JOURNAL OF COMPUTATIONAL BIOLOGY
Volume 10, Number 2, 2003

Pp. 171–185

Discovering Simple Regions in Biological Sequences Associated with Scoring Schemes

HONGHUI WAN,[1] LUGANG LI,[2] SCOTT FEDERHEN,[3] and JOHN C. WOOTTON[3]

ABSTRACT

Let A denote an alphabet consisting of n types of letters. Given a sequence S of length L with v_i letters of type i on A, to describe the compositional properties and combinatorial structure of S, we propose a new complexity function of S, called the reciprocal complexity of S, as

$$C(S) = \prod_{i=1}^{n} \left(\frac{L}{nv_i} \right)^{v_i}.$$

Based on this complexity measure, an efficient algorithm is developed for classifying and analyzing simple segments of protein and nucleotide sequence databases associated with scoring schemes. The running time of the algorithm is nearly proportional to the sequence length. The program DSR corresponding to the algorithm was written in C++, associated with two parameters (window length and cutoff value) and a scoring matrix. Some examples regarding protein sequences illustrate how the method can be used to find regions. The first application of DSR is the masking of *simple* sequences for searching databases. Queries masked by DSR returned a manageable set of hits below the *E*-value cutoff score, which contained all true positive homologues. The second application is to study *simple* regions detected by the DSR program corresponding to known structural features of proteins. An extensive computational analysis has been made of protein sequences with known, physicochemically defined nonglobular segments. For the SWISS-PROT amino acid sequence database (Release 40.2 of 02-Nov-2001), we determine that the best parameters and the best BLOSUM matrix are, respectively, for automatic segmentation of amino acid sequences into nonglobular and globular regions by the DSR program: Window length $k = 35$, cutoff value $b = 0.46$, and the BLOSUM 62.5 matrix. The average "agreement accuracy (sensitivity)" of DSR segmentation for the SWISS-PROT database is 97.3%.

Key words: complexity, biological sequence, simple region, scoring scheme, search masking, nonglobular domain.

[1]National Center for Genome Resources, Santa Fe, NM 87505.
[2]Department of Protective Medicine, Nanjing Army Medical College, Second Army Medical University, Nanjing, Jiangsu 210099, China.
[3]Computational Biology Branch, National Center for Biotechnology Information, National Library of Medicine, National Institutes of Health, Bethesda, MD 20894.

1. INTRODUCTION

NATURAL BIOLOGICAL MOLECULAR SEQUENCES are very different from random symbolic strings as shown in Appendix A. Numerous segments, regions, or patches of contrasting compositional bias occur in both nucleotide and amino acid sequences (Salamon and Konopka, 1992; Wootton and Federhen, 1993; Wootton, 1994). Such a property commonly contrasts in regions of distinct molecular function and mutational history. Diagnosing, mapping, and classifying these local features are essential aspects of assigning biological functions to sequences. In particular, an important problem in sequence analysis is to find those "simple" (low-complexity) segments of a sequence or "simple" sequences in a sequence database associated with scoring schemes. Here, the term "simple" conveys only part of the diversity and richness of variegation in natural sequences, which may be very subtle. This has profound implications for the understanding of molecular structure, function, variation, and evolution.

Simple DNA sequences usually show complicated subpatterns arising from mutational processes such as unequal crossing over, replication slippage, biased substitution, and transposition. Examples are microsatellites, VNTRs, long tandemly repeated satellite DNAs, telomeric sequences, irregular low-complexity sequences in intergenic regions, CpG islands, and recombinational hotspots. In addition, many segments of nucleic acid sequences show more subtle compositional biases. A typical example is the trinucleotide quasi-periodicity in coding sequences arising from the different biases in the three codon positions.

Recent statistical analyses of protein sequence databases show that approximately one-quarter of the residues are in compositionally biased regions and more than one-half of proteins have at least one such region (Wootton, 1994). Biased regions include nonglobular structural domains, interspersed simple polymers, and regular or irregular repeats. These occur especially in large multidomain proteins with crucial biological functions in transcriptional regulation, RNA processing, signal transduction, morphogenesis, embryonic development, and both intracellular and extracellular structure and integrity. Many have molecular interactions with important biological consequences, as shown by protein engineering experiments, variations in human disease, and locations of autoimmune epitopes. Investigation of local compositional complexity and periodicity is really informative at an early stage of the analysis of a new protein sequence, especially when results can be interpreted together with local matches from database searches (Altschul *et al*., 1994; Wootton and Federhen, 1996). In natural protein sequences, there is a strong tendency for compact globular folded domains to have a high complexity of composition that resembles a "random" distribution of amino acid frequencies (Finkelstein, 1994). In contrast, compositionally biased segments of lower complexity correlate in many cases (some exceptions are known) with nonglobular, extended, or conformationally mobile structure. Many simple protein segments are known to be involved in crucial molecular functions and interactions, but in general they are relatively intractable to structural study by crystallographic and NMR methods, in contrast to globular domains.

Given this range of important aspects of research on simple segments of biological sequences, it is best to base the computational analysis and quantitative description on precise formalisms of compositional complexity. In most previous work on biological sequences, complexity measures have been based on well-known functions in information theory or statistical theory such as informational entropy, information content, or data compression schemes.

The search for simple regions associated with scoring schemes in DNA and protein sequences or sequence databases is important in sequence analysis. The rapid increase in available sequences, in particular from large-scale genome sequencing projects, makes it significant to develop sensitive automatic methods for the identification of simple regions. In this paper, we develop some new algorithms that extend the utility of compositional complexity measures in biological sequence analysis and present an efficient algorithm for classifying and analyzing simple segments of protein and nucleotide sequence databases associated with real-valued similarity scores.

2. A NEW COMPLEXITY MEASURE OF SEQUENCES

A DNA sequence is considered here as a sequence from the four-letter alphabet {A, C, G, T}. Similarly, a protein sequence can be considered as a sequence from a 20-letter alphabet. Other alphabets can also be used, e.g., the two-letter purine–pyrimidine alphabet. We therefore consider the general case of sequences

from an arbitrary finite alphabet $\mathbf{A} = \{a_1, a_2, \ldots, a_n\}$ of n letters, in which a_i is called the letter of type i $(1 \leq i \leq n)$. Symbolic sequences are characterized by $\mathbf{A}$ and (usually) by a finite length L. One-dimensional strings play an important role in various fields such as genetics, dynamic systems, biology, linguistics, and psychology.

Let $\mathbf{A}^L$ denote the set of sequences of length L over $\mathbf{A}$ for a positive integer L. Given a sequence S in $\mathbf{A}^L$, we denote by v_i the number of letters of type i in S, that is, the number of occurrences of a_i in S $(1 \leq i \leq n)$. Obviously, $v_1 + v_2 + \cdots + v_n = L$ and $0 \leq v_i \leq L$. The vector $V = (v_1, v_2, \ldots, v_n)$ is called the *frequency vector* and $U = (\frac{v_1}{L}, \frac{v_2}{L}, \ldots, \frac{v_n}{L})$ is called the *normalized frequency vector* of S. The set of sequences with the frequency vector V is called the *V-equivalence class*, denoted by $\mathcal{S}(V)$.

