A Simple Statistical Algorithm for Biological Sequence Compression

Minh Duc Cao Trevor I. Dix Lloyd Allison
Chris Mears
Faculty of Information Technology,
Monash University, Australia
Email: {minhc,trevor,lloyd,cmears}@infotech.monash.edu.au

Abstract

This paper introduces a novel algorithm for biological sequence compression that makes use of both statistical properties and repetition within sequences. A panel of experts is maintained to estimate the probability distribution of the next symbol in the sequence to be encoded. Expert probabilities are combined to obtain the final distribution. The resulting information sequence provides insight for further study of the biological sequence. Each symbol is then encoded by arithmetic coding. Experiments show that our algorithm outperforms existing compressors on typical DNA and protein sequence datasets while maintaining a practical running time.

1. Introduction

Modelling DNA and protein sequences is an important step in understanding biology. Deoxyribonucleic acid (DNA) contains genetic instructions for an organism. A DNA sequence is composed of nucleotides of four types: adenine (abbreviated A), cytosine (C), guanine (G) and thymine (T). In its double-helix form, two complementary strands are joined by hydrogen bonds joining A with T and C with G. The reverse complement of a DNA sequence is also considered when comparing DNA sequences. Certain regions in a DNA sequence are translated to proteins, which control the development of organisms. The alphabet of protein sequences consists of 20 amino acids, each of which is determined by a triplet of nucleotides called a codon.

The amount of DNA sequenced from organisms is increasing rapidly. Compression of biological sequences is useful, not primarily for managing the genome database, but for modelling and learning about sequences. Work by Stern et al. [21] recognizes the importance of mutual compressibility for discovering patterns of interest from genomes. Chen et al. [6] and Powell et al. [19] show that compressibility is a good measurement of relatedness between sequences and can be effectively used in sequence alignment and evolutionary tree construction.

Since DNA is the "instruction of life", it is expected that DNA sequences are not random and should be compressible. Some DNA sequences are highly repetitive. It is estimated that 55% of the human genome is repeat DNA. A repeat subsequence is a copy of a previous subsequence in the genome, either forward or reverse complement. Most DNA repeats are not exact as nucleotides can be changed, inserted or deleted. As an example, the ALU family are repeats of length about 300 bases, and any one is only about 87% similar to a consensus sequence.

Interestingly, most general purpose text compression algorithms fail to compress DNA to below the naive 2 bits per symbol. That is because DNA regularities are different from those in text and are rarely modelled by those compressors. A number of special purpose compression algorithms for DNA have been developed recently. Most of these search for repeat subsequences and encode them by reference to a previous instance. As a DNA subsequence could be (approximately) repeated many times, using information from many of those repeat positions is expected to give better compression ratios.

In this paper, we present the *expert model* (XM) and an algorithm for biological sequence compression. Our compressor encodes each symbol by estimating the probability based on information obtained from previous symbols. If the symbol is part of a repeat, the information from one or more previous occurrences is used. Once the symbol's probability distribution is determined, it is encoded by a primary compression algorithm such as arithmetic coding.

This paper is organized as follows. Section 2 reviews current research on biological compression. Our expert model is described in section 3 and experimental results are presented in section 4. Finally, section 5 concludes our work.

2. Background

Most compression algorithms fall into one of two categories, namely *substitutional* compression and *statistical* compression. Those in the former class replace a long repeated subsequence by a pointer to an earlier instance of the subsequence or to an entry in a dictionary. Examples of this category are the popular Lempel-Ziv compression algorithms [25, 26] and their variants. As DNA sequences are known to be highly repetitive, a substitutional scheme is a natural approach to take. Indeed, most DNA compressors to date are in this category.

On the other hand, a statistical compression encoder such as *prediction by partial match* (PPM) [8] predicts the probability distribution of each symbol. Statistical compression algorithms depend on assumptions about how the sequence is generated to calculate the distribution. These assumptions are said to be the *model* of the sequence. If the model gives a high probability to the actual value of the next symbol, good compression is obtained. A model that produces good compression makes good predictions and is a good description of the data.

