IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661, p- ISSN: 2278-8727Volume 10, Issue 5 (Mar. - Apr. 2013), PP 46-51
www.iosrjournals.org

A New Approach of Protein Sequence Compression using Repeat Reduction and ASCII Replacement

S.M. Rafizul Haque, Tania Mallick, Iffat Sania Kabir
CSE Discipline, Khulna University, Bangladesh
CSE Discipline, Khulna University, Bangladesh
CSE Discipline, Khulna University, Bangladesh

Abstract: *Protein sequences have been considered for a long time as highly complex sequences from the informational content point of view. We study about the randomness of protein and difficulty of protein compression. In this report, we propose a lossless compression method to reduce the random or repeated sequence of protein. Our proposed method is the enhancement of Dictionary Based Algorithm proposed by D. Adjeroh and F. Nan. They have shown better result in compressing protein. Our repeat reduction method shows improved results in compressing three types of protein sequences from MJ (Methanococcus jannaschii), HI (Haemophilus influenzae) and HS (Homo sapiens).* The experimental results demonstrate that the performance of the proposed method is superior compared to Dictionary based algorithm.
Keywords: *Proteins, protein compression, repeat reduction, protein sequence, repeat, ASCII replacement.*

I. Introduction

Now a day's bioinformatics is an important area of research and development. So, storing and manipulating of the data related to bioinformatics using computers as a compressed form is an important task. Compression is an art of reducing the size of a file by removing redundancy in its structure. Data compression offers an attractive approach to reducing communication costs by using available bandwidth effectively. Compression techniques are broadly divided into two categories: lossless and lossy. If the recovery of data is exact, the compression algorithms are called lossless. The lossless compression algorithms are used for all kinds of text, scientific and statistical databases, bioinformatics, medical and biological images, and so on. If the recovery of data is approximate, the compression algorithms are called lossy. Lossy algorithms are useful in image and video processing. There are two different perspectives of protein compression. They are practical perspective and scientific perspective. In practical perspective, compression permits the use of resources such as storage and bandwidth. In scientific perspective, compression presents a way of capturing and computing structure in a protein sequence [1]. Total number of protein sequences will be increased day by day. So, protein should be accumulated as a compressed shape. Typical compression techniques are not appropriate for protein compression [3]. It is very difficult to compress protein due to its multiplicity and individuality of protein sequence. The evident randomness of the signs in a protein sequence is another difficulty to compress protein.
In 1999, C. G. Nevill-Manning and I. H Witten said "Protein is incompressible" [1]. According to them protein is difficult to compress since there is little Markov dependency in protein. In 1999, Nevill and Witten have developed a special compression algorithm called CP (Compress Protein). The basic concept of CP is to weight alternative contexts like CTW (Context Tree Weighting) function. Unfortunately, this technique produces negative result. In 2004, A. Hategan and I. Tabus proved "Protein is Compressible" [2]. They developed an algorithm called ProtComp which is based on optimal building of the substitution probability matrix. This algorithm gives the compression ratio of 1.3:1. In 2006, D. Adjeroh and F. Nan developed an algorithm that uses a term SCP (Sorted Common Prefix) [3]. This term is used to separate approximate repeats or overlapping repeats. This technique gives a high compression ratio. This technique can compress protein 3.50 bits per symbol in average.

Here we have proposed a new algorithm for protein sequence compression based on repeat reduction and ASCII Replacement. The proposed method can compress protein 2.89 bits per symbol in average, which is the best result found for protein compression.

II. Protein

Proteins are the most abundant biological macromolecules, occurring in all cells and all parts of cells. Protein is one of the based components of life. It is also called the machinery of life because the existence of living being is impossible without protein. Protein sequence is the combination of 20-amino acids. The alphabet structures of 20 amino acids are shown in Table-1.

Table 1: Codes for 20-amino acids

Code	Full Name	Code	Full Name
A	Alanine	M	Methionine
C	Cysteine	N	Aspartamine
D	Aspartate	P	Praline
E	Glutamate	Q	Glutamine
F	Phenylalanine	R	Arginine
G	Glycine	S	Serine
H	Histidine	T	Threonine
I	Isoleucine	V	Valine
K	Lysine	W	Tryptophan
L	Leucine	Y	Tyrosine

