

# A block coding method that leads to significantly lower entropy values for the proteins of *Haemophilus Influenzae* and its coding sections

G. Sampath

Department of Computer Science, The College of New Jersey, Ewing, NJ 08628  
sampath@tcnj.edu

**Abstract.** A simple statistical block code in combination with the LZW-based compression utilities *gzip* and *compress* has been found to increase by a significant amount the level of compression possible for the proteins encoded in *Haemophilus Influenzae* (*hi*), the first fully sequenced genome. The method yields an entropy of 3.665 bits per symbol (bps), which is 0.657 bps below the maximum of 4.322 bps. This is an improvement of 0.452 bps over the best known to date of 4.118 bps using the *lza-CTW* algorithm of Matsumoto, Sadakane, and Imai, the next best being 4.143 bps using Nevill-Manning and Witten's *cp* algorithm. Using an efficient inverse map from the 20 amino acids to the 61 triplets that code for them, the genome too is found to be compressible, although the gain is not as high. Calculated estimates based on this latter method yield an entropy of 1.757 bps for the coding portions of the genome, with a possibly lower actual entropy. Both of these results, which flow from the sequential use of statistics-based encoding techniques followed by substitution-based text compression algorithms (*gzip*, *compress*), hint at the existence of hitherto unexplored redundancies at both the global and the local level in *hi* and the proteins coded by it. Further work may help determine whether such decreases in entropy are possible with other proteins and genomes or *hi* is a rarity (perhaps even unique) in this respect.

**Keywords.** Protein compression; DNA compression; *Haemophilus Influenzae*

## 1. Introduction

The rapid increase in available biological sequence data in several publicly accessible databanks [13-18] and the use of advanced distributed computing methods [5] have led to the use of increasingly complex computational methods designed to extract biological, chemical, and physical correlates from those sequences. On the one hand, the growing volume of sequence data available makes it possible to search for deep structural properties in the sequences and relate them to biological function at several levels, from the molecular to the cellular, in ways that are more promising and statistically more significant than was possible earlier with smaller data sets. Thus new algorithms have been designed explicitly for this purpose or existing ones modified to work with biological data [9], and special data structures designed to work efficiently with the massive amounts of data available [12]. On the other hand, there is a growing interest in reducing the amounts of storage required to cope with the exponential rise in the sizes of the databanks. This has led to the development or adaptation of a wide range of data compression techniques [10, 11], new and old, tailored to the task of compressing genome and protein sequences. Currently, however, the results obtained from the application of these methods appear only to reinforce the long-standing belief that biological sequences are not compressible in a significant way. Such a belief, which is reflected in the title of a recent communication [8], stems in part from the fact that the vast majority of those sequences are random and

their measured entropies are not very far from those predicted by information theory [1]. In spite of this negative result, interest in biological sequence compression has not slackened because any gains, however small, are likely a reflection of constraints in internal structure which may not otherwise be evident in a sequence but may help understand the physical and chemical properties of the macromolecules coded by it.

The purpose of the present communication is to report on a significant amount of compression achieved with the proteins coded by *Haemophilus Influenzae (hi)* (the first genome to be fully sequenced [13]). An appreciable amount of compression has also been obtained with the genome itself, although it is not as large as with the proteins. One possible consequence of this result is that it holds out hope for similar successes with other sequences, perhaps through the use of the methods outlined here or with new approaches. Thus, further investigation can help determine if the compression achieved with *hi* is to be regarded as a fluke (which would make *hi* unique, itself a subject worth studying) or significant decreases in entropy are possible with other proteins and genomes.

In Section 2, a brief introduction is given to the data compression problem as it affects biological sequences and its relation to entropy measures. In Section 3, a block code is developed for the proteins coded by *hi* based on their statistical distribution. Section 4 describes a procedure to compress the protein sequences and presents results comparing its performance with that of other methods as reported in the literature. Section 5 presents an efficient way of mapping residues (that is, amino acids) in the protein sequence to the codons (triplets) in the parent DNA that leads to a non-trivial decrease in entropy of the latter. Section 6 offers some conclusions and directions for future work.

