The Shannon Information Entropy of Protein Sequences

Bonnie J. Strait and T. Gregory Dewey
Department of Chemistry, University of Denver, University Park, Denver, Colorado 80208 USA

ABSTRACT A comprehensive data base is analyzed to determine the Shannon information content of a protein sequence. This information entropy is estimated by three methods: a *k*-tuplet analysis, a generalized Zipf analysis, and a "Chou-Fasman gambler." The *k*-tuplet analysis is a "letter" analysis, based on conditional sequence probabilities. The generalized Zipf analysis demonstrates the statistical linguistic qualities of protein sequences and uses the "word" frequency to determine the Shannon entropy. The Zipf analysis and *k*-tuplet analysis give Shannon entropies of approximately 2.5 bits/amino acid. This entropy is much smaller than the value of 4.18 bits/amino acid obtained from the nonuniform composition of amino acids in proteins. The "Chou-Fasman" gambler is an algorithm based on the Chou-Fasman rules for protein structure. It uses both sequence and secondary structure information to guess at the number of possible amino acids that could appropriately substitute into a sequence. As in the case for the English language, the gambler algorithm gives significantly lower entropies than the *k*-tuplet analysis. Using these entropies, the number of most probable protein sequences can be calculated. The number of most probable protein sequences is much less than the number of possible sequences but is still much larger than the number of sequences thought to have existed throughout evolution. Implications of these results for mutagenesis experiments are discussed.

INTRODUCTION

A basic tenet of the protein folding problem is that the information contained in the primary sequence is sufficient to dictate the three-dimensional structure. From an information theoretical point of view, protein folding can be envisioned as a communication process by which the sequence information is transmitted to the three-dimensional structure. There are a number of questions one can ask about such information transfer: How much information is transferred from sequence to structure? How redundant is the information? Is information transfer via protein folding a "noisy" or "noiseless" communication channel? To approach such questions, one must first establish, in a quantitative fashion, what the total information content of a protein sequence is. Additionally, one would like to develop techniques to describe the information in a three-dimensional structure, the folded protein. The goal of the present work is to establish the information content of the sequence. In a separate work, we address the issue of the information content of a protein's three-dimensional structure (Dewey, 1996).

Since the pioneering work of Gatlin (1972) on nucleic acid sequences, there has been a large body of literature on information theory and sequence data. Work on protein sequences has been discussed by Yockey (1977, 1992). Information theory approaches has also been extensively used in discussions of molecular evolution (Yockey, 1992; Eigen, 1992; Kauffman, 1993; Volkenstein, 1994). Despite this body of work, an accurate measure of the information content of a protein remains elusive. This is because the sequence libraries studied to date are simply too small to yield accurate estimates. A similar problem arises in determining the information content of the English language (Shannon, 1951). Even in this case, the library is too small to gain accurate estimates of the Shannon information entropy. Following the example of Shannon (1951), we devise a number of approaches to circumvent the small library problem. As will be seen, we obtain similar information entropies for protein sequences using three very different methods. This provides added confidence in the results.

The implications of the information content of a sequence for molecular evolution have been discussed extensively and are worth noting. Generous scenarios for molecular evolution suggest that the number of protein sequences that could have existed throughout terrestrial history is far less than the number of possible sequences (Eigen, 1992; Kauffman, 1993; Yockey, 1992). For a small protein that is 100 amino acids long, there are 20^{100} or 10^{130} possible sequences. There have been numerous estimates of the number of proteins that could have existed throughout molecular evolution. One such estimate places this number between 10^{40} and 10^{50} proteins (Eigen, 1992). This difference suggests that only very small regions of "sequence space" have been explored. This implies that the requirement for generating a functional protein is not stringent and has prompted the concept of a protein as a "slightly edited random polymer" (Volkenstein, 1994). This idea has been supported by numerical simulations of the folding of model polymer systems that consist of random hydrophobic sequences (Shakhnovich and Gutin, 1993). However, recent evidence shows that proteins are not completely random and that correlations do exist within sequences (Pande et al., 1994; Balafas and Dewey, 1995; Strait and Dewey, 1995; Dewey and Strait, 1996). Such correlations may point to a physical

Received for publication 15 March 1996 and in final form 16 April 1996.
Address reprint requests to T. Gregory Dewey, Department of Chemistry, University of Denver, University Park, Denver, CO 80208. Tel.: 303-871-3100; Fax: 303-871-2254; E-mail: gdewey@du.edu.

0006-3495/96/07/148/08 $2.00

driving force for the "editing" process (Pande et al., 1994). Doubtless, there are genetic driving forces as well. Such influences result in sequences that have nonrandom distributions.

The enormous disparity between the number of possible sequences and the number that could have existed has important implications for both molecular evolution and the protein folding problem. However, this comparison is not so meaningful because it neglects the information content of protein sequences. An analogy can be made with the astronomical number of letter sequences that can be generated from the English alphabet. This number has little significance for the structure and meaning of the language. In the present work, a quantitative description of the "number of most probable protein sequences" is made using a fundamental theorem of information theory. Shannon showed that the number of most probable messages of a source, $\langle\Omega\rangle$, could be calculated from the information entropy, I, according to (Shannon and Weaver, 1962)

$$\langle\Omega\rangle = 2^{NI}, \tag{1}$$

where N is the number of letters in the text and I is the information content in binary digits (bits) per letter. For a protein "text," Eq. 1 can be represented in the convenient form $\langle\Omega\rangle = 20^{0.2314NI}$, which allows direct comparison with the total number of sequences, Ω, given by $\Omega = 20^N$. Equation 1 requires that the messages be ergodic, and this assumption is made for protein sequences (cf. Dewey and Strait, 1996). For a determination of the number of most probable sequences, an accurate value of I must be obtained from protein sequence data.

