On scattered subword complexity

Zoltán KÁSA
Sapientia Hungarian University of Transylvania
Department of Mathematics and Informatics,
Tg. Mureș, Romania
e-mail: kasa@ms.sapientia.ro

Abstract. Special scattered subwords, in which the gaps are of length from a given set, are defined. The scattered subword complexity, which is the number of such scattered subwords, is computed for rainbow words.

1 Introduction

Sequences of characters called words or strings are widely studied in combinatorics, and used in various fields of sciences (e.g. chemistry, physics, social sciences, biology [2, 3, 4, 11] etc.). The elements of a word are called letters. A contiguous part of a word (obtained by erasing a prefix or/and a suffix) is a subword or factor. If we erase arbitrary letters from a word, what is obtained is a scattered subword. Special scattered subwords, in which the consecutive letters are at distance at most \( d \) (\( d \geq 1 \)) in the original word, are called \( d \)-subwords [7, 8]. In [9] the super-\( d \)-subword is defined, in which case the distances are of length at least \( d \). The super-\( d \)-complexity, as the number of such subwords, is computed for rainbow words (words with pairwise different letters).

In this paper we define special scattered subwords, for which the distance in the original word of length \( n \) between two letters which will be consecutive in the subword, is taken from a subset of \( \{1,2,\ldots,n-1\} \).

Computing Classification System 1998: G.2.1, F.2.2
Mathematics Subject Classification 2010: 68R15
Key words and phrases: word complexity, scattered subword, \( d \)-complexity, super-\( d \)-complexity
The complexity of a word is defined as the number of all its different subwords. Similar definitions are for \(d\)-complexity, super-\(d\)-complexity and scattered subword complexity.

The scattered subword complexity is computed in the special case of rainbow words. The idea of using scattered words with gaps of length between two given values is from József Bukor [1].

Another point of view of scattered complexity in the case of non-primitive words is given is [5].

## 2 Definitions

Let \(\Sigma\) be an alphabet, \(\Sigma^n\), as usually, the set of all words of length \(n\) over \(\Sigma\), and \(\Sigma^*\) the set of all finite word over \(\Sigma\).

**Definition 1** Let \(n\) and \(s\) be positive integers, \(M \subseteq \{1, 2, \ldots, n-1\}\) and \(u = x_1x_2 \ldots x_n \in \Sigma^n\). An \(M\)-subword of length \(s\) of \(u\) is defined as \(v = x_{i_1}x_{i_2} \ldots x_{i_s}\) where

- \(i_1 \geq 1\),
- \(i_{j+1} - i_j \in M\) for \(j = 1, 2, \ldots, s-1\),
- \(i_s \leq n\).

**Definition 2** The number of \(M\)-subwords of a word \(u\) for a given set \(M\) is the scattered subword complexity, simply \(M\)-complexity.

The \(M\)-subword in the case of \(M = \{1, 2, \ldots, d\}\) is the \(d\)-subword defined in [7], while in the case of \(M = \{d, d+1, \ldots, n-1\}\) is the super-\(d\)-complexity defined in [9].

**Examples.** The word abcd has 11 \(\{1, 3\}\)-subwords: a, ab, abc, abcd, ad, b, bc, bcd, c, cd, d. The \(\{2, 3, \ldots, n-1\}\)-subwords of the word abcdedef are the following: a, ac, ad, ae, af, ace, acf, adf, b, bd, be, bf, bdf, c, ce, cf, d, df, e, f.

Hereinafter instead of \(\{d_1, d_1 + 1, \ldots, d_2 - 1, d_2\}\)-subword we will use the simple notation \((d_1, d_2)\)-subword.

## 3 Computing the scattered complexity for rainbow words

Words with pairwise different letters are called rainbow words. The \(M\)-complexity of a rainbow word of length \(n\) does not depend on what letters it contains,
and is denoted by $K(n, M)$.

Let us recall two results for special scattered words, as $d$-subwords and super-$d$-subwords.

For a rainbow word of length $n$ the super-$d$-complexity [9] is equal to

$$K(n, \{d, d+1, \ldots, n-1\}) = \sum_{k \geq 0} \binom{n-(d-1)k}{k+1},$$

and the $(n-d)$-complexity [8] is

$$K(n, \{1, 2, \ldots, n-d\}) = 2^n - (d-2) \cdot 2^{d-1} - 2, \text{ for } n \geq 2d-2.$$

For special cases the following propositions can be easily proved.

