DATA COMPRESSION FOR DNA SEQUENCE

Assist.Lec. Asaad Sumoom Daghal
Technical College-Najaf
Department of Technical Communication Engineering
Asaadasaad2000@yahoo.com

ABSTRACT
DNA Sequences making up any organism comprise the basic blueprint of that organism so that understanding and analyzing different genes within sequences has become an extremely important task. Biologists are producing huge volumes of DNA sequences every day that makes genome sequence database growing exponentially. The databases such as Gen-Bank represent millions of DNA sequences filling many thousands of gigabytes computer storage capacity. Hence an efficient algorithm to compress DNA sequence is required. In this paper compression algorithm which is called "Huffman code tree" is used to code and compress DNA sequences. Depending upon this algorithm we assigning binary bit codes (0 and 1) for each base (A, T, C, and G). After assigning the bases by bit codes, we determine the code for each base. Code for each base is determined by tracing out the path from the root of the tree to the leaf that represents that base.
Huffman code provides a variable code length. In fact the codes for characters having a higher frequency of occurrence are shorter than those codes for characters having lower frequency. So this algorithm compress DNA sequences better than from old method (fixed length) if we assigning 2 bits per base. From analysis the results, average code length (1.62 bits/base) can be achieved using this algorithm. For a higher compression ratio advised to use other compression method with the proposed method such as the learning automata.

KEY WORDS: DNA, Huffman Code, Compression

ضغط البيانات لمتوالية الحمض النووي

مدرس مساعد . أسعد سموم دغل
الكلية التقنية –نجف / قسم هندسة تقنية الاتصالات

الخلاصة
تشكل متواليات الحمض النووي لأي كائن حي يشمل المخطط الأساسي لهذا الكائن بحيث أن فهم وتحليل الجينات المختلفة ضمن المتواليات أصبحت مهمة هامة للغاية. في كل يوم ينتج علماء الأحياء كميات ضخمة من تسلسل الحمض النووي ، هذا ما يجعل قاعدة بيانات الجينوم متزايدة باطراد. قواعد البيانات مثل بنك الجينات تمثل الملايين من تسلسل الحمض النووي التي تحتاج سعة خزن قد تصل إلى عدة آلاف من الغيغابايت من سعة الكمبيوتر. وبالتالي مطلوب خوارزمية فعالة لضغط تسلسل الحمض النووي. في هذا

البحث خوارزمية الضغط المستخدمة التي تسمى "رمز شجرة هوفمان " يستخدم لترميز وضغط تسلسل الحمض النووي. اعتمادا على هذه الخوارزمية يتم تأشير رمز قطعة ثنائي (0 و 1) لكل قاعدة (A, T, C, G). بعد تأشير الرموز للقواعد نجد الرمز لكل قاعدة. يتم إيجاد الرمز لكل قاعدة عن طريق تتبع المسار من جذر الشجرة إلى الورقة التي تمثل تلك القاعدة.

ترميز هوفمان يوفر طول رمز متغير. في الحقيقة الرموز لشخصيات التي لها تكرار عالي للظهور يكون تمثيلها بطول أقصر من الرموز للحروف التي لها تكرار قليل. لذلك فأن الخوارزمية المقترحة تضغط متواليات الحمض النووي أفضل من الطرقة القديمة(طول ثابت) إذا نحن استخدمنا 2 قطعة لكل قاعدة. ومن تحليل النتائج، معدل طول رمز 1.62 بت لكل قاعدة يمكن انجازه باستخدام الخوارزمية المقترحة . للحصول على نسبة ضغط أفضل، ينصح باستخدام طريقة أخرى مثلا التعلم الآلي مع الطريقة المستخدمة.

1. INTRODUCTION

The compression of DNA (Deoxyribonucleic Acid) sequences is one of the most challenging tasks in the field of data compression. Since DNA sequences are "the code of life" we expect them to be non-random and to present some regularities. It is natural to try taking advantage of such regularities in order to compactly store the huge DNA data bases which are routinely handled by molecular biologists. **[2]**

The amount of DNA being extracted from organisms and sequenced is increasing exponentially. This yields two problems: storage and comprehension. Despite the prevalence of broadband network connections, there still exists a need for compact representation of data to speed up transmission. Transferring a single sequence that is millions of characters long to may take ten to fifteen minutes over a dial up connection. Compression of genomic sequence can decrease the storage requirements, and increase the transmission speed.

