A Biological Sequence Compression Based on Cross Chromosomal Similarities Using Variable length LUT

Rajendra Kumar Bharti raj05_kumar@yahoo.co.in
Ph.D. scholar, UTU,
Astt. Professor, CSE Deptt.
B. C. T. Kumaon Engineering College,
Dwarahat, Almora, Uttarakhand, India

Archana Verma vermarchana05@gmail.com
Ph.D. scholar, UTU,
Astt. Professor, CSE Deptt.
B. C. T. Kumaon Engineering College,
Dwarahat, Almora, Uttarakhand, India

Prof. R.K. Singh rksinghkec12@rediffmail.com
Professor, ECE Deptt.
B. C. T. Kumaon Engineering College,
Dwarahat, Almora, Uttarakhand, India

Abstract

While modern hardware can provide vast amounts of inexpensive storage for biological databases, the compression of Biological sequences is still of paramount importance in order to facilitate fast search and retrieval operations through a reduction in disk traffic. This issue becomes even more important in light of the recent increase of very large data sets, such as meta genomes.

The present Biological sequence compression algorithms work by finding similar repeated regions within the Biological sequence and then encode these repeated regions together for compression. The previous research on chromosome sequence similarity reveals that the length of similar repeated regions within one chromosome is about 4.5% of the total sequence length. The compression gain is often not high because of these short lengths of repeated regions. It is well recognized that similarities exist among different regions of chromosome sequences. This implies that similar repeated sequences are found among different regions of chromosome sequences. Here, we apply cross-chromosomal similarity for a Biological sequence compression. The length and location of similar repeated regions among the different Biological sequences are studied. It is found that the average percentage of similar subsequences found between two chromosome sequences is about 10% in which 8% comes from cross-chromosomal prediction and 2% from self-chromosomal prediction. The percentage of similar subsequences is about 18% in which only 1.2% comes from self-chromosomal prediction while the rest is from cross-chromosomal prediction among the different Biological sequences studied. This suggests the significance of cross-chromosomal similarities in addition to self-chromosomal similarities in the Biological sequence compression. An additional 23% of storage space could be reduced on average using self-chromosomal and cross-chromosomal predictions in compressing the different Biological sequences.

Keywords: Biological Sequences, Chromosome, Cross Chromosomal Similarity, Compression Gain, Prediction.

1. INTRODUCTION

The Deoxyribonucleic acid(DNA) constitutes the physical medium in which all properties of living organisms are encoded. Molecular sequence databases (e.g.,EMBL,Genbank, DDJB, Entrez, SwissProt, etc) currently collect hundreds or thousands of sequences of nucleotides and amino acids reaching to thousands of gigabytes and are under continuous expansion. Need for Compression arises because approximately 44,575,745,176 bases in 40,604,319 sequence records are there in the GenBank database[12].

Increasing genome sequence data of organism lead Biological database size two or three times bigger annually[1]. Thus, Efficient compression may also reveal some biological functions and helps in phylogenic tree reconstruction etc[2,3,4]. Compression is desirable to uncover similarities among sequences, and provide a means to understand their properties in addition to reduce storage requirement[13].

There are many text compression algorithms available having quite a high compression ratio[7]. But they have not been proved well for compressing Biological sequences as the algorithm does not incorporate the characteristics of Biological sequences even through sequences can be represented in simple text form. Biological sequences are comprised of just four different bases e.g. for DNA only A,T,G and C[1,2,3,4,7]. Each base can be represented by two bits in binary. So, It has been observed that no file-compression program achieves benchmark of the compression ratio for Biological sequences. Several compression algorithms specialized for Biological sequences have been developed in the last decade and some of these are: Biocompress-2, Gen Compress and CTW+LZ. One knows that all such algorithms take a long time (essentially a quadratic time search or even more) but at the same time achieving high speed and best compression ratio remains to be a challenging task[2,3,4,7].

In this paper, it has been tried to cope the above said problem. In this work it has been tried to achieve a better compression ratio and runs significantly faster than any existing compression program for Biological sequences. A lot of research work has already been carried out for developing programmes for Biological sequence compression. It is seen that all Biological sequence compression algorithms find repetition with in the sequence. Longer repetitive length implies higher compression gain. The compression ratio gain is high if highly similar subsequences are found. It is well known that there are similarities among different chromosome sequences. However, cross chromosomal similarities are seldom exploited in sequence compression[11]. The objective of this paper is to exploit self chromosomal similarity and cross chromosomal similarities as well. It should be noted that similar subsequences located within the chromosome sequence are called self similar while located in other chromosome sequence are called cross chromosome similar sequences.