For a sequence $S = s_1 s_2 \ldots s_L \in \mathcal{S}(v_1, v_2, \ldots, v_n)$, the Shannon entropy (Shannon, 1948) of S is defined as (Staden, 1984; Wootton and Federhen, 1993)

$$H(S) = -\sum_{i=1}^{n} \frac{v_i}{L} \log \frac{v_i}{L}. \tag{1}$$

This can be regarded as the average information per position in the sequences. Here, the logarithm is taken to base n (usually, $n = 4$ for DNA sequences and $n = 20$ for protein sequences), so that $0 \leq H(S) \leq 1$. To express complexity in frequently used information units, logarithms may be taken to base 2, giving bits, or to base e, giving nats. However, the applications in biological sequence analysis make it more natural to use base n. Accordingly, all logarithms throughout this paper are taken to the base n, unless otherwise stated.

The probability of occurrence of s_i in S is the proportion of letters of j-th type in S, i.e., $\frac{v_i}{L}$, whenever $s_i = a_j$. The probability of occurrence of all v_j letters of type j is $\left(\frac{v_j}{L}\right)^{v_j}$. The product of these probabilities is their joint probability (P) and is the probability associated with the whole sequence. Hence,

$$P(S) = \prod_{i=1}^{n} \left(\frac{v_i}{L}\right)^{v_i}. \tag{2}$$

Obviously,

$$H(S) = -\frac{1}{L} \log(P(S)). \tag{3}$$

In other words, the Shannon entropy of a sequence can be written as the negative logarithm to some base of the geometric mean of the probabilities of occurrence of residues in the sequence. Expression (3) can thus be regarded as another definition of the Shannon entropy of a sequence by means of the probability-product.

The inverse of the function $P(S)$ is called the *reciprocal probability* of S:

$$C(S) = \prod_{i=1}^{n} \left(\frac{L}{v_i}\right)^{v_i}. \tag{4}$$

As shown in Appendix B, $C(S)$ achieves the maximum value n^L whenever $v_1 = v_2 = \cdots = v_n$. We derive the following normalized form of $C(S)$, which will be used instead as definition of the reciprocal probability of S in expression (4):

$$C(S) = \left(\frac{1}{n}\right)^L \prod_{i=1}^{n} \left(\frac{L}{v_i}\right)^{v_i}. \tag{5}$$

$C(S)$ is called the *reciprocal complexity* of S.

3. ALGORITHM FOR LOCATING SIMPLE SEGMENTS

We focus in this section on algorithm development and software design for using our new complexity measure defined in (5) to delineate simple segments of biological sequences.

Shannon entropy is effective only as "local" measures in the context of biological sequences, but it cannot be rigorously comparable across sequences of substantially different size (Wan and Wootton, 1999, 2000). In particular, it is hard to use it to identify or compare those sequences of different length that have equivalent complexity. For instance, we always have that $H(S) = 0$ for any sequence S in the equivalence class $\mathcal{S}(L, 0, \ldots, 0)$, $L = 1, 2, \ldots$. Let us look at a typical example. For the DNA sequence $S =$ `gtgtgtac`, it is obvious that the longest "simplest segment" of S should be `gtgtgt`. Table 1 lists all distinct segments of S and their frequency vectors along with Shannon complexity values. Clearly, `gtgtgt` cannot be found by exhaustive search for the subsequence of S with the minimum value of the Shannon complexity function.

As an example, the rightmost column in Table 1 lists the reciprocal complexity values of all distinct segments of the DNA sequence `gtgtgtac`. The segment `gtgtgt` achieves the minimum value of the reciprocal complexity function. The resulting segmentation corresponds well to intuitive views of the detection of these "simple" regions of sequences. In fact, the reciprocal complexity function is particularly suitable for this optimized segmentation because it gives closely similar expected values at all window lengths.

For any sequence $S \in \mathbf{A}^L$, we denote by $S[i..j]$ the (contiguous) segment of S which starts at position i and ends at position j in S and write $S[i..j] \sqsubset S$. In particular, $S[i..i+k-1]$ is a *k-gram* or a *k-window* of S that starts at position i and ends at position $i+k-1$. For simplicity, we write $S_i = S[i..(i+k-1)]$ for $1 \le i \le L-k+1$. To explore and investigate nonrandom compositional heterogeneity in sequence databases, we use the reciprocal complexity function as our "objective measure" to find the "simple segments" of a sequence.

We now present a two-pass algorithm to find simple segments of a sequence using two parameters. The first stage uses the reciprocal complexity function to form an approximate description of "raw" simple segments, and the second stage employs the reciprocal complexity function for local optimization within each raw segment to refine the simple region boundaries.

A sliding window (k-gram) of length k is moved in single-residue steps along the sequence, and complexity $C(S_i)$ is computed at each step. *Trigger windows* with respect to a real number t are those of

TABLE 1. SHANNON ENTROPY VALUES OF ALL (PAIRWISE) DISTINCT SEGMENTS OF `gtgtgtac`

			Shannon entropy		*Reciprocal complexity*	
Length	*Segment*	*Frequency vector*	*Value*	*Rank*	*Value*	*Rank*
1	`a, c, g, t`	(1, 0, 0, 0)	0.0000	1	0.2500	7
2	`gt, tg, ta, ac`	(1, 1, 0, 0)	0.5000	4	0.2500	7
3	`gtg, tgt`	(2, 1, 0, 0)	0.4591	2	0.1055	5
3	`gta, tac`	(1, 1, 1, 0)	0.7925	9	0.4219	9
4	`gtgt, tgtg`	(2, 2, 0, 0)	0.5000	4	0.0625	3
4	`tgta`	(2, 1, 1, 0)	0.7500	7	0.2500	7
4	`gtac`	(1, 1, 1, 1)	1.0000	14	1.0000	13
5	`gtgtg, tgtgt`	(3, 2, 0, 0)	0.4855	3	0.0283	2
5	`gtgta`	(2, 2, 1, 0)	0.7610	8	0.1907	6
5	`tgtac`	(2, 1, 1, 1)	0.9610	13	0.7629	12
6	`gtgtgt`	(3, 3, 0, 0)	0.5000	4	0.0156	1
6	`tgtgta`	(3, 2, 1, 0)	0.7296	6	0.1055	5
6	`gtgtac`	(2, 2, 1, 1)	0.9591	12	0.7119	11
7	`gtgtgta`	(3, 3, 1, 0)	0.7244	5	0.0690	4
7	`tgtgtac`	(3, 2, 1, 1)	0.9212	11	0.4654	10
8	`gtgtgtac`	(3, 3, 1, 1)	0.9056	10	0.3512	8

complexity not greater than t. A *raw segment* with respect to t may consist of a single trigger window or a contig of overlapping windows. The *contig* with respect to t is constructed by merging each trigger window in both directions with any overlapping trigger windows. Each nonoverlapping contig is called a *raw segment* with respect to t.