III. Protein Sequencing

Protein sequence is the combination of 20-amino acids. Protein sequence is highly responsible for determination of protein structure. Protein sequence can be easily obtained from DNA/RNA sequence. Each protein contains a unique amino acid sequence which has a set of generic codes called 'codon'. 'Codon' is a set of three nucleotide as for example AUG is a code that contains adenine-uracil-guanine nucleotides respectively. On the other hand, DNA sequence contains four nucleotides such as adenine (A), cytosine (C), guanine (G), thymine (T) and the number of possible codon in DNA is 64. At first, DNA sequence is transcribed into messenger RNA or mRNA .Then, it is loaded onto the ribosome and is read three nucleotides at a time by matching each codon. After that translation procedure is being processed and 20-amino acid is generated per second as a sequence. In fact, Protein sequence is referred to as amino acid sequence. The protein sequence or amino-acid sequence determines the three dimensional structure of a protein. Proteins have static, unchanging three dimensional structures.

IV. Related Works

CP algorithm is proposed by, C. G. Nevill-Manning and I. H. Witten in 1999 [1]. They mentioned that protein is difficult to compress, since there is little Markov dependency in protein. They developed an algorithm with the help of PPM (Prediction by Partial Matching) algorithm. The technique used by CP to accommodate mutation probabilities may conceivably be applicable to other domains. The main thought of CP algorithm is to weight alternative contexts. This scheme includes exact repetition in protein sequences and applies all contexts up to a positive length and weighted by their similarity to the current context. This method proposed the probability prediction for amino acid exact repetition. Another technique named blending is used for replacement in CP algorithm. PPM algorithm is used to switch this replacement. This approach is related to Loewenstern and Yainilos algorithms for DNA compress (1997). Here various context lengths are inserted with various Hamming distances. The author utilized four genomes in their investigation-HI (Haemophilus influenzae), SC (Saccharomyces cerevisiae), MJ (Methanococcus jannaschii) and HS (Homo sapiens). Regardless of the fact that they establish a technique of getting improved compression than is specified by generic compression techniques. But like other method this method also failed to compress all proteins. The technique provides very poor compression ratio. Due to this failure, the inventor came to a conclusion that protein was incompressible.

In 2004, A. Hategan and I. Tabus proposed a lossless compression algorithm called ProtComp [2]. This algorithm is adaptive and uses approximate repeats as well as mutation probability for amino acids. The algorithm is based on optimal building of the substitution probability matrix. ProtComp algorithm is two pass algorithms. The purpose of the first pass is to construct the substitution probability matrix, M and the purpose of the second pass is encoding the symbols. First, each current block having the largest number of matches and compare with fixed parameters and updated the substitution matrix. Then, transmitting the side information and transmitting the symbols. Finally, total code words are obtained. In one pass version, a fixed substitution matrix is used in ProtComp algorithm. Here, first step is skipped and ProtComp2, ProtComp1Huff and ProtComp1Arithm are applied to get the result. The algorithm is tested using four proteomes - HI, SC, MJ and HS and each sequence contains very long repetition. When using two pass algorithm, this is builds a substitution matrix and transmits the matrix as side information. When using one pass algorithm, the results are very close with two pass algorithm. ProtComp algorithm gives the compression ratio 1.3:1. But this algorithm cannot calculate gaps. Finally, it is also a poor compression technique but the results improved the previous one.

A dictionary based protein compression algorithm is proposed by D. Adjeroh and F. Nan [3]. The algorithm uses Sorted Common Prefix (SCP) which is used to separate approximate repeats or overlapping repeats. The SCP is symmetric and SCP is usually sparse. Protein sequence has different forms of repetitions- overlapped repetitions, direct repetitions, palindrome repetitions, adjourning repetitions. At first, the protein sequences are processed and repetitions are detected. Then the repetitions are parsed using SCP and two outputs can be obtained such as (1) Dictionary items, D which contains repeated patterns, repetition length, position number of occurrences and (2) Remaining sequence, R which contains non-repeated subsequences. After that, remove the repeated substring from the input sequence, and move it to an external dictionary. In the dictionary, record the positions in the sequence where each repetition occurred, along with the repetition types. It stops when the compression gain, G(S) is negative or less than a threshold. Each leaf in the tree is part of the final output. In this algorithm inventors considered the problem of compressibility of protein sequences. This technique gives a high compression ratio and compress protein up to 3.453 bits per symbol. The improved performance was observed on all the protein sequences in the protein corpus.

