Annals of Combinatorics 4 (2000) 247-255
0218-0006/00/040247-9$1.50+0.20/0

Annals of Combinatorics

Avoided Strings in Bacterial Complete Genomes and a Related Combinatorial Problem*

Bailin Hao, Huimin Xie †, Zuguo Yu, and Guoyi Chen

Institute of Theoretical Physics, Academia Sinica, P. O. Box 2735, Beijing 100080, China
hao@itp.ac.cn

Received December 12, 1998

AMS Subject Classification: 05A15, 92C40

Abstract. The visualization of avoided and under-represented strings in some bacterial complete genomes raises a combinatorial problem which may be solved either by using the Goulden–Jackson cluster method or by construction of the minimal finite automaton defined by the set of forbidden words of the corresponding language.

Keywords: complete genomes, avoided strings, language, enumeration, fractal

1. Introduction

The heredity information of organisms (except for so-called RNA-viruses) is encoded in their DNA sequence which is a one-dimensional unbranched polymer made of four different kinds of monomers (nucleotides): adenine (a), cytosine (c), guanine (g), and thymine (t). As far as the encoded information is concerned, we can ignore the fact that DNA exists as a double helix of two "conjugated" strands and only treat it as a one-dimensional symbolic sequence made of the four letters from the *alphabet* $\Sigma = \{a, c, g, t\}$. Since the first complete genome of a free-living bacterium *Mycoplasma genitalium* was sequenced in 1995, an ever-growing number of complete genomes has been deposited in public databases. The availability of complete genomes opens the possibility to ask some global questions on these sequences. One of the simplest conceivable questions consists of checking whether there are short strings of letters that are absent or under-represented in a complete genome. The answer is in the affirmative and the fact may have some biological meaning [4].

The reason why we are interested in absent or under-represented strings is twofold. First of all, this is a question that can only be asked in the present day when complete genomes are at our disposal. Second, the question makes sense as one can derive a *factorizable* language from a complete genome which would be entirely defined by the set of forbidden words.

We start by considering how to visualize the avoided and under-represented strings in a bacterial genome whose length is usually the order of a million letters.

* Partially Supported by the Chinese Natural Science Foundation and the Project on Nonlinear Science

† On leave from the Department of Mathematics, Suzhou University, Jiangsu 215006, China.

2. Visualization of Under-Represented Strings

There are 4^K different strings of length K made of four letters. In order to check whether all these strings appear in a genome we use 4^K counters to be visualized as a $2^K \times 2^K$ square array on a computer screen. These can be realized as a direct product of K identical 2×2 matrices:

$$M^{(K)} = M \otimes M \otimes \cdots \otimes M,$$

where

$$M = \begin{pmatrix} g & c \\ a & t \end{pmatrix}.$$

We call this $2^K \times 2^K$ square a K-frame. In practice it is convenient to use binary subscripts for this 2×2 matrix and it is easy to develop an algorithm that depends only on the total length of the genome but not on the string length K. Put in a frame of fixed K and described by a color code biased towards small counts, each bacterial genome shows a distinctive pattern which indicates on absent or under-represented strings of certain types [4]. For example, many bacteria avoid strings containing the string *ctag*. Any string that contains *ctag* as a substring will be called a *ctag*-tagged string. If we mark all *ctag*-tagged strings in frames of different K, we get pictures as shown in

