Undergraduate Journal of Mathematical Modeling: One + Two

Volume 4 | 2012 Spring Article 3

2012

Complexity of Mitochondrial Genome Sequences

Brandon Toun
University of South Florida

Advisors:
Arcadii Grinshpan, Mathematics and Statistics
Egor Dolzhenko, Princeton University: Evolutionary Biology VSRC

Problem Suggested By: Egor Dolzhenko

Recommended Citation
Toun, Brandon (2012) "Complexity of Mitochondrial Genome Sequences," *Undergraduate Journal of Mathematical Modeling: One + Two*: Vol. 4: Iss. 2, Article 3.
DOI: http://dx.doi.org/10.5038/2326-3652.4.2.3
Available at: https://digitalcommons.usf.edu/ujmm/vol4/iss2/3

Complexity of Mitochondrial Genome Sequences

Abstract

The purpose of this project is to compare the complexities of different species' mitochondrial genome sequences. Using an implementation of Deflate compression algorithm from Java standard library, we were able to compress mitochondrial genomes of nine different species. The complexity of each sequence is estimated as a ratio of the original sequence length to the length of the compressed sequence. In addition, we show how a notion of topological entropy from symbolic dynamics can be used as another complexity measure of nucleotide sequences.

Keywords

Mitochondrial Genome, LZ77, Huffman Coding, Deflater Algorithm

TABLE OF CONTENTS

PROBLEM STATEMENT

Compare complexities of different species' mitochondrial genome sequences.

MOTIVATION

Any abnormalities in mitochondrial genome can be deleterious for the rest of the cell. In particular, various mutations in mitochondrial genomes are associated with human diseases (Ruiz-Pesini, Lott and Procaccio). Furthermore, it is conceivable that in a near future analysis of mitochondrial genomes becomes a routine way to diagnose such diseases. This calls for novel computational techniques for studying nucleotide sequences. In particular, comparing sequence complexity could be used to detect abnormalities in the mitochondrial genomes and can also be used to compare the genomes on an evolutionary scale.

MATHEMATICAL DESCRIPTION AND SOLUTION APPROACH

We are interested in comparing complexities of mitochondrial genome sequences of different species. Using sequences from a genomics database and a Java program to compress them we were able to estimate the complexity of these sequences (see Appendix A). We observed that the code works well even for large (6 – 38kb) sequences.

The program prompts the user to enter a sequence and subsequently calculates the sequence length. It then compresses the sequence using the Deflater algorithm (Rose India Technologies; Feldspar), which is a part of the Java standard library.

The Deflater algorithm is a combination of Huffman coding (Huffman) and LZ77 (Ziv and Lempel) compression. Huffman coding gives each distinct value in a piece of data a code.

https://digitalcommons.usf.edu/ujmm/vol4/iss2/3
DOI: http://dx.doi.org/10.5038/2326-3652.4.2.3

For example, the word $w = AACCCTTTTGGGGG$ consists of four distinct letters: A, C, T, and G that can be encoded by strings 1101, 000, 010, 111, respectively. Which implies that the word w can be encoded using $2 \times 4 + 3 \times 3 + 4 \times 3 + 5 \times 3 = 44$ whereas with 8 bit ASCII encoding for each symbol, w requires $16 \times 8 = 128$ bits.

LZ77 algorithm encodes strings as a sequence of triples. For example, a string $w = ACTACTG$ is encoded by triples $(0,0,A)$, $(0,0,C)$, $(0,0,T)$, $(3,3,G)$. The string w can be reconstructed from these triples as follows: (1) first three triples mean that A, C, and T must be appended to a word to get ACT, then (2) triple $(3,3,G)$ instructs to first add tree symbols starting at position 3 (from right to left) and then append a symbol G. The result is the original word w. It's clear that repetitive sequences can be encoded by this algorithm very efficiently.

The complexity is calculated according to the following expression

$$C(w) = \frac{|w|}{|w_i|}$$

Where $C(w)$ is the complexity of the sequence, $|w|$ is the length of the original sequence, and $|w_i|$ is the length of the sequence after compression. Note that $C(w)$ is never less than 1; the higher $C(w)$ is, the less complex the sequence is and vice versa.

