On Compressibility of Protein Sequences*

Donald Adjeroh and Fei Nan
Department of Computer Science and Electrical Engineering
West Virginia University, Morgantown, WV 26506-6109 [don, fein] @csee.wvu.edu

Abstract

We consider the problem of compressibility of protein sequences. Based on an observed genome-scale long-range correlation in concatenated protein sequences from different organisms, we propose a method to exploit this unusual redundancy in compressing the protein sequences. The result is a significant reduction in the number of bits required for representing the sequences. We report results in bits per symbol (bps) of 2.27, 2.55, 3.11 and 3.44 for protein sequences from *M. jannaschii, H. influenzae, S. cerevisiae*, and *H. sapiens* respectively, the same protein sequences used by Nevill-Manning and Witten in the "*Protein is incompressible*" paper [23]. The observed long-range correlations could have significant implications beyond compression and complexity analysis of protein sequences.

1. Introduction

Proteins are large molecules and are responsible for differing functions in an organism. There are different types of protein, and the exact function of a given protein depends on its three dimensional structure. Proteins are made up of amino acids, which are often joined in chains to form polypeptides (or simply peptides for smaller sequences). Proteins are formed from an alphabet of 20 amino acids. Each amino acid has a corresponding DNA triple, taken from the 4-symbol DNA (or RNA) alphabet. According to the Central Dogma of molecular biology [8], the information flow is unidirectional: from DNA to RNA to Proteins. That is, the exact sequence of amino acids in a give protein sequence is determined by the primary DNA sequence of the genes that produced the protein. The exact rules used in this mapping from DNA bases to amino acids is defined in the Genetic Code[11]. The genetic code is a non-overlapping block code, whereby three successive RNA nucleotides code for one amino acid, or for a stop translation signal. The areas of the DNA that contain genes are thus called **coding** regions, while the remaining parts are called **non-coding** regions.

Given the recent exponential growth of available biological sequences, the need for efficient storage and communication of such data has heightened significantly. Compression methods have emerged to address this challenge, with varying degrees of success. Protein in particular is known to be very difficult to compress, especially when compared with the general DNA which could have various forms of repetition. The interest in compression is, however, not just for efficient storage and data transport. For biological sequences in particular, the intrinsic relationship between complexity, redundancy and compression imply that an algorithm that can compress biological sequences could provide a means for analyzing such sequences for hidden structures. Such hidden structures could be exploited for various applications, from sequence classification to construction of phylogenic trees, to comparative genomics.

2 Protein Sequence Compression

Given that we represent the information in protein sequences using 20 symbols, if the sequences were to be completely random (that is, completely unpredictable or incompressible [20]), then, we should need $\log_2(20)$ or about 4.32 bits to code each amino acid. Traditional compression methods that have done very well on text often find it difficult to compress biological sequences, such as DNA or protein sequences. In fact, using the classical compression methods (such as word-based Huffman, arithmetic coding, or LZ-based methods) directly on such sequences often result in ***data expansion***, rather than compression [13, 27]. The major reason is the fact that these methods often use models that were derived for traditional text, and hence fail to consider certain special characteristics of biological sequences. Biological sequences are however known to convey important purposeful information between different generations of an organism. Moreover, biological sequences are known to contain different types of repetitions and other hidden regularities. Long runs of

* This work was partially supported by a DOE CAREER Grant, No.: DE-FG02-02ER25541.

Proceedings of the Data Compression Conference (DCC'06)
0-7695-2545-8 /06 $20.00 © 2006 IEEE

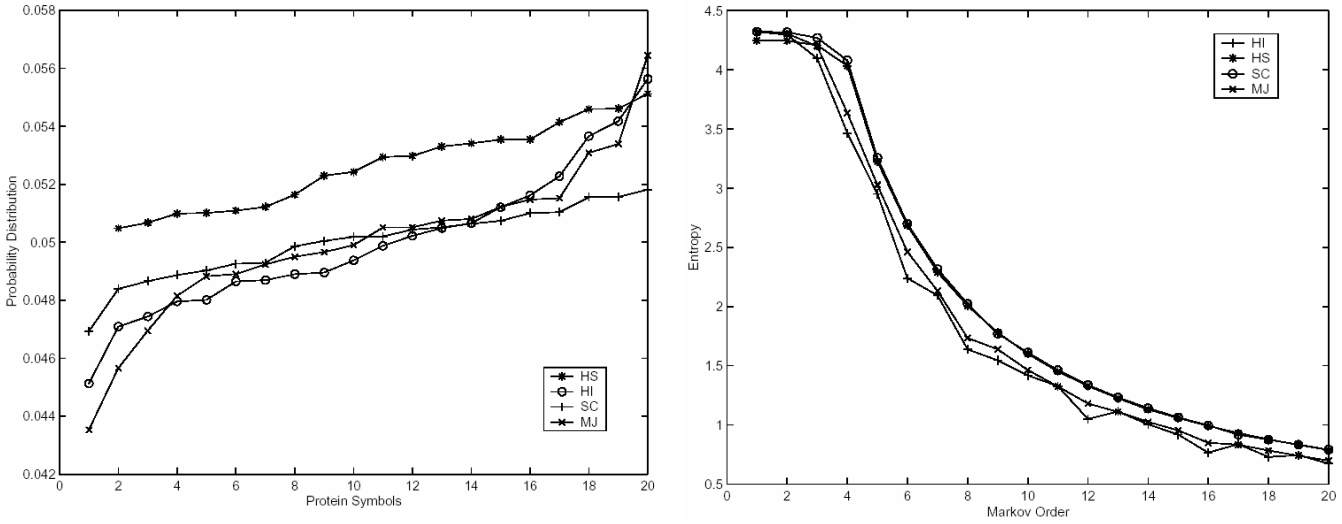
tandem repeats and of randomly interspersed repeats are prominent features of DNA sequences. Thus, from the viewpoint of compression and sequence understanding, the repetitions inherent in biological sequences imply redundancies which can provide an avenue for a signifcant compaction. The identification of such dependencies is the starting point for biological sequence compression.