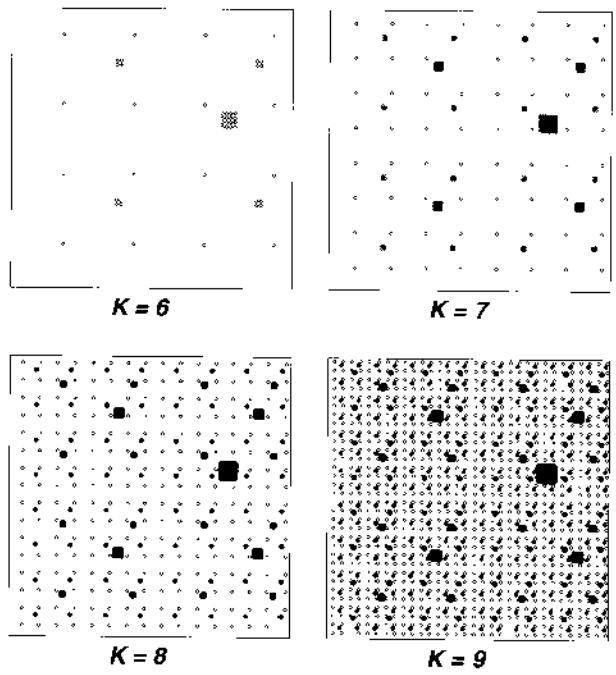

Figure 1: *Ctag*-tagged strings in $K = 6$ to 9 frames.

Figure 1. The large scale structure of these pictures persists but more details appear

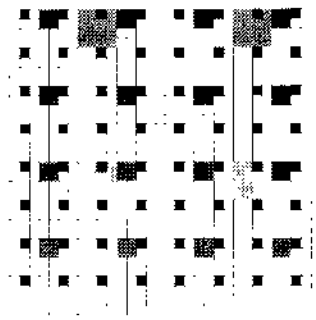

Figure 2: The pattern of cg-tagged strings showing the overlaps.

with growing K. Excluding the area occupied by these tagged strings, one gets a fractal in the $K \to \infty$ limit. It is natural to ask what is the dimension of this fractal for a given tag.

In fact, this is the dimension of the complementary set of the tagged strings. The simplest case is that of g-tagged strings. As the pattern has an apparently self-similar structure, the dimension is easily calculated to be

$$D = \frac{\log 3}{\log 2}.$$

Moreover, the dimension of all other cases must lie in between $\log 3/\log 2$ and 2. However, the calculation of these dimensions is somewhat tricky as one must take into account the overlap of patterns precisely (see, for example the case of cg-tagged strings shown in Figure 2).

Now let a_K be the number of all strings of length K that do not contain the given tag. As the linear size δ_K in the K-frame is $1/2^K$, the dimension may be calculated as:

$$D = \lim_{K\to\infty} \frac{\log a_K}{-\log \delta_K} = \lim_{K\to\infty} \frac{\log a_K{}^{1/K}}{\log 2}.$$

Suppose the generating function of a_K is known:

$$f(s) = \sum_{K=0}^{\infty} a_K s^K.$$

Then, according to the Cauchy criterion of convergence, we have

$$\lim_{K\to\infty} a_K^{1/K} = |\lambda| = \frac{1}{|s_0|},$$

where λ is the radius of convergence of series expansion of $f(s)$ and s_0 is the minimal module zero of $f^{-1}(s)$. This finally determines the dimension

$$D = -\frac{\log |s_0|}{\log 2}.$$

Table 1: Generating function and dimension for some single tags.

Tag	$f(s)$	D	Tag	$f(s)$	D
g	$\frac{1}{1-3s}$	$\frac{\log 3}{\log 2}$	ggg	$\frac{1+s+s^2}{1-3s-3s^2-3s^3}$	1.98235
gc	$\frac{1}{1-4s+s^2}$	1.89997	$ctag$	$\frac{1}{1-4s+s^4}$	1.99429
gg	$\frac{1+s}{1-3s-3s^2}$	1.92266	$ggcg$	$\frac{1+s^3}{1-4s+s^3-3s^4}$	1.99438
gct	$\frac{1}{1-4s+s^3}$	1.97652	$gcgc$	$\frac{1+s^2}{1-4s+s2-4s^3+s^4}$	1.99463
gcg	$\frac{1+s^2}{1-4s+s^2-3s^3}$	1.978	$gggg$	$\frac{1+s+s^2+s^3}{1-3s-3s^2-3s^3-3s^4}$	1.99572

The generating function for the numbers of strings of various length made of the four letters that do not contain certain designated strings ("bad words" as called in [6]) may be calculated by using the Goulden–Jackson cluster method [2], well-described by Noonan and Zeilberger [6]. In particular, the case of a single tag—one "bad word" only—is easily treated and some of the results are shown in Table 1.