## 2. Biological sequence compression, information theory, and the Shannon limit

Almost all known biological functions can be traced to the sequence of nucleotides known as DNA that is found in the cells of every organism. The DNA is transcribed one-to-one in the cell to an RNA sequence, and the latter in the presence of a number of catalysts is translated into a set of proteins, which are sequences of amino acids that contribute to a host of cellular and higher-level functions of the organism. At each of these levels, the macromolecules involved (DNA, RNA, and protein) can be viewed as sequences of symbols in an alphabet, and the transformation steps as string operations. In information theory [1], a sequence is a string on an alphabet whose information content can be related to its entropy. The entropy  $E$  of a sequence is closely related to the alphabet on which the sequence is based and is defined as

$$E = - \sum_{i=1}^N p_i \log_2 p_i \quad \text{bits per symbol (bps)}$$

where  $p_i$  is the probability (or normalized frequency) of occurrence in the sequence of the  $i$ -th symbol in an alphabet of  $N$  symbols. When the probabilities are all equal and equal to  $1/N$ , the entropy is simply  $\log_2 N$  bps. This is commonly referred to as the *Shannon entropy* of the alphabet. DNA's alphabet is the set  $\{T, C, A, G\}$ , RNA's is  $\{U, C, A, G\}$ , and the protein alphabet is a set of 20 amino acids  $\{A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y\}$ . The Shannon entropy values are respectively 2, 2, and 4.322 bps.

When a sequence of symbols is formed from the alphabet, the measured entropy of the sequence is a measure of redundancy in the sequence. If the sequence is random, it tends

to have symbols that occur with the same frequency, and hence leads to an entropy close to the Shannon value. From the information theoretic point of view, the farther away (towards 0) the measured entropy is from the Shannon value, the greater the redundancy and hence the lower the information content. This also means that the sequence has dependencies among the symbols in it, which may sometimes be correlated to physical properties such as, in the case of RNA and protein, the shape that the macromolecule attains in space.

Biological sequences can also be approached from the point of view of their storage requirements: the presence of redundancies indicates the possibility of using less storage. Thus the entropy of a sequence can also be viewed as the number of bits of storage per symbol, this number being computed as an average over the whole sequence. The lower the entropy, the less the storage required. One can then consider ways of recoding the sequence to yield a compressed form, whose length in turn is a measure of the redundancy in the sequence. This view has led to concerted attempts at compressing biological sequences using a wide range of available algorithms. However, such attempts have generally had limited success. The measured entropies of the sequences that result have either been not far from the Shannon limit or, worse, in many cases higher, with the original sequence expanding in size. This resistance of both DNA and protein to compression has led to the use of pattern recognition methods and complex Markov models [2-8] that examine a sequence for patterns and dependencies between a symbol and the symbols preceding it in a variable size window. When applied to the proteins encoded by a set of fully sequenced genomes the resulting decreases in entropy are on the order of 0.2 to 0.3 bps [7, 8]. With the genome *Haemophilus Influenzae* (*hi*), the lowest entropy obtained so far has been 4.118 bps using the *Iza-CTW* algorithm of Matsumoto, Sadakane, and Imai [7], which is 0.204 bps below the Shannon limit. In comparison, in the work reported here an entropy level of 3.665 bps (which is 0.657 bps below the maximum) has been obtained for the proteins of *hi*. The details are given in the following sections.

### 3. A block code for the proteins coded by *Haemophilus Influenzae* (*hi*)

Compression of the proteins coded by the genome sequence of *hi* is based on a process of coding each residue with a 2-part code.

**Partitioning the amino acids.** First a frequency distribution of occurrence of the 20 amino acids in the proteins is obtained and sorted in increasing order. Let this sorted distribution be  $\mathbf{F} = [f_1, \dots, f_{20}]$  and the corresponding amino acid array  $\mathbf{A} = [a_1, \dots, a_{20}]$ .  $\mathbf{A}$  is divided into  $N$  groups  $\mathbf{G} = [\mathbf{G}_1, \dots, \mathbf{G}_N]$  where  $\mathbf{G}_i = [i_1, \dots, i_k]$  and  $\mathbf{G}_i \leq \mathbf{G}_{i+1}$ . Here  $\leq$  is a total ordering defined on the groups: the frequency of occurrence of every residue in  $\mathbf{G}_i$  is less than or equal to that of every residue in  $\mathbf{G}_{i+1}$ ,  $1 \leq i \leq N-1$ . A residue in the protein is then coded with a number  $g$ ,  $1 \leq g \leq N$ , corresponding to the group  $\mathbf{G}_g$  that it belongs to.

**Coding the elements in each partition.** Next the same residue is assigned a number  $p$  that codes the position of the corresponding amino acid in the group. If  $|\mathbf{G}_g|$  is 1, this number is omitted as there is no need to distinguish the single amino acid from another in the group.