The earliest special purpose DNA compression algorithm found in the literature is *BioCompress* developed by Grumbach and Tahi [11]. BioCompress detects an exact repeat in DNA using an automaton, and uses Fibonacci coding to encode the length and position of its previous location. If a subsequence is not a repeat, it is encoded by the naive 2 bits per symbol technique. The improved version, *BioCompress-2* [12] uses a Markov model of order 2 to encode non-repeat regions. The *Cfact* DNA compressor developed by Rivals et al. [20] also searches for the longest exact repeats but is a two-pass algorithm. It builds the suffix tree of the sequence in the first pass, and does the actual encoding in the second pass. Regions not repeated are also encoded by 2 bits per symbol. The *Off-line* approach by Apostolico and Lonardi [3] iteratively selects repeated substrings for which encoding would gain maximum compression.

A similar substitution approach is used in *GenCompress* by Chen et al. [6] except that approximate repeats are exploited. An inexact repeat subsequence is encoded by a pair of integers, as for *BioCompress-2*, and a list of edit operations for mutations, insertions and deletions. Since almost all repeats in DNA are approximate, *GenCompress* obtains better compression ratios than *BioCompress-2* and *Cfact*. The same compression technique is used in the *DNACompress* algorithm by Chen et al. [7], which finds significant inexact repeats in one pass and encodes these in another pass.

Most other compression algorithms employ similar techniques to *GenCompress* to encode approximate repeats. They differ only in the encoding of non-repeat regions and in detecting repeats. The *CTW+LZ* algorithm developed by Matsumoto et al. [16] encodes significantly long repeats by the substitution method, and encodes short repeats and non repeat areas by context tree weighting [23]. At the cost of time complexity, *DNAPack* Behzadi and Fessant [4] employs a dynamic programming approach to find repeats. Non-repeat regions are encoded by the best choice from an order 2 Markov model, context tree weighting, and naive 2 bits per symbol methods.

Several DNA compression algorithms combine substitution and statistical styles. An inexact repeat is encoded using (i) a pointer to a previous occurrence and (ii) the probabilities of symbols being copied, changed, inserted or deleted. In the *MNL* algorithm by Tabus et al. [22] and its improvement, *GeMNL* by Korodi and Tabus [14], the DNA sequence is split into fixed size blocks. To encode a block, the algorithm searches the history for a regressor, which is a subsequence having the minimum Hamming distance from the current block, and represents it by a pointer to the match as well as a bit mask for the differences between the block and the regressor. The bit mask is encoded using a probability distribution estimated by the normalized maximum likelihood of similarity between the regressor and the block.

Probably the only two pure statistical DNA compressors published so far are *CDNA* by Loewenstern and Yianilos [15] and *ARM* by Allison et al. [2]. In the former algorithm, the probability distribution of each symbol is obtained by approximate partial matches from history. Each approximate match is with a previous subsequence having a small Hamming distance to the context preceding the symbol to be encoded. Predictions are combined using a set of weights, which are learnt adaptively. The latter ARM algorithm forms the probability of a subsequence by summing the probabilities over all explanations of how the subsequence is generated. Both these approaches yield significantly better compression ratios than those in the substitutional class and can also produce information content sequences. *CDNA* has many parameters which do not have biological interpretations. Both are very computationally intensive.

The expert model presented in this paper is a statistical algorithm. The encoder maintains a panel of experts and combines them for prediction but a much simpler and computationally cheaper mechanism is used than in those above. The framework allows any kind of expert to be used, though we report here only experts obtained from statistics and repetitivenes of sequences. Weights for expert combination are based on expert performance. Our compressor is found to be superior to any compression algorithms to date and its speed is practical. The algorithm is capable of biological knowledge discovery based on per element information content sequences [10]. This is a purpose of our compressibility research.