DNA sequences are comprised of just four different bases labeled A, T, C, and G (for adenine, thymine, cytosine, and guanine respectively). T pairs with A, and G pairs with C, and can be coded using two bits per base. According to functionality, DNA manifests different properties from other kinds of data. Standard compression algorithms for text or image files exploit small repeated patterns and contextual similarities to achieve compression. However, repeated patterns in DNA sequences are typically much longer and less frequent, so standard compression algorithms perform poorly on DNA. The most popular general-purpose encoders of today, such as **gzip [4]**, which is based on the Lempel-Ziv algorithm **[5]**, and bzip2, based on the Burrows-Wheeler Transform **[7]**, which usually produces more than two bits per base to achieve the un-encoded representation. Hence, the quest for efficient DNA compression programs started to become popular in the competition-driven community of data compression enthusiasts. DNA sequences are compressible, so they are not random. But they are not highly compressible. It is therefore necessary for coding methods to be as efficient as possible. In the context of compression, missing structure will lead to inefficient compression. **[3]**

2. DATA COMPRESSION:

In computer science and information theory, data compression or source coding is the process of encoding information using fewer bits (or other information- bearing units) than an un-encoded representation would use, through use specific encoding schemes. Compression is useful because it helps reduce the consumption of expensive resources, such as hard disk space or transmission bandwidth. Compressed data must be decompressed to be used, and this extra processing may be detrimental to some applications. Therefore, the design of data compression schemes therefore involves trade-offs among various factors, including the degree of compression and the computational resources required to compress and uncompress the data. Further some compression

algorithms can introduce distortion in data which are known as lossy compression. Lossless compression algorithms usually exploit statistical redundancy in such a way as to represent the sender's data more concisely without error **[6]**. Lossless compression is possible because most real-world data has statistical redundancy. For example, in English text, the letter "e" is much more common than the letter "z", and the probability that the letter "q" will be followed by the letter "z" is very small. Lossless compression exploits the repeats, palindromes and patterns present in the digital to reduce the over all size.

The ratio of the original, uncompressed data file and the compressed file is referred to as the compression ratio. The compression ratio is denoted by Equation (1). **[1]**

$$\text{Compression ratio} = \frac{\text{size of the original file}}{\text{size of the compressed file}} \tag{1}$$

3. HUFFMAN CODING

In computer science and information theory, Huffman coding is an entropy encoding algorithm used for lossless data compression. The term refers to the use of a variable-length code table for encoding a source symbol (such as a character in a file) where the variable-length code table has been derived in a particular way based on the estimated probability of occurrence for each possible value of the source symbol. It was developed by David A. Huffman.

Huffman coding uses a specific method for choosing the representation for each symbol, resulting in a prefix code (sometimes called "prefix-free codes"), that is, the bit string representing some particular symbol is never a prefix of the bit string representing any other symbol that expresses the most common source symbols using shorter strings of bits than are used for less common source symbols. Huffman was able to design the most efficient compression method of this type: no other mapping of individual source symbols to unique strings of bits will produce a smaller average output size when the actual symbol frequencies agree with those used to create the code. A method was later found to design a Huffman code in linear time if input probabilities (also known as weights) are sorted.

Huffman tree generated from the exact frequencies of the text "this is an example of a Huffman tree". The frequencies and codes of each character are shown in figure 1 and **Table 1**. Encoding the sentence with this code requires 135 bits, as opposed to 288 bits if 36 characters of 8 bits were used **[6].**

4. THE AVERAGE CODE LENGTH OR WEIGHTED LENGTH

The average number of bits per base (length) is measured according to the following Equation (2). **[1]**

$$L_{avg} = \sum_{i=1}^{L} l_i \times p_i \tag{2}$$

Where

L_{avg}= average code length

l_i = code length in bits.

p_i = Histogram probability.

We can calculate histogram probability (p_i) from Equation. (3).

$$p_i = \frac{\text{Base length}}{\text{Sequence length}} \tag{3}$$

Or we can calculate weighted length from Equation (4). **[8]**

$$w_i = \sum_{i=1}^{n} l(i) * f(i) \quad (4)$$

Where

w_i = Weighted Extended Path (code length)

$l(i)$ = length of path (number of edges on path) from root to external node labeled i

$f(i)$ = frequency of occurrences of bases

5. CODE EXISTING FOR DNA SEQUENCE

For a DNA (A, C, G, T), in classical code method we can represent it in length by 2 bits per base, and assigning as (A=00), (C=01), (G=10), (T=11).This yields when the string A, C ,G, T whose sequence length is 1200, in classical code method (fixed length) , this requires 2400 bits of space. That's mean the storage of encoded sequence is almost double its original sequence length **[9].**

6. PROPOSED ALGORITHM:

The process of proposed algorithm is achieved by constructed Huffman tree, and then derived codes from it, that are used to calculate the number of bits in the encoded sequence length.