V. Our Approach

From the analysis of protein structure, we can conclude that protein is a sequence of 20-amino acids. This sequence can be represented by alphabet. Multiplicity and individuality exists in the protein sequence; it is difficult to compress protein. Moreover, different types of repetitions occur in protein sequence. Depending on this assumption, we have proposed our method based on repeat reduction.
Our proposed model contains three basic parts. These parts are mentioned as follows:

1. Repetition analysis
2. Encoding or Compression
3. Decoding or Decompression

5.1 REPEAT ANALYZATION PROCESS

The first phase of proposed method analyzes all repeats containing exact 6 residues as a unit and total number of each repeat and all staring position of each repeat. These criteria will be saved in temporary list during execution time.
The algorithm of our repeat analyzation phase is as following:

1. Set rptstrng = 0 and exist = 0.
2. If repeat is found Then do exist = exist + 1.
3. If exist = 0 then do findposition (curstr, nextstr, curpos).
4. Record string position and number.
5. Sort all record in decreasing order according to total number of a repeat found in sequence.
6. Count all repeats which has repeated more than 1.

Here, in this algorithm we analyze all repeats combined with different number of residues. Then we sort total list according to number of repeats in decreasing order. Then we count only unique repeats and our method gives the highest priority to a repeat which has highest number of repetitions. In the list of repeat, we assign a value 97 as a default to the first repeat. Then we increase this value to 1 and save as a character in ASCII table.

5.2 COMPRESSION PROCESS

1. Replace all repeats with ASCII character.
2. Separate remaining repeats with overlap form with a first bracket replace each repeats with its first residue.
3. After that we encode the sequence with memory stream and define a location to save the encoded file.
4. Finally, we will get compressed file in zip format.

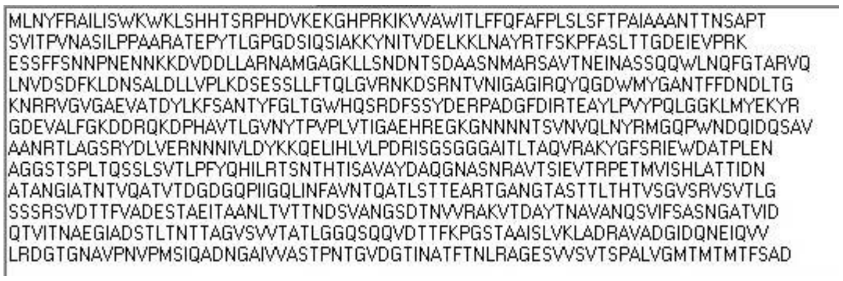
Here we compute total compression in KB = $\sum$ RN (RL-1) – const
Where, RN = Number of Repeats
RL = Repeat Length
Const = Total number of repeated string independent when we compare internal exchange of different repeat length = $\sum 1$.
Now, Compression ratio = (total compression in KB / original sequence in KB) * 8
Where, we consider 8 residues in per unit symbol.

5.3 DECOMPRESSION PROCESS

1. In this process, input will be zipped file.
2. At first, it decode using memory stream buffer.
3. Then, we get all repeats from the replacement with ASCII according with position.
4. After replacement process will search where first bracket is located.
5. Finally it will cancel first bracket and decode remaining parts and get original sequence.

VI. Illustration Of The Proposed Method With An Example

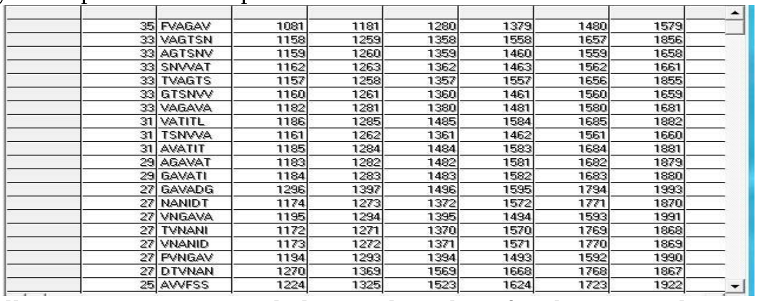
Consider the entire brain protein sequence from ***H. Influenza*** as input sequence. The sample sequence is given below:



```
MLNYFRAILISWKWKLSHHTSRPHDVKEKGHPRKIKVVAWITLFFQFAFPLSLSFTPAIAAANTTNSAPT
SVITPVNASILPPAARATEPYTLGPGDSIQSIAKKYNITVDELKKLNAYRTFSKPFASLTTGDEIEVPRK
ESSFFSNNPNENNKKDVDDLLARNAMGAGKLLSNDNTSDAASNMARSAVTNEINASSQQWLNQFGTARVQ
LNVDSDFKLDNSALDLLVPLKDSESSLLFTQLGVRNKDSRNTVNIGAGIRQYQGDWMYGANTFFDNDLTG
KNRRVGVGAEVATDYLKFSANTYFGLTGWHQSRDFSSYDERPADGFDIRTEAYLPVYPQLGGKLMYEKYR
GDEVALFGKDDRQKDPHAVTLGVNYTPVPLVTIGAEHREGKGNNNNTSVNVQLNYRMGQPWNDQIDQSAV
AANRTLAGSRYDLVERNNNIVLDYKKQELIHLVLPDRISGSGGGAITLTAQVRAKYGFSRIEWDATPLEN
AGGSTSPLTQSSLSVTLPFYQHILRTSNTHTISAVAYDAQGNASNRAVTSIEVTRPETMVISHLATTIDN
ATANGIATNTVQATVTDGDGQPIIGQLINFAVNTQATLSTTEARTGANGTASTTLTHTVSGVSRVSVTLG
SSSRSVDTTFVADESTAEITAANLTVTTNDSVANGSDTNVVRAKVTDAYTNAVANQSVIFSASNGATVID
QTVITNAEGIADSTLTNTTAGVSVVTATLGGQSQQVDTTFKPGSTAAISLVKLADRAVADGIDQNEIQVV
LRDGTGNAVPNVPMSIQADNGAIVVASTPNTGVDGTINATFTNLRAGESVVSVTSPALVGMTMTMTFSAD
```