Different specialized algorithms have been proposed for compressing biological sequences, with varying degrees of success. Examples here include BIOCOMPRESS1 and BIOCOMPRESS2 [13], CFACT [27], GENCOMPRESS [6], and GTAC [19], and the context-tree weighting method of Masumoto *et al* [22]. Maximum likelihood approaches to DNA sequence compression were proposed in [18], while methods that use offline dictionaries were reported in [3] and [2]. Loewenstern and Yainilos [21] proposed a method to estimate the entropy of DNA sequences by using inexact matches based on a family of distance measures. The different algorithms differ primarily in the type(s) of repetitions they consider, and in how the repetitions are represented and exploited to achieve compression.

Compression methods have also been proposed with specific consideration of protein sequences. In a well-cited paper [23], the PPM algorithm [7] was modified by considering mutation probabilities for the amino acids that make up protein sequences. Though the results produced were relatively better than those from the original PPM (which led to data expansion), the compression was not significant. This led the authors to conclude that protein is not compressible by any appreciable degree. In [22] context tree weighting was combined with simple LZ77 parsing to provide a better compression over the same data corpus used in [23]. While this is a significant result, the method did not do well on 2 of the 4 sequences in the corpus. More recently, Sampath [28] presented a block-based method that could provide a significant compression on one sequence in the corpus (the HI sequence, from *Haemophilus influenzae*), with compression ratios of up to 3.66bps. However, attempts to use the same method on the other three sequences lead to data expansion.

2.1 Why protein is difficult to compress

In the following, we use the term "protein sequence" to refer to the entire collection of all the proteins in an organism (as was done in [23]). This is different from the use of the terminology in other related work (such as in [11]) to refer to the sequence of amino acids in one polypeptide.



(a) **(b)**

Fig 1: Sorted probabilities (a) and higher order entropy (b) for the four sequences in the protein corpus. The protein sequences are taken from four organisms: HI: *H. influenzae*; MJ: *M.jannaschii*; HS: *H. sapiens*; SC: *S. cerevisiae*

Even with the degeneracy of the genetic code, it is well established that protein sequences are particularly difficult to compress. One reason is the problem of multiplicity and individuality of protein sequences [10]. Given the relation between sequence complexity and complexity of organisms, one can expect that for complex higher organisms, their protein sequence should be difficult to model, and hence compress. Another

major difficulty lies in the apparent randomness of the symbols in a protein sequence. In Fig. 1(a) we show the individual probabilities for the four protein sequences in the Protein Corpus used by Nevill-Manning and Witten [23]. For each sequence, the symbols (x-axis) are sorted in increasing order of their probability. The first observation is the very small range of values for the probabilities (from around 0.043 ~0.057) for any given sequence. The amino acid Tryptophan (*Trp*, symbol *W*) did not occur in the HS sequence. A smaller range for the probabilities imply more closeness to the uniform distribution, and hence less compressibility. It can be observed that HS (*H. Sapiens*) and SC (*S. cerevisiae*) seem to have a higher complexity. This difficulty in compression is further illustrated in Fig. 1 (b), which shows the higher order entropy for the sequences.

We can also consider the problem of compressibility of protein sequences by looking at the performance of current state of the art compression algorithms on such sequences. Table 1 shows that protein is in fact difficult to compress. Traditional compression algorithms (BWT, PPM, WinZip (LZ-based)) all expand, rather than compress the sequences. That is, they require more than 4.32 bits per symbol (bps) to represent the protein sequences. A better result is produced by specific algorithms for protein sequence compression, but none was able to produce a universally good compression across all the sequences. Most of them still require more than 4bps. (The biological sequence compression algorithms in the table are GenCompress[6], CP [23], lzaCTW [22] and blockCode [28]).

2.2 Why protein should be compressible

Given the Central Dogma, we can say that the protein sequence (at least at the primary sequence level) is a function of the original DNA sequence. From information theory, we know that the entropy of a function of a random variable cannot be any greater than the entropy of the original random variable. Thus, the protein sequence cannot have more entropy than the original DNA sequence from which it is formed [11]. But this is only with respect to the coding regions of the DNA sequence. Further, with the assumption that organisms are purposeful, and not just a random collection of symbols, we can expect that there will be constraints placed on how the DNA sequences (hence protein sequences) are ordered. Such constraints are likely to lead to some form of redundancy, which could be exploited for compression.