2. METHODOLOGY

We use eight sequences to find chromosomal similarities. First, we search all cross similarities and then compress with the help of variable length LUT based compression algorithm. Large compression gain means that two subsequences are similar to each other.

2.1 Similarities Between Two Biological Sequences Chromosome

The potential gain in cross chromosomal compression is obtained by finding the total lengths of subsequence in the current chromosome sequence that is predicted from other chromosome sequence. The length of these cross reference subsequences determines the potential compression gain in multiple Biological sequence compression. Long length implies a high compression ratio.

2.2 Algorithm

Consider a finite sequence over the DNA alphabet {a, c, g, t}. Initialize an (n+1)x(m+1) array, L, for the boundary cases when i=0 or j=0. Namely, we initialize L[I,-1]=0 for i=-1,0,1,…,n-1 and L[-

1,j]=0 for j=-1,0,1,….,m-1. Then, we iteratively build up values in L until we have L[n-1,m-1], the length of a longest common subsequence of a finite sequence.

```
LCS(X,Y)
for  i ← -1 to n-1   do
          L[i,-1] ← 0
for  j ← 0 to m-1 do
          L[-1,j] ← 0
for  i ← 0 to n-1 do
          for j ← 0 to m-1 do
                    if X[i] = Y[j] then
                              L[i, j ] ← L[ i-1, j-1] + 1
                    else
                              L[i, j] ← max { L [i-1, j], L[i, j-1]}
return array L.
```

3. ENCODING REPEATS

Here, we use two step algorithms for compression: in first step we identify the all repetitions defined as Pre coding routine and in second step encoding algorithm is applied on both unique repeat and repeated subsequence.

Step 1: Pre Coding Routine

As discussed earlier the Pre coding routine help us to find the all repeats within a sequence, now method of pre coding routine algorithm will be presented.

i. Look Up Table(LUT)

The coding routine is based on variable length LUT, In which the initialization of table will be made first. We take all possible combination of three characters {a, t, g, or c} of the sequence which has been mapped onto a character chosen from the character set which consists of 8 bit ASCII characters. The generated LUT is given in table 1. It has been observed that with the help of above said generated table the implementation of pre coding routine becomes easy in handling the pre coding routine. Here it has been also observed that characters a, t, g, c and A, T, G, C will have same meaning. As an example if a segment “ACTGTCGATGCC” appeared in the LUT, in the destination file, it is represented as “j2X6”. By doing so the generated output will become case-sensitive.

It has been also observed that in Biological sequences some time a special character “N” appears. But it is exceptional. It will be necessary to consider this character also. Now handling this particular character will be dealt.

ii. Handling With the N

It has been realized that where ever the occurrence of N is repeated, there will be several Ns all together. To cater out this situation whenever the occurrence of Ns will take place in the sequences than the provision has been made in such a way that for its recognition a special character “/” inserted in the beginning and ending of Ns. All these Ns will be replaced by the characters representing total number of Ns. For example, if segment ”NNNNNN” will appear it will be replace by “/6/”.

We have taken three characters all together at a time in an input sequence. There is a possibility that while forming destination file by making a set of three characters all together, quite often less than three characters will be left in source file, for example, ATGATGATGCATTG and ATGATGATGCATTGC in which after making a set of three characters in first example TG and in second only character C is left over. So, how to handle such situation will be dealt now.

iii. Segment Which Consists of Less Than 3 Characters

In the Table 1 there is no appearance of `TC`. So to this situation just the original segment is written to destination file.

Step 2: Now, the development of algorithm will be described. There are seven steps in the designed algorithm. The steps from 1 to 6 do the task of approximate repeat and the last step signifies encoding. Here, w, wk and k represent character string,

Initialize: w=Nil.
Initialize: wk=Nil.
Initialize: k=Nil.

Step 1: read first three unprocessed characters (k). If k!= NULL, go along to step 2.Else (the EOF step 3 is reached),process the last one or two characters by step 5.