Thus every one of the residues in the protein has a code pair  $(g, p)$ . ( $p$  may be absent, as mentioned above). A protein with  $L$  residues can then be coded as a sequence of such pairs. Alternately, it can be represented by two strings: a group membership string  $\mathbf{g} = g_1$

$g_2 \dots g_L$  and a position string  $\mathbf{p} = p_1 p_2 \dots p_{L'}$ , where  $L'$  can be  $< L$  because, as noted above, an amino acid may be the only one in its group in the partition.

#### 4. Compressing the proteins coded by the *hi* genome

It is shown next that compression can be achieved by applying well-known utilities such as *compress* and *gzip* separately to the  $\mathbf{g}$  and  $\mathbf{p}$  strings. The value  $g$  in each pair can be stored in  $\lceil \log_2 N \rceil$  bits. The value  $p_k$  ( $k = 1, \dots, L'$ ) requires  $s_k$  bits, where

$$0 \leq s_k \leq g_{\max} = \max \{ \lceil \log_2 |\mathbf{G}_1| \rceil, \lceil \log_2 |\mathbf{G}_2| \rceil, \dots, \lceil \log_2 |\mathbf{G}_N| \rceil \}.$$

The storage requirement for the protein is then

$$N \lceil \log_2 |\mathbf{G}_i| \rceil + \sum_{k=0}^{L'} s_k$$

bits.

The partitioning of the amino acid array  $\mathbf{A}$  as described in Section 3.1 is done to reflect the frequencies of occurrence of the amino acids in the protein. This is effectively a Huffman-like code in which the more frequently occurring amino acids are in smaller groups, which require fewer bits to encode membership in the group. To avoid wasting bits in coding  $g$ , it is of advantage to choose  $N$  as a power of 2. For example, the 20 amino acids can be divided into 8 groups of  $i_1, \dots, i_8$  amino acids, with the frequencies of occurrence decreasing to the right in the list, or 4 groups  $i_1, \dots, i_4$ . Then the  $g$  in each residue's code pair ( $g, p$ ) is coded with 3 or 2 bits respectively. The sizes of the groups in a partition can be chosen similarly, but a choice of a power of 2 for each  $|\mathbf{G}_i|$  once again tends to give the best results. For example, the partition could be  $\{1, 1, 2, 2, 2, 4, 4, 4\}$  in the first case, or  $\{4, 4, 4, 8\}$  in the second. Experiments with the protein sequence of *hi* led to the first of these choices as the one giving the best results. The decoding is straightforward: the group for the  $i$ -th residue is the  $i$ -th symbol in the  $\mathbf{g}$  file, and its position in the group is either not required (if the group has only one member) or the value corresponding to the next symbol in the  $\mathbf{p}$  file.

The individual strings  $\mathbf{g}$  and  $\mathbf{p}$  are now compressed using a number of available utilities. The following results were obtained on a Sun Ultra-5 workstation with a 500 MHz processor and 512 Mbytes running Solaris:

File	File size (bytes)
Protein file (see [16])	509519
$\mathbf{g}$ file	509508
$\mathbf{p}$ file	453560
$\mathbf{g}$ compressed with <i>gzip</i> (switch -9) A	205192
$\mathbf{p}$ compressed with <i>gzip</i> (switch -9) B	45485
$\mathbf{g}$ compressed with <i>compress</i> C	187836
$\mathbf{p}$ compressed with <i>compress</i> D	117301

**Table 1. Results of compressing the block coded files  $\mathbf{g}$  and  $\mathbf{p}$**

Data for *hi* were taken from the Protein Corpus [17]. The block coding time was negligible and the compression time very small, neither was recorded. In the latter case, it ranged from near zero with *compress* to a few seconds with *gzip* (switch -9).

Since the alphabet for each of the strings is limited to symbols used to represent small integers one would expect their behavior with compression algorithms to be similar to that shown by DNA (with its alphabet size of 4). However the outcome is different from the expectation. Because the decoding is to be done after decompression, the **g** and **p** files can be compressed separately and by algorithms that are different. As seen in Table 2 below, this mixing and matching leads to an unexpected drop in the total size of the compressed data when **p** is compressed with *gzip* and **g** with *compress*. (A small amount of storage, 44 bytes, used for 11 gene region separators in the source, is not included.)

Shannon entropy for amino acids (bps)	4.321928
Shannon storage for protein file (bytes)	275140
Storage required by combining B with C from Table 1 (bytes)	233321
Entropy resulting (bps)	3.665030

**Table 2. Entropy resulting from combining different compressed versions of g and p**

Table 3 (adapted from [7]) gives an idea of how significant the change in entropy from the Shannon value is in comparison with values reported for other methods. (Entropy values above the Shannon value are italicized, that for the current method is in bold.)