Since DNA sequences are often extremely large, they could be studied with tools developed for infinite sequences in symbolic dynamics. One such tool is the topological entropy (Lind and Marcus) which could be calculated through the following the limit

$$E(w) = \lim_{n\to\infty} \frac{1}{n} \log(F_n).$$

Here, F_n stands for the number of all subsequences of w of length n. For example, if $w = ABCDE$, then $F_1 = 5$ since A, B, C, D, E are exactly the subsequences of w of length 1. Similar

calculation shows that $F_2 = 4$, and so on.

Although the entropy is defined for infinite sequences, you can still apply this notion to extremely long finite sequences, by calculating F_n up to some fixed parameter k that would depend of the length of the original sequence, or extending the original finite sequence into an infinite sequence by assuming that the extended sequence inherits the statistical properties of the original. In future, it would be interesting to develop the notion of topological entropy for finite sequences and determine how it compares to other measures of complexity.

DISCUSSION

Our estimates of complexities for the mitochondrial genomes of 9 species are presented in Table 1. Note that the complexity may be affected by the size of the original sequence. For example, *Laminaria digitata* (large brown algae) had the largest mitochondrial genome sequence of the nine species and its complexity turned out to be quite high. The next two largest sequences are of *Lottia digitalis* (a mollusk) and *Leishmania tarentolae* (a virus) respectively. These were the least complex of the nine species, possibly because their mitochondria need fewer genes to function. The smallest and the least compressible sequence was that of *Elaeis guineensis* (an oil palm).

CONCLUSION AND RECOMMENDATIONS

In the future, it may be possible to quickly and efficiently sequence mitochondrial genomes. Thus, if a patient is believed to be developing a disease associated with mitochondrial mutations, the doctor may test whether the patient's mitochondrial genome is different from the normal (reference) genome. One possible way of doing this is to compress the sequence and look

https://digitalcommons.usf.edu/ujmm/vol4/iss2/3
DOI: http://dx.doi.org/10.5038/2326-3652.4.2.3

for a difference in complexity. If the complexity difference is indeed found, more elaborate computational techniques (such as sequence alignment) could be used to find out what exactly what the differences are.

NOMENCLATURE

Symbol	Description
$\|W\|$	length of the sequence W
$\|W_i\|$	length of the sequence W
$C(w)$	Complexity of the sequence w
F_n	number of all subsequences of w of length n

REFERENCES

Feldspar, Antaeus. An Explanation of the DEFLATE Algorithm. 23 August 1997. 1 May 2012 <http://www.cs.ucdavis.edu/~martel/122a/deflate.html>.

Huffman, D.A. "A Method for the Construction of Minimum-Redundancy Codes." Proceedings of the I.R.E. (1952): 1098-1102.

Lind, Douglas A and Brian Marcus. An Introduction to Symbolic Dynamics and Coding. Cambridge: Cambridge University Press, 1995.

Rose India Technologies. String Compression Using Deflater Class in Java. n.d. 27 April 2012 <http://www.roseindia.net/tutorial/java/corejava/zip/compression.html>.

Ruiz-Pesini, E., et al. "An enhanced MITOMAP with a global mtDNA mutational phylogeny." Nucleic Acids Research 35 (Database issue) (2007): D823-D828.

Schaffer, S W and M.-Saadeh Suleiman. Mitochondria: The Dynamic Organelle. New York: Springer, 2007.

Ziv, Jacob and Abraham Lempel. "A Universal Algorithm for Sequential Data Compression." IEEE Transactions on Information Theory, 23(3) (1977): 337-343.