Fig 1: Input protein sequence

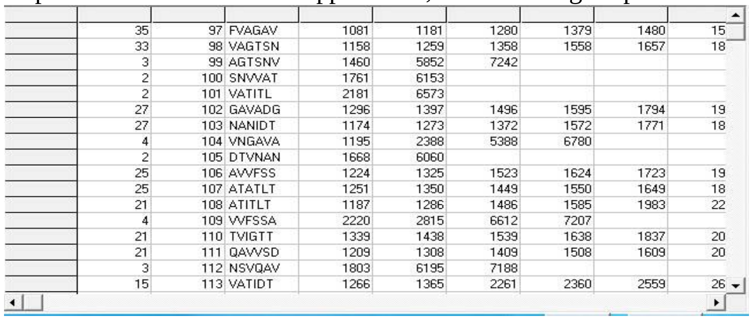
After the repeat analyzation process the output is as follows:



	35	FVAGAV	1081	1181	1280	1379	1480	1579	
	33	VAGTSN	1158	1259	1358	1558	1657	1856	
	33	AGTSNV	1159	1260	1359	1460	1559	1658	
	33	SNVVAT	1162	1263	1362	1463	1562	1661	
	33	TVAGTS	1157	1258	1357	1557	1656	1855	
	33	GTSNVV	1160	1261	1360	1461	1560	1659	
	33	VAGAVA	1182	1281	1380	1481	1580	1681	
	31	VATITL	1186	1285	1485	1584	1685	1882	
	31	TSNVVA	1161	1262	1361	1462	1561	1660	
	31	AVATIT	1185	1284	1484	1583	1684	1881	
	29	AGAVAT	1183	1282	1482	1581	1682	1879	
	29	GAVATI	1184	1283	1483	1582	1683	1880	
	27	GAVADG	1296	1397	1496	1595	1794	1993	
	27	NANIDT	1174	1273	1372	1572	1771	1870	
	27	VNGAVA	1195	1294	1395	1494	1593	1991	
	27	TVNANI	1172	1271	1370	1570	1769	1868	
	27	VNANID	1173	1272	1371	1571	1770	1869	
	27	PVNGAV	1194	1293	1394	1493	1592	1990	
	27	DTVNAN	1270	1369	1569	1668	1768	1867	
	25	AVVFSS	1224	1325	1523	1624	1723	1922	

Fig 2: All repeats in sequence including total number of each repeat and start positions

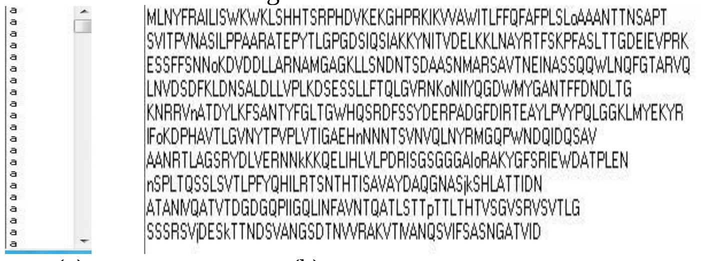
Counting only unique repeats those have no overlapped form, the following output is obtained:



