u_0 = \mathtt{a}$, $u_1 = \mathtt{bab}$, $u_2 = \mathtt{bba}$ and $u_3 = \mathtt{bb}$, and 3 is an Abelian period of w.

This definition of Abelian period matches the one of *weak repetition* (also called *Abelian power*) when u_0 and u_k are the empty word and $k > 2$ [7].

In recent years, several efficient algorithms have been designed for an Abelian version of the classical pattern matching problem, called the *Jumbled Pattern Matching* problem [5,2,3,14,4,15,1,11], defined as the problem of finding the occurrences of a substring in a text up to a permutation of the letters in the substring, i.e., the occurrences of any substring of the text having the same Parikh vector as the pattern. However, apart from the greedy offline algorithm given in [7], no efficient algorithms are known for computing all the Abelian periods of a given word.[1]

[☆] Some of the results in this paper were presented at Prague Stringology Conference 2011 (Fici et al. (2011) [9]).

[*] Corresponding author. Tel.: +39 09123891111; fax: +39 091 23891024.
E-mail addresses: Gabriele.Fici@unipa.it (G. Fici), Thierry.Lecroq@univ-rouen.fr (T. Lecroq), Arnaud.Lefebvre@univ-rouen.fr (A. Lefebvre), Elise.Prieur@univ-rouen.fr (É. Prieur-Gaston).

[1] During the publication process of the present article, some papers dealing with the computation of the Abelian periods of a word have been published [10,12].

0166-218X/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.dam.2013.08.021

```
AbelianPeriod-BruteForce($w$, $n$)
1  for $h \leftarrow 0$ to $\lfloor (n-1)/2 \rfloor$ do
2    $p \leftarrow h+1$
3    while $h+p \leq n$ do
4      if $(h,p)$ is an Abelian period of $w$ then
5        Output($h$, $p$)
6      $p \leftarrow p+1$
```

Fig. 1. Brute-Force algorithm for computing all the Abelian periods of a word w of length n.

In this article, we present several offline and online algorithms for computing all the Abelian periods of a given word. In Section 2 we give some basic definitions and fix the notation. Section 3 presents offline algorithms, while Section 4 presents online algorithms. In Section 5 we give some experimental results on execution times. Finally, Section 6 contains conclusions and perspectives.

2. Definitions and notation

Let $\Sigma = \{a_1, a_2, \ldots, a_\sigma\}$ be a finite ordered alphabet of cardinality σ and Σ^* the set of words over Σ. We set $ind(a_i) = i$ for $1 \leqslant i \leqslant \sigma$. We denote by $|w|$ the length of w. We write $w[i]$ the i-th symbol of w and $w[i..j]$ the factor of w from the i-th symbol to the j-th symbol included, with $1 \leqslant i \leqslant j \leqslant |w|$. We denote by $|w|_a$ the number of occurrences of the letter $a \in \Sigma$ in the word w.

The *Parikh vector* of a word w, denoted by $\mathcal{P}_w$, counts the occurrences of each letter of Σ in w, i.e., $\mathcal{P}_w = (|w|_{a_1}, \ldots, |w|_{a_\sigma})$. Notice that two words have the same Parikh vector if and only if one is obtained from the other by permuting letters (in other words, one is an anagram of the other). We denote by $\mathcal{P}_w(i, m)$ the Parikh vector of the factor of length m beginning at position i in the word w.

Given the Parikh vector $\mathcal{P}_w$ of a word w, we denote by $\mathcal{P}_w[i]$ its i-th component and by $|\mathcal{P}_w|$ its norm, that is the sum of its components. Thus, for $w \in \Sigma^*$ and $1 \leqslant i \leqslant \sigma$, we have $\mathcal{P}_w[i] = |w|_{a_i}$ and $|\mathcal{P}_w| = \sum_{i=1}^{\sigma} \mathcal{P}_w[i] = |w|$. Finally, given two Parikh vectors $\mathcal{P}$, $\mathcal{Q}$, we write $\mathcal{P} \subset \mathcal{Q}$ if $\mathcal{P}[i] \leqslant \mathcal{Q}[i]$ for every $1 \leqslant i \leqslant \sigma$ and $|\mathcal{P}| < |\mathcal{Q}|$.

Definition 1 ([6]). A word w has an Abelian period (h, p) if $w = u_0 u_1 \cdots u_{k-1} u_k$ such that:

- $\mathcal{P}_{u_0} \subset \mathcal{P}_{u_1} = \cdots = \mathcal{P}_{u_{k-1}} \supset \mathcal{P}_{u_k}$,
- $|\mathcal{P}_{u_0}| = h$, $|\mathcal{P}_{u_1}| = p$.

We call u_0 and u_k resp. the *head* and the *tail* of the Abelian period. Notice that the length $t = |u_k|$ of the tail is uniquely determined by h, p and $|w|$, namely $t = (|w| - h) \bmod p$.

The following lemma gives an upper bound on the number of Abelian periods of a word.

Lemma 2.1. *A word of length n over an alphabet Σ of cardinality σ can have $\Theta(n^2)$ different Abelian periods.*

Proof. The word $w = (a_1 a_2 \cdots a_\sigma)^{n/\sigma}$ has Abelian period (h, p) for any $p \equiv 0 \mod \sigma$ and every h such that $0 \leqslant h \leqslant \min(p-1, n-p)$. Therefore, w has $\Theta(n^2)$ different Abelian periods. □

A natural order can be defined on the Abelian periods of a word.

Definition 2. Two distinct Abelian periods (h, p) and (h', p') of a word w are ordered as follows: $(h, p) < (h', p')$ if $p < p'$ or ($p = p'$ and $h < h'$).

We are interested in computing all the Abelian periods of a word. However, the algorithms we present in this paper can be easily adapted to give the smallest Abelian period only.

3. Offline algorithms

3.1. Brute-Force algorithm

In Fig. 1, we present a Brute-Force algorithm computing all the Abelian periods of an input word w of length n. For each possible head of length h from 1 to $\lfloor (n-1)/2 \rfloor$ the algorithm tests all the possible values of p such that $p > h$ and $h + p \leqslant n$. It is a reformulation of the algorithm given in [7].

Example 1. For $w =$ abaababa the algorithm outputs $(1, 2)$, $(0, 3)$, $(2, 3)$, $(1, 4)$, $(2, 4)$, $(3, 4)$, $(0, 5)$, $(1, 5)$, $(2, 5)$, $(3, 5)$, $(0, 6)$, $(1, 6)$, $(2, 6)$, $(0, 7)$, $(1, 7)$ and $(0, 8)$. Among these periods, $(1, 2)$ is the smallest.

Theorem 3.1. *The algorithm* AbelianPeriod-BruteForce *computes all the Abelian periods of a given word of length n in time $O(n^2 \times \sigma)$ with $O(n \times \sigma)$ space.*