To calculate the Shannon information entropy, two representative data sets of 190 and 155 proteins were considered. Selection of the data set is very important when considering sequence statistics. One hopes to cover the protein data bank to obtain a representative sampling of a variety of proteins. At the same time, it is important not to include too many proteins because of possible sequence redundancies. The data sets used in this work were generated from two different algorithms whose aim was to maximize the coverage while minimizing redundancy (Hohohm et al., 1992). The first algorithm gave a data base of 195 "representative" proteins, and the second one gave 155 proteins. Although larger data bases of sequences exist, the data sets from the Hohohm work have the advantage of having readily accessible x-ray structures. This allows us to use a context-based guessing algorithm (see Chou-Fasman Gambler, below).

A *k*-TUPLET ANALYSIS

Using the concatenated sequences of these two data bases, the kth order information entropy, I_k, is determined from (Shannon and Weaver, 1962)

$$I_k = -\sum_{i=1}^{20}\sum_{s=1}^{20^{k-1}} p_i p(i|s)\log_2 p(i|s), \tag{2}$$

where p_i is the probability of finding the ith amino acid and $p(i|s)$ is the conditional probability of the ith amino acid occurring after the s string. The k-tuplet entropies were determined using Eq. 2 and are shown in Table 1. For comparison, the entropies calculated for the English language are also included (Shannon, 1951; Shannon and Weaver, 1962). The intent here is not to suggest any analogies; rather, it is to show the generality of the methods. Although these statistical approaches were first applied to language problems, their mathematical basis is more general (Cover and Thomas, 1991). As can be seen, in both cases the entropies decrease with increasing string size. This is a result of the calculation, capturing more of the long-range order of the message. It should also be pointed out that I_1 is the information entropy based on the nonuniform distibution on amino acids. Its value of 4.18 is sometimes quoted as the information content of an amino acid in a protein.

The information entropy, I_k, as defined in Eq. 2 is a conditional information entropy. Shannon rigorously showed (Shannon and Weaver, 1962) that

$$I = \lim_{k\to\infty} I_k. \tag{3}$$

The information entropy, I, can then be determined from the limiting behavior of the conditional entropy. Often Markov models are used in describing information transfer, and it is

TABLE 1 Information entropy of protein sequences from frequency distributions

Method	I_1	I_2	I_3	I_4
Data set I				
k-tuplet analysis	4.18	4.16	3.99	2.36
Zipf plot*	—	2.72 (2.58)	2.54 (2.44)	2.61 (2.57)
Drop and search algorithm#	4.19	4.15	3.99	2.48
Drop and search algorithm§	4.19	4.15	3.97	2.31
Data set II				
k-tuplet analysis	4.17	4.16	3.96	2.20
Zipf plot*	—	2.49 (2.39)	2.42 (2.34)	—
Drop and search algorithm#	4.18	4.14	4.00	2.59
Drop and search algorithm§	4.18	4.14	3.97	2.32
English language				
k-tuplet analysis	4.14	3.56	3.30	—

All information entropy are in bits/symbol. Data set I contains 190 protein sequences (37,101 amino acids). Data set II contains 155 protein sequences (30,309 amino acids). See Hohohm et al. (1992) for specific proteins chosen. Sequences were obtained from the Brookhaven Protein Data Bank. English language data are from Shannon and Weaver (1962).

*Normal entries are determined from the sums of word frequencies. Entries in parenthesis are determined using Eq. 5 and the slope of the Zipf plot and assuming a cut-off rank.

#A sequence of 1 million amino acids was generated. The computer's random number generator was reseeded every 1,000th iteration.

§A sequence of 100,000 amino acids was generated.

helpful to relate Eq. 2 to such models. Markov chains are discussed in more detail in Appendix A. For a stationary Markov process of N steps or independent variables, X_i, one has (Cover and Thomas, 1991): $I(X_1, \ldots, X_N) = NI(X_1)$. It can also be shown for such a process that $I(X_1) = \lim_{k \to \infty} I(X_k|X_{k-1}, \ldots, X_1) = I_k$, which establishes Shannon's result (Eq. 3) for this special case. For Markov processes, one can also show that $I_k \leq I_{k-1}$. This is a commonly observed condition but is not sufficient to establish that the process is Markovian.

The problem with the k-tuplet analysis is that an enormously large text is required to reach a limiting entropy. In the fifth-order entropy, there are 3.2×10^6 possible quintuplets, and the number of amino acids in the data set is not sufficient to provide accurate probabilities. Consequently, the value of the entropy drops precipitously as a result of the finite size of the data set. Thus, one could not have full confidence in I_4 being the "limiting value" (see top line, Table 1). To circumvent such problems, two different devices are used: the "drop and search" algorithm described in this section and the Zipf analysis described in the following section.

The "drop and search" algorithm was devised to generate an arbitrarily large sequence from the protein data base. This algorithm is illustrated in Fig. 1. In this method, a random number generator chooses a single amino acid out of the concatenated sequence. A second random location in the array is chosen, and the position marker is moved, right or left, at random until an amino acid is found that is identical to the previously generated one. Upon locating this amino acid, the neighbor to the right is added to the computer-generated sequence. This process is repeated until a sequence of the desired length is generated. This approach preserves the statistics of the original sequence while allowing large, new sequences to be generated. In Appendix A the mathematical basis of this algorithm is discussed in more detail.