**Proposition 3** For $n, d_1 \leq d_2$ positive integers

$$K(n, \{d_1, d_1+1, \ldots, d_2\}) \leq n + \sum_{k \geq 1} \binom{n-(d_1-1)k}{k+1} - \sum_{k \geq 1} \binom{n-d_2k}{k+1}.$$

**Proof.** This can be obtained from (1) and the formula

$$K(n, \{d_1, d_1+1, \ldots, n-1\}) \leq K(n, \{d_1, d_1+1, \ldots, n-1\}) - K(n, \{d_2+1, d_2+2, \ldots, n-1\}) + n.$$

For example, $K(7, \{2, 3, 4, 5, 6\}) = 33$, $K(7, \{4, 5, 6\}) = 13$, and from the proposition $K(7, \{2, 3\}) \leq 27$. The exact value is $K(7, \{2, 3\}) = 25$, the two words $acg$ and $aeg$ are not eliminated (here the original distances are 2 and 4 in $acg$, and 4 and 2 in $aeg$).

**Proposition 4** For the integers $n, d \geq 1$, where $n = hd + m$

$$K(n, \{d\}) = \frac{(h+1)(n+m)}{2}.$$

**Proof.**

$$K(n, \{d\}) = n + \sum_{i=1}^{n-d} \left\lfloor \frac{n-i}{d} \right\rfloor = n + d(1 + 2 + \ldots + h - 1) + mh$$

$$= n + \frac{dh(h-1)}{2} + mh = \frac{(h+1)(n+m)}{2}.$$
To compute the M-complexity of a rainbow word of length n we will use graph theoretical results. Let us consider the rainbow word $a_1a_2\ldots a_n$ and the correspondig digraph $G = (V,E)$, with

$V = \{a_1, a_2, \ldots, a_n\}$,

$E = \{(a_i, a_j) \mid j - i \in M, i = 1, 2, \ldots, n, j = 1, 2, \ldots, n\}$.

For $n = 6, M = \{2, 3, 4, 5\}$ see Figure 1.

The adjacency matrix $A = (a_{ij})_{i=1\ldots n, j=1\ldots n}$ of the graph is defined by:

$$a_{ij} = \begin{cases} 1, & \text{if } j - i \in M, \\ 0, & \text{otherwise}, \end{cases} \quad \text{for } i = 1, 2, \ldots, n, j = 1, 2, \ldots, n.$$

Because the graph has no directed cycles, the entry in row $i$ and column $j$ in $A^k$ (where $A^k = A^{k-1}A$, with $A^1 = A$) will represent the number of directed paths of length $k$ from $a_i$ to $a_j$. If $I$ is the identity matrix (with entries equal to 1 only on the first diagonal, and 0 otherwise), let us define the matrix $R = (r_{ij})$:

$$R = I + A + A^2 + \cdots + A^k,$$

where $A^{k+1} = O$ (the null matrix).

The $M$-complexity of a rainbow word is then

$$K(n, M) = \sum_{i=1}^{n} \sum_{j=1}^{n} r_{ij}.$$

Matrix $R$ can be better computed using a variant of the well-known Warshall algorithm (for the original Warshall algorithm see for example [12]):
Warshall\( (A,n) \)
1 \( W \leftarrow A \)
2 \( \text{for } k \leftarrow 1 \text{ to } n \)
3 \( \hspace{1em} \text{do for } i \leftarrow 1 \text{ to } n \)
4 \( \hspace{2em} \text{do for } j \leftarrow 1 \text{ to } n \)
5 \( \hspace{3em} \text{do } w_{ij} \leftarrow w_{ij} + w_{ik}w_{kj} \)
6 \( \text{return } W \)

From \( W \) we obtain easily \( R = I + W \).

For example let us consider the graph in Figure 1. The corresponding adjacency matrix is:

\[
A = \begin{pmatrix}
0 & 0 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}
\]

After applying the Warshall algorithm:

\[
W = \begin{pmatrix}
0 & 0 & 1 & 1 & 2 & 3 \\
0 & 0 & 0 & 1 & 1 & 2 \\
0 & 0 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}, \quad R = \begin{pmatrix}
1 & 0 & 1 & 1 & 2 & 3 \\
0 & 1 & 0 & 1 & 1 & 2 \\
0 & 0 & 1 & 0 & 1 & 1 \\
0 & 0 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1
\end{pmatrix}
\]

and then \( K(6,\{2,3,4,5\}) = 20 \), the sum of elements in \( R \).