3. Algorithm description

As a statistical method, our XM algorithm compresses each symbol by forming the probability distribution for the symbol and then using a primary compression scheme to code it. The probability distribution at a position is based on symbols seen previously. Correspondingly, the decoder, also having seen all previous decoded symbols, is able to compute the identical probability distribution and can recover the symbol at the position.

In order to form the probability distribution of a symbol, the algorithm maintains a set of experts, whose predictions of the symbol are combined into a single probability distribution. An expert is any entity that can provide a probability distribution at a position. Expert opinions about a symbol are blended to give a combined prediction for the symbol.

The statistics of symbols may change over the sequence. One expert may perform well on some region, but could give bad advice on others. A symbol is likely to have similar statistical properties to the context surrounding, particularly the context preceding the symbol. The reliability of an expert is evaluated from its recent predictions. A reliable expert has high weight for combination while an unreliable one has little influence on the final prediction or may be ignored.

3.1. Type of experts

An expert can be anything that provides a reasonably good probability distribution for a position in the sequence. A simple expert can be a Markov model (*Markov expert*). An order-k Markov expert gives the probability of a symbol in a position given k preceding symbols. Initially, the Markov expert does not have any prior knowledge of the sequence and thus gives a uniform distribution to a symbol. The probability distribution adapts as the encoding proceeds. Essentially, the Markov expert provides the background statistical distribution of symbols over the sequence. Here we use an order-2 Markov expert for DNA, and order-1 for protein.

Different areas of a DNA sequence may have differing functions and thus may have different symbol distributions. Another type of expert is the *context Markov expert*, whose probability distribution is not based on the entire history of the sequence but on a limited preceding context. In other words, the context Markov expert bases its prediction on the local statistics. The context Markov expert currently used by XM is order-1 with a context of 512 previous symbols.

The compressibility of biological sequences comes mainly from repeated subsequences. Therefore, it is important to include experts that make use of this feature. XM employs a *copy expert* that considers the next symbol to be part of a copied region from a particular offset. A copy expert with offset f suggests that the symbol at position i is likely to be the same as the symbol at position $i - f$.

A copy expert does not blindly give a high probability to its suggested symbol. It uses an adaptive code [5], over some recent history, for correct/incorrect predictions. The copy expert gives a probability to its predicted symbol of:

$$p = \frac{r+1}{w+2} \tag{1}$$

where w is the window size over which the expert reviews its performance and r is the number of correct predictions the expert has made. The remaining probability, $1 - p$, is distributed evenly to the other symbols in the alphabet.

For complementary reverse repeats, a similar *reverse expert* is used. This works exactly the same as the copy expert, except that it suggests the complementary symbol to the one from the earlier instance and it proceeds in the reverse direction.

3.2. Proposing experts

At position i of the sequence, there are $O(i)$ possible copy and reverse experts. This is too many to combine efficiently and anyway most would be ignored. To be efficient, the algorithm must use at most a small number of copy and reverse experts at any one time. We currently employ a simple hashing technique to propose likely experts. Every position is stored in a hash table with the hash key composed of h symbols preceding the position. If there is an opening for a new expert at any point, the hash table is consulted.

3.3. Combining expert predictions

The core part of our XM algorithm is the evaluation and combination of expert predictions. Suppose a panel of experts E is available to the encoder. Expert θ_k gives the prediction $P(x_{n+1}|\theta_k, x_{1..n})$ of symbol x_{n+1} based on its observations of preceding n symbols. A sensible way to combine experts' predictions is based on Bayesian averaging:

$$\begin{aligned} P(x_{n+1}|x_{1..n}) &= \sum_{k \in E} P(x_{n+1}|\theta_k, x_{1..n}) w_{\theta_k,n} \\ &= \sum_{k \in E} P(x_{n+1}|\theta_k, x_{1..n}) P(\theta_k|x_{1..n}) \end{aligned} \tag{2}$$