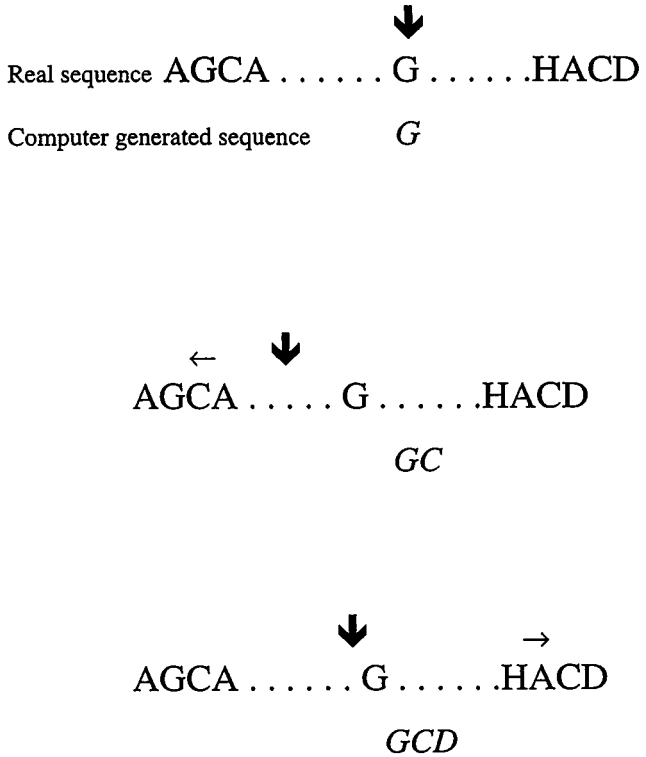

FIGURE 1 Diagram illustrating the "drop and search" algorithm used to generate large protein sequences. The original data bank of sequences was ordered on an array (bold letters). Using a random number generator, a single amino acid out of the sequence was chosen (*top*). A second random location in the array is chosen (thick downward arrow), and the marker is moved, right or left (thin arrow), at random until the last generated amino acid is located. When the last amino acid is located, the neighbor to the right is added to the growing sequence (italicized letters). This approach preserves the statistics of the original sequence while allowing large, new sequences to be generated.

The larger sequences generated by the "drop and search" method provide for more accurate determinations of entropies of higher order k-tuplets. The results of the k-tuplet analysis for these computer-generated sequences are shown in Table 1, where data for sequences of 100,000 and 1 million amino acids are compared. For these larger sequences, the value of information entropy is not strongly dependent on the sequence size, indicating that the data set is large enough to give reliable results. At this stage, it is still not clear whether I_4 is a limiting entropy, but it is independent of sequence length. This indicates that this estimate has not been effected by the finite size of the sequence. As can be seen, the drop and search algorithm gives quadruplet Shannon entropies in the range of 2.3–2.6 bits/amino acid; these are approximately the same as those determined for the original protein data base.

GENERALIZED ZIPF ANALYSIS

In addition to using letter frequencies, one can also use word frequencies to calculate the information entropy. In this case, the information entropy per letter of words of length k is given by $I_k = (-1/k)\Sigma_i p_i \ln p_i$, where p_i is the probability of finding a given word and the sum is over all possible words. Linguistic texts have been shown to follow the Zipf law (Mantegna et al., 1995):

$$p(R) \sim \frac{1}{R^\alpha}, \tag{4}$$

where R is the rank or position of the word in a list ordered according to word frequency. For most natural languages the exponent α is 1. For "words" created from letter sequences, $0 \leq \alpha \leq 1$. The Zipf plot for amino acid triplets in proteins for data set I (195 proteins) is shown in Fig. 2. For comparison, a random sequence of amino acids of the same composition was created and its Zipf plot is shown. A Zipf plot is a log-log plot of frequency of occurrence versus rank. As can be seen, the protein sequence has linguistic features, i.e., Eq. 4 is followed for a significant portion of the plot. The random sequence shows considerable curvature and at low rank appears to have a limiting α of zero.

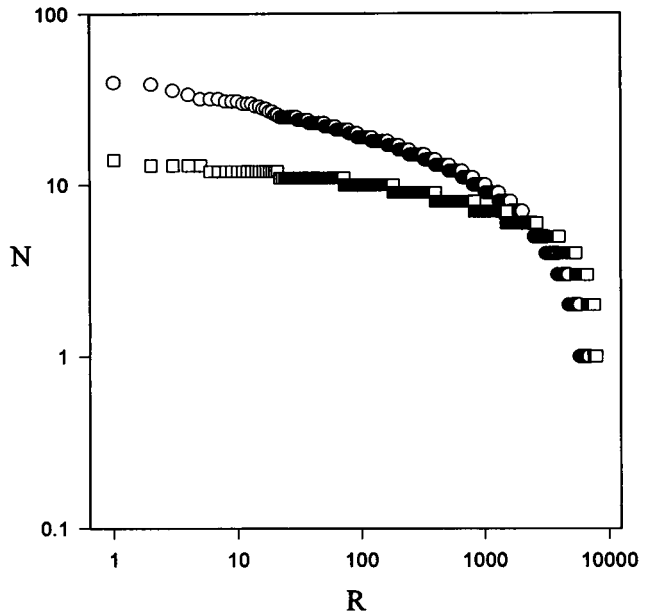

FIGURE 2 Zipf plot showing the number of occurrences of all amino acid triplets occurring in data set I (see Table 1) versus the rank. This log-log plot shows a linear power law scaling region (see Eq. 3) for protein sequences (○) with an α of 0.18 determined from the slope. A randomly generated sequence of the same amino acid composition does not show such behavior (□).