The Warshall algorithm combined with the Latin square method can be used to obtain all nontrivial (with length at least 2) \( M \)-subwords of a given rainbow word \( a_1a_2\cdots a_n \). Let us consider a matrix \( A \) with the entries \( A_{ij} \), which are set of words. Initially this matrix is defined as:

\[
A_{ij} = \begin{cases}
\{a_ia_j\}, & \text{if } j - i \in M, \\
\emptyset, & \text{otherwise},
\end{cases} \quad \text{for } i = 1, 2, \ldots, n, j = 1, 2, \ldots, n.
\]

If \( A \) and \( B \) are sets of words, \( AB \) will be formed by the set of concatenation of each word from \( A \) with each word from \( B \):

\[
AB = \{ab \mid a \in A, b \in B\}.
\]

If \( s = s_1s_2\cdots s_p \) is a word, let us denote by \( \prime s \) the word obtained from \( s \) by erasing the first character: \( \prime s = s_2s_3\cdots s_p \). Let us denote by \( A'_{ij} \) the set \( A_{ij} \)
in which we erase the first character from each element. In this case \( \mathcal{A} \) is a matrix with entries \( \mathcal{A}_{ij} \).

Starting with the matrix \( A \) defined as before, the algorithm to obtain all nontrivial \( M \)-subwords is the following:

\[
\text{Warshall-Latin}(A, n)
\]

1. \( W \leftarrow A \)
2. \( \text{for } k \leftarrow 1 \text{ to } n \)
3. \( \text{do for } i \leftarrow 1 \text{ to } n \)
4. \( \text{do for } j \leftarrow 1 \text{ to } n \)
5. \( \text{do if } W_{ik} \neq \emptyset \text{ and } W_{kj} \neq \emptyset \)
6. \( \text{then } W_{ij} \leftarrow W_{ij} \cup W_{ik}'W_{kj} \)
7. \( \text{return } W \)

The set of nontrivial \( M \)-subwords is \( \bigcup_{i,j \in \{1,2,...,n\}} W_{ij} \).

For \( n = 8 \), \( M = \{3,4,5,6,7\} \) the initial matrix is:

\[
\begin{pmatrix}
\emptyset & \emptyset & \emptyset & \{ad\} & \{ae\} & \{af\} & \{ag\} & \{ah\} \\
\emptyset & \emptyset & \emptyset & \{be\} & \{bf\} & \{bg\} & \{bh\} \\
\emptyset & \emptyset & \emptyset & \emptyset & \{cf\} & \{cg\} & \{ch\} \\
\emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \{dg\} & \{dh\} \\
\emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \{eh\} \\
\emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset \\
\emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset \\
\emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset
\end{pmatrix}
\]

The result of the algorithm \text{Warshall-Latin} in this case is:

\[
\begin{pmatrix}
\emptyset & \emptyset & \emptyset & \{ad\} & \{ae\} & \{af\} & \{ag,adg\} & \{ah,adh,aeh\} \\
\emptyset & \emptyset & \emptyset & \{be\} & \{bf\} & \{bg\} & \{bh,beh\} \\
\emptyset & \emptyset & \emptyset & \emptyset & \{cf\} & \{cg\} & \{ch\} \\
\emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \{dg\} & \{dh\} \\
\emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \{eh\} \\
\emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset \\
\emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset \\
\emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset
\end{pmatrix}
\]

The algorithm \text{Warshall-Latin} can be used for nonrainbow words too, with the remark that repeating subwords must be eliminated. For the word \text{aabbbaa} and \( M = \{3,4,5,6,7\} \) the result is: \text{aa, ab, aba, ba}.
4 Computing the \((d_1, d_2)\)-complexity

Let us denote by \(a_i\) the number of \((d_1, d_2)\)-subwords which terminate at position \(i\) in a rainbow word of length \(n\). Then

\[
a_i = 1 + a_{i-d_1} + a_{i-d_1-1} + \cdots + a_{i-d_2},
\]

with the remark that for \(i \leq 0\) we have \(a_i = 0\). Subtracting \(a_{i-1}\) from \(a_i\) we get the following simpler equation.

\[
a_i = a_{i-1} + a_{i-d_1} - a_{i-1-d_2}.
\]

The \((d_1, d_2)\)-complexity of a rainbow word of length \(n\) is

\[
K(n, \{d_1, d_1+1, \ldots, d_2\}) = \sum_{i=1}^{n} a_i
\]