6.1 Structure Of Huffman Tree Codes:

Huffman coding is the based on building a binary tree that holds all characters in source at its leaf nodes, and with there corresponding characters frequencies (probabilities) at the side. The nodes from the original tree are called internal nodes. The special nodes are called external nodes. The following tree shown in **Figure 2** is extended binary tree. Empty circles represent internal nodes and boxes represent external nodes. Every internal node consists of exactly 2 children and every external node is a leaf.

The tree is built by going through the following steps**. [10]**

1- Combine the two lowest frequencies (probabilities) and continue this procedure.
2- Assign "0" to higher frequency (prob.) and "1" to lower frequency (prob.) of each pair, or vice versa.
3- Trace the path for each character frequency from lower to upper point. Recording the one's and zeros along the path.
4- Assign each character (message) codes sequentially from right to left.

6.2 Tree Of DNA Sequence {A, C, G, T}:

Because the DNA sequence have only four latter (A, C, G, T), we show that from the **Figure 2**, the number of internal node is (3) and the number of external node is (4). Using **Figure 2** and procedure of Huffman code tree, we can built the code of DNA sequence and get the results as: (A=1, T=00, C=010, G=011)

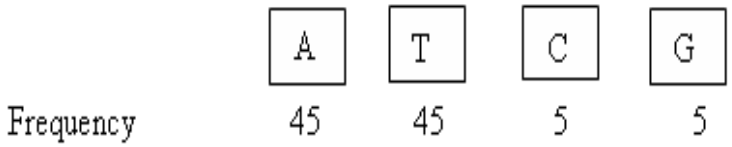
From the result of code DNA by Huffman tree we show that the four latter always represented by 7 bits and we can calculate weighted length from Equation (4) as:

Calculation of Bits in Encoded sequence = 1*f(a)+2*f(t)+3*f(g)+3*f(c).

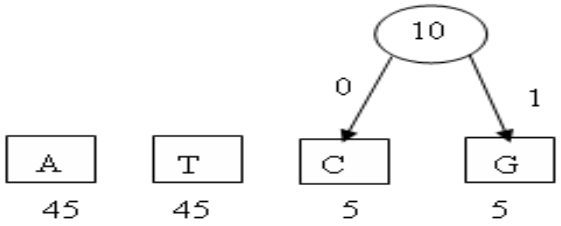
6.3 Construct Tree Of DNA Sequence:

We can construct DNA tree and encoded being code step by step as shown below.

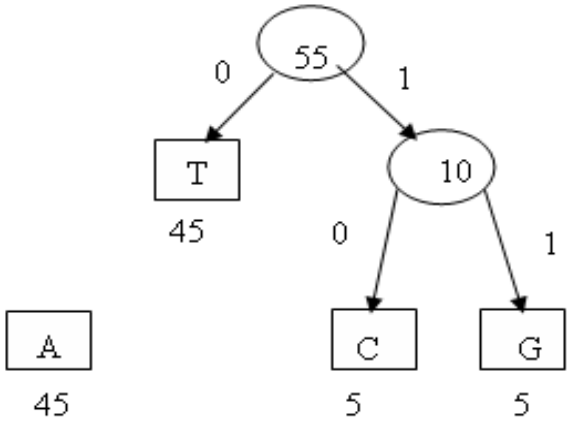
(Step1)

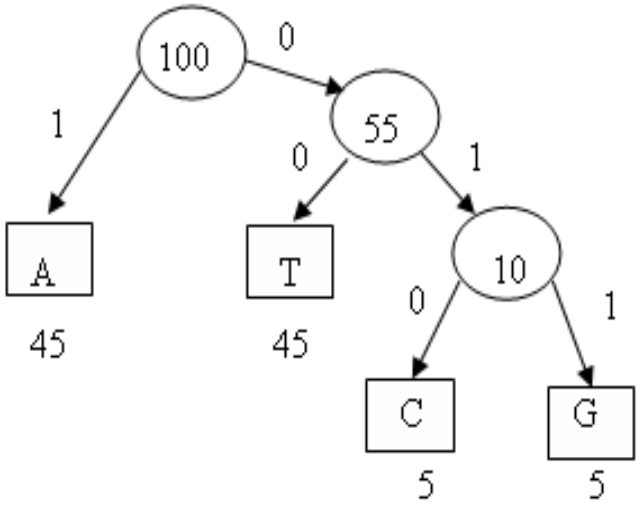


(Step 2) Combining the lowest frequency values and assigns code



(Step 3) Repeat step 2 until the end





7. EXAMPLES AND CALCULATIONS:

Example 1: Given Sub Sequence: (aataaaataaaacaaaaaaattaaaagaaaccaaagaattaaaaatta…)
Sequence length = 600.