<b>Compressor</b>	<b>Entropy (bps)</b>
(Shannon)	4.32198
<i>Compress</i>	<i>4.7702</i>
<i>Bzip2</i>	<i>4.324</i>
<i>Gzip -9</i>	<i>4.6712</i>
Arith	<i>4.1557</i>
<i>lz-ari (1M)</i>	<i>4.1270</i>
<i>Normal PPMD+</i>	<i>4.862</i>
<i>Adapted PPMD+</i>	<i>4.151</i>
<i>CTW20(8)</i>	<i>4.1381</i>
<i>CTW20(16)</i>	<i>4.1378</i>
<i>lz-CTW(8)</i>	<i>4.1185</i>
<i>lza-CTW(8)</i>	<i>4.1177</i>
<i>CP(1)</i>	<i>4.149</i>
<i>CP(2)</i>	<i>4.146</i>
<i>CP(3)</i>	<i>4.143</i>
<b>Current method</b>	<b>3.665</b>

**Table 3. Comparison of results with earlier work (see [7] for details)**

## 5. Using compressed proteins to compress DNA

Attempts to compress DNA sequences have generally been unsuccessful. Although there have been instances [6, 7] in which entropies as low as 1.1048 have been found, these

lower values are usually obtained with non-coding segments (which are either introns or 'junk' DNA). For coding segments, values in the range 1.84 through 1.95 are more typical [6]. Based on empirical data available, it is generally the case that the coding sections of a genome are very resistant to compression, while the intron/'junk' segments appear to be more susceptible. This is often attributed to the small alphabet and the tendency of coding DNA sequences to be uniformly distributed on the alphabet. (It is widely believed that this uniform distribution is a result of DNA having evolved over a long period of time, a consequence, perhaps, of the third law of thermodynamics.) But here too *hi* runs counter to the general behavior of DNA. The following paragraphs show how the coding sections of *hi* can be compressed to 1.7209 bps.

As mentioned in Section 2, the genetic code is degenerate, with more than one codon coding for an amino acid. The following table lists the number of codons for each amino acid, it also contains information on the reverse code needed to go from amino acid to codon as discussed in the next paragraph.

No of codons (X)	Amino acids	No of amino acids	Reverse code size = $\lceil \log_2 X \text{ bits} \rceil$
1	M, W	2	0
2	F, Y, C, H, Q, N, L, D, E	9	1
3	I	1	2
4	V, P, T, A, G	5	2
6	L, S, R	3	3

**Table 4. Reverse code requirements for the genetic code**

If the coding segments of a genome are translated to amino acid sequences, they can be recovered only if the codon coding for a residue is known. This requires storing a coded form of the source codon for each residue, which requires from 0 to 3 bits, leading to a third reverse code file **r** if the genome is to be compressed (see Table 4). If the storage required for the compressed sequence protein ( $= |g| + |p|$ , obtained from the translation of genome to protein sequence) plus the storage for **r** is less than the Shannon storage value of the genome sequence, then compression of the genome is considered to occur. Consideration must also be given to allocation of storage to the gene separators in the DNA so that the gene boundaries can be recovered during decompression. There are 1709 genes, with 85% of them having a coding density of 1070 base pairs per gene [13]. With no compression and at 4 bytes per gene (2 for the length, 2 for the offset within the genome) this generates a fourth file **b** with 6836 bytes. The total storage for the translated compressed gene data is then 336301 bytes. (This is in contrast with the Protein Corpus [17], where the objective of compression seems only to be decreasing the entropy. The corpus stores the residues in the form of 12 strings corresponding to the 12 gene regions in the genome, rather than as individual proteins. This is apparently based on the assumption that the gene boundaries are accessible through the parent DNA, which is assumed to be available separately.) The following are the results from applying this method to the protein sequences from *hi*, with **r** and **b** stored uncompressed:

Compressed storage for <i>hi</i> proteins	A	233321 bytes
Reverse code for 509519 residues	B	96144 bytes
Gene boundary information (file <b>b</b> )	C	6836 bytes
Total storage required for genome	D	336301 bytes
Shannon storage required for $3 \times 509519$ bases (assuming 2 bps for DNA)	E	382889 bytes
Entropy of compressed translated genome with reverse codes and gene boundaries	F	1.757 bps
Reduction in entropy level	G	0.243 bps

**Table 5. Results of compressing the coding sections of *hi***

As noted in [6] if the fact that only 61 codons code for amino acids (the other three coding for terminators) is taken into account, then the Shannon entropy of DNA is  $\log_2 61 / 3 = 1.977$  bps. Using this in the computation above, the entropy for the compressed translated genome is lower than the Shannon value by 0.22 bps, which is still an appreciable reduction. A slight improvement is possible if the reverse code for Isoleucine (I) is stored as a prefix code. The reverse code file **r** and the gene boundary file **b** also may be compressible, results are incomplete at this time.