For example, if \(d_1 = 2, d_2 = 4\), the following values are obtained

<table>
<thead>
<tr>
<th>(n)</th>
<th>(a_n)</th>
<th>(K(n, {2, 3, 4}))</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>7</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>11</td>
</tr>
<tr>
<td>6</td>
<td>7</td>
<td>16</td>
</tr>
<tr>
<td>7</td>
<td>11</td>
<td>24</td>
</tr>
<tr>
<td>8</td>
<td>16</td>
<td>35</td>
</tr>
<tr>
<td>9</td>
<td>24</td>
<td>52</td>
</tr>
<tr>
<td>10</td>
<td>35</td>
<td>76</td>
</tr>
<tr>
<td>11</td>
<td>52</td>
<td>112</td>
</tr>
<tr>
<td>12</td>
<td>76</td>
<td>157</td>
</tr>
<tr>
<td>13</td>
<td>112</td>
<td>233</td>
</tr>
</tbody>
</table>

If we denote by \(A(z) = \sum_{n \geq 1} a_n z^n\) the generating function of the sequence \(a_n\), then from (2) we obtain

\[
\sum_{n \geq 1} a_n z^n = \sum_{n \geq 1} z^n + \sum_{n \geq 1} a_{n-d_1} z^{n-d_1} + \cdots + \sum_{n \geq 1} a_{n-d_2} z^{n-d_2},
\]

and

\[
A(z) = \frac{z}{1-z} + z^{d_1} A(z) + \cdots + z^{d_2} A(z).
\]

From this we obtain

\[
A(z) = \frac{z}{z^{d_2+1} - z^{d_1} - z + 1}.
\]

For \(d_1 = 2, d_2 = 4\) the sequence \((a_n)_{n \geq 0}\) ([10] sequence A023435) corresponds to a variant of the dying rabbits problem [6].
To compute the generating function for the complexity $K(n, \{d_1, d_1 + 1, \ldots, d_2\})$, let us denote this complexity simply by $K_n$ only, and its generating function by $K(z) = \sum_{n \geq 1} K_n z^n$. We remark that $K_n = 0$ for $n \leq 0$, and $K_1 = 1$.

From (3) and (4) we can immediately conclude that

$$K(z) = \frac{1}{1 - z} A(z) = \frac{z}{(1 - z)(z^{d_2+1} - z^{d_1} - z + 1)}.$$ 

5 Correspondence between $(d, n + d - 1)$-subwords and $\{1, d\}$-subwords

The following result is inspired from the sequence A050228\(^1\) of [10].

**Proposition 5** The number of $\{1, d\}$-subwords of a rainbow word of length $n$ is equal to the number of $(d, d + 1, \ldots, n + d - 1)$-subwords of length at least 2 of a rainbow word of length $n + d$.

**Proof.** By the generalization of the sequence A050228 [10] the number of the $\{1, d\}$-subwords of a rainbow word of length $n$ is equal to

$$K(n, \{1, d\}) = \sum_{k \geq 0} \binom{n + 1 - (d - 1)k}{k + 2}.$$ 

From (1) we have

$$K(n + d, \{d, d + 1, \ldots, n + d - 1\}) - (n + d) = \sum_{k \geq 1} \binom{n + d - (d - 1)k}{k + 1}.$$ 

By changing $k$ to $k + 1$ in the sum, we obtain

$$\sum_{k \geq 0} \binom{n + 1 - (d - 1)k}{k + 2},$$

and this proves the theorem. □

**Example.** For $abcde$ the 19 $\{1, 3\}$-subwords are:

$a, b, c, d, e, ab, abc, abcd, ad, ade, abce, abce, abce, bc, bc, bc, bc, cde, cde, de$.

For $abcdefgh$ the 19 $\{3, 4, 5, 6, 7\}$-subwords of length at least 2 are:

$ad, ae, af, ag, adg, ah, aeh, be, bf, bg, bh, beh, cf, cg, ch, cg, dh, dh, eh$.

\(^1\)A050228: $a_n$ is the number of subsequences $\{s_k\}$ of $\{1, 2, 3, \ldots, n\}$ such that $s_{k+1} - s_k$ is 1 or 3.
Conclusions

A special scattered subword, the so-called M-subword is defined, in which the distances (gaps) between letters are from the set M. The number of the M-subwords of a given word is the M-complexity. Graph algorithms are used to compute the M-complexity and to determine all M-subwords of a rainbow word. This notion of M-complexity is a generalization of the d-complexity [7] and of the super-d-complexity [9]. If M consists of successive numbers from $d_1$ to $d_2$ then the so-called $(d_1, d_2)$-complexity is computed by recursive equations and generating functions.

Acknowledgements

This work was supported by the project under the grant agreement no. TÁMOP 4.2.1/B-09/1/KMR-2010-0003 (Eötvös Loránd University, Budapest) financed by the European Union and the European Social Fund.

References


*Received: December 4, 2010 • Revised: March 12, 2011*