Base	Frequency	Code
A	275	1
T	275	00
C	25	010
G	25	011

From equation (4), Number of Bits in Encoded sequence
= 1*f (a) +2*f (t) +3*f (g) +3*f(c)
= 1*275 + 2* 275 + 3*25 + 3*25
= 275 + 550 + 75 + 75 = (975 bits)

From Equation(1): $\text{Compression Ratio} = \frac{\text{number of total bits represent bases}}{\text{number of bits encoded}}$

$= \frac{1200}{975} = 1.23$

∴ Compression ratio in this case (1.23: 1)

Example 2: Given sub sequence: (ttttcagtgt tagattgctc taattctttg agctgttctc tcagctcctc atatttttct)
Sequence length = 60

Base	Frequency	Code
T	30	0
C	13	11
A	9	100
G	8	101

Number of Bits in Encoded sequence
= 1*f(t)+2*f(c)+3*f(a)+3*f(g).
= 1*30 + 2* 13 + 3*9 + 3*8
= 30+26+27+24 = (1 0 7 bits).

$$\text{Compression Ratio} = \frac{120}{107} = 1.1214$$

∴ Compression ratio in this case (1.1214: 1)

Example3: let us consider the sub sequence: (aggcgtatgcgatcctgaccatgcaa…)
Sequence length = 400

Base	Frequency	Code
A	50	1
T	50	01
C	50	000
G	50	001

Number of Bits in Encoded sequence
= 1*f (a) +2*f (t) +3*f (g) +3*f(c) = 1*50 + 2* 50 + 3*50 + 3*50
= 50 + 100 + 150 + 150 = (450 bits)

$$\text{Compression Ratio} = \frac{400}{450} = 0.8889$$

In this example (3) the probability (frequency) occurrence of all bases are equally, and that is represent the worst case. So the Huffman code method is not effective for compression ratio when the probability occurrences of letter are equally.
Table 2 shows all results (weight length, average code length and compression ratio)

8. COMPARISON WITH OTHER DNA COMPRESSORS

The most natural benchmark for our algorithms is a comparison with the other algorithms designed to compress DNA sequences. Unfortunately, such a comparison turned out to be a rather difficult task. The first reason is that the source or executable code of DNA compressors is usually not available. The second reason is that the huge space and time requirements of most DNA compressors make it difficult (or impossible) to test them on sequences of significant length

9. CONCLUSION:

The need for effective DNA compression is evident in biological applications where storage and transmission of DNA are involved. Algorithm using the concept of Huffmn tree is proposed to compress DNA sequences. The binary trees are used to derive the variable length codes that satisfy the prefix property. Since Huffman's code satisfies Prefix property, it's an efficient way to encode and decode DNA sequences. Using Huffman code algorithm, it will be easier to compress DNA sequences but it is not efficient when we use alone to get good compression ratio, because the DNA sequence is always as pair. This means the likelihood of the occurrence of (A and T) as well as equal and so on C & G. Therefore, this affects in compression ratio because the Huffman algorithm depends on the occurrence one of the elements with a high probability and the result is a short length of bits and this does not occur in DNA sequence**.** So this is representing the major limitations of using Huffman code to compress DNA sequence.

10. FUTURE WORK:

DNA sequences may be repetitive or non repetitive, For this we can take advantage of this property and use another compression techniques such as run length encoding or use Another way such as learning automata with proposed algorithm (Binary tree Huffman code)

11. REFERENCES:

[1] E. U. Scoot, " Computer Vision and Image Processing: A practical Approach Using CVIP Tools ", prentice Hall, Inc., 1998

[2] G. Manzini and M. Rastero " A Simple and Fast DNA Compression" Feburuary 17, 2004

[3] H. Afify,M.Islam and M.Abdel wahedL."DNA LossLess Differential Compression Algorithm Based on similarity of genomic sequence data base.(IJCSIT) Vol 3, No4, August 2011.

[4] J.Gailly, M.Adler,"gzip(GNUzip) compression utility", http://www.gnu.org/soft-ware/gzip/

[5] J.Ziv and A.Lempel,"A Universal Algorithm for Sequential Data Compression". IEEE Trans Information Theory, Vol.23,pp.337-343,1977

[6] K. Huffman." Data Compression and Huffman Coding". http://en.wikipedia.org/wiki/ data compression, Huffman coding, Sept. 1991

[7] M.Burrows and D.J.Wheeler,"A Block Sorting LossLess Data Compression Algorithms", Technical report124,Digital System Research center,1994.