A related question is the number $G(n)$ of different types of generating functions for a given tag length n. These numbers turn out to be independent upon the size of the alphabet Σ as long as there are more than two letters in Σ [3]:

n	1	2	3	4	5	6	7	8	9	10	11
$G(n)$	1	2	3	4	6	8	10	13	17	21	27

In fact, these $G(n)$ are so-called correlations of n as given by the integer sequence $M0555$ in [7] (see also [3]).

3. Redundant and True Avoided Strings

Once we know that there are avoided strings in the complete genomes from the visualization scheme, one can perform a direct search for these strings. The direct search has the merit not being significantly limited by the string length K. However, another combinatorial problem arises which is closely related to the problem discussed in the previous section. Take, for example, the complete genome of *E. coli*. At $K = 7$, the first avoided string *gcctagg* is discovered. At the next $K = 8$ level, a total of 173 avoided strings are identified. However, these 173 strings are not all true avoided strings as some must be the consequence of the absence of the $K = 7$ string *gcctagg*. A naive estimate of the redundant avoided strings without taking into account any possible overlap of substrings would lead to $4^i(i+1)$: If there is only one avoided string at the $K+0$ level, it would take away 8, 48, 256, 1280, ... strings at the next $K+i$ levels. This estimate works well for *E. coli* until $K = 13$ when the overlap of the first and the last letter g in the true avoided string *gcctagg* would show off. Applying the Goulden–Jackson cluster method to the case of only one "bad word" *gcctagg* leads to the following generating

function:

$$f(s) = \frac{1+s^6}{1-4s+s^6-3s^7}.$$

The number of redundant avoided strings are given by

$$\frac{1}{1-4s} - f(s) = s^7 + 8s^8 + 48s^9 + 256s^{10} + 1280s^{11} + 6144s^{12} + 28671s^{13} + 131063s^{14} + \cdots.$$

The deviation from the naive estimate appears from s^{13}.

For a non-trivial example, we consider the newly published complete genome of the hyperthermophilic bacterium *Aquifex aeolicus* [1]. For this, 155 1335-letter sequence four avoided strings are identified at $K = 7$. They form the set B of "bad words":

$$B = \{gcgcgcg,\ gcgcgca,\ cgcgcgc,\ tgcgcgc\}.$$

As there are significant overlaps among these strings, the naive estimate of redundant avoided words can hardly work. The application of the Goulden–Jackson cluster method requires the solution of a system of four linear equations and leads to the following generating function:

$$f(s) = \frac{1+s^2+s^4+s^6+s^8+s^{10}+s^{12}}{1-4s+s^2-4s^3+s^4-4s^5+s^6-4s^8-4s^{10}-4s^{12}}.$$

The numbers of redundant avoided strings are given by:

$$\frac{1}{1-4s} - f(s) = 4s^7 + 27s^8 + 152s^9 + 784s^{10} + 3840s^{11} + \cdots.$$

In what follows we show that these results may be obtained by an entirely different method, namely, by making use of formal language theory. For convenience of presentation, we first collect a few notions from language theory without proofs. The details may be found, e.g., in [9] and references therein.

4. Some Notions from Formal Language Theory

In formal language theory one starts with an alphabet, e.g., $\Sigma = \{a, c, g, t\}$. Let Σ^* denote the collection of all possible strings made of letters from Σ, including the empty string ε. Any subset $L \subset \Sigma^*$ is called a *language* over the alphabet Σ. The set $L' = \Sigma^* - L$ defines the complementary language. A language L is a factorizable language if any substring of a word $x \in L$ also belongs to L. A factorizable language has a minimal set of forbidden words or *Distinctive Excluded Blocks* [8] (DEBs) L'' such that, if $x \in L''$, then any proper substring of x belongs to L. A factorizable language is completely determined by its set of DEBs:

$$L = \Sigma^* - \Sigma^* L'' \Sigma^*.$$

A prominent example of factorizable language is given by the admissible symbolic sequences in the symbolic dynamics of a dynamical system (see, e.g., [5, 9]). Another class of factorizable languages may be obtained from a complete genome as follows.

