Genome Data Compression using Digital Chaos

Sai Venkatesh Balasubramanian
Sree Sai Vidhya Mandhir, Mallasandra, Bengaluru-560109, Karnataka, India
saivenkateshbalasubramanian@gmail.com

Abstract:

Efficient techniques of Genome Data handling and storage are the need of the hour in the present genetic engineering era. The present work purports to the design and implementation of a Genome Sequence Data Compression Technique without the use of references and lookup. This is achieved by first generating a digital chaotic bit stream, formed by performing XOR operations on three square waves with mismatched frequencies. The generated bit stream is XORed with the Genome Sequence bit stream after necessary data conditioning, and the result is stored as a 2D array (image). The png format is chosen, owing to its inherent lossless properties. It is seen that the perfectly reversible operations of compression and decompression result in compression ratios of around 2.6-3.5 being achieved with absolute zero error. The use of digital chaos provides an additional layer of security, since the frequencies of the input square wave signals form a secure key, which when mismatched during decompression even by 1 percent, can result in error rates of upto 60 percent.

Keywords: Genome Sequence Data, Data Compression, Digital Chaos, LabVIEW

1. Introduction

In the ongoing era of genetic engineering, the importance of genome data handling and storage cannot be over-emphasized [1-4]. Applications of genome sequencing and data handling range from personalized medicine using genetics and epigenetics, by identifying key alleles, to genographic applications, where specific mitochondrial DNA (mtDNA) Haplogroups are used to trace human ancestry and migration patterns [5-10]. Fortunately, the information explosion that has occurred in recent years, along with the advances in Big Data Technologies have provided a convenient platform to handle huge amounts of genome data [1-4,11]. The issues most prominent in DNA data handling are often twofold. First, to store genome data for posterity, efficient data compression techniques are required [12]. Second, the recording of individual human genome data requires a high level security to prevent unwanted abuse and to protect privacy [13,14].

Conventionally, Genome Compression has been performed using the standard coding techniques such as Huffman Coding and Lempel-Ziv Coding, arising out of information theory [15-22]. To make the process more effective, reference-based Genome compression have been proposed, which can achieve high compression ratios, provided a “dictionary” is required during decompression [17-19].

The present work purports to the design and implementation of a new technique, which is able to compress Genome Data almost threefold, without the use of any reference data. Specifically, the base for storing the compressed genome data is chosen as a “digital chaotic image”. This is obtained by first performing an exclusive or (XOR) operation on three discrete square wave signals, with mismatched frequencies. The output signal is characterized using standard measures such as Lyapunov Exponent and Kolmogorov Entropy, and the presence of chaos is ascertained. The genome sequence to be compressed is converted to a byte stream using appropriate mappings and is then XOR-ed with the digital chaotic output obtained earlier. The resultant byte array is converted and reshaped into a two-dimensional matrix, saved as grayscale portable network graphics (png) image. It is observed that the size of this image is much lesser than the raw genome sequence text file. The decompression is achieved by performing the exact inverse operation and it is seen that the data is restored intact without any error. The use of digital chaos to compress the genome data provides an additional layer of security, since the frequencies of the input signals used to create the digital chaos act as a form of secure key, which needs to be reproduced exactly to ensure correct decompression. The simplicity of the proposed genome compression technique coupled with its ability to compress genome data threefold standalone forms the highlight of the present work. It is noted that this technique can be used in conjunction with other established techniques to increase the efficiency and compression ratio even further [15-22].

2. Methodology and Results

The block diagram of the proposed genome compression technique is illustrated in Fig. 1. The four essential components, namely digital chaos generation, genome data conditioning, compression and decompression are elaborated below:

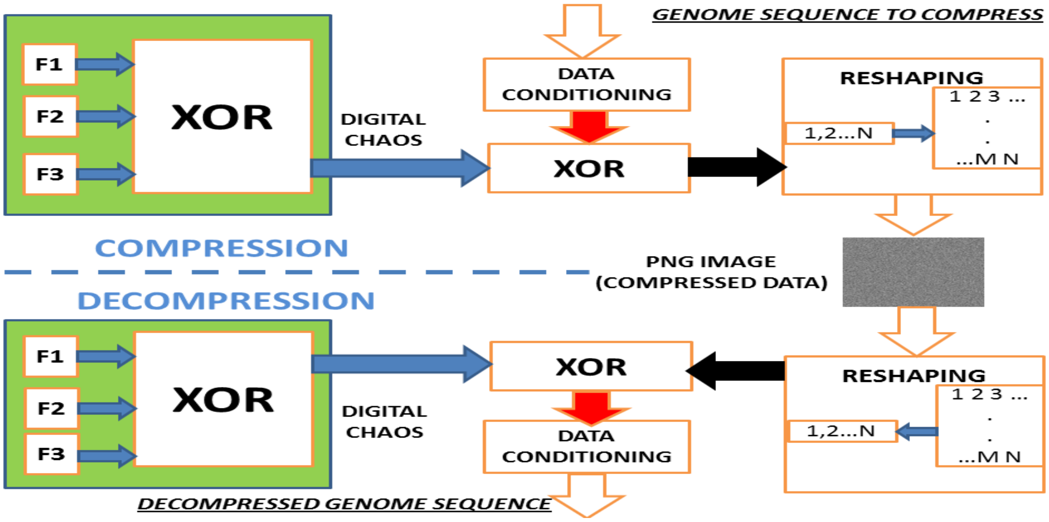

Figure 1 Block Diagram illustrating the proposed Genome Data Compression Technique

A. Generation of Digital Chaos

The Digital Chaos generation, represented by the green blocks in Fig. 1, essentially consists of a simple XOR circuit [23]. The premise is that the XOR, being a difference function, will amplify any mismatches in its input signals, and by setting the mismatched parameter as frequency, the out of sync lead/lag will increase with each cycle, progressively leading to chaos, the implementation being a real-time visualization of the butterfly effect [24-27]. Thus, the three input signals denoted by *F1*, *F2* and *F3* to the XOR Gate are chosen as discrete square waves, with the time periods set to 2.7, 3.86 and 9 samples respectively. The use of such non-integral values is essential to keep the three signals out of sync, causing the required mismatches, which are promptly amplified by the XOR Gate. The obtained signal C is plotted for the first 1000 samples in Fig. 2.

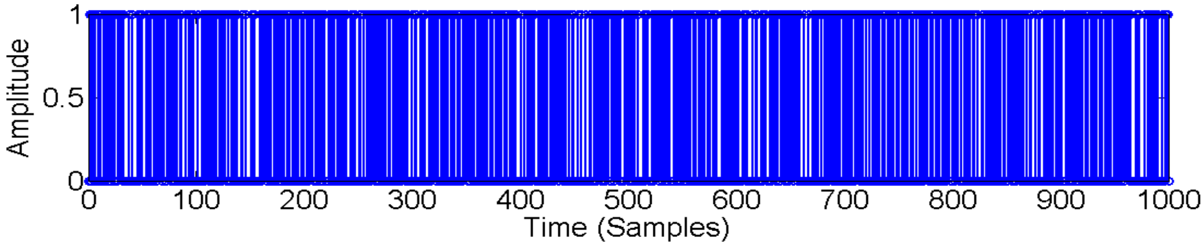

Figure 2 The first 1000 samples of the generated digital chaotic output

The most suitable measure to ascertain the presence of and to characterize chaos in the obtained output is the Largest Lyapunov Exponent (LLE), a measure of a system's sensitive dependence on initial conditions [25]. In the present work, Rosenstein's algorithm is used to compute the Lyapunov Exponents λ_i from the voltage waveform, where the sensitive dependence is characterized by the divergence samples $d_j(i)$ between nearest trajectories represented by i given as follows, C_j being a normalization constant [29]:

$$d_j(i) = C_j e^{\lambda_1 (i\Delta t)}; d(t) = Ce^{\lambda_1 t} \qquad (1)$$

The second parameter used to characterize chaos is the Kolmogorov Entropy, which is essentially a statistical measure of the uncertainty of the signal [30]. By assigning each of the two quantifiable states (0 and 1) of the output amplitude as an event i, the Kolmogorov Entropy K_2 obtained depends on their probabilities p_i [30].

For the output signal shown in Fig. 1, a positive LLE of 2.43 is obtained, along with Kolmogorov Entropy of 1.23 bits/symbol, both the values testifying to the presence of chaos. Since the key contributors to the chaos are the three input frequencies, and since the obtained chaotic signal is a bit stream of 1's and 0's, the output is termed ‘Frequency Controlled Digital Chaos'.