In other words, the weight $w_{\theta_k,n}$ of expert θ_k for encoding x_{n+1} is the posterior probability $P(\theta_k|x_{1..n})$ of θ_k after encoding n symbols. $w_{\theta_k,n}$ can be estimated by Bayes's theorem:

$$\begin{aligned} w_{\theta_k,n} &= P(\theta_k|x_{1..n}) \\ &= \frac{\prod_{i=1}^{n} P(x_i|\theta_k, x_{1..i-1}) P(\theta_k)}{\prod_{i=1}^{n} P(x_i|x_{1..i-1})} \end{aligned} \tag{3}$$

If we assume that every expert has the same prior probability $P(\theta_k)$ then normalizing equation 3 by a common factor M we have:

$$w_{\theta_k,n} = \frac{1}{M} \prod_{i=1}^{n} P(x_i|\theta_k, x_{1..i-1}) \tag{4}$$

The normalization factor M, in fact does not matter as equation 2 could be again normalized to have $\sum P(x_{n+1}|x_{1..n}) = 1$. Take the negative log of equation 4 and ignore the constant term:

$$-log_2(w_{\theta_k,n}) \sim -\sum_{i=1}^{n} log_2 P(x_i|\theta_k, x_{1..i-1}) \tag{5}$$

Since $-log_2 P(x_i|\theta_k, x_{1..i-1})$ is the cost of encoding symbol x_i by expert θ_k, the right hand side of equation 5 is the length of encoding of subsequence $x_{1..n}$ by expert θ_k. As we want to evaluate experts on a recent history of size w, only the message length of encoding symbols $x_{n-w+1..n}$ is used to determine weights of experts. We find that, the algorithm works best when negative log 2 of the expert weight varies as three times the average code length over a window of size $w = 20$:

$$\begin{aligned} -log_2(w_{\theta_k,n}) &\sim -\frac{3}{w} \sum_{i=n-w+1}^{n} log_2 P(x_i|\theta_k, x_{1..i-1}) \\ &= 3AveMsgLen(x_{n-w+1..n}|\theta_k) \end{aligned} \tag{6}$$

or

$$w_{\theta_k,n} \propto 2^{-3AveMsgLen(x_{n-w+1..n}|\theta_k)} \tag{7}$$

Suppose there are three hypotheses about how a symbol is generated: by the distribution of the species genome; by the distribution of the current subsequence; or by repeating from an earlier subsequence. We therefore entertain three experts for these hypotheses: (i) a Markov expert for the species genome distribution, (ii) a context Markov expert for the local distribution, and (iii) a repeat expert, which is the combination of any available copy and reverse experts, for the third hypothesis. The experts' predictions are blended as in equations 2 and 7.

If a symbol is part of a significant repeat, the copy or reverse expert of that repeat must predict significantly better than a general prediction such as that from the Markov expert. We therefore define a *listen threshold*, T, to determine the reliability of a copy or reverse expert. A copy or reverse expert is considered reliable if its average code word length is smaller than $C_{mk} - T$ bits where C_{mk} is the average code word of the Markov expert. T is a parameter of the algorithm.

The algorithm can be used as an entropy estimator or a compressor for biological sequences. The information content of every single symbol is estimated by the negative log of its probability. To compress the sequence, we use arithmetic coding [24] to code each symbol based on the probability distribution combined from experts.

4. Experimental results

We implemented the encoder and decoder of XM in Java and ran experiments on a workstation with Pentium IV 2.4Ghz CPU and 1GB of RAM, using the Sun Java run-time environment 1.5. The compression results are calculated from the size of real encoded files. Note that the figures for actual compression and information content are similar up to four decimal places. The subtle difference between the information content computation and the actual compression is due to rounding in arithmetic coding and padding the last byte of the encoded files.