Step 2: Check that k has all non-N characters. If it is true, go to step 3.Else if k has N characters, go to step 4 Else go to step 5.

Step 3: If wk exists in the LUT then

```
        w=wk
Else
{
Output the code (character) for w
// ASCII code (character) that are mapped in LUT.
Add wk to the LUT table;
w=k;
}
End if;
```

Step 4: Search first N and successive Ns in the string and count total number of appearing in successive Ns, replace all the such Ns with "/n/" into destination file. After this, go to step 6. If number of successive Ns appears more than one time repeat the step 4.

Step 5: write non-N characters whose number is less than three into destination file directly without any modification. After that, go to step 6.

Step 6: Return to step 1 and repeat all process until EOF is reached.

Step 7: compress the output file by LZ77 algorithm.

Character	Base	Character	Base	Character	Base	Character	Base
! (33)	A A A	q (113)	T A A	' (39)	C A A	W(87)	G A A
b(98)	A A T	r (114)	T A T	H(72)	C A T	X(88)	G A T
" (34)	A A C	s (115)	T A C	I (73)	C A C	Y(89)	G A C
d(100)	A A G	$(36)	T A G	J (74)	C A G	Z(90)	G A G
e (101)	A T A	u (117)	T T A	K(75)	C T A	0 (48)	G T A
f (102)	A T T	v(118)	T T T	L(76)	C T T	1(49)	G T T
# (35)	A T C	w(119)	T T C	M(77)	C T C	2 (50)	G T C
h(104)	A T G	x(120)	T T G	N(78)	C T G	3(51)	G T G
i (105)	A C A	y (121)	T C A	O(79)	C C A	4 (52)	G C A
j (106)	A C T	z (122)	T C T	P(80)	C C T	5(53)	G C T
k (107)	A C C	%(37)	T C C	Q(81)	C C C	6 (54)	G C C
l (108)	A C G	B(66)	T C G	R(82)	C C G	7(55)	G C G
m(109)	A G A	&(38)	T G A	S (83)	C G A	8 (56)	G G A
n(110)	A G T	D(68)	T G T	((40)	C G T	9(57)	G G T
o (111)	A G C	E(69)	T G C	U(85)	C G C	+ (43)	G G C
p(112)	A G G	F(70)	T G G	V(86)	C G G	- (45)	G G G

TABLE 1: Initial Look up Table (LUT)

4. EXPERIMENTAL RESULT

Besides considering the total length of subsequences within a chromosome that can be referenced from other chromosomes, their distribution with in the sequence are also important. Let the subsequence in a sequence X that is similar to a subsequence in sequence i be X(i) and the subsequence j be X(j), the total length of subsequences within X that can be referenced from

i and j is given by T= | X(i) | + | X(j) | - | X(i) ∩ X(j) |. Obviously if these subsequences are well spread out such that | X(i) ∩ X(j) | is zero, i.e., they do not overlap in position, T maximized. This implies that a high proportion of the nucleotides within X can be predicted by cross referencing among chromosomes, resulting in a high compression gain.

The Biological sequences may be compressed by applying an algorithm based on fixed length look up table. But in this paper a different approach has been used to develop/ design the algorithm which is based on variable length look up table. Different Biological sequences have been taken as a sample and passed to developed program. The results obtained so have been given in tabular form as in Table 2. In the Table 2 the compressibility of some Biological sequences obtained by the method of algorithm based on variable length look up table has been also presented.

From Table-2 we can easily conclude the result that variable length length LUT compression method produces better compression, hence in the proposed algorithm we are using variable length LUT.

The analysis of Table-3 shows that the compression ratio is 91.87% which is the highest gain achieved by other existing algorithms.

Type of the Sequences	Original Size(bits) before compression	Size of the sequence after applying various compression algorithm			
		DNACompress	GenCompress	Fixed LUT	Variable LUT
Gallus β globin	752	272	360	256	248
Goat alanine β globin	732	256	352	248	232
Human β globin	752	272	360	256	248
Lemur β globin	760	280	376	264	256
Mouse β globin	776	280	376	264	256
Opossum β hemoglobin β- M gene	760	272	376	264	256
Rabbit β globin	736	264	352	256	248
Rat β globin	752	272	360	256	248
Avg	752.5	271	364	258	249

TABLE 2: Comparaisons between Different Biological Sequence Compression Techniques.