B. Data Conditioning of the Genome Sequence

A genome sequence typically consists of nucleotides, such as *A*, *C*, *G* and *T*, representing Adenine, Cytosine, Guanine and Thymine respectively [31]. Apart from the nucleotides, genome sequences also contain spaces, line feeds and null characters [31]. Thus, it is seen that the total types of characters in a genome sequence is 7, and thus, each character can be mapped to a 3-bit value as shown in Table 1.

Table 1 Mapping the Genome Sequences to 3-bit values

Character	ASCII Value	3-bit Value
Null	00	000
Line feed	10	110
Space	32	101
A	65	001
C	67	010
G	71	011
T	84	100

Mapping a genome sequence of length *N* into 3-bit values result in a bit stream of size *3N*. However, these bits have to be converted to bytes before they can be stored as an image. To do so, the size of binary data N_b should be divisible by 8 and with the square root of the quotient being an integer. To achieve this, appropriate zero padding is performed, with the number of zeroes *Z* given as follows:

$$Z = \frac{\left(\left\lceil\sqrt{\frac{n}{8}}\right\rceil\right)^2}{8} - N \qquad (2)$$

The total number of bits after appending the zeros is given by $N_b=8(Z+N)$.

C. Compression

To achieve compression, N_b samples of the digital chaotic bit stream *C* are generated, and this signal is XORed with the genome data bit sequence mentioned above. The N_b bits of the resulting output are converted to bytes by grouping 8 successive bits together. The resultant bytes are rearranged such that it will form a 2D array, with a size of $\sqrt{\frac{N}{8}}\,\mathrm{X}\sqrt{\frac{N}{8}}$.

The 2D array is saved as an image. The Portable Network Graphics (png) format is chosen, since it is known to exhibit a near-to lossless compression. The advantage of saving the data as a png image is the effective use of Lempel-Ziv based coding techniques in the png algorithm to achieve high compression ratios [32].

In the present work, all the DNA genome samples are obtained from the genome database of the National Center for Biotechnology Information (http://www.ncbi.nlm.nih.gov) [33]. As an example, a genome sequence of size 1166631 bytes is taken as the input, and the implementation is done using the virtual instrumentation software LabVIEW [34]. The compressed png image of size 657x657 is shown in Fig. 3. It is seen that the compressed image size is 356722 bytes, thus resulting in a compression ratio of 3.2704.

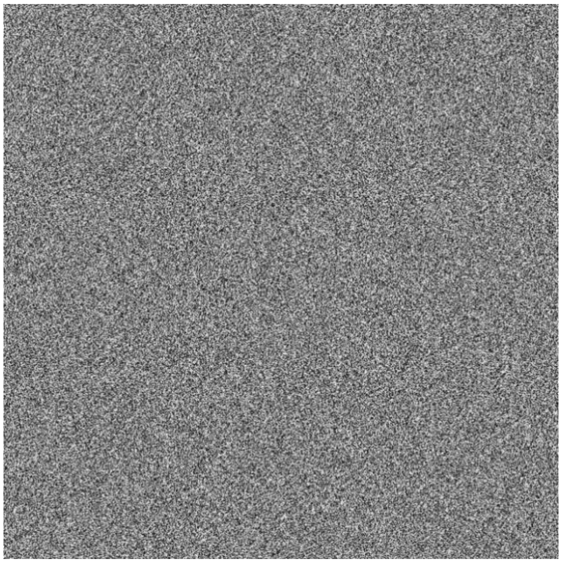

Figure 3 The compressed image containing encoded genome sequence of size 1166631 bytes

D. Decompression

The decompression essentially involves operations inverse to that of compression. Thus, the image is read as a 2D array and after a 2D to 1D reshaping as shown in Fig. 1, each byte is converted to a stream of bits. This bit stream is XORed with a digital chaotic signal generated in the receiver using 3 square waves, similar to the one generated in the transmitter. It must be noted that the time periods used for the three signals, *F1*, *F2* and *F3* should exactly match the ones used in the transmitter to achieve error-free decoding. It is observed that even a 1 percent mismatch in the time period values causes an error rate of 62 percent. After the XORing is done, the DNA bit stream is broken into groups of three, and each 3-bit value is mapped to the original genome character using Table 1. Owing to the perfectly reversible operations used and the inherent lossless nature of the png algorithm, the error rate in the decompressed genome data is found to be an absolute zero. The Compression and Decompression mentioned above have been tested for multiple Genome data samples, and the compression ratio is found to vary around the range between 2.6 and 3.5. The statistics for three such evaluations are tabulated in Table 2.