For comparison, we applied our algorithm on a standard dataset of DNA sequences that has been used in most other DNA compression publications. The dataset contains 11 sequences including two chloroplast genomes (CHMPXX and CHNTXX), five human genes (HUMDYSTROP, HUMGHCSA, HUMHBB, HUMHDABCD and HUMHPRTB), two mitochondria genomes (MPOMTCG and MTPACG) and genomes of two viruses (HEHCMVCG and VACCG). For DNA compression, we use hash key of length 11 and listen threshold of 0.5 bits.

Sequence	BioC	GenC	DNAC	DNAP	CDNA	GeMNL	XM
CHMPXX	1.6848	1.6730	1.6716	1.6602	-	1.6617	**1.6577**
CHNTXX	1.6172	1.6146	1.6127	1.6103	1.65	1.6101	**1.6068**
HEHCMVCG	1.8480	1.8470	1.8492	**1.8346**	-	1.8420	1.8426
HUMDYSTROP	1.9262	1.9231	1.9116	1.9088	1.93	1.9085	**1.9031**
HUMGHCSA	1.3074	1.0969	1.0272	1.039	**0.95**	1.0089	0.9828
HUMHBB	1.8800	1.8204	1.7897	1.7771	1.77	-	**1.7513**
HUMHDAB	1.8770	1.8192	1.7951	1.7394	1.67	1.7059	**1.6671**
HUMHPRTB	1.9066	1.8466	1.8165	1.7886	**1.72**	1.7639	1.7361
MPOMTCG	1.9378	1.9058	1.8920	1.8932	**1.87**	1.8822	1.8768
MTPACG	1.8752	1.8624	1.8556	1.8535	1.85	**1.8440**	**1.8447**
VACCG	1.7614	1.7614	1.7580	1.7583	1.81	**1.7644**	**1.7649**
Average	1.7837	1.7428	1.7254	1.7148	-	-	**1.6940**

Table 1. Comparison of DNA compression.

Table 1 compares the compression results, in bits per symbol (bps), of XM to that of other DNA compressors on the dataset. Due to space limitations, we present here the most efficient algorithms, including BioCompress-2 (BioC) [12], GenCompress (GenC) [6], DNACompress (DNAC) [7], DNAPack (DNAP) [4], CDNA [15] and GeMNL [14]. Comparison with other DNA compressors can be found on the website: `ftp://ftp.infotech.monash.edu.au/software/DNAcompress-XM/` The results of CDNA are reported for only 9 sequences in precision of two decimal places. The GeMNL results are also reported without the sequence HUMHBB and in two decimal place precision but we are able to obtain higher precision by downloading the encoded files from the author's website. We include the average compression results of each algorithm in the last row.

XM outperforms all other algorithms in most sequences from the standard dataset. The average compression ratio is also significantly better. For CDNA and GeMNL, due to missing compression results of several sequences, we are unable to compute the same average. Instead, we compare the average of the only available results. The average compression ratio of nine sequences reported for CDNA is 1.6911 bps, while XM's average performance on the same set is 1.6815 bps. On the ten sequences excluding HUMHBB, GeMNL averages 1.6980 bps, compared to XM's 1.6883 bps. Total time for XM to encode these 11 sequences is about 8 seconds. Decoding time is similar since both encoder and decoder do essentially the same computation.