Type of Sequences	With reference to	Original Size(bits) before compression	Size of the sequence after applying compression algorithm	
			Cross chromosomal Similarity	Cross chromosomal Similarity Using variable length LUT
Gallus β globin	Human β globin	752	192	56
Goat alanine β globin	Rat β globin	732	192	56
Human β globin	Mouse β globin	752	176	56
	Rabbit β globin	752	176	56
Lemur β globin	Gallus β globin	760	40	16
	Opossum β hemoglobin β- M gene	760	56	16
Mouse β globin	Human β globin	776	208	72
Opossum β hemoglobin β- M gene	Goat alanine β globin	760	240	72
Rabbit β globin	Human β globin	736	136	48
Rat β globin	Mouse β globin	752	340	72
Avg		752.5	165.6	52

TABLE 3: Cross Chromosomal Similarity Using Variable length
Compression ratio=91.87%.
Bits/ Base=.5526 bits/base.

5. CONCLUSION AND DISCUSSION

There are several algorithms for the compression of biological sequence/genome. The widely used algorithms GenCompress, DNA Compress have the characteristics of simplicity & flexibility. Our proposed algorithm is also simpler and more flexible. All the existing algorithms are either statistics based or dictionary based. In this regard our algorithm is dictionary based. One more characteristic for our algorithm is that unlike other algorithm our algorithm is trying to compress whole genome structure.

The proposed algorithm is also very helpful to find the relatedness among different sequences. Also it is very useful in multiple sequence compression for both repetitive and non-repetitive. The result analysis of the application of our algorithm shows high compression ratio to other exiting Biological Sequence Compression. This algorithm also uses less memory compared to the other algorithm and its implementation is comparatively simple.

The proposed algorithm compresses all the Biological sequences having self chromosomal and cross chromosomal similarities. While all other algorithms only use one of the properties of sequences. If the sequence is compressed using proposed algorithm it will be easier to make sequence analysis between compressed sequences. It will also be easier to make multi sequence alignment. High compression ratio also suggests a highly similar sequences.

6. REFERENCES

1. Ateet Mehta , 2010, et al., *" DNA Compression using Hash Based Data Structure*", IJIT&KM, Vol2 No.2, pp. 383-386.

2. B.A., 2005, " *Genetics: A comceptual approach*." Freeman, PP 311.

3.Choi Ping Paula Wu, 2008, et al., " *Cross chromosomal similarity for DNA sequence compression*", Bioinformatics 2(9): 412-416.

4. Gregory Vey, 2009, "*Differential direct coding: a compression algorithm for nucleotide sequence data*", Database, doi: 10.1093/database/bap013.

5. J. Ziv and A., 1977, et al, "*A universal algorithm for sequential data compression*," IEEE Transactions on Information Theory, vol. IT-23.

6. K.N. Mishra, 2010, " *An efficient Horizontal and Vertical Method for Online DNA sequence Compression*", IJCA(0975-8887), Vol3, PP 39-45.

7. P. raja Rajeswari, 2010, et al., " *GENBIT Compress- Algorithm for repetitive and non repetitive DNA sequences*", JTAIT, PP 25-29.

8. Pavol Hanus, 2010, et al., "*Compression of whole Genome Alignments*", IEE Transactions of Information Theory, vol.56, No.2Doi: 10.1109/TIT.2009.2037052.

9. R. Curnow, 1989, et al. *"Statistical analysis of deoxyribonucleic acid sequence data-a review*," J Royal Statistical Soc., vol. 152, pp. 199-220.

10. Sheng Bao, 2005, et al. "A *DNA Sequence Compression Algorithm Based on LUT and LZ77*", IEEE International Symposium on Signal Processing and Information Technology.

11. U. Ghoshdastider, 2005, et al., "*GenomeCompress: A Novel Algorithm for DNA Compression*", ISSN 0973-6824.

12. Xin Chen, 2002, et al.," DNA *Compress: fast and effective DNA sequence Compression*" BIOINFORMATICS APPLICATIONS NOTE, Vol. 18 no. 12, Pages 1696–1698.

13. X. Chen, 2002, et al., "*Dnacompress:fast and effective dna sequence compression*," Bioinformatics, vol. 18,.

14. Voet & Voet, Biochemistry, 3rd Edition, 2004.