We also expect that nature will find a way to protect important genes or gene products in a given organism. This means that there is likely to be some other form of redundancy in biological sequences, at least, for the purpose of more reliable transmission of genetic information from one generation to the other. The various forms of repetitions in biological sequences could be one way to ensure this reliability. More importantly, the phenomenon of gene duplications, multiple gene copies, and histone clusters in genomes of higher organisms imply that their protein sequences (at least, at the primary sequence level) will have some redundancy. For instance, [14] reports of 1509 duplicate paralogous genes from 5766 yeast open reading frames for baker's yeast, *S. cerevisiae.* Similar observations on the extent of duplicate genes in other model organisms were reported in [15]. A number of experiments have shown that the inactivation of certain genes has no apparent phenotypical effect on the fitness of certain species [9, 12, 14, 24, 25]. This observation relates to the functional redundancy of genes in different organisms, and it has been observed that most of these functionally similar genes tend to be quite similar at both the transcription level and at the sequence level [26].

Table 1: Failure of compression algorithms on protein sequences. Results in bits/symbol (smaller values imply better performance).

	General Compression Algorithms					Biological Sequence Compression Algorithms						
	PPM	BWT	WinZip	Gzip	PPMD	Gen Compress	CP(0)	CP(1)	CP(2)	CP(3)	Lza CTW-8	Block Code
HI	4.881	4.49	4.671	4.672	4.151	4.156	4.156	4.149	4.146	4.143	4.118	3.665
MJ	4.734	4.45	4.589	4.588	4.061	4.062	4.068	4.06	4.056	4.051	4.028	5.102
SC	4.854	4.49	4.638	4.64	4.157	3.97	4.163	4.158	4.152	4.146	3.952	5.175
HS	4.639	4.43	4.605	4.605	4.119	3.972	4.133	4.126	4.12	4.112	3.920	5.087

Initial empirical evidence that protein sequences could indeed be significantly compressed can be seen in the partial success of the block coding algorithm of Sampath [28] and the context-tree weighting methods of Matsumoto et al [22]. See Table 1. The methods were able to compress some protein sequences in the corpus

below the 4.0bps mark. In particular, the block coding method was able to compress the HI sequence to about 3.66bps, although the method performed very poorly on all the other sequences. This is still a very significant result: it dispels the view that protein is incompressible, and shows that a carefully constructed algorithm could work well on a restricted set of input sequences. As was suggested by the author, this ability to compress only the sequences from HI could mean that *H.influenzae* is a unique organism. We note that, from the viewpoint of compressibility, neither the higher-order entropy, nor the simple probability distribution of the symbols supports this view that HI is unique. Although generality of the algorithm is still a problem, especially when viewed from the definition of "incompressibility" in [23], the fact that such a relatively low compression can be achieved on a protein sequence is significant in its own right. This raises the hope that some unknown constraints may be at play in such sequences, which if identified could have implications not only for the compaction of the sequences, but on general analysis of such sequences.

2.3 Problems with current approaches

In our view, one major problem with current approaches to protein sequence compression in particular, and biological sequence compression in general is the focus on statistical compression methods. As seen from the practical results from existing algorithms, simple Markovian models clearly are not adequate. The statistical significance of repetitions may not be enough to build an adequate model for the symbol probabilities. Further, most methods seem to perform simple pattern searches, while biological sequences are known to have various types of repetition, whether one considers approximate or exact repetition. More importantly, the methods tend to ignore the fact that biological phenomena such as gene duplication could lead to long-range genome-wide correlation in rotein sequences, with possible large-scale duplications occurring between different chromosomes. Our approach is to exploit these genome-wide long-range correlations in protein sequences.

3. Long-Range Correlation in Protein Sequences

The coding regions in genomic sequences are generally less repetitive than the non-coding regions [11, 21]. Thus, it is widely agreed that protein sequences generally have more entropy, and hence exhibit more randomness than general DNA sequences. A repetition-based complexity profile has recently been proposed for prokaryotic sequences [29]. In fact, protein has been touted as being incompressible [23]. In studying the SCP (sorted common prefix) statistics of genomic sequences, we bumped unto an unusual observation: an unprecedented redundancy in protein sequences! (See Table 3, compare with Table 2). The protein sequences[†] were the same used by Nevill-Manning and Witten in the "*Protein is Incompressible*" paper [23]. Each of the four files in the corpus contains a concatenated sequence of **all** the proteins in the genome of one single organism (except for that of *H.Sapiens*, which is not complete). The observation is the unusually high values of K_{max} in the SCP statistics for the protein sequences. (K_{max} is the maximum common prefix for a given sequence). This means that, certain (set of) genes in some area of the genome are repeated *in exactly the same order* at some other point(s), further down the genome. Apart from the mere size of the repeated subsequences (as given by K_{max}), an equally important aspect is the relatively long range of separation between the repeats. For *H. sapiens* and *S. cerevisiae*, the occurrences with K_{max} are separated by more than 350,000 protein symbols (or > 1,050, 000 base pairs). Since the protein sequences are concatenated, this means that the repetition could involve genes located in different chromosomes. For protein sequences (see Table 3), we can observe the overlapping nature: K_{max} tends to be larger than the difference between the starting indices.