Fig. 3 shows Zipf plots for three different word lengths; doublets, triplets, and quartets for data set 1. Similar behavior is seen for data set 2 (155 proteins). It is interesting to note that whereas Zipf behavior breaks down at different ranks for the three different word lengths, the linear regions in Fig. 3 all have similar slopes. The slope in the linear regions of these plots give an α value of approximately 0.18. The generalized Zipf law (Eq. 4) is seen in other "linguistic texts" (Mantegna et al., 1995). Typical α values are 1.0 for English words, 0.57 English letter sequences, 0.77 UNIX binary code, and 0.34 DNA base sequences. For the calculation of information entropies, Eq. 4 can be taken as an empirical observation. However, it should be noted that expressions of this form can be derived for Markov processes (Mantegna et al., 1995; Brillouin, 1962). Such models show similar regions of curvature as well. It is possible to derive approximate expressions for α in terms of the parameters of such models. Unfortunately, processes involving long-range correlations also generate similar Zipf plots. Thus, it is difficult to discriminate models on the basis of Zipf behavior. For the present purposes, the Zipf behavior is merely used as an empirical description that provides a convenient probability function (Eq. 4).

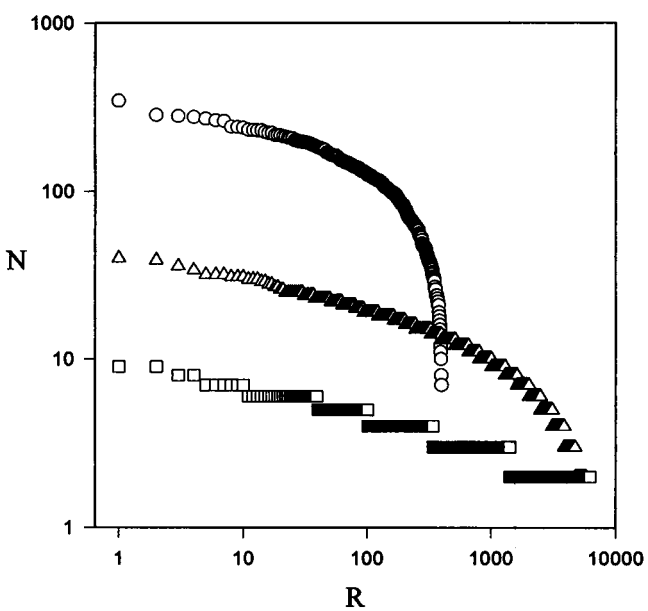

FIGURE 3 Zipf plot showing the number of occurrences of all amino acid doublets (○), triplets (△), and quadruplets (□) versus rank occurring in Data Set I. The rank is the ordering from large to small number of occurrences of all "words." Each data set has similar slopes at low rank (high occurrence frequency) and follows the same form for the Zipf equation (Eq. 3).

Information entropy is determined directly from word frequencies, and these results are given in Table 1 for different length words. Note that the region that deviates from linearity on the Zipf plot has low word frequencies and, therefore, makes a small contribution to the sum over all frequencies required to calculate the Shannon entropy. Unlike the k-tuplet analysis and the drop and search algorithm, the Zipf plots give results that are independent of string size. These entropies are approximately 2.5 bits/amino acid and agree well with the values of I_4 determined from the k-tuplet and "drop and search" algorithm. With the Zipf analysis, a limiting value is achieved at low word size. Its agreement with I_4 for the k-tuplet analysis suggests that a limiting value may indeed have been reached in this later case.

The Shannon entropy can also be calculated by assuming a probability of the form of Eq. 4 and using the fitted value for α obtained from the linear region of the Zipf plot. To use this approach, one must define a cutoff rank, R_c, to normalize the resulting sums. The probability function is now defined as

$$p(R) = R^{-\alpha} \Big/ \sum_{R=1}^{R_c} R^{-\alpha}. \tag{5}$$

The Shannon entropy is given by

$$I = -\sum_{R=1}^{R_C} p(R)\log_2 p(R) \approx \log_2 R_c - \frac{\alpha}{1-\alpha}\ln_2 e, \tag{6}$$

where the second equality in Eq. 6 holds for large R_C. The cutoff, R_C, is taken as the rank at which deviation from Zipf behavior occurs and which is the point where the Zipf plots curve over. The values calculated for the Shannon entropies determined from Eq. 6 are given in Table 1 (numbers in parentheses). Good agreement is seen between these values and those obtained from directly summing the word frequencies.

CHOU-FASMAN GAMBLER

Gambling and guessing strategies have played an important role in information theory since the theory's inception. In Shannon's work in the 1950s, he used human subjects to better estimate the probability distribution of letters in the English language (Shannon, 1951; Shannon and Weaver, 1962). In this experiment, the subject was presented with an English text and asked to guess the next letter. An optimal subject will guess the most probable letter first and the second most probable letter second and so on. From the number of guesses it took to guess the correct letter, one can calculate the empirical frequency distribution. This allows the information entropy to be determined. Using this method, Shannon obtained a value of 1.3 bits/letter for the information entropy of the English language. This is significantly smaller than the 2.8 bits/letter obtained from a k-tuplet analysis and the 1.7 bits/letter from the Zipf analysis. Gambling strategies also result in low information entropies. In these experiments, the subject essentially assigns a probability or likelihood to each guess in the form of a wager. The number of guesses is then weighted by these probabilities. This approach also yields an entropy for the English language of approximately 1.3 bits/letter.