At the second stage of the algorithm, each raw segment is reduced to a single optimal simple segment, which may be the entire raw segment but is usually a shorter subsequence. This is found by exhaustive search for the subsequence of the raw segment with the minimum value of the reciprocal complexity function C.

Algorithm 1. Given a sequence $S = s_1 s_2 \dots s_L \in \mathbf{A}^L$, a proper positive integer $k \leq L$, and an appropriate real number b between 0 and 1, identify simple segments of S.

Step 1. Determine the frequency vector $V_1 = (v_1, v_2, \dots, v_n)$ of $S_1 = s_1 s_2 \dots s_k$ and compute $C(V_1)$ using the formula

$$C(V_1) = \frac{1}{n^k} \prod_{i=1}^{n} \left(\frac{k}{v_i} \right)^{v_i}.$$

Step 2. Recognize the letter-types (on $\mathbf{A}$) of residues s_1 and s_{k+1}, say p and q, respectively. Then the frequency vector V_2 of S_2 can be expressed as $V_2 = V_1 - e_p + e_q$, where e_j denotes the j-th n-dimensional unit vector

$$e_j = (\underbrace{0, \dots, 0, 1}_{j}, \underbrace{0, \dots, 0}_{n-j}).$$

Step 3. Calculate $C(V_2)$ based on $C(V_1)$, e_p, and e_q, using the following formula

$$C(V_2) = C(V_1) \frac{v_p^{v_p} v_q^{v_q}}{(v_p - 1)^{v_p - 1} (v_q + 1)^{v_q + 1}}.$$

Step 4. Repeat Step 3 until $C(V_{L-k+1})$ is calculated.

Step 5. Find the minimum value m_k and the maximum value M_k of $C(V_i)$ by exhaustive search for all i with $1 \leq i \leq L - k + 1$, where V_i is the frequency vector of S_i.

Step 6. Locate all trigger windows of length k and complexity not greater than $t = m_k + b(M_k - m_k)$, $k = 1, 2, \dots, l$.

Step 7. Construct contigs $W_1, W_2, \dots, W_r$ by merging each trigger window in both directions with any overlapping trigger windows.

Step 8. For each contig W_i, look for the leftmost, say T_i, of the longest of those subsequences of W_i that attain the minimum value of C, defined in Equation (5), by exhaustive search ($i = 1, \dots, r$); i.e., first filter on C, then filter on length, then choose the leftmost of the remaining. Finally, output $T_1, T_2, \dots, T_r$, which are simple segments of S.

We now turn to computational complexity analysis of Algorithm 1. Recall that for a given alphabet $\mathbf{A}$, n is fixed (usually, $n = 4$ or 20) in biological applications. Step 1 takes $O(k)$ time and space. Both Steps 2 and 3 require $O(1)$ time and space. Steps 4, 5, and 6 need $O(L - k + 1)$ time and space. For all k with $1 \leq k \leq L$, Steps 1, 2, 3, 4, 5, and 6 totally require $O(L^2)$ time and space. Obviously, Step 7 takes $O(k)$ time and space. To analyze Step 8 in Algorithm 1, we denote by l_i the length of W_i ($i = 1, \dots, r$). Detecting T_i by exhaustive search in contig W_i needs $O(l_i^2)$ time and space. Note that $l_1 + l_2 + \dots + l_r \leq L$. Thus, Step 8 requires at most $O(L^2)$ time and space. Therefore, Algorithm 1 can be executed in at most $O(L^2)$ time in the worst case for a fixed n.

4. EXTENSION TO ARBITRARY ALPHABETS AND REAL-VALUED SIMILARITY SCORES

We now consider extension to arbitrary alphabets and real-valued similarity scores. To analyze realistic biological data, similarity scores, or distances, between pairs of letters may be based on the δ-function

defined in the alphabet $\mathbf{A} = \{a_1, \ldots, a_n\}$ (as considered above):

$$\delta(a_i, a_j) = \begin{cases} 1 & \text{if } a_i = a_j, \\ 0 & \text{otherwise.} \end{cases}$$

In this case, the letters occur independently and with equal probability. However, more commonly are real-valued matrices (or ratios in some cases) as in PAM (Dayhoff *et al.*, 1978) or BLOSUM matrices (Henikoff and Henikoff, 1992). A special simple case of scoring schemes, frequently used in sequence analysis, is to consider the letters of the alphabet to be associated with weights. These weights may represent a physiochemical property, such as hydrophobicity. The vector of the n weights, $w(i)$, for each letter a_i in the alphabet $\mathbf{A}$, are normalized. Then the "score" assigned to (a_i, a_j) is $\frac{w(i)+w(j)}{2}$ (Tomii and Kanehisa, 1996). In addition, several authors have developed an elegant theory of scoring schemes in the context of database search (Altschul, 1991; Altschul *et al.*, 1994).

Now we develop a new algorithm for classifying and analyzing segments of protein and nucleotide sequence databases associated with real-valued similarity scores, to show compositional features. To this end, we need some new definitions.

We define a real-valued function σ in $\mathbf{A} \times \mathbf{A}$ by $\sigma(a_i, a_j) = \sigma_{ij}$ for $1 \le i, j \le n$. The matrix $M(\mathbf{A}) = (\sigma_{ij})_{n\times n}$ is called the *(similarity) scoring scheme* of the alphabet $\mathbf{A}$, in which σ_{ij} is called a *score*. Obviously, the δ-function is the special case of the σ-function when $\sigma_{ij} = \delta(a_i, a_j)$ for $1 \le i, j \le n$, and the scoring scheme in this case is the $n \times n$ identity matrix.

We denote by h and l the highest score and the lowest score in the scoring scheme $M(\mathbf{A}) = (\sigma_{ij})_{n\times n}$, respectively; i.e., $h = \max\{\sigma_{ij} \mid 1 \le i, j \le n\}$, $l = \min\{\sigma_{ij} \mid 1 \le i, j \le n\}$. Then $M^*(\mathbf{A}) = \left(\sigma^*(a_i, a_j)\right)_{n\times n}$ is called the *normalized scoring scheme* of $\mathbf{A}$, where

$$\sigma^*(a_i, a_j) = \frac{\sigma_{ij} - l}{h - l}.$$

For simplicity and convenience, we use the logarithm of $C(S)$ as a complexity measure of sequences rather than its original expression in (5). Thus, the reciprocal complexity of a sequence $S = s_1 s_2 \ldots s_L$ associated with the normalized scoring scheme $M^*(\mathbf{A})$ is defined as

$$C^*(S) = L \log L - L \log n - \sum_{i=1}^{n} v_i^* \log v_i^*, \tag{6}$$

where

$$v_i^* = \sum_{j=1}^{L} \sigma^*(a_i, s_j).$$

Algorithm 2. Given a sequence $S = s_1 s_2 \ldots s_L \in \mathbf{A}^L$, a scoring scheme $M(\mathbf{A})$, a proper positive integer $k \le L$, and a suitable positive real number b between 0 and 1, detect simple segments of S.