To our knowledge, this scale of redundancy has never been observed in protein sequences of a genome. Although multiple gene copies and repeated histone clusters are known to be present in most eukaryotic genomes [16], their number and their size are not enough to explain the above observation. More importantly, the ***orders*** in which the genes are arranged in the genome tend to be conserved as they are being repeated at a different location along the genome. Apparently, this redundancy has not been previously observed with computational methods, because protein sequences are not usually considered in the way it was considered by Nevill-Manning and Witten [23]. The reason could come from the following: (I) Most computational analysis of protein sequences treat each gene independent of the other, and not in the

[†] Available at: http://www.data-compression.info/Corpora/ProteinCorpus/proteincorpus.htm

concatenated form as was done in [23]. (II) They are usually based on protein sequences from different species, and not the complete set of protein sequences from a large genome, such as the human genome. (III) They do not expect to find such long range repetitions in protein, and hence, choice of parameters may exclude such unusual cases. (IV) Most repeat finding algorithms do not consider overlapping repeats. (V) Large complete genomes have only recently become available. Thus, simple methods to find repeats would fail to identify such long-range redundancies that could span chromosomal boundaries.

Table 2: SCP Statistics for Common DNA Sequences

Seq	size, u	Kmax	Kmax/u	Start Index1	Start Index2	Diff
A	172280	32442	0.188	12145	15217	3072
B	66495	408	0.00614	8572	1958	6614
C	186609	191	0.00102	121038	38876	82162
D	316613	1682	0.00531	12335	199487	187152
E	1531929	3573	0.00233	528048	531933	3885
F	4638690	2815	0.00061	4165800	4207196	41396

A: humEpsBarr. B: HUMGHCSA. C: MitoMPOMTCG.
D: YSCCHRIII. E: YSCCHRIV. F: E.coli.

Table 3: SCP Statistics for Concatenated Protein Sequences

Seq	size, u	# of genes	Kmax	Kmax/u	Start Index1	Start Index2	Diff
HI	448770	1740	220685	0.492	53200	8	53192
MJ	509508	1680	343105	0.673	34899	3	34896
SC	2900346	8220	886531	0.306	480296	29	480267
HS	3295749	5733	392004	0.119	358676	24	358652

HI: *H. influenzae*; MJ: *M. jannaschii*; HS: *H. sapiens*; SC: *S. cerevisiae*
Diff=|Index1-Index2|

This observation goes against the grain of conventional assumptions about protein sequences in the data compression community. The biological implications are numerous, for instance in the regulation and control of gene expressions. Protein sequences from the same genome could exhibit very long-range periodicities. But these are interrupted and masked by the introns – the non-coding areas in the genome. One could use this as a basis for a check for correct splice sites of subsequent genes, (after the correct identification of some initial genes). There could also be some evolutionary implications: nature not only conserves the important genes by replications in the same genome, but it also conserves their orders**.** Biological and evolutionary evidence of the prevalence of genome-wide gene duplications and their implications have been studied in [12, 17].

3.1 Efficient analysis of sequence correlation

We use the SCP data structure described in [1] as the basic structure to analyze the protein sequences. Let T be an input sequence of length $u = |T|$. Suppose we pre-compute the longest-common-prefix (LCP) between *all* pairs of the sorted suffixes from a sequence, T. Using the relationship between the BWT [5] and the suffix tree, in addition to the auxiliary arrays previously described in [2, 4], we can obtain these sorted suffixes in linear time. We store this information in a table. Since the table is based on the sorted suffixes, we call this the **sorted common prefix (SCP)**. Although the SCP and traditional LCP store the same basic information, the structure of the SCP is completely different. Also, the sorted nature of the suffixes for the SCP has implications in the computation of this table, and its diverse uses. Given the i-th and j-th sorted suffixes, (SS_i and SS_j respectively), if SCP(i,j)=k, it means that the first k positions in the i-th suffix and the j-th suffix are exact matches (i.e. $SS_i[1...k] = SS_j[1...k]$, and $SS_i[k+1] \neq SS_j[k+1]$). We observe the following properties about the SCP structure. Let i,j,k, be indices for the sorted suffixes, such that $i<j$, and $j<k$. Then, $SCP(i,j) \geq SCP(i,k), \forall k > j$. Also, if $SCP(i,j) \geq 0$ and $SCP(i,k) = 0$, then $\forall k > j$, $SCP(j,k) = 0$. More generally, let $x = SCP(i,k)$. Then, $\forall k > j$, $SCP(j,k) = SCP(i,k) + SCP(SS_j[x+1,...,u], SS_k[x+1,...,u])$ $= x + SCP(SS_j[x+1,...,u], SS_k[x+1,...,u])$. The SCP is symmetric: SCP (j,k)= SCP(k,j). It is also usually sparse, although not always. Example, with $T = AAAAA$, we will have a full SCP table, where the row entries are of the form **1 1 1 1**; **2 2 2**; **3 3**; **4**. Essentially, any pair of suffixes with an SCP value different from 0 has some repetition which may or may not lead to a compression gain. Further, the block nature of the SCP means that we can compute the SCP for each symbol block, without reference to the other symbol blocks, leading to a lower complexity in the calculations.