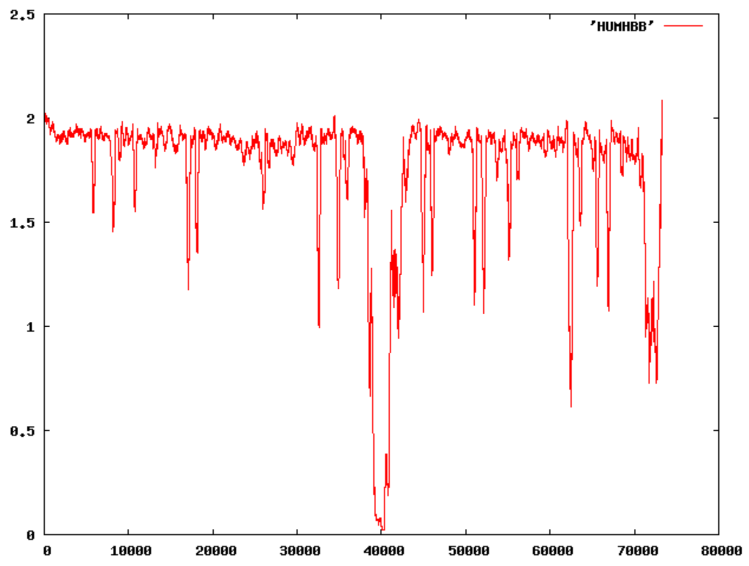

Figure 1. Information content of the HUMHBB sequence.

As a statistical compressor, the expert model is able to produce the information content sequence from DNA or protein. This is important when we want to analyze areas of interest [21, 9, 10]. For example, figure 1 shows a graph of information content along the HUMHBB sequence. The data in the graph is smoothed with a window size of 300 for viewing purposes. One can notice spikes in the graph corresponding to areas of repeats in the sequence.

The alphabet for proteins consists of 20 symbols and thus the base line of protein entropy is $log_2 20 = 4.322$ bps. Similar to DNA, most general purpose compressors fail to compress to less than that base line. Nevill-Manning and Witten [18] designed CP, a protein-oriented compression algorithm based on PPM. However, compression ratios obtained by CP are only marginally better than the base line entropy. Several other attempts such as ProtComp [13], LZ-CTW [16] and BW [1] show that protein sequence are indeed compressible with better compression ratios.

We experimented with compressing protein using XM on a protein corpus gathered by [18] which consists of proteomes of four species: Haemophilus Influenzae (HI), Saccharomyces Cerevisiae (SC), Methanococcus Jannaschii (MJ) and Homo Sapiens (HS). As an amino acid is coded by three nucleotides, we use a shorter hash key for

Sequence	CP	ProtComp	LZ-CTW	XM
HI	4.143	4.108	4.1177	**4.1022**
SC	4.146	3.938	3.9514	**3.8850**
MJ	4.051	4.008	4.0279	**4.0002**
HS	4.112	3.824	4.0055	**3.7860**
Average	4.113	3.9695	4.0256	**3.9434**

Table 2. Comparison of protein compression.

protein, of length 6. The listen threshold is raised to 1.0 bit as the upper bound entropy of protein is 4.322 bps instead of 2.0 bps in DNA. Table 2 shows the compression ratios of CP, ProtComp, LZ-CWT and XM of the four protein sequences. Note that an incorrect protein corpus that was more compressible was made available at some point resulting in a significantly lower compression ratios being reported in ProtComp [13] and BW [1]. We obtained the compression results of ProtComp on the correct protein corpus from the author's website but were unable to do so for BW as the authors have moved to new projects [17]. We found that our algorithm is able to compress proteins better than CP and LZ-CWT and marginally better than ProtComp for all sequences in the corpus.

5. Conclusion

We have presented the expert model, XM, which is simple and based on biological principles. The associated compression algorithm is efficient and effective for both DNA and protein sequence compression. The algorithm utilizes approximate repeats and statistical properties of the biological sequence for compression. As a statistical compression method, XM is able to compute the information content of every symbol in a sequence which is useful in knowledge discovery [21, 9, 10]. Our algorithm is shown to outperform all published DNA and protein compressors to date while maintaining a practical running time.

References

[1] D. Adjeroh and F. Nan. On compressibility of protein sequences. *DCC*, pages 422–434, 2006.

[2] L. Allison, T. Edgoose, and T. I. Dix. Compression of strings with approximate repeats. *ISMB*, pages 8–16, 1998.

[3] A. Apostolico and S. Lonardi. Compression of biological sequences by greedy off-line textual substitution. *DCC*, pages 143–152, 2000.