Step 1. Get the normalized scoring scheme $M^*(\mathbf{A})$ from $M(\mathbf{A})$ and determine $v_i = \sum_{j=1}^{k} \sigma^*(a_i, s_j)$ for $1 \le i \le n$ and compute $C^*(S_1)$ using the formula

$$C^*(S_1) = k \log k - k \log n - \sum_{i=1}^{n} v_i \log v_i.$$

Step 2. Determine $r_i = v_i - \sigma^*(a_i, s_1) + \sigma^*(a_i, s_{k+1})$ for $1 \le i \le n$ and calculate $C^*(S_2)$ based on the following formula

$$C^*(S_2) = k \log k - k \log n - \sum_{i=1}^{n} r_i \log r_i.$$

Step 3. Compute $C^*(S_3), \ldots, C^*(S_{l-k+1})$, using procedure similar to that described in Step 2.

Step 4. Find the minimum value m_k and the maximum value M_k of $C^*(S_i)$ for $1 \le i \le L - k + 1$.

Step 5. Locate all trigger windows of length k and complexity not greater than $t = m_k + b(M_k - m_k)$.

Step 6. Construct contigs $W_1, W_2, \ldots, W_r$ by merging each trigger window in both directions with any overlapping trigger windows.

Step 7. For each contig W_i, look for the leftmost, denoted by T_i, of the longest of those subsequences of W_i that attain the minimum value of C^*, defined in Equation (6), by exhaustive search ($i = 1, \ldots, r$):

$$C^*(T_i) = \min\{C^*(T) \mid T \sqsubset W_i\}.$$

$T_1, T_2, \ldots, T_r$ are simple segments of S.

The parameter b is called the *cutoff value*. Algorithm 2 has been implemented in the computer language C++. The code is under the name "DSR" and is portable to most computers with a *C* compiler. It can be easily used by the biologists to do complex trait analysis of biological sequences.

5. APPLICATIONS

The first application of the program is the masking of *simple* sequences for searching databases. Both proteins and nucleic acids contain *simple* regions, which can lead to confusing results during the performance of sequence database searches. The residue frequencies within such regions differ dramatically from the database as a whole, disrupting the statistics used by BLAST. This can produce alignments that are based purely on compositional bias rather than a significant position-by-position alignment (Wootton and Federhen, 1993, 1996). Actually, in most cases, it does not make sense from either structural or mutational viewpoints to attempt to align *simple* sequences position-by-position. This problem is solved by automatic masking, using DSR, of the *simple* segments in database search query sequences, replaced by a stretch of N's (nucleotide) or X's (protein). The masking is associated with a scoring scheme which will be used in the database search.

Figure 1 shows *Thermus thermophilus* seryl tRNA synthetase filtered in this way using DSR associated with the BLOSUM 62 matrix. This sequence had 31 true positive homologues in BLASTP searches of the NCBI nonredundant databases of December 20, 2000. Run unfiltered, it gave an overwhelming output list of 1,125 database sequences below the *E*-value cutoff score. Almost all of these "hits" were spurious *simple* matches. DSR masking reduced the BLASTP output list to a manageable 32 sequences and revealed some previously buried true matches to weaker homologous partial scrapie sequences in addition to the strong synthetase protein homologues. All 31 true positive homologues are in this BLASTP output list, in which only one is not a true positive homologue. As an alternative to DSR, SEG (*www.ncbi.nlm.nih.gov/Education/BLASTinfo/Seg.html*), which was developed by Wootton and Federhen (1993) and is associated without the BLOSUM 62 matrix (i.e., with the identity matrix) in the masking, reduced the output list to 92 sequences. Obviously , compared to the output list reduced by DRS, this list contains too many "noises" and most sequences in it are not true positive homologues.

We give another complete example for the use of DSR associated with the BLOSUM 62 matrix in the masking of *simple* sequences for searching databases. Let us look at the human prion protein sequence (GI number: 130912, SWISS-PROT accession number: P04156, name: PRIO_HUMAN). PRIO_HUMAN had 34 true positive homologues in BLASTP searches of the NCBI nonredundant databases of December 20, 2000. Run unmasked below the *E*-value threshold score, it yielded 2,856 hits. PRIO_HUMAN when masked by DSR returned 37 hits, containing all 34 true positive homologues, verus 156 hits when masked by SEG. The extra hits found by SEG contained a much higher ratio of apparently unspecific matches, due to excessive filtering. Generally, where large discrepancies were observed, queries masked by SEG returned many more hits below the *E*-value cutoff score. A lot of computational experiments have shown that the difference in numbers was very striking between using SEG and using DSR.

Masking variable loop or "quasi-loop" regions helps to filter out many "noises" and get only true positive homologues in BLAST searches. On the other hand, if the periodic regions of some proteins, e.g., 7TM regions, are masked, that could substantially hurt identification of GPCRs—although admittedly, this would be hard to show using SCOP. Recently, Wan and Song (2002) defined a quasi-periodic region in terms of "nearness to periodicity" based on the average Hamming distance to the nearest perfect-periodic sequence

(a)

Simple regions		Complex regions
	1-7	MVDLKRL
rqepevfhrairekg	8-22	
	23-29	VALDLEA
llaldrevqelkkrlqevqternqvakrvp kappeekealiargkalgeeakrleealre kearlealllqvplppwpgapvggeeanre ikrvggppef	30-129	
	130-164	SFPPLDHVALMEKNGWWEPRISQVSGSRSY ALKGD
lalyelallrfamdf	165-179	
	180-295	MARRGFLPMTLPSYAREKAFLGTGHFPAYR DQVWAIAETDLYLTGTAEVVLNALHSGEIL PYEALPLRYAGYAPAFRSEAGSFGKDVRGL MRVHQFHKVEQYVLTEASLEASDRAF
qellenaeeilrllelpyrlveva	296-319	
	320-355	TGDMGPGKWRQVDIEVYLPSEGRYRETHSC SALLDW
qarranlryrdpegrvr	356-372	
	373-392	YAYTLNNTALATPRILAMLL
enhqlqdgrvrvpqa	393-407	
	408-421	LIPYMGKEVLEPCG

(b)