Fig. 2 shows the different forms of large scale repetitions (gene duplications) we observed in the protein sequences. S_1 is the repeated pattern. Some repetitions are overlapped with previous repetitions (case

A), or are completely contained in some previous repetition (case B), while others could occur thousands of amino acids down the protein sequence (case C). In some other cases, the repeated subsequences could span two different adjourning repetitions (case D). We observed that in all the sequences in the protein corpus, the large scale repetitions tend to be tandem duplications.

In [1], it was shown that, for a fixed alphabet, the SCP can be computed using an $O(u)$ number of comparisons. We, however, note that the compression results are independent of ***how*** we extract the long range tandem duplications or other repetitions in the sequence. The important issue is that we extract and exploit them in the compression. But the compression time clearly depends on the method we use to extract these.

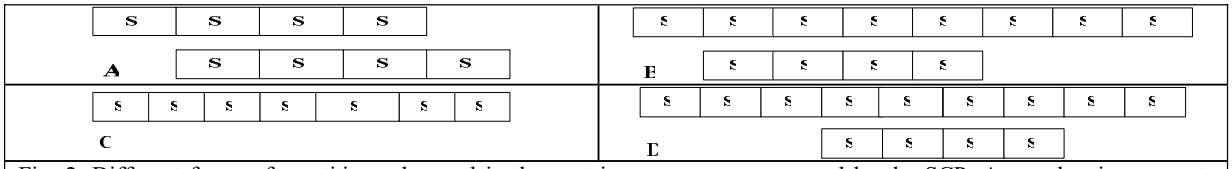


Fig. 2: Different forms of repetitions observed in the protein sequences, as exposed by the SCP. A: overlapping repeats (pattern S_1 occurred 5 times); B: One subsequence covers the other subsequence. We break them down into smaller sequences S_1; C: repeated sequences are separated by possibly long stretches of amino acids between them. This is the most common case; D: The triple overlap case. We break the overlaps down into subsequence S_1.

4. Compressing Protein Sequences

Fig. 3 is a flowchart of the proposed compression algorithm. After each parse, we obtain two outputs – the dictionary items and the remaining sequences. The dictionary consists of information about the repeated patterns, such as repetition length, position number of occurrence, etc, needed for later decoding. Since both of these may contain redundancy, to achieve maximum compression gain, we pass these two to the core algorithm again until no compression could be achieved. For two repeating sequences that overlap (cases A, B and D in Fig. 2), we cannot replace one of them without cutting the length of the second. In this scenario, we will construct a shorter subsequence to represent these two, as shown in Fig. 2. After identifying the various forms of correlation in the sequence, the next problem becomes how to use these to compress the protein sequence. This will typically involve performing some form of substitution using the discovered repetition structures. Thus, we need to parse the input sequence to indicate the positions of the repeats and where and how they are to be substituted. We use an off-line dictionary based approach. First, we remove the repeats from the original sequence, and move them into an off-line dictionary (or index) of repeats. Then we code all occurrences of the repeat with reference to the position of the repeat in the dictionary. The size of the pointers is likely to be generally smaller than when we use on-line dictionary (especially with absolute references), since the number of repeats should be much smaller than the size of the input sequence. But we have to compress and send the off-line dictionary as part of the compressed string. To guarantee compression, we can enforce a condition that whenever an item is inserted in the dictionary, it should not lead to an expansion of the data. We also have to consider whether referencing using the pointers should be relative or absolute.

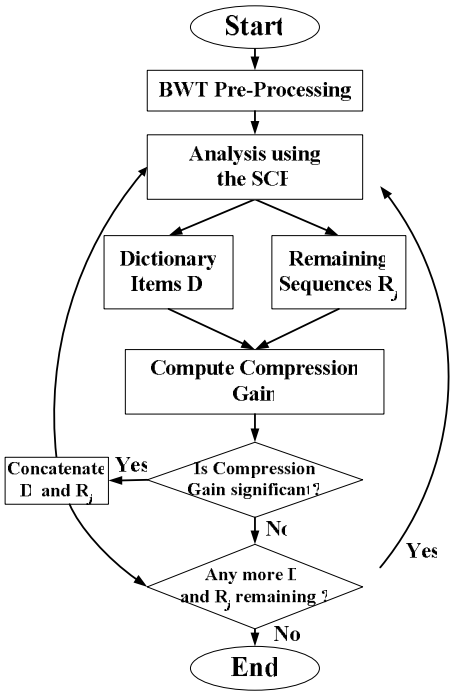


Fig. 4: Flowchart for proposed compression algorithm.

4.2 Parsing and encoding

Our approach here is to use the vocabulary parsing scheme (**vps**) proposed in [2]. For simplicity we use the **VPS1 algorithm**, which provides offline dictionary compression with pointers in the dictionary. The compression performance will depend on a number of factors, such as: the size of the pointers and the way they are coded; the type of referencing used - absolute or relative; the size of the dictionary (i.e. number of distinct repetition structures); and the type of referencing used in the dictionary (if any).

Under algorithm VPS1, we remove each repeated substring from the input sequence, and move it to an external dictionary. In the dictionary, we record the positions in the sequence where each repetition occurred, along with the repetition type. Thus there is no reference or pointer information in the original sequence. The parsed string is just a concatenation of the remaining subsequences after the repeats have been removed. To capture the case of different repetition types, we include repetition codes for each type of repetition in the dictionary.