The method when applied to the proteins coded by the three other genomes in the Protein Corpus [17] (being the two fully sequenced ones, *Methanococcus Jannischii* and *Saccharomyces Cervisiae*, and the third partially sequenced one, *Home Sapiens*) yielded negative results (the sequences expanded). Work is ongoing on modified versions, with tests underway on these genomes as well as a range of other proteins in various protein databases [13-16, 18]. Firmer conclusions about their compressibility should be possible after results are available.

## 6. Conclusions and directions

The following are some conclusions that can be drawn from the above study:

- 1) It does not seem reasonable to infer from the failure or success of a compression method working on one or more sequences that it will fail or succeed with others [8]. As shown above, it is possible that existing methods in combination or suitably modified (or perhaps in conjunction with others yet to be devised) might succeed in lowering the entropy of one or more protein sequences or their parent genes, or both, which have been resistant to known compression schemes. It is also possible that attempts to devise general compression schemes may be frustrated because an algorithm that works with a genome or protein may not always work with another (and vice versa).
- 2) If tests on a wider range of data show that the method described here works only on *hi* but not on other sequences, it would make *hi* unique, which then raises the question what makes it so.
- 3) Based on the method described a sufficient level of compressibility in protein would imply compressibility of the parent DNA as long as the **r** and **g** files are not too big.
- 4) The fact that *hi* has a significantly lower entropy than the maximum (for both the proteins and the set of its encoding genes) implies a level of redundancy that does not show up in Markov models. This in turn hints at the existence of more internal structure

than suspected at both the protein and the genome level and points to the need to look for its causes.

## References

- [1] Ash, R. B. *Information theory*. New York, Interscience Publishers, 1965.
- [2] Chen, X., Kwong, S., and Li, M. "A compression algorithm for DNA sequences and its applications to genome comparison." *Genomic Informatics* 10, 52-61, 1999.
- [3] Grumbach, S. and Tahi, F. "A new challenge for compression algorithms: genetic sequences." *Information Processing and Management* 30, 875-866, 1994.
- [4] Lanctot, J. K., Li, M., and Yang, E. "Estimating DNA sequence entropy." *Proceedings 11th Annual ACM-SIAM Symposium on Discrete Algorithms*, 409-418, 2000.
- [5] Larson, S. M., Snow, C. D., Shirts, M., and Pande, V. S. "Folding@Home and Genome@Home: Using distributed computing to tackle previously intractable problems in computational biology." In: *Computational Genomics*, R. Grant (ed.), Horizon Press, (2002).
- [6] Loewenstern, D. and Yianilos, P. N. "Significantly lower entropy estimates for natural DNA sequences." *Journal of Computational Biology*, volume 6, number 1, 1999.
- [7] Matsumoto, T., Sadakane, K., and Imai, H. "Biological Sequence Compression Algorithms." *Genomic Informatics* 11, 43-52, 2000.
- [8] Nevill-Manning, C. and Witten, I. H. "Protein is incompressible." *Proceedings of IEEE Data Compression Conference*, 257-266, 1999.
- [9] Pevzner, P. A. *Computational Molecular Biology: An Algorithmic Approach*. Cambridge (Mass.), MIT Press, 2000.
- [10] Sayood, K. *Introduction to Data Compression*. San Francisco, Morgan Kaufman, 1996.
- [11] Solomon, D. *Data Compression: The Complete Reference*. New York, Springer Verlag, 1997.
- [12] J. S. Vitter. "External memory algorithms and data structures: dealing with massive data." *ACM Computing Surveys*, **33**(2), June 2001, 209-271.

## Sources on the Web

- [13] Center for Biological Sequence Analysis (CBS). <http://www.cbs.dtu.dk/index.html>, <http://www.cbs.dtu.dk/services/GenomeAtlas/Bacteria/Haemophilus/influenzae/Rd/>.
- [14] GenomeNet. <http://www.genome.ad.jp>.
- [15] National Center for Biotechnology Information (NCBI). <http://www.ncbi.nlm.nih.gov>.
- [16] Swiss-Prot. <http://us.expasy.org/srs5>.
- [17] Protein.Corporus. <http://www.Data-Compression.info>.
- [18] TIGR Database (TDB). <http://www.tigr.org/tdb>.