>1SRY
MVDLKRLxxxxxxxxxxxxxxxVALDLEAxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxSFPPLDHVALMEKNG
WWEPRISQVSGSRSYALKGDxxxxxxxxxxxxxxxxMARRGFLPMTLPSYAREKAFLGTGHF
PAYRDQVWAIAETDLYLTGTAEVVLNALHSGEILPYEALPLRYAGYAPAFRSEAGSFGK
DVRGLMRVHQFHKVEQYVLTEASLEASDRAFxxxxxxxxxxxxxxxxxxxxxxxxxTGDMGPGK
WRQVDIEVYLPSEGRYRETHSCSALLDWxxxxxxxxxxxxxxxxxYAYTLNNTALATPRILAM
LLxxxxxxxxxxxxxxxLIPYMGKEVLEPCG

FIG. 1. Automated segmentation of the *Thermus thermophilus* seryl tRNA synthetase by compositional complexity using the DSR software program associated with the BLOSUM 62 matrix. (**a**) A readable display of the simple and complex rigons, represented by lower and upper case, respectively. The sequence regions all read from left to right and their order in the polypeptide is top to bottom, as shown by the central column of residue numbers. (**b**) The masked sequence ready for use as a database search query sequence. This is in FASTA format with the simple sequences replaced with "x" characters. These results were obtained by the DSR software program using the parameters window length $k = 15$, cutoff value $b = 0.5$. The scoring scheme is the BLOSUM 62 matrix.

and developed an efficient algorithm for finding quasi-periodic regions in sequences. Using this algorithm, we recognize those quasi-periodic subregions in the simple regions detected by the DSR program and then keep them in search masking which can help searches for really remote homologs such as the evaluations of search methods that are applied to the SCOP database. A detailed treatment will appear in a separate paper.

DSR rigorously explores nonrandom compositional heterogeneity as represented in real genomic and protein sequences, independently of prior expectations based on precedents. It is useful for discovery of subtle repeating properties in proteins, especially for those representing a physiochemical property such as hydrophobicity or polarity. Figure 2 shows segmentation of the globular and nonglobular (coiled-coil) domains of the seryl tRNA synthetase protein (*www.rcsb.org/pdb/cgi/explore.cgi?pid=161710052757 79&pdbId=1SES*) on the basis of sequence complexity. In this case, the BLOSUM 62 matrix was used. The

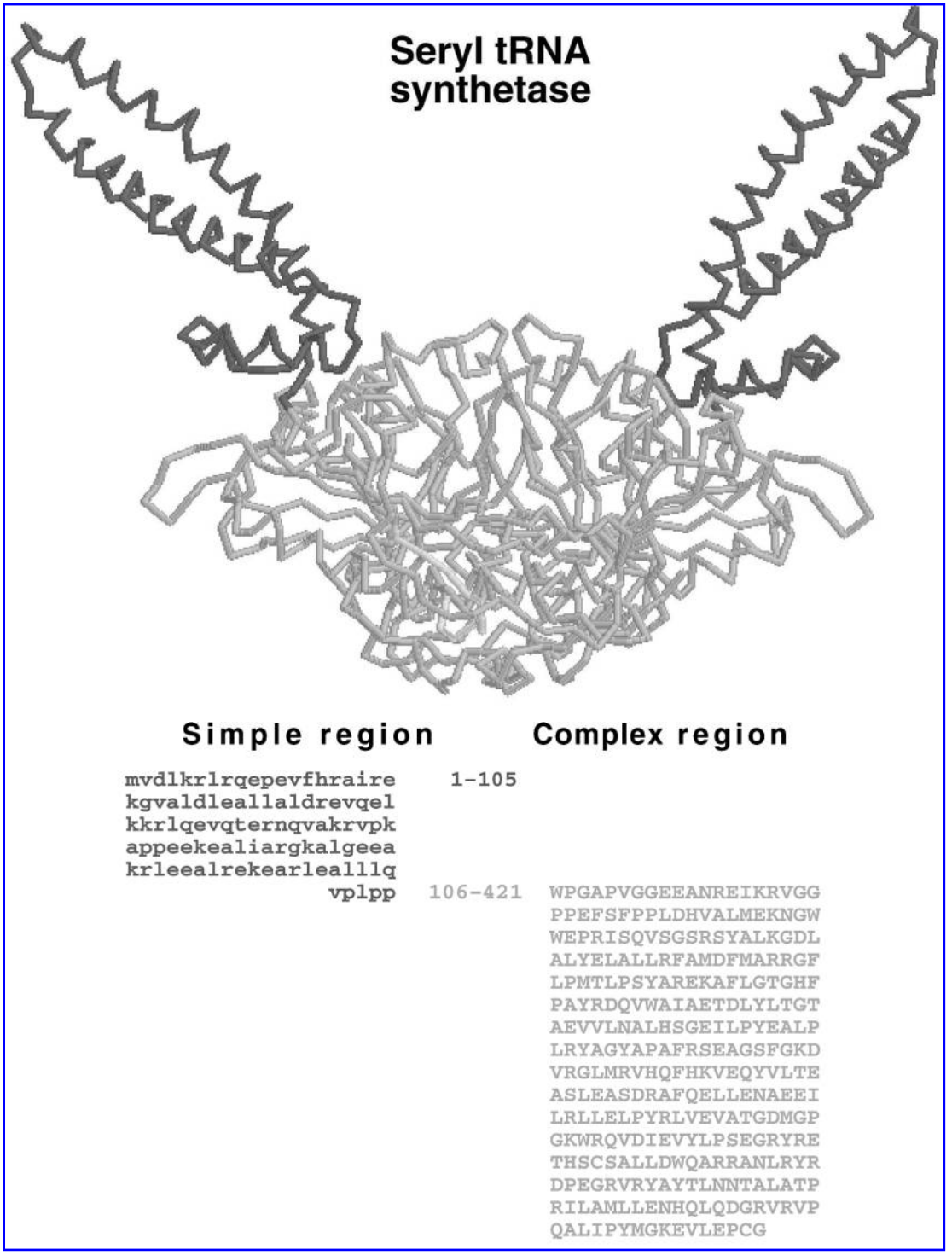

FIG. 2. Segmentation of the globular and nonglobular (coiled-coil) domains of the seryl tRNA synthetase protein on the basis of sequence complexity, associated with the BLOSUM 62 matrix. The parameters used in this study are window length $k = 35$, cutoff value $b = 0.46$.

sequence was segmented automatically by the DSR algorithm, and the corresponding parts of the protein structure (from X-ray crystallography) are shown by red (lower-complexity segment, with weak 7-residue repeat) and green (high complexity segment, globular domain).