Example. Consider the following sample sequence, **S**:

	P_1		P_2		P_3		P_4		P_5		P_6		P_7
x_1	MMCTGTCMM	x_2	MM	x_3	GTCMM	x_4	TG	x_5	MMCTG	x_6	TTGMCMGTT	x_7	MM

When we move the repeated sequences into the dictionary, the remaining parsed sequence will be:

Parse(S) : $x_1x_2x_3x_4x_5x_6x_7$

The x_i's represent some other parts of the sequence that are not included in the repetition structure. Thus, the parsed sequence from Algorithm VPS1 is very simple. The dictionary however is a little more complicated. Using the following notations: r_i = i-th repetition pattern, $l(r)$=$|r|$=length of repeat pattern r, $\eta(r)$ = total number of occurrences of r, $t(r)$ = repetition code for current occurrence of r, we can represent the dictionary structure for the sample sequence as follows:

index	Repeat Pattern	$l(r)$	$t(r)$	$\eta(r)$	positions
1	MMCTGTCMM	9	1	1	P_1
			3	1	P_6
2	GTCMM	5	1	1	P_3
			2	1	P_5
3	MM	2	1	2	P_2, P_7
4	TG	2	1	2	P_4

$t(r)$: 1: direct repeat; 2: reverse repeat; 3: complimented palindrome

The performance of the above scheme depends critically on the internal representations used for the dictionary. We use the term ***vocabulary*** to refer to the ensemble of repeat structures without reference to their specific locations in the sequence. For simplicity, we consider only the case with one type of repeat. The analysis can be extended to the general case with different types of repeats. We analyze the expected compression gain for the offline scheme using the above dictionary organization. We note that we do not need to explicitly encode $l(r)$ and $\eta(r)$ since these can be computed on the fly. By using self-delimiting and uniquely decodable codes, we avoid direct coding of the delimiters in the above representations.

We use the following additional notations: $P_{r,j}$ = position of the j-th occurrence of repeat pattern r; κ= dictionary size (i.e. number of distinct repetitions); u = |S| = size of the sequence, Σ = input alphabet, ($|\Sigma|$=20 for protein sequences); $\beta = \lceil \log|\Sigma| \rceil$ = number of bits required to code each amino acid[‡]; $b(n) = \lceil \log n \rceil$= number of bits required to represent an integer n; $C(X)$ cost of coding sequence X; $L = \max_i \{l(r_i)\}$- the maximum length of the repeats; **0** and **1** are flag bits.

Vocabulary Encoding:

Vocabulary: $r_1\ \mathbf{0}\ r_2\ \mathbf{0} \ldots \mathbf{0}\ r_\kappa\ \mathbf{0}$

Positions:

$$P_{1,1}, P_{1,2}, \ldots P_{1,\eta(1)}\ \mathbf{0}\ P_{2,1}, P_{2,2}, \ldots P_{2,\eta(2)}\ \mathbf{0} \ldots \mathbf{0} P_{\kappa,1}, P_{\kappa,2}, \ldots P_{\kappa,\eta(\kappa)}\ \mathbf{0}$$

Using the above representation, we have the following:

[‡]All logarithms are to **base 2**, unless otherwise noted.

Cost of **original sequence**: $C(S) = |S|.\lceil \log|\Sigma| \rceil = u.\beta$

Cost of **parsed sequence** (i.e. remaining sequence after removing repeats):

$$C(parse(S)) = u\beta - \beta\sum_{i=1}^{\kappa} l(r_i)\eta(r_i)$$

Cost of **vocabulary**: $C(V_A(S)) = \beta_1 \sum_{i=1}^{\kappa}(l(r_i)+1) = \kappa\beta_1 + \beta_1 \sum_{i=1}^{\kappa} l(r_i)$, where $\beta < \beta_1 \leq \beta + 1$.

Essentially, the addition of the flag bit **0** forms a new alphabet with the original alphabet Σ.

Cost of **positions**: $C(Posn(S)) = \sum_{i=1}^{\kappa}\sum_{j=1}^{\eta(i)} b(P_{i,j}) + \sum_{i=1}^{\kappa} 1 = \kappa + \sum_{i=1}^{\kappa}\sum_{j=1}^{\eta(i)} \lceil \log P_{i,j} \rceil$

The cost of dictionary representation is then the combined cost of the vocabulary and the positions:

$$C(D(S)) = C(V_A(S)) + C(Posn(S))$$

Compression Gain:

$$G(S) = C(S) - [C(D(S)) + C(Parse(S))]$$

$$= u.\beta - \left[\left(\kappa\beta_1 + \beta_1\sum_{i=1}^{\kappa} l(r_i) + \kappa + \sum_{i=1}^{\kappa}\sum_{j=1}^{\eta(i)} \lceil \log P_{i,j} \rceil\right) + \left(u\beta - \beta\sum_{i=1}^{\kappa} l(r_i)\eta(r_i)\right)\right]$$

With $\beta_1 = \beta + 1$, we have an underestimation of the gain:

$$G(S) = \beta\sum_{i=1}^{\kappa} l(r_i)(\eta(r_i)-1) - \kappa(\beta+2) - \sum_{i=1}^{\kappa} l(r_i) - \sum_{i=1}^{\kappa}\sum_{j=1}^{\eta(i)} \lceil \log P_{i,j} \rceil$$

Using the compression gain above, we can reduce the time required for parsing and computation of compression gain by performing the required computations based on the length of the repeat. We see that with $\kappa = 1$, the minimal length that can guarantee a positive compression gain will be given by:

$$l(r) \geq \frac{\eta(r)\lceil \log u \rceil + \beta + 2}{\beta(\eta(r)-1) - 1}$$

Thus, for a protein sequence with 1 million symbols, say, using the simplest case of repeats that appear at least 2 times, we get a minimal length for the repeat sequence to be 14.