In this section, a gambling algorithm is presented that is based on the Chou-Fasman rules for secondary structure prediction (Chou and Fasman, 1978). The Chou-Fasman parameters can be used to assign probabilities to each guess, and therefore, this is a gambling, rather than a guessing, algorithm. The use of gambling strategies is accepted in information theory (Cover and Thomas, 1991) as a valid means of estimating information entropy. The implementation of such algorithms does not imply that a sequence or "text" has similarities to the English language. Rather, it should be viewed as a purely mathematical device that yields the information content of the sequence. Gambling algorithms reflect the context of the sequence more than simple statistical analysis of letter or word frequencies. A brief mathematical description of the use of gambling algorithms to determine information content is presented in Appendix B.

One could use more recent and sophisticated sets of rules than Chou-Fasman (cf. Srinivasan and Rose, 1995). In the present work, we restrict ourselves, for simplicity, to the Chou-Fasman rules. Even within this set of rules, gambling strategies of different levels of detail can be developed. Again, for simplicity, we consider a fairly simple strategy. This initial example will allow a first estimate of "context-based" information in protein sequences. As higher level rules and strategies are developed, one would anticipate further restrictions on the number of possible guesses for an amino acid in a given location. This will have the effect of reducing the information entropy. Thus, our present results must be viewed as upper limits to the information entropy that can be obtained by a gambling strategy.

To implement the Chou-Fasman gambler, an amino acid is chosen randomly from our protein data base. The amino acids on the amino-terminal side of the chosen residue are examined. The goal is to develop a gambling strategy to correctly guess, given the preceding sequence, what amino acid will follow. With the Chou-Fasman rules, a guess is made whether the next amino acid (carboxy side) will be an α helix, β sheet, or β turn. With this guess, the specific amino acid is "bet on" using Chou-Fasman stakes, i.e., the probability distribution of amino acids in a given secondary structural category is used. For an α helix, this will be given by

$$\Omega_\alpha = 2^{I(aa|\alpha)}, \tag{7}$$

where Ω_α is the average number of guesses required to give the proper amino acid in an α helix and $I(aa|\alpha)$ is the conditional information entropy for an amino acid given that it is in an α helix. This latter quantity is given by

$$I(aa|\alpha) = \sum_{i=1}^{20} p(\alpha)p(i|\alpha)\ln p(i|\alpha), \tag{8}$$

where the sum is over all 20 amino acids, $p(\alpha)$ is the probability of finding an α helical amino acid in the sequence, and $p(i|\alpha)$ is the conditional probability of finding the ith amino acid given that it is an in an α helix.

In Chou and Fasman's original work, p(α) is determined. However, the conditional probability of finding a helix given a specific amino acid, $p(\alpha|i)$, is calculated and not $p(i|\alpha)$. To use Eq.8, one assumes the Bayesian relationship for the joint probability:

$$p(i, \alpha) = p(\alpha)p(i|\alpha) = p(i)p(\alpha|i). \tag{9}$$

The probability of an amino acid occurring in a sequence, $p(i)$, is determined from the composition of our data base of 195 proteins. The remaining probabilities, $p(\alpha|i)$, are given in Chou and Fasman (1978). With Eq. 9, $p(\alpha|i)$ is determined and $I(aa|\alpha)$ can be calculated.

Using equations analogous to Eqs. 7–9, one can determine the parameters for β sheets and β turns. These calculations give $I(aa|\alpha) = 1.56$, $\Omega_\alpha = 2.95$; $I(aa|\beta) = 0.82$, $\Omega_\beta = 1.77$; and $I(aa|\beta\mathrm{T}) = 1.29$, $\Omega_{\beta\mathrm{T}} = 2.45$, where α, β, and βT stand for α helix, β sheet, and β turn, respectively, and all conditional entropies are in units of bits/amino acid. A random coil entropy is not used in the calculation because when it is not possible to guess the secondary structure of the next amino acid, the probability distribution of all amino acids is used. This corresponds to the use of I_1 from the k-tuplet analysis. This value is 4.18, giving $\Omega_1 = 18.1$.

Returning to the algorithm, sites in the concatenated sequence of 195 proteins are chosen at random. The previous amino acids on the amino side of the chosen site are examined, and the Chou-Fasman rules are used to guess the secondary structure of the first amino acid to the right (carboxy side) of the chosen site. If this guess is an α helix, the weighted probability guess will be given by Ω_α. On average, it will take 2.95 guesses before the right amino acid

is found. This process is repeated N times to generate a statistical distribution. The total number of guesses on average is

$$\Omega(N) = \prod_{i=1}^{N} \Omega_i, \tag{10}$$

where Ω_i is one of four choices: Ω_α, Ω_β, $\Omega_{\beta T}$, Ω_1. The information entropy in bits/amino acid determined by the Chou-Fasman gambler is now given by

$$I_{CF} = \frac{\log_2 \Omega(N)}{N}. \tag{11}$$

Note that the "gambling payoff" has not been explicitly introduced into the present treatment (cf. Cover and Thomas, 1991), as it adds nothing to the problem.

To generate the sequence in Eq. 10, the following rules are used. Note that the algorithm differs somewhat from the Chou-Fasman rules because the x-ray structure of all of the proteins in the data base is known. Because the secondary structure of the preceding amino acids is determined from the $\Phi\Psi$ angles, the rules can be simplified somewhat. Furthermore, the goal is to predict the next amino acid from the secondary structural setting. Chou-Fasman was designed to predict secondary structure from sequence. The rules are as follows:

1. If four of the preceding six residues are α helical, the following amino acid will also be α helical and Ω_α will be used. However, if the immediately preceding amino acid is an α-tetrapeptide breaker and is not in an α state, the helix is broken. Furthermore, prolines cannot occur in the inner helix at the C-terminal end.