Let G be a complete genome of an organism; it may consist of one or more linear or circular sequences. All possible substrings of G, including the empty string ε and G itself, obviously form a subset of Σ^* and thus define a language which is factorizable by construction.

Any language $L \subset \Sigma^*$ introduces an equivalence relation R_L in Σ^* with respect to L. For any pair $x, y \in \Sigma^*$ xR_Ly if and only if for each $z \in \Sigma^*$, either both $xz, yz \in L$ or both $xz, yz \notin L$. The number of equivalence classes in Σ^* with respect to L defines the *index* of R_L, denoted by index(R_L).

An important theorem (Myhill–Nerode) says that L is a regular language if and only if index(R_L) is finite and L being regular implies that the minimal deterministic automaton corresponding to L, minDFA(L), is unique up to an isomorphism, i.e., to renaming of the states. Moreover, the number of states in minDFA(L) equals to index(R_L).

Let L be a factorizable language and L'' its set of all DEB's. Define a set

$$V = \{v | v \text{ is a proper prefix of some } y \in L''\}.$$

For each word $x \in L$, there exists a string $v \in V$ such that xR_Lv. In other words, all equivalence classes of L are represented in the set V. In order to find all equivalence classes of Σ^* with respect to L, it is enough to start from L''. In addition, L' is an equivalence class of Σ^*. For two given strings $u, v \in V$, uR_Lv if and only if for each $z \in \Sigma^*$ uz contains a DEB as its suffix $\Leftrightarrow$ $vz \in L'$ and *vice versa*. This statement sets the computation rule to identify all equivalence classes. Each equivalence class may be named after a member $x_i \in L$ and be denoted as $[x_i]$. The transfer function between states of minDFA(L) is defined as $\delta([x_i], s) = [x_is]$ for $x_i \in L$ and $s \in \Sigma$.

5. Finite Automaton and Incidence Matrix

Now we apply what has just been said to the complete genome of *Aquifex aeolicus* with its set B of four avoided strings at length $K = 7$. Although there are longer avoided strings we take B to be its L'' for the time being. From the proper suffixes of these strings, we get the set

$$V = \{g, gc, gcg, gcgc, gcgcg, gcgcgc, c, cg, cgc, cgcg, \\ cgcgc, cgcgcg, t, tg, tgc, tgcg, tgcgc, tgcgcg\}.$$

By checking the equivalence class of these strings, only 13 out of these 18 strings are kept as representatives of each class. Adding the class $[L'] \subset \Sigma^*$, we get the following 14 equivalence classes of Σ^*:

$$[\varepsilon]\ [g]\ [gc]\ [gcg]\ [gcgc]\ [gcgcg]\ [gcgcgc] \\ [c]\ [cg]\ [cgc]\ [cgcg]\ [cgcgc]\ [cgcgcg]\ [L'].$$

The transfer function $\delta([x_i], s) = [x_is]$, $x_i \in V$ and $s \in \Sigma$, is determined by attributing $[x_is]$ to the existing equivalence classes. It is listed in Table 2. The particular transfer function $\delta([x_i], s) = [L']$ leads to a "dead end".

By counting the number of lines leading from one state to another, we write down

Table 2: The transfer function for the minimal deterministic automaton for *Aquifex aeolicus*.