4.3 Multi-level hierarchical decomposition

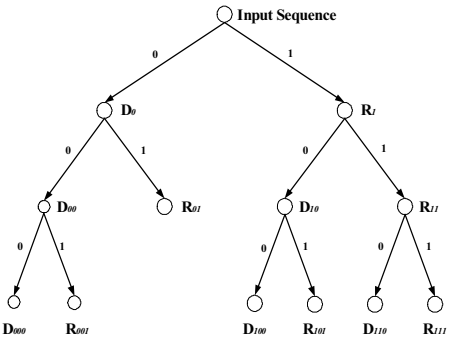


Fig.2: Multilevel hierarchical decomposition

To further exploit the potential redundancy, we perform the parsing and decomposition in a hierarchical manner. That is, after the first parse, the remaining sequence is used again as the input to the parsing algorithm to discover other potential correlations that may have been missed by the previous parsing stages. Similarly, given that the dictionary entries could be very long (potentially tens of thousands of amino acids, see Table 3), we also feed the dictionary entries for further decomposition and parsing. The result is a recursive multi-level decomposition strategy, (see Fig. 4). At each level, the input sequence is decomposed into two components, the dictionary D_i and the remaining sequence R_j. Both D_i and R_j are further passed to the parsing algorithm for further decomposition. Decomposition stops when the compression gain $G(S)$ for the input sequence is negative or less than a threshold. Each internal node except the root in the tree is a provisional output. Each leaf in the tree is part of the final output. For instance, R_{01} in Fig. 4 is terminated as a leaf in the tree because we cannot achieve any more compression gain by decomposing it further. Thus, it is coded without further parsing. It is obvious that the remaining sequences *R* typically require more rounds of passing than the dictionary items *D,* since the remaining sequences normally have longer lengths.

The compression results typically improve with higher levels of decomposition. Also, the time required for parsing and compression gain analysis, decreases very fast as the levels of decomposition increase, although the overall time increases with more decomposition levels.

4.4. Results

Table 4 shows the statistics of maximal repeats for the protein sequence and compression results using the above parsing scheme, and the observed long-range correlations in the sequences. Table 5 shows a comparison with the results from Table 1 for the biological sequence compression algorithms on the same sequence. We have ignored the bits required to represent the special position symbols used to indicate gene region separators in the protein sequences (usually less than 100 bytes per sequence).

Table 4: Statistics of maximal repeats and compression results using the observed long-range correlations.

	Size	Kmax	repeat length $l(r)$	number of occurrence $\eta(r)$	Compression Results (bps)
HI	509508	220685	34896	7	2.546
MJ	448770	343105	53192	5	2.273
SC	2900346	886531	406239	3	3.111
HS	3295749	392004	338359	3	3.435

Table 5: Comparative compression performance using the observed long-range correlations. Compression results in bits/symbol (smaller values imply better performance).

	Gen Compress	CP(0)	CP(1)	CP(2)	CP(3)	lzaCTW (8)	Block Code	Proposed Method
HI	4.156	4.156	4.149	4.146	4.143	4.118	3.665	2.546
MJ	4.062	4.068	4.06	4.056	4.051	4.028	5.102	2.273
SC	3.97	4.163	4.158	4.152	4.146	3.952	5.175	3.111
HS	3.972	4.133	4.126	4.12	4.112	3.920	5.087	3.435

The superior performance of the proposed method is evident in the above tables. We notice that the improved performance is consistent across all the different protein sequences. This is expected, given the consistently high values of K_{max} in Table 3. It was observed that the long range correlations that produced the highest compression gains are often due to large tandem duplications, with thousands of symbols in the primitive pattern.

In terms of compression time, after the SCP is computed, the parsing and compression stage take about 7 minutes for MJ sequence, on a dual-Athlon 1.8 GHz processor server running Ubuntu Linux 5.10.

5. Conclusion

We have considered the problem of compressibility of protein sequences. Based on the observation of long-range genome scale correlations in protein sequences, we proposed a simple scheme to exploit such correlations. Using a method based on the sorted common prefix we identify the correlated protein sequences, and then use a simple dictionary-based parsing and encoding scheme to provide compression. The results show that the approach can provide a consistent compression of the protein sequences, at times down to less than 2.3bps. The improved performance was observed on all the protein sequences in the Protein Corpus.

References

[1] Adjeroh D.A., and Feng J., "The SCP and compressed domain analysis of biological sequences", *Proc., IEEE Bioinformatics Conference*, August 2003, pp. 587-592.