APPENDIX-CODE

```
import java.io.*;
import java.util.zip.*;
import java.util.zip.Deflater;
import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.util.Scanner;

public class SequenceCompressor {
  public static void main(String args[]) {
  Scanner scan = new Scanner(System.in);
  System.out.println("Enter sequence: ");

    String sequence = scan.nextLine();

    byte[] dataByte = sequence.getBytes();
    System.out.println("Mitochondrial Genome Sequence Compressor");
    System.out.println("Actual Size of Sequence : " + dataByte.length);
    Deflater deflater = new Deflater();
    deflater.setLevel(Deflater.BEST_COMPRESSION);
    deflater.setInput(dataByte);
    deflater.finish();
    ByteArrayOutputStream byteArray = new ByteArrayOutputStream(
        dataByte.length);
    byte[] buf = new byte[1024];
    while (!deflater.finished()) {
      int compByte = deflater.deflate(buf);
      byteArray.write(buf, 0, compByte);
    }
    try

    {
      byteArray.close();
    }
```

https://digitalcommons.usf.edu/ujmm/vol4/iss2/3
DOI: http://dx.doi.org/10.5038/2326-3652.4.2.3

```
        catch (IOException ioe) {
          System.out.println("When we will close stream error : " + ioe);
        }

        byte[] comData = byteArray.toByteArray();

        System.out.println("Compressed size of sequence : " + comData.length);
      }
    }
```

Listing 1. Java source code of the compression algorithm.

APPENDIX-FIGURES

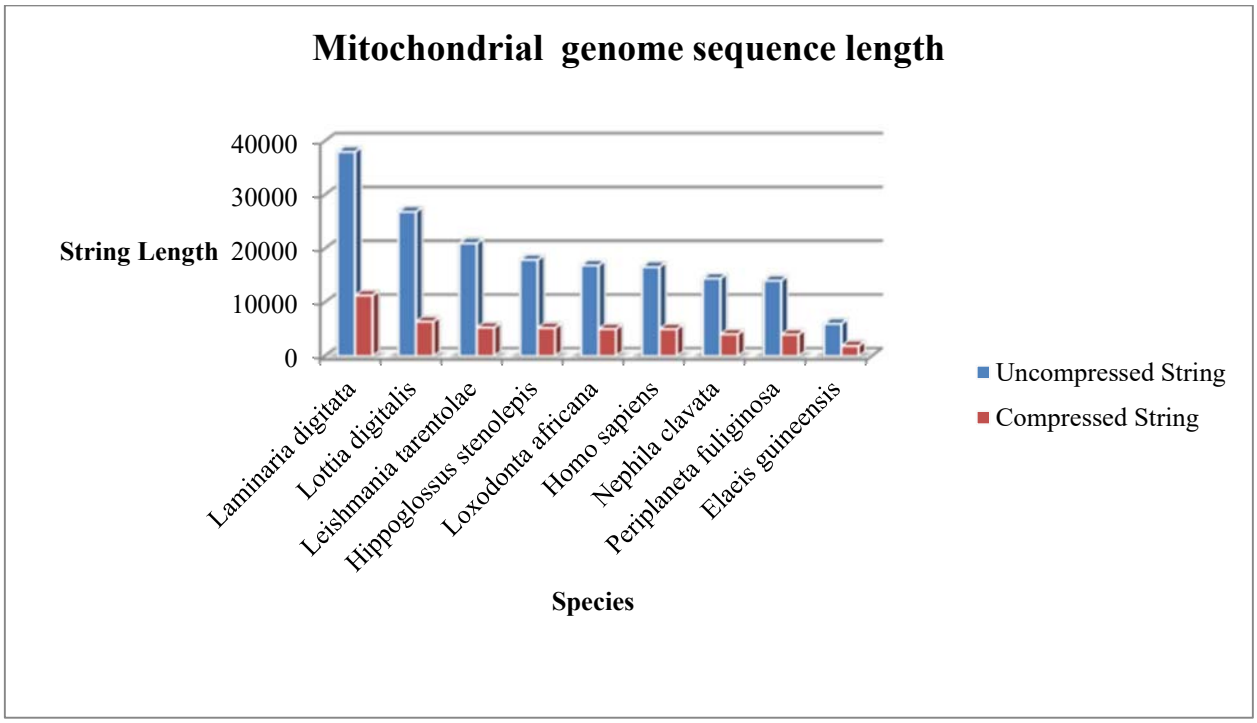

Figure 1: Bar graph comparing original sequence lengths to compressed versions.