	35	97	FVAGAV	1081	1181	1280	1379	1480	15
	33	98	VAGTSN	1158	1259	1358	1558	1657	18
	3	99	AGTSNV	1460	5852	7242			
	2	100	SNVVAT	1761	6153				
	2	101	VATITL	2181	6573				
	27	102	GAVADG	1296	1397	1496	1595	1794	19
	27	103	NANIDT	1174	1273	1372	1572	1771	18
	4	104	VNGAVA	1195	2388	5388	6780		
	2	105	DTVNAN	1668	6060				
	25	106	AVVFSS	1224	1325	1523	1624	1723	19
	25	107	ATATLT	1251	1350	1449	1550	1649	18
	21	108	ATITLT	1187	1286	1486	1585	1983	22
	4	109	VVFSSA	2220	2815	6612	7207		
	21	110	TVIGTT	1339	1438	1539	1638	1837	20
	21	111	QAVVSD	1209	1308	1409	1508	1609	20
	3	112	NSVQAV	1803	6195	7188			
	15	113	VATIDT	1266	1365	2261	2360	2559	26

Fig 3: All unique repeats which have no overlapped form

Then our proposed system produce ASCII table and replace all repeats with ASCII character. The output is given as follows:



(a)

```
a
a
a
a
a
a
a
a
a
a
a
a
a
a
a
a
a
a
a
a
a
```

(b)

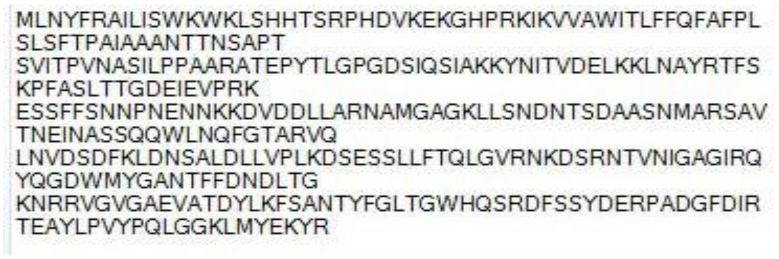
```
MLNYFRAILISWKWKLSHHTSRPHDVKEKGHPRKIKVVAWITLFFQFAFPLSLoAAANTTNSAPT
SVITPVNASILPPAARATEPYTLGPGDSIQSIAKKYNITVDELKKLNAYRTFSKPFASLTTGDEIEVPRK
ESSFFSNNoKDVDDLLARNAMGAGKLLSNDNTSDAASNMARSAVTNEINASSQQWLNQFGTARVQ
LNVDSDFKLDNSALDLLVPLKDSESSLLFTQLGVRNKoNIIYQGDWMYGANTFFDNDLTG
KNRRVnATDYLKFSANTYFGLTGWHQSRDFSSYDERPADGFDIRTEAYLPVYPQLGGKLMYEKYR
lFoKDPHAVTLGVNYTPVPLVTIGAEHnNNNTSVNVQLNYRMGQPWNDQIDQSAV
AANRTLAGSRYDLVERNNkKKQELIHLVLPDRISGSGGGAIoRAKYGFSRIEWDATPLEN
nSPLTQSSLSVTLPFYQHILRTSNTHTISAVAYDAQGNASjkSHLATTIDN
ATANIVQATVTDGDGQPIIGQLINFAVNTQATLSTTpTTLTHTVSGVSRVSVTLG
SSSRSVjDESkTTNDSVANGSDTNVVRAKVTIVANQSVIFSASNGATVID
```

Fig 4: Output after ASCII replacement (a) ASCII table and (b) sequence encoded with ASCII

Then we save the final encoded file using memory stream. The final output is as follows:

Compressed size: 2671
Original size: 6332
File is Compressed D drive in zipped form.
Compression ratio is: 3.37 bits per symbol

For decompression process, the input file is input.txt.zip. At first our processes unzip the file using memory stream and array byte and decode the file with replacement of ASCII character. At the end of this process the following output is obtained:



```
MLNYFRAILISWKWKLSHHTSRPHDVKEKGHPRKIKVVAWITLFFQFAFPL
SLSFTPAIAAANTTNSAPT
SVITPVNASILPPAARATEPYTLGPGDSIQSIAKKYNITVDELKKLNAYRTFS
KPFASLTTGDEIEVPRK
ESSFFSNNPNENNKKDVDDLLARNAMGAGKLLSNDNTSDAASNMARSAV
TNEINASSQQWLNQFGTARVQ
LNVDSDFKLDNSALDLLVPLKDSESSLLFTQLGVRNKDSRNTVNIGAGIRQ
YQGDWMYGANTFFDNDLTG
KNRRVGVGAEVATDYLKFSANTYFGLTGWHQSRDFSSYDERPADGFDIR
TEAYLPVYPQLGGKLMYEKYR
```

Fig 5: Output after decompression process

VII. Experimental Result

We have tested our method with the amino acids protein sequences. Here, we have shown the compression ratios for each file in Table 8 and compression ratios for bit per symbol in table 9. Table 10 shows the compression results in bits per symbol for dictionary based method and our proposed method.