[2] Adjeroh D.A., Zhang Y, Mukherjee A., Powell M., and Bell , T.C., "DNA sequence compression using the Burrows-Wheeler Transform ", *Proc., IEEE Bioinformatics Conference*, August 2003, pp. 303-313, 2002

[3] Apostolico A and Lonardi S, "Offline compression by greedy textual substitution", *Proceedings of the IEEE*, 88 (11), 1733--1744, 2000

[4] Bell T.C, Powell M., Mukherjee A. and Adjeroh D. A. "Searching BWT compressed text with the Boyer-Moore algorithm and binary search", *Proc., IEEE Data Compression Conference*, Snowbird, Utah, April 2 - 4, 2002.

[5] Burrows M. and Wheeler D.J., "A block-sorting lossless data compression algorithm", *Technical Report*, Digital Equipment Corporation, Palo Alto, CA, 1994.

[6] Chen X., Kwong S.and Li M, "A compression algorithm for DNA sequences and its applications in genome comparison", *Proc., 10th Workshop on Genome Informatics (GIW'99)*, pp. 52--61, 1999.

[7] J.G. Cleary and I.H. Witten, "Data compression using adaptive coding and partial string matching", *IEEE Transactions on Communication*, 32(4), 396-402, 1984
[8] Clote P and Backofen R, *Computational Molecular Biology*, Wiley, 2002
[9] Cutler s and McCourt P, "Dude, where's my phenotype dealing with redundancy in signaling networks", *Vision Statement of Plant Physiology*. 138:558-559, 2005
[10] Fruton, J.S, *Proteins, Enzymes, Genes: The Interplay of Chemistry and Biology*, Yale University Press, 1999
[11] Gatlin, L, *Information Theory and the Living System*, Columbia University Press, New York, 1972.
[12] Gibson, T. J. and Spring, J. "Genetic redundancy in vertebrates: polyploidy and persistence of genes encoding multidomain proteins." *Trends Genet*. 14:46–49.1998
[13] Grumbach S and Tahi F., "A new challenge for compression algorithms: genetic sequences", *Info. Processing & Management*, 30: 875-886, 1994.
[14] Gu Z, Steinmetz L, Gu X, Scharle, C, Dacis, R and Li W. "Role of duplicate genes in genetic robustness against null mutations". *Nature* 421:63-66 2003
[15] Gu Z, Cavalcanti A, Chen F, Bouman, P and Li W. "Extent of gene deplication in the genomes of drosophila, nematode, and yeast". *Molecular Biology and Evolution*, 19(3):256-262 2002.
[16] Hawkins JD, *Gene Structure and Expression*, Cambridge University Press, 1985.
[17] M. Hurles, "Gene duplication: The genomic trade in spare parts", *PLoS Biology*, 2(7).e205:S13-S19, 2004.
[18] Korodi G and Tabus I, "An efficient normalized maximum likelihood algorithm for DNA sequence compression", *ACM Transactions on Information Systems*, 23(1):3-34 2005
[19] Lanctot J. K., Li M. and Yang E., "Estimating DNA sequence entropy", *Proc., 11th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA2000)* , pp. 409-418, 2000.
[20] Li M and Vit'anyi P, *An Introduction to Kolmogorov Complexity and its Applications*, Springer--Verlag, 1993.
[21] Loewenstern, D. and Yainilos, P., "Significantly lower entropy estimates for natural DNA sequences", *Proc., IEEE Data Compression Conference*, Snowbird, UT, pp. 151-161, 1997.
[22] Matsumoto T., Sadakane K. and Imai H, "Biological sequence compression algorithms", Department of Information Science, University of Tokyo, 2000 . http://citeseer.nj.nec.com/457261.html
[23] Nevill-Manning C.G and Witten I.H. "Protein is incompressible", *Proc., the IEEE Data Compression Conference*, 257-269, 1999
[24] Normanly J and Barteltl B, "Redundancy as a way of life - IAA metabolism", *Current Opinion in Plant Biology*, 2:207-213, 1999
[25] Nowak, M. A., Boerlijst, M. C., Cooke, J. and Smith, J.M, "Evolution of genetic redundancy", *Nature* 388:167–171.1997
[26] Maier D, Marte BM, Schafer W, Yu Y and Preiss A, "Drosophila evolution challenges postulated redundancy in the E(spl) gene complex", *Proceedings of the National Academy of Sciences*, 90:5464-5468.1993
[27] Rival, E., Delgrange, O., Delahaye, J.P., Dauchet, M., Delorme, M. Henaut, A. and Ollivier, E., "Detection of significant patterns by compression algorithms: the case of approximate tandem repeats in DNA sequences", *CABIOS*, 13: 131-136, 1997
[28] Sampath G, "A block coding method that leads to significantly lower entropy values for the proteins and coding sections of Haemophilus influenzae", *Proc., IEEE Bioinformatics Conference*, August 2003, pp.287-293.
[29] Troyanskaya OG, Arbell O, Koren Y, Landau GM, and Bolshoy A, "Sequence complexity profiles of prokaryotic genomic sequences: A fast algorithm for calculating linguistic complexity", *Bioinformatics*, 18 (5), 679-688, 2002;