2. If three of five preceding residues are β sheets, the following residue is guessed to be a β sheet and Ω_β is used. If the immediately preceding amino acid is a β-tetrapeptide breaker and is not in a β state, the sheet is terminated.

3. When the preceding region contains both α- and β-forming residues that are not inconsistent with rules 1 and 2, then the average Chou-Fasman propensities of the region are used to make a choice.

4. When the amino acid is preceded by three or fewer β turns, it is assumed to be a β turn. In this case, $\Omega_{\beta T}$ is used.

5. When no prediction can be made from the above rules, the amino acid is guessed according to the compositional distribution, i.e., Ω_1 is used.

With these rules, the entropy was calculated for a large number (1,000–100,000) sites chosen at random in the concatenated sequence. Any Monte Carlo experiment above 1,000 sites gave comparable entropies of 2.0 bits/amino acid. As in the case of the English language, the gambling algorithm gives a much lower entropy than k-tuplet analyses. These methods incorporate knowledge of the structure and correlations in the sequence and, therefore, give better guesses. Any single amino acid carries less information because it is intrinsically more predictable. This lower entropy should be considered a more realistic estimate of the information content of the protein sequence because it incorporates existing knowledge on relationships between sequence and structure. The estimates based on statistical distributions (k-tuplet and Zipf) do not incorporate any a priori knowledge of protein sequences and therefore do not fully capture the information content. In principle, these methods would yield the same result as large string lengths are examined, but this would require a prohibitively large data base.

The gambler method is similar in spirit to the one presented by Yockey (1977, 1992). In this previous approach, a large number of mutants of cytochrome c were examined and chemical intuition was used to assign the number of possible amino acids that could occur in a given site. This algorithm does not use as sophisticated a set of rules as Chou-Fasman and gives a higher number of guesses per position. The entropy calculated from an equation of the form of Eq. 11 was 2.95 bits/amino acid. Because only a single sequence was considered, the entropy determined in this previous work is really the conditional entropy. It represents the conditional entropy of an amino acid given that it must exist in the three-dimensional structure of cytochrome c. The conditional entropy cannot be greater than the sequence entropy. The fact that the cytochrome c number (2.95 bits/amino acid) is higher than the present sequence entropies (2.5 bits/amino acid) suggests that a finite size problem may occur when considering mutagenesis data for a single protein sequence.

DISCUSSION

From the combined Zipf and k-tuplet analysis, the limiting information entropy of a protein sequence is seen to range from 2.4 to 2.6 bits/amino acid. Using the Chou-Fasman gambler algorithm, a somewhat smaller entropy of 2.0 bits/amino acid was obtained. For a protein on 100 amino acids, there are on average 200–250 bits of information contained in the sequence. Is this enough information to dictate the final structure of a protein? Certainly it would take more than 200 bits of information to encode all of the $\Phi\Psi$ angles of the peptide backbone. However, one can "encode" a protein structure more efficiently than this. Recently, the information content of the three-dimensional structure of a protein was estimated by calculating the Kolmogorov entropy (Dewey, 1996). The Kolmogorov entropy or algorithmic complexity is defined as the length in bits of the shortest computer program required to describe a given object. In previous work (Dewey, 1996), it was argued that the Kolmogorov entropy of a protein was less than 1.0 bits/amino acid. Thus, it is seen that there is more than enough information available in the protein sequence to determine the protein structure. Another, perhaps more interesting, parameter is the amount of shared information between sequence and structure. Work on determining the shared or mutual information is currently under way.

It is interesting to consider the implications of these results for molecular evolution. Assuming a value of 2.5 for I, the

number of most probable protein sequences is given by $\langle\Omega\rangle = 20^{0.5785N}$. For a protein with $N = 100$, there are 10^{73} most probable messages, far less than the 10^{130} possible messages. The probability of randomly finding a "most probable" sequence in sequence space is 10^{-57}. Thus, enormous regions of sequence space will not contain a most probable protein. Scenarios have been developed for massive screening of proteins in the hope of finding new proteins (Kauffman, 1992). These results would suggest that such efforts are unlikely to succeed. The number of proteins that could possibly have existed throughout evolution has been estimated at 10^{40}-10^{50} (cf. Eigen, 1992). With these estimates, the fraction of "most probable" sequence space that has been explored by evolution is 10^{-33}-10^{-23}. The conclusion is that there are many more "most probable" sequences that have not been explored in sequence space, but unfortunately they are embedded in a vaster space of all possible protein sequences.

Although these results may have important implications for molecular evolution, they are also of potential use in designing mutagenesis experiments. By using cassette mutagenesis, random combinations of mutations can be introduced at two or three positions in a region of a protein. The number of acceptable mutations in the DNA binding region of the λ repressor has been explored (Reidhaar-Olson and Sauer, 1988). For a stretch of eight amino acids, the number of possible replacements suggests that 2.5×10^6 functional proteins could be generated. One must bear in mind that in the mutagenesis experiment, only double and triple mutants were produced, so the 2.5×10^6 number will clearly overestimate the number of possible functional proteins. It is interesting to compare this result to the number of most probable segments. The number of most probable segments is 1.0×10^6, predicted from Zipf and *k*-tuplet analysis and 2.4×10^3 predicted from the gambler algorithm. The number of most probable sequences is that which is consistent with the proteins taken from the data bank. These sequences will fold into three-dimensional structures consistent with this data bank. However, they will not necessarily be functional proteins, as functionality may be conferred by local properties. Thus, the most probable prediction should be an overestimate of the number of functional proteins. Yet this number is significantly smaller than the mutational estimates. This suggests that there may not be a robust mutational tolerance for systems with more than two or three mutations. Similar considerations apply to evolutionary mutations in a family of proteins (Yockey, 1977).