To study *simple* regions detected by the DSR program corresponding to known structural features of proteins, an extensive computational analysis has been made of protein sequences with known, physicochemically defined nonglobular segments. Tables 2, 3, and 4 show, respectively, *simple* and *complex* segments identified by DSR compared with experimentally-determined nonglobular and globular regions of three typical proteins: (1) Collagen alpha 2(I) chain [Precursor] (*www.expasy.ch/cgi-bin/niceprot.pl?P08123*), (2) Myosin heavy chain, cardiac muscle beta isoform (*www.expasy.ch/cgi-bin/niceprot.pl?P12883*), and (3) Aggrecan core protein [Precursor] (*www.expasy.ch/cgi-bin/niceprot.pl?P16112*). The parameters used in these studies are: Window length $k = 35$, cutoff value $b = 0.46$. The scoring scheme is the BLOSUM 62 matrix.

To "measure" the efficiency of the DSR program for the particular investigation of segmentation of nonglobular and globular regions, we mathematically define an "agreement accuracy (sensitivity)." For any sequence $S \in \mathbf{A}^L$, suppose that $S[i..j]$ is a *simple* (*complex*) segment of S detected by DSR, which corresponds to known nonglobular (globular) region $S[k..l]$. Then $S[r..t]$ is called a *mutual region* of S, where $r = max\{i, k\}$ and $t = min\{j, l\}$. A segment of S is called an *obscure region* of S if it corresponds

TABLE 2. COMPARISON OF DSR SEGMENTATION AND EXPERIMENTAL DETERMINATION FOR COLLAGEN ALPHA 2(I) CHAIN [PRECURSOR]

DSR segmentation		*Experimental domains*		
Simple	*Complex*	*Non-globular*	*Globular*	*Comments*
	1-24		1-22	Signal peptide
25-1104		23-79		Amino-terminal propeptide (largely nonglobular)
		80-90		Telopeptide
		91-1102		Collagen alpha 2(i) chain (nonglobular)
	1105-1366		1103-1366	Globular carboxyl-terminal propeptide

TABLE 3. COMPARISON OF DSR SEGMENTATION AND EXPERIMENTAL DETERMINATION FOR MYOSIN HEAVY CHAIN, CARDIAC MUSCLE BETA ISOFORM

DSR segmentation		*Experimental domains*		
Simple	*Complex*	*Non-globular*	*Globular*	*Comments*
	1-836		1-838	Globular domain
837-1935		839-1935		Rodlike tail (S2 and LMM domains/Coiled coil (potential))

TABLE 4. COMPARISON OF DSR SEGMENTATION AND EXPERIMENTAL DETERMINATION FOR AGGRECAN CORE PROTEIN [PRECURSOR]

DSR segmentation		*Experimental domains*		
Simple	*Complex*	*Nonglobular*	*Globular*	*Comments*
	1-351		1-349	Globular domains G1-A, G1-B, G1-B'
352-480		350-476		Unknown structure
	481-674		477-672	Globular domains G2-B, G2-B'
675-2165		673-2162		Nonglobular extended domain
	2166-2415		2163-2415	Globular domain G3

to an unknown structural feature of protein. Let p denote total length of all mutual regions of S, and let q denote total length of all obscure regions of S. The *agreement accuracy* ω of DSR segmentation is defined as

$$\omega(S) = \frac{p}{L - q}.$$

For example, as shown in Table 4, the sequence S of aggrecan core protein [Precursor] has two *simple* segments: S[352..480] and S[675..2165], and three *complex* segments: S[1..351], S[481..674], and S[2166..2415]. On the other hand, S has one nonglobular region: S[673..2162], and three globular regions: S[1..349], S[477..672], and S[2163..2415]. Then, S has four mutual regions S[1..349], S[481..672], S[675..2162], and S[2166..2415], and one obscure region S[350..476]. In this case, $p = 2279$ and $q = 127$. Therefore, the agreement accuracy is $\omega = 99.6\%$. (See Table 5 for agreement accuracy of the three proteins.)

Now we switch our attention to considering the agreement accuracy of a sequence database. Let $\mathcal{D}$ denote a protein sequence database or a protein sequence family. Then the *agreement accuracy* $\Omega_{\mathcal{D}}(k, b, m)$ of DSR segmentation for $\mathcal{D}$ associated with parameters k (window length) and b (cutoff value), and the BLOSUM m matrix, is defined as the average sum of agreement accuracy values of all individual sequences in $\mathcal{D}$:

$$\Omega_{\mathcal{D}}(k, b, m) = \frac{1}{|\mathcal{D}|} \sum_{s \in \mathcal{D}} \omega(S).$$

To determine the best parameters k and b, and the best BLOSUM m matrix for $\mathcal{D}$, we have to mathematically find an "extreme point" (k^*, b^*, m^*) such that $\Omega_{\mathcal{D}}(k, b, m)$ achieves its maximum in (k^*, b^*, m^*):

$$\Omega_{\mathcal{D}}(k^*, b^*, m^*) = \max\{\Omega_{\mathcal{D}}(k, b, m) | k \in \mathcal{K}, b \in \mathcal{B}, m \in \mathcal{M}\}.$$

Based on our computational experiments and experience, it is good enough to test window sizes from 5, 6, to 100, the cutoff values from 0.01, 0.02, to 0.99, and the BLOSUM matrices from BLOSUM-30.0, BLOSUM-30.5, BLOSUM-31.0, BLOSUM-31.5, to BLOSUM-100.5. That is, we take $\mathcal{K} = \{5, 6, \ldots, 100\}$, $\mathcal{B} = \{n\% | \ n$ is an integer between 1 and 99$\}$, and $\mathcal{M} = \{30.0, 30.5, 31.0, 31.5, \ldots, 100.0, 100.5\}$. We have developed a software program for finding an extreme point (k^*, b^*, m^*) in the set $\mathcal{K} \times \mathcal{B} \times \mathcal{M}$ and calculating maximum $\Omega_{\mathcal{D}}(k^*, b^*, m^*)$ for $\mathcal{D}$.

Users can freely choose a protein functional family as a training set to which the protein sequence of interest belongs. In general, we choose the SWISS-PROT amino acid sequence database as a typical training set (Release 40.2 of 02-Nov-2001, *http://tw.expasy.org/sprot/*), which contains 102,164 sequence entries, comprising 37,554,368 amino acids abstracted from 92,269 references. Our computational experiments have shown that in this case the extreme point $(k^*, b^*, m^*) = (35, 0.46, 62.5)$ and the maximum $=$ 97.3%. In other words, the best parameters and the best BLOSUM matrix are, respectively, for automatic segmentation of amino acid sequences into nonglobular and globular regions by the DSR program (Wan and Wootton, 2002*a*):

Window length $k = 35$, cutoff value $b = 0.46$, the BLOSUM 62.5 matrix.

The average "agreement accuracy (sensitivity)" of DSR segmentation for the SWISS-PROT database is 97.3%

TABLE 5. AGREEMENT ACCURACY OF DSR SEGMENTATION

Protein	*Agreement accuracy*
Collagen alpha 2(I) chain [Precursor]	99.7%
Myosin heavy chain, cardiac muscle beta isoform	99.9%
Aggrecan core protein [Precursor]	99.6%

6. DISCUSSION

This paper has developed an efficient algorithm and a software program based on the reciprocal complexity measure which allows comparison of segments of different lengths. This novel method can be used as a powerful computational tool to find *simple* patterns associated with scoring schemes in DNA and protein sequences. In particular, it demonstrates utility for search masking and labeling of nonglobular domains based on the primary protein structure.