Table 2: Compressed file size in Bytes using our method

Protein sequence	Original size in KB	Compressed size in KB
MJ	8805	2715
HI	6332	2671
HS	560	199
Total	15697	5585
Average	5232.33	1861.67

Table 3: Compression ratio in bit per symbol (bps) using our method

Protein sequence	Original size in KB	Compression ratio in bps (bit per symbol)
MJ	8805	2.47
HI	6332	3.37
HS	560	2.84
Average	5232.33	2.89

From the above experimental results, we find that our method can compress 2.89bits per symbol in average which is the best result for protein compression that had been ever found.

Table 4: Comparison results for Dictionary based compression and our method

Sequence	Size (KB)	Dictionary-based compression	Our Method
MJ	8805	2.54	2.47
HI	6332	3.43	3.37
HS	560	3.11	2.84
Average	5232.33	3.03	2.89

From Table 10 we can see that our proposed method can compress in average 303 bits per 100 symbols and Dictionary based compression proposed by D. Adjeroh and F. Nan in average 289 bits per 100 symbols. So, the improvement by our method is 14 bits per 100 symbols.
The following graph compares the performance among Dictionary-based and our proposed method.

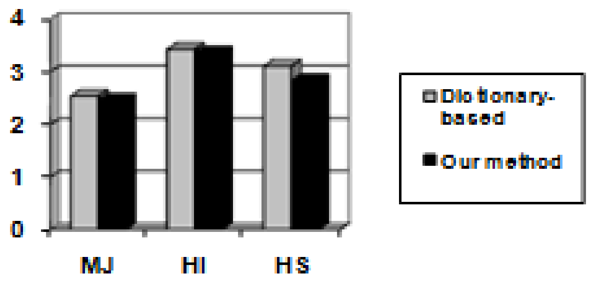


Fig 6: Performance comparison among Dictionary-based and our proposed method for bits per symbol.

The superior performance of the proposed method is evident in the above graph. We notice that the improved performance is consistent across different protein sequences. In terms of compression time our total compression process takes about 4 minutes for MJ sequence on a Pentium-D 2.66 GHz processor running on Microsoft Windows 7.

VIII. Conclusion

Protein sequence compression is one of the most difficult tasks in the area of bioinformatics. The existing algorithms have faced difficulty in compressing protein due to its repetition. Each of them is the improvement of previous one. Here, we proposed a protein compression method based on repeat reduction and ASCII replacement. Our proposed method gives better result than that of the previous algorithms. For real time application manually insertion of protein is not good approach. We hope that we will be able to remove previous mistakes and provide better facility in real time application than previous techniques.

IX. Future Work

In future, we want to improve compression process as well as compression ratio so that any kind of protein sequence can be compressed efficiently. Moreover, we want to include gaps in a protein sequence.

Reference

[1] Craig G. Nevill-Manning and Ian H. Witten, ”Protein is incompressible”, in proceeding of The Data Compression Conference (DCC’99), pp 257-266, Snowbird, Utah, USA, March 1999.

[2] Andrea Hategan and Ioan Tabus,“Protein is compressible”, in proceeding of the 6th Nordic Signal Processing Symposium (NORSIG 2004), pp 192-195, Espoo, Finland, June 2004.

[3] Donald Adjeroh and Fei Nan, ”On compressibility of Protein sequence”, in proceeding of The Data Compression Conference (DCC’06), pp 422-434, Snowbird, Utah, USA, March 2006.

[4] T. Matsumoto, K. Sadakane and H. Imai, ”Biological Sequence Compression Algorithms” In proceeding of The Data Compression Conference (DCC’07), pp 43-52,Snowbird,Utah, USA, March 2007.

[5] A. Hategan and I. Tabus, “Detecting local similarity based on lossless compression of Protein sequences”, in International workshop on Genomic Signal Processing 95, 2005.