[8] R. Rajeswari, A. Apparao and R. Kiran " Huffbit compress-algorithm to compress DNA

Sequences using extended binary trees. Jornal of theoretical and applied IT, pp.101-106, 2010.

[9] S. Grumach and F.Tahi ," A New Challenge for Compression Algorithms: Genetic Sequences", I.N.R.I.A, Recquencourt, Bp105, 78153 Le Chesnay France

[10] Y.Q. Shi and H. Sun. "Image and video compression for multimedia engineering Fundamentals, Algorithms and standars". Boca Raton London New York Washington, D.C.2000.

Table 1

Char.	Freq.	Code	Char.	Freq.	Code.
a	4	010	n	2	0010
e	4	000	o	1	00110
f	3	1101	p	1	10011
h	2	1010	r	1	11000
i	2	1000	S	2	1011
l	1	11001	Space	7	111
m	2	0111	t	2	0110
u	1	00111	x	1	10010

Table 2

DNA Sequence length	Base	Frequent base	Probability Eq,.(3)	Code base	Weight length Eq.(4)	Average code length Bits/base Eq.(2)	Compression Ratio Eq.(1)
Sub Sequence 1 100	A	45	0.45	1	165	1.65	1.212: 1
	T	45	0.45	00			
	C	5	0.05	010			
	G	5	0.05	011			
Sub Sequence 2 600	A	275	0.4584	0	975	1.6248	1.23: 1
	T	275	0.4584	00			
	C	25	0.0416	010			
	G	25	0.0416	011			
Sub Sequence 3 400	A	50	0.25	0	450	2.25	0.8889: 1
	T	50	0.25	11			
	C	50	0.25	100			
	G	50	0.25	101			
Sub Sequence 4 60	A	30	0.5	1	107	1.7833	1.1214:1
	C	13	0.2167	01			
	T	9	0.15	000			
	G	8	0.1333	001			

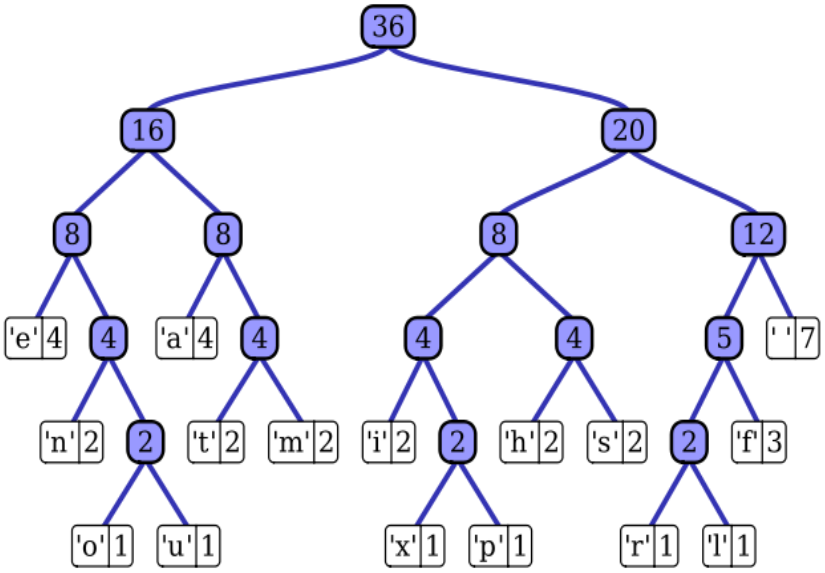


Figure 1 Huffman tree.

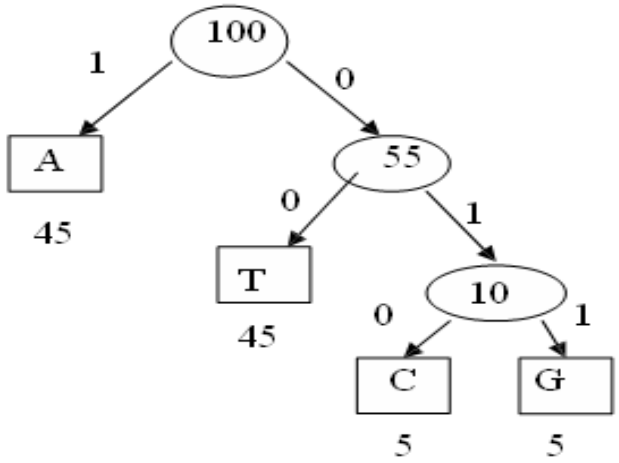


Figure 2 Huffman tree codes for DNA