$[x_i]\backslash s$	a	c	g	t
$[\varepsilon]$	$[\varepsilon]$	$[c]$	$[g]$	$[c]$
$[g]$	$[\varepsilon]$	$[gc]$	$[g]$	$[c]$
$[gc]$	$[\varepsilon]$	$[c]$	$[gcg]$	$[c]$
$[gcg]$	$[\varepsilon]$	$[gcgc]$	$[g]$	$[c]$
$[gcgc]$	$[\varepsilon]$	$[c]$	$[gcgcg]$	$[c]$
$[gcgcg]$	$[\varepsilon]$	$[gcgcgc]$	$[g]$	$[c]$
$[gcgcgc]$	$[L']$	$[c]$	$[L']$	$[c]$
$[c]$	$[\varepsilon]$	$[c]$	$[cg]$	$[c]$
$[cg]$	$[\varepsilon]$	$[cgc]$	$[g]$	$[c]$
$[cgc]$	$[\varepsilon]$	$[c]$	$[cgcg]$	$[c]$
$[cgcg]$	$[\varepsilon]$	$[cgcgc]$	$[g]$	$[c]$
$[cgcgc]$	$[\varepsilon]$	$[c]$	$[cgcgcg]$	$[c]$
$[cgcgcg]$	$[\varepsilon]$	$[L']$	$[g]$	$[c]$

an *incidence matrix*:

$$M = \begin{bmatrix}
1 & 1 & & & & & 2 & & & & \\
1 & 1 & 1 & & & & 1 & & & & \\
1 & & & 1 & & & 2 & & & & \\
1 & 1 & & & 1 & & 1 & & & & \\
1 & & & & & 1 & 2 & & & & \\
1 & 1 & & & & & 1 & 1 & & & \\
 & & & & & 2 & & & & & \\
1 & & & & & & 2 & 1 & & & \\
1 & 1 & & & & & 1 & & 1 & & \\
1 & & & & & & 2 & & & 1 & \\
1 & 1 & & & & & 1 & & & & 1 \\
1 & & & & & & 2 & & & & \\
1 & 1 & & & & & 1 & & & &
\end{bmatrix}.$$

One draws the minimal deterministic automaton according to the above transfer function. As it is no longer a planar graph, we do not show it here. The columns

and rows of the matrix M are ordered as elements in the first column in Table 2 of the transfer function.

To make connection with the generating function

$$f(s) = \sum_{0}^{\infty} a_K s^K,$$

obtained by using the Goulden–Jackson cluster method, we note that the sum of elements in the first row of the Kth power of M is nothing but a^K [8]:

$$a_K = \sum_{j=1}^{13} (M^K)_{1j}.$$

The summation runs over all equivalence classes except for L'. We list the elements of the first row of M^K in columns of Table 3.

The negative numbers in the last row of Table 3 show the difference of a_K and 4^K. They are precisely the coefficients in the expansion of $1/(1-4s) - f(s)$ as shown at the end of Section 3. We see that the transfer function and the incidence matrix contain more detailed information on the combinatorial problem than the generating function alone. The consequence of this approach has to be further elucidated in the future.

Table 3: Elements of the first rows of M_K and their sum.

$K=$	1	2	3	4	5	6	7	8	9	10	11
	1	4	16	64	256	1024	4095	16378	65501	261960	1047664
	1	2	8	32	128	512	2048	8190	32756	131002	523920
	0	1	2	8	32	128	512	2048	8190	32756	131002
	0	0	1	2	8	32	128	512	2048	8190	32756
	0	0	0	1	2	8	32	128	512	2048	8190
	0	0	0	0	1	2	8	32	128	512	2048
	0	0	0	0	0	1	2	8	32	128	512
	2	7	28	112	448	1792	7168	28665	114640	458483	1833624
	0	2	7	28	112	448	1792	7168	28665	114640	458483
	0	0	2	7	28	112	448	1792	7168	28665	114640
	0	0	0	2	7	28	112	448	1792	7168	28665
	0	0	0	0	2	7	28	112	448	1792	7168
	0	0	0	0	0	2	7	28	112	448	1792
Sum:	4	16	64	256	1024	4096	16380	65509	261992	1047792	4190464
							-4	-27	-152	-784	-3840

Acknowledgement. The first author thanks Professor Zeilberger for calling his attention to the excellent presentation of the Goulden–Jackson cluster method in [6] during the CAP98 conference.