It is well known that 25% of biological sequences are compositionally biased and only a few of them are suitable for structure experiments. At the genomic level, many important genomes are biased, such as pathogens, parasites, and maize. Most proteins important for pathogenesis are compositionally biased and are not alignable. It is impossible to use traditional bioinformatics approaches, such as the BLAST search and sequence comparison, for finding "functional" hits from protein sequence databases for these compositionally biased sequences. The DSR program provides a useful bioinformatics tool for delineating functional and structural features of compositionally biased sequences.

The DSR method can be applied to classifying and analyzing segments of protein and nucleotide sequence databases associated with real-valued scores. We are continuing to use the program for showing compositional features of protein sequence databases associated with real-valued similarity scores. For instance, we use it to analyze realistic biological data related to similarity scores, or distances, between pairs of letters based on real-valued matrices (or ratios in some cases) as in PAM or BLOSUM matrices, or similarity in a physiochemical property such as hydrophobicity.

DSR can be utilized with every problem in which either the nucleic acid composition or the amino acid composition is important. For instance, one can mention the compositional difference between exons and introns, coding and noncoding regions, various dependences in DNA, and RNA composition associated with structural motifs, such as loops. DSR can also be used for detecting the compositional difference between α helices, β sheets, $\alpha+\beta$, α/β, domains, families, superfamilies, pathways, functional categories, whole genomes, and those genes or proteins in a biological network.

APPENDIX

A. *Comparison between random strings and biological sequences*

We generated 1,000 random sequences of length 300 to 600 over {A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y}. No significant similarity was found for all such sequences in BLASTP searches of the NCBI nonredundant databases of December 20, 2000. Actually, the average lowest BLASTP expectation value (E-value) for these 100 random sequences is 13.6, reported by BLASTALL, based on BLAST version 2.0 (Altschul *et al.*, 1997), in a search against the NCBI non-redundant databases. Here, we list only two representatives in FASTA format of 100 random sequences in the following. The first one has length 300 and the second has length 400. The lowest E-values are 11 and 13, respectively.

> random sequence 1

FLWIQALLDVPMVCRSILWAGFRLEWKMLCKRMWQSWDTPNADLDTTCWHRSNKTRWFVKVGDYSSHGLDVAEYCVK
VMWFFQIFLCEGMESIGRDPCICSFCMTEYTPMRTRYSYAVRVYVPIRYLHALIIGAWQSVLKISTRHPGWCYTAKA
IDYLKNWIYSIVLEHWNRHYKPNGEPFYQPYPRNCRDNCCIAWTEGHHNPHNVVTADPNSFDYMQCTTFVVNVKYQG
HNKMVYYIINLYRTSVSVMKQWGDTKNTKTQHKCKGPAFYNGMHCVSLRGLYFRQPSFAECQCCRLWHC

> random sequence 2

EVQKINDFHEMEPLYYRNHWELNTDFDEEAGHVMHTCKPYEDDKEDIVGQKLRNVKTANNATVMWSWYTNWYGQICI
HMPNMQGDNGMNVCEGMGFVFSQIHGFYPGHYTMDWFAASGDWRSQAVYTSSNKDVQHKWPHLAVNWCNNTVGHECY
EWWYQQERIPYRCVWIWMTMAVQEGDLNHDELAQCHYTPHANPCCDCYPLLGISKPVLTENYGFFWDEHRMHGRAHV
QWCDYHCRIHFTDYAQTFVHHQPAENVVNTHTHKWGRPPQWTMASDGPHRWFIETDHGPVQWGYHSGQIVHGQKVAM
DFKLSFTWPNFWEQNCMNAGKGQGEWMPKDRMISNDNAPFNKDRQGIEKYKTFCQKPSPAWWDGGRYVHNCVNSEFP
DICDSLWLSWCMMQA

B. The maximum of $C(S)$

We derive the maximum of $C(S)$ defined in expression (3) as necessary for normalization. We begin with some definitions. For any $x = (x_1, \ldots, x_n) \in \mathcal{R}^n$ where $\mathcal{R} = (-\infty, \infty)$, let

$$x_{[1]} \geq \cdots \geq x_{[n]}$$

denote the components of x in decreasing order, and let

$$x_{\downarrow} = (x_{[1]}, \ldots, x_{[n]})$$

denote the *decreasing rearrangement* of x. For $x, y \in \mathcal{R}^n$, we say y *majorizes* x and write it as $x \prec y$ or $y \succ x$, if

$$x_{[1]} + x_{[2]} + \cdots + x_{[k]} \leq y_{[1]} + y_{[2]} + \cdots + y_{[k]}, \quad (k = 1, 2, \ldots, n-1),$$
$$x_{[1]} + x_{[2]} + \cdots + x_{[n]} = y_{[1]} + y_{[2]} + \cdots + y_{[n]}.$$

This notation and terminology was introduced by Hardy *et al.* (1952). It has been shown to have many applications to problems in the field of computational biology and bioinformatics (Wan and Wootton, 1999).

A real-valued function ϕ defined on a set $\mathcal{A} \subset \mathcal{R}^n$ is said to be *strictly monotonic* if it is *strictly increasing*, i.e., $x \prec y$ implies $\phi(x) < \phi(y), x, y \in \mathcal{A}$; or it is *strictly decreasing*, i.e., $x \prec y$ implies $\phi(x) > \phi(y), x, y \in \mathcal{A}$. The monotonic functions are a class of significant functions and fulfill a remarkable role in axiomatic construction of complexity measures of biological sequences on an integral partition lattice (Wan and Li, 1986; Wan and Wootton, 1999). Further explanations of these terms are found in Wan (1986). To investigate the monotonicity of the function $C(S)$, we need the following propositions.

Proposition 1 (Wan, 1986). *If $I \subset \mathcal{R}$ is an interval and $f : I \to \mathcal{R}$ is strictly concave, then*

$$\phi(x) = \sum_{i=1}^{n} f(x_i)$$

is strictly decreasing on I^n.

Based on Proposition 1, we prove that the complexity function $C(S)$ is a strictly decreasing function of n variables: $u_1, \ldots, u_n$, where $U =: (u_1, \ldots, u_n)$ is the normalized frequency vector of S.

Proposition 2. *Let*

$$F(x) = -\sum_{i=1}^{n} x_i \log x_i$$

be a function with domain D:

$$D = \{x = (x_1, \ldots, x_n) \mid x_1 + \cdots + x_n = 1, x_i \geq 0, 1 \leq i \leq n\}.$$

Then $F(x)$ is strictly decreasing on D and achieves the maximum value $\log n$ *at the point* $x^0 =: (\frac{1}{n}, \ldots, \frac{1}{n})$.