[4] B. Behzadi and F. L. Fessant. DNA compression challenge revisited: A dynamic programming approach. *CPM*, pages 190–200, 2005.

[5] D. M. Boulton and C. S. Wallace. The information content of a multistate distribution. *Theoretical Biology*, 23(2):269–278, 1969.

[6] X. Chen, S. Kwong, and M. Li. A compression algorithm for DNA sequences and its applications in genome comparison. *RECOMB*, page 107, 2000.

[7] X. Chen, M. Li, B. Ma, and T. John. DNACompress: Fast and effective DNA sequence compression. *Bioinformatics*, 18(2):1696–1698, Dec 2002.

[8] J. G. Cleary and I. H. Witten. Data compression using adaptive coding and partial string matching. *IEEE Trans. Comm.*, COM-32(4):396–402, April 1984.

[9] T. I. Dix, D. R. Powell, L. Allison, S. Jaeger, J. Bernal, and L. Stern. Exploring long DNA sequences by information content. *Probabilistic Modeling and Machine Learning in Structural and Systems Biology, Workshop Proc*, pages 97–102, 2006.

[10] T. I. Dix, D. R. Powell, L. Allison, S. Jaeger, J. Bernal, and L. Stern. Comparative analysis of long DNA sequences by per element information content using different contexts. *BMC Bioinformatics*, to appear, 2007.

[11] S. Grumbach and F. Tahi. Compression of DNA sequences. *DCC*, pages 340–350, 1993.

[12] S. Grumbach and F. Tahi. A new challenge for compression algorithms: Genetic sequences. *Inf. Process. Manage.*, 30(6):875–866, 1994.

[13] A. Hategan and I. Tabus. Protein is compressible. *NORSIG*, pages 192–195, 2004.

[14] G. Korodi and I. Tabus. An efficient normalized maximum likelihood algorithm for DNA sequence compression. *ACM Trans. Inf. Syst.*, 23(1):3–34, 2005.

[15] D. Loewenstern and P. N. Yianilos. Significantly lower entropy estimates for natural DNA sequences. *Computational Biology*, 6(1):125–142, 1999.

[16] T. Matsumoto, K. Sadakane, and H. Imai. Biological sequence compression algorithms. *Genome Informatics*, 11:43–52, 2000.

[17] F. Nan. Personal communication, 2006.

[18] C. G. Nevill-Manning and I. H. Witten. Protein is incompressible. *DCC*, pages 257–266, 1999.

[19] D. R. Powell, L. Allison, and T. I. Dix. Modelling-alignment for non-random sequences. *Advances in Artificial Intelligence*, pages 203–214, 2004.

[20] E. Rivals, J.-P. Delahaye, M. Dauchet, and O. Delgrange. A guaranteed compression scheme for repetitive DNA sequences. *DCC*, page 453, 1996.

[21] L. Stern, L. Allison, R. L. Coppel, and T. I. Dix. Discovering patterns in plasmodium falciparum genomic DNA. *Molecular & Biochemical Parasitology*, 118:175–186, 2001.

[22] I. Tabus, G. Korodi, and J. Rissanen. DNA sequence compression using the normalized maximum likelihood model for discrete regression. *DCC*, page 253, 2003.

[23] F. M. J. Willems, Y. M. Shtarkov, and T. J. Tjalkens. The context-tree weighting method: Basic properties. *IEEE Trans. Info. Theory*, pages 653–664, 1995.

[24] I. H. Witten, R. M. Neal, and J. G. Cleary. Arithmetic coding for data compression. *Comm ACM*, 30(6):520–540, June 1987.

[25] J. Ziv and A. Lempel. A universal algorithm for sequential data compression. *IEEE Trans. Inf. Syst.*, 23(3):337–342, May 1977.

[26] J. Ziv and A. Lempel. Compression of individual sequences via variable-rate coding. *IEEE Trans. Inf. Syst.*, 24(5):530–536, 1978.