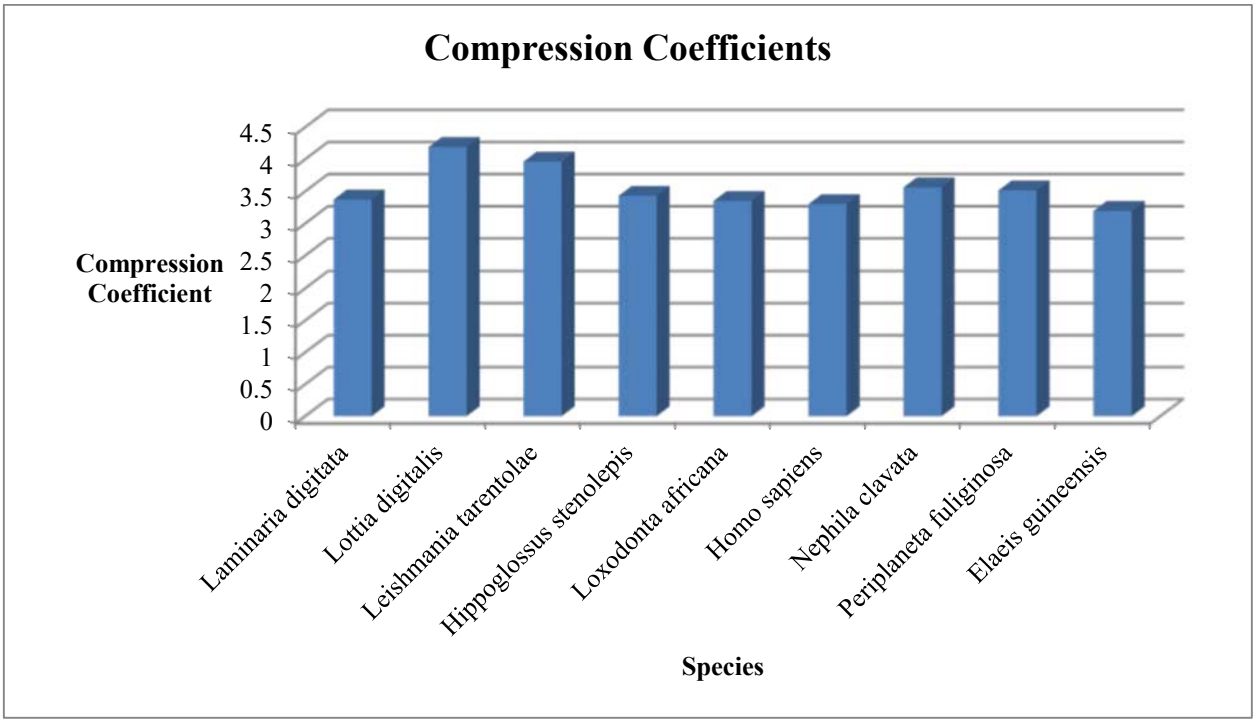

Figure 2: Comparison of complexity of different species' mitochondrial genome sequences.

APPENDIX-TABLES

Species	Original Sequence Length	Compressed Sequence Length	Compression Coefficient
Laminaria digitata	38007	11272	3.37
Lottia digitalis	26835	6403	4.19
Leishmania tarentolae	20992	5297	3.96
Hippoglossus stenolepis	17902	5221	3.43
Loxodonta africana	16866	5034	3.35
Homo sapiens	16569	5019	3.30
Nephila clavata	14436	4054	3.56
Periplaneta fuliginosa	14021	3989	3.51
Elaeis guineensis	6009	1881	3.19

Table 1: Lengths of mitochondrial genomes and compression coefficients.

https://digitalcommons.usf.edu/ujmm/vol4/iss2/3
DOI: http://dx.doi.org/10.5038/2326-3652.4.2.3