Proof. Let $f(t) = -t \log t$. It is easy to see that $f(t)$ is strictly convex on the unit interval $[0, 1]$. Thus, by Proposition 1, $F(x)$ is strictly decreasing on D. Since x^0 is the bottom element in D under the partial order relation $\prec$, i.e., $x \succ x^0$ for any $x \in D$ (Wan and Li, 1986), we have

$$F(x) \leq F(x^0) = \log n.$$

In other words, $F(x)$ has the maximum value $\log n$ at the point x^0. This completes the proof of Proposition 2. ■

We can also apply the method of Lagrange multipliers to find the maximum of $F(x)$ subject to $g(x) =: \sum_{i=1}^{n} x_i - 1 = 0$.

It is clear that $C(S)$ can be rewritten as

$$C(S) = e^{lF(\frac{v_1}{L},\ldots,\frac{v_n}{L})}.$$

By Proposition 2,

$$F\left(\frac{v_1}{L}, \ldots, \frac{v_n}{L}\right) \leq \log n.$$

Thus,

$$C(S) \leq n^L.$$

Obviously, $C(S)$ achieves the maximum value n^L whenever the components of the frequency vector $V = (v_1, v_2, \ldots, v_n)$ of S are equal: $v_1 = v_2 = \cdots = v_n$.

ACKNOWLEDGMENTS

We are very grateful to the anonymous referees for many valuable comments. This work was supported in part by NIH grant R01-GM0002511 and US National Research Council grant 971066.

REFERENCES

Altschul, S.F. 1991. Amino acid substitution matrices from an information theoretic perspective. *J. Mol. Biol.* 219, 555–565.

Altschul, S.F., Boguski, M.S., Gish, W., and Wootton, J.C. 1994. Issues in searching molecular sequence databases. *Nature Genet.* 6, 119–129.

Altschul, S.F., Madden, T.L., Schaffer, A.A., Zhang, J., Zhang, Z., Miller, W., and Lipman, D.J. 1997. Gapped BLAST and PSI-BLAST: A new generation of protein database search programs. *Nucl. Acids Res.* 25, 3389–3402.

Dayhoff, M.O., Schwartz, R.M., and Orcutt, B.C. 1978. A model of evolutionary change in proteins. *Atlas of Protein Sequence and Structure* 5, 345–352.

Finkelstein, A.V. 1994. Implications of the random characteristics of protein sequences for their three-dimensional structure. *Curr. Opin. Struct. Biol.* 4, 422–428.

Gibbs, A.J., and McIntyre, G.A. 1970. The diagram: A method for comparing sequences: Its use with amino acid and nucleotide sequences. *Eur. J. Biochem.* 16, 1–11.

Hardy, G.H., Littlewood, J.E., and Polya, G. 1952. *Inequalities*, Cambridge University Press, London.

Henikoff, S., and Henikoff, J.G. 1992. Amino acid substitution matrices from protein blocks. *Proc. Natl. Acad. Sci. USA* 89, 10915–10919.

Milosavljevic, A., and Jurka, J. 1993. Discovering simple DNA sequences by the algorithmic significance method. *Comput. Appl. Biosci.* 9, 407–411.

Salamon, P., and Konopka, A.K. 1992. A maximum entropy principle for distribution of local complexity in naturally occurring nucleotide sequences. *Comput. Chem.* 16, 117–124.

Shannon, C.E. 1948. A mathematical theory of communication. *Bell Sys. Tech. J.* 27, 379–423, 623–656.

Staden, R. 1984. Graphic methods to determine the function of nucleic acid sequences. *Nucl. Acids Res.* 12, 521–538.

Tomii, K., and Kanehisa, M. 1996. Analysis of amino acid indices and mutation matrices for sequence comparison and structure prediction of proteins. *Protein Eng.* 9, 27–36.

Wan, H. 1986. *Combinatorial and Computing Theory of Nonnegative Integral Matrices Associated with Majorization*, Dalian University of Technology Press, Dalian.

Wan, H., and Li, Q. 1986. On the number of tournaments with prescribed score vector. *Discrete Math.* 61, 213–219.

Wan, H., and Song, E. 2002. Quasiperiodic biosequences and modulo incidence matrices. *Proc. IEEE Conference on High Performance Comput. Bio.*, 1–8.

Wan, H., and Wootton, J.C. 1999. Axiomatic foundations of complexity functions of biological sequences. *Ann. Comb.* 3, 105–127.

Wan, H., and Wootton, J.C. 2000. A global compositional complexity measure for biological sequences: AT-rich and GC-rich genomes encode less complex proteins. *Comput. Chem.* 24, 71–94.

Wan, H., and Wootton, J.C. 2002a. Algorithms for computing lengths of chains in integral partition lattices. *Theoretical Computer Sci.* 289, 783–800.

Wan, H., and Wootton, J.C. 2002b. *The points of contact between globular and non-globular domains in protein sequences.* In preparation.

Wootton, J.C. 1994. Sequences with 'unusual' amino acid compositions. *Curr. Opin. Struct. Biol.* 4, 413–421.

Wootton, J.C., and Federhen, S. 1993. Statistics of local complexity in amino acid sequences and sequence databases. *Comput. Chem.* 17, 149–163.

Wootton, J.C., and Federhen, S. 1996. Analysis of compositionally biased regions in sequence databases. *Methods Enzymol.* 266, 554–571.

Address correspondence to:
Honghui Wan
National Center for Genome Resources
2935 Rodeo Park Drive East
Santa Fe, NM 87505

E-mail: hw@ncgr.org

This article has been cited by:

1. S MAHONY, P BENOS, T SMITH, A GOLDEN. 2006. Self-organizing neural networks to support the discovery of DNA-binding motifs. *Neural Networks* **19**:6-7, 950-962. [CrossRef]
2. YURIY L. ORLOV, RENE TE BOEKHORST, IRINA I. ABNIZOVA. 2006. STATISTICAL MEASURES OF THE STRUCTURE OF GENOMIC SEQUENCES: ENTROPY, COMPLEXITY, AND POSITION INFORMATION. *Journal of Bioinformatics and Computational Biology* **04**:02, 523-536. [CrossRef]
3. Shaun Mahony, David Hendrix, Terry J. Smith, Aaron Golden. 2005. Self-Organizing Maps of Position Weight Matrices for Motif Discovery in Biological Sequences. *Artificial Intelligence Review* **24**:3-4, 397-413. [CrossRef]
4. S. Mahony, D. Hendrix, A. Golden, T. J. Smith, D. S. Rokhsar. 2005. Transcription factor binding site identification using the self-organizing map. *Bioinformatics* **21**:9, 1807-1814. [CrossRef]