APPENDIX A: JUSTIFICATION FOR THE DROP AND SEARCH ALGORITHM

To provide a justification for the drop and search algorithm, two specific cases are considered. They are strings or symbol sequences generated by a random Markov process and strings that are highly ordered. The highly ordered strings could be generated by models that show long-range correlations or "persistence."

For a detailed description of the information entropy of Markov models see Cover and Thomas (1991). A Markov model generates a sequence of states, $X_1, \ldots, X_N$, from a simple probabilistic transition rule. To determine the information entropy, $I(X_1, \ldots, X_N)$, one must calculate the joint probability density:

$$\Pr\{(X_1,\ldots,X_N) = (x_1,\ldots,x_N)\} = p(x_1,\ldots,x_N), \quad \text{(A1)}$$

where x_i are specific instances of the random variables. The Markov sequence or chain is generated from the rule

$$p(x_{i+1}) = \sum_{x_i} p(x_i) P_{x_i x_{i+1}}, \quad \text{(A2)}$$

where $P_{x_i x_{i+1}}$ is an element of the transition probability matrix. A consequence of Eq. A2 is that the joint probability may be represented as conditional probabilities by

$$p(x_1,\ldots,x_N) = p(x_1)p(x_2|x_1)p(x_3|x_2)\cdots p(x_N|x_{N-1}). \quad \text{(A3)}$$

The drop and search algorithm described (see *k*-tuplet Analysis, above) generates a large sequence, $x'_1, \ldots, x'_M$, from a smaller sequence, $x_1, \ldots, x_N$, with $M \gg N$. It is asserted that with this algorithm, one has

$$\frac{I(x'_1,\ldots,x'_M)}{M} = \frac{I(x_1,\ldots,x_N)}{N}. \quad \text{(A4)}$$

To see this, one should view the drop and search algorithm as a shuffling process that preserves nearest-neighbor frequencies. Consider an L repeated array of the original N members of the Markov chain such that $LN = M$. The drop and search algorithm creates a new array from the original as follows:

$$\{x_1,\ldots,x_N, x_1,\ldots,x_N,\ldots,x_1,\ldots x_N\} \rightarrow \{x_k,\ldots x_i, x_{i+1},\ldots x_M\}. \quad \text{(A5)}$$

Because the algorithm is a random search, it preserves the individual probabilities in the shuffling transformation. Additionally, because of the nearest-neighbor requirements, i.e., that x_i and x_{i+1} are nearest neighbors in the original and in the shuffled sequence, the conditional probability, $p(x_{i+1}|x_i)$, will also be preserved. With Eq. A3 it is seen that

$$\begin{aligned} p(x_1,\ldots,x_N, x_1,\ldots x_N,\ldots x_1,\ldots,x_N) &= p(x_k,\ldots,x_i, x_{i+1},\ldots x_M) \quad \text{(A6)} \\ &= p(x_1)p(x_2|x_1)p(x_3|x_2)\cdots p(x_M|x_{M-1}). \end{aligned}$$

The Markov probability is the same for the original and shuffled sequences. This establishes that Eq. A4 will hold for Markovian processes. For a statistical sampling of the chain, one must observe all 20^2 doublets for the conditional probabilities. Assuming that 100 such observations are adequate, sequences of length 40,000 are required.

The second example moves from random, Markovian sequences to highly ordered sequences. For simplicity, strings of a binary alphabet consisting of zeroes and ones are considered. In the extreme example of the highly ordered string {01010 ... 01010}, the drop and search algorithm exactly reproduces the string. Other examples of order include strings that show persistence, i.e., long stretches of ones followed by long stretches of zeroes. These are examples of long range correlations because a stretch of 1s are more likely to follow a one. However, once a zero is obtained, a stretch of zeroes will follow it. As a simple example, a string consisting of $N/2$ zeroes followed by $N/2$ ones is considered. The information content of such a string is actually just the total information needed to convey the length of each block of zeroes or ones (Cover and Thomas, 1991) and is given by $I \approx 2\log_2(N/2)$. The information per digit is now given by

$$I \approx \frac{2\log_2(N/2)}{N}, \quad \text{(A7)}$$

which approaches zero as N becomes large. Applying the drop and search algorithm to this unusual sequence gives an interesting result. Initially a

few zeroes may be generated at random. Before long it will drop in the one region, the pointer will then move to the interface (01) to find the preceding zero. Thus, a one will be generated. After this one is generated, all of the other numbers will be ones. For a large number of drops M, the drop and search essentially transforms $\{000 \ldots 01 \ldots 111\} \rightarrow \{111 \ldots 111\}$, where the first few zeroes that might have been generated by chance are ignored. The information entropy of the newly generated sequence is $I \approx \log_2 M/M$, which also approaches zero as M becomes large. The number of observations required to show that the sequence has minimal information content is actually quite small. This can be estimated by $\log_2 M/M = \epsilon$, where ϵ is a small number, say 10^{-2}-10^{-3}. With a change of variables and a simple series expansion, one sees that $M \approx 1/\epsilon$ and M are approximately 100-1000.

APPENDIX B: ESTIMATING INFORMATION ENTROPY FROM GAMBLING ALGORITHMS

In this section it is shown how the information entropy can be calculated by using gambling algorithms. For a more comprehensive treatment see Cover and Thomas (1991). Consider a sequence of events that are gambled on. The probability of the ith choice being a "winner" is given by p_i, and the "payoff" is o_i. The payoff says that 1 dollar on i results in o_i dollars. The bet, b_i, is the fraction of the gambler's wealth that is put on the ith outcome. The fractional increase in the gambler's wealth, S, at the end of a game is $S = \Sigma_{i=1}^{m} o_i b_i$. Assuming the gambler bets all of his wealth on each game, the increasing wealth at the end of N games is

$$S_N = \prod_{j=1}^{N} S_j = 2^{NW}, \tag{B1}$$

where

$$W = \sum_{i=1}^{m} p_i \log_2 b_j o_j. \tag{B2}$$

The quantity W is the expectation value of $\log_2 S$. Using Lagrangian multipliers, it can be shown that the wealth is maximized when $b_j = p_j$. This gives

$$W = \sum_{i=1}^{m} p_i \log_2 o_i - I(p_i), \tag{B3}$$

where I is the Shannon information of the probability of winning. For even payoffs, $o_i = 1/m$ and

$$S_N = 2^{-N(\log_2 m + I)}. \tag{B4}$$

In the development of the Chou-Fasman gambling algorithm, the payoffs are of no significance. In this instance, the average number of correct "wagers," W, is defined as

$$\Omega = \frac{2^{-mN}}{S} = 2^{NI}, \tag{B5}$$

where N is now the sequence length and m is the number of amino acids. However, Eq. B5 is still not complete. The Chou-Fasman gambler represents a case in which pertinent prior knowledge (the preceding sequence and the Chou-Fasman rules) is obtained. This represents a case of gambling with side information, i.e., the racetrack tip. If the variable x represents the outcome of the gamble and the variable y represents prior information, then to calculate the increase in wealth, one uses a conditional expectation value:

$$W(x|y) = \sum_{i=1}^{m} \sum_{y} p(x_i, y) \ln_2 b(x_i|y) o(x_i), \tag{B6}$$

where $b(x_i|y)$ is the ith bet given the side information, y. Again, the conditional expectation is optimized and the optimal bet is given by $b(x_i|y) = p(x_i|y)$. As before, the wager term is removed. The Chou-Fasman gambler used above is then defined as

$$\Omega_y = \frac{2^{-mN}}{S} = 2^{I(x|y)}, \tag{B7}$$

where $I(x|y)$ is the conditional information entropy. The x variables are the 20 amino acids and the y variables are the secondary structural units (see Chou-Fasman Gambler, above).

This work was supported in part by National Institutes of Health grant 1R15GM51019.

REFERENCES

Balafas, J. S., and T. G. Dewey. 1995. Multifractal analysis of solvent accessibilities in proteins. *Phys. Rev. E.* 52:880–887.

Brillouin, L. 1962. Science and Information Theory. Academic Press, New York.

Chou, P. Y., and G. D. Fasman. 1978. Empirical predictions of protein conformation. *Annu. Rev. Biochem.* 47:251–276.

Cover, T. M., and J. A. Thomas. 1991. Elements of Information Theory. John Wiley and Sons, New York.

Dewey, T. G., and B. J. Strait. 1996. Multifractals, decoded walks and the ergodicity of protein sequences. *In* Pacific Symposium on Biocomputing. L. Hunter and T. E. Klein, editors. World Scientific, Singapore. 216–229.

Dewey, T. G. 1996. The algorithmic complexity of a protein. *Phys. Rev. E.* In press.

Eigen, M. 1992. Steps Toward Life. Oxford University Press, Oxford, England.

Hohohm, U., M. Scharf, R. Schneider, and C. Sander. 1992. Selection of representative protein data sets. *Protein Sci.* 1:409–417.

Kauffman, S. A. 1992. Applied molecular evolution. *J. Theor. Biol.* 157:1.

Kauffman, S. A. 1993. The Origins of Order. Oxford University Press, New York.

Mantegna, R. N., S. V. Buldyrev, A. L. Goldberger, S. Havlin, C.-K. Peng, M. Simons, and H. E. Stanley. 1995. Systematic analysis of coding and noncoding DNA sequences using methods of statistical linguistics. *Phys. Rev. E.* 52:2939–2950.

Pande, V. S., A. Y. Grosberg, and T. Tanaka. 1994. Nonrandomness in protein sequences: evidence for a physically driven stage of evolution? *Proc. Natl. Acad. Sci. USA.* 91:12972–12975.

Reidhaar-Olson, J. F., and R. T. Sauer. 1988. Combinatorial cassette mutagenesis as a probe of the information content of a protein sequence. *Science.* 241:53–57.

Shakhnovich, E. I., and A. M. Gutin. 1993. Engineering of stable and fast-folding sequences of model proteins. *Proc. Natl. Acad. Sci. USA.* 90:7195–7199.

Shannon, C. E. 1951. Prediction and entropy of printed english. *Bell Sys. Tech. J.* 30:50–64.

Shannon, C. E., and W. Weaver. 1962. The Mathematical Theory of Communication. University of Illinois Press, Urbana, IL.

Srinivasan, R., and G. D. Rose. 1995. LINUS: a hierarchic procedure to predict the fold of a protein. *Proteins Struct. Funct. Genet.* 22:81–99.

Strait, B. J., and T. G. Dewey. 1995. Multifractals and decoded walks: applications to protein sequence correlations. *Phys. Rev. E.* 52: 6588–6592.

Volkenstein, M. V. 1994. Physical Approaches to Biological Evolution. Springer-Verlag, Berlin.

Yockey, H. P. 1977. On the information content of cytochrome c. *J. Theor. Biol.* 67:345–376.

Yockey, H. P. 1992. Information Theory and Molecular Biology. Cambridge University Press, Cambridge, England.