Table 2 Compression Ratios and File Sizes for Selected Genome Compression Implementations

Genome File Size (Bytes)	PNG File size (Bytes)	Image Size (Pixels)	Compression Ratio (Dimensionless)
1166631	356722	657x657	3.2704
4669566	1729831	1315x1315	2.6994
8915249	3277977	1816x1816	2.7197

Conclusion

A novel and simple technique to perform genome data compression is proposed and implemented. The backbone of the proposed technique is the use of digital chaos, formed by performing XOR operation on three square waves with mismatched frequencies. The digital chaos is XORed with the genome data bit stream after suitable data conditioning is done, and the result is stored as a grayscale image (a 2D array). PNG format is chosen owing to its inherent lossless capabilities, and it is seen that a compression ratio between 2.6 and 3.5 is observed. The decompression is performed using the inverse operations of compression and it is seen that if the three input signals exactly match that of the compression case, decompression is achieved with absolute zero error rate. The frequency values used in the generation of digital chaos thus acts as a key, with even one percent mismatch in the values giving rise to error rates of nearly 60 percent. Thus, it is seen that the use of digital chaos adds an extra layer of security. The compression technique is completely compatible and can be used in conjunction with existing and established compression techniques to boost the compression ratio even further. The simplicity of the proposed design coupled with the high compression rates achieved without using any form of reference/lookup form the significant highlights of the present work.

References

[1] Marx, V. (2013). Biology: The big challenges of big data. Nature, 498(7453), 255-260.

[2] Mayer-Schnberger, V., and Cukier, K. (2013). Big data: A revolution that will transform how we live, work, and think. Houghton Mifflin Harcourt.

[3] Howe, D., Costanzo, M., Fey, P., Gojobori, T., Hannick, L., Hide, W. and Rhee, S. Y. (2008). Big data: The future of biocuration. Nature, 455(7209), 47-50.

[4] Trelles, O., Prins, P., Snir, M., and Jansen, R. C. (2011). Big data, but are we ready?. Nature Reviews Genetics, 12(3), 224-224.

[5] Ginsburg, G. S., and Willard, H. F. (2009). Genomic and personalized medicine: foundations and applications. Translational research, 154(6), 277-287.

[6] Hamburg, M. A., and Collins, F. S. (2010). The path to personalized medicine. New England Journal of Medicine, 363(4), 301-304.

[7] Molidor, R., Sturn, A., Maurer, M., and Trajanoski, Z. (2003). New trends in bioinformatics: from genome sequence to personalized medicine. Experimental gerontology, 38(10), 1031-1036.

[8] Turchi, C., Buscemi, L., Giacchino, E., Onofri, V., Fendt, L., Parson, W., and Tagliabracci, A. (2009). Polymorphisms of mtDNA control region in Tunisian and Moroccan populations: an enrichment of forensic mtDNA databases with Northern Africa data. Forensic Science International: Genetics, 3(3), 166-172.

[9] Costa, M. D., Cherni, L., Fernandes, V., Freitas, F., El Gaaied, A. B. A., and Pereira, L. (2009). Data from complete mtDNA sequencing of Tunisian centenarians: testing haplogroup association and the golden mean to longevity. Mechanisms of ageing and development, 130(4), 222-226.

[10] Congiu, A., Anagnostou, P., Milia, N., Capocasa, M., Montinaro, F., and Destro Bisol, G. (2012). Online databases for mtDNA and Y chromosome polymorphisms in human populations. J Anthropol Sci, 90, 201-215.

[11] Burton, P. R., Hansell, A. L., Fortier, I., Manolio, T. A., Khoury, M. J., Little, J., and Elliott, P. (2009). Size matters: just how big is BIG? Quantifying realistic sample size requirements for human genome epidemiology. International journal of epidemiology, 38(1), 263-273.

[12] Grumbach, S., and Tahi, F. (1994). A new challenge for compression algorithms: genetic sequences. Information Processing and Management, 30(6), 875-886.

[13] Zhou, X., Peng, B., Li, Y. F., Chen, Y., Tang, H., and Wang, X. (2011).
To release or not to release: Evaluating information leaks in aggregate human-genome data. In Computer SecurityESORICS 2011 (pp. 607- 627). Springer Berlin Heidelberg.

[14] Church, G. M. (2005). The personal genome project. Molecular Systems Biology, 1(1).

[15] Brandon, M. C., Wallace, D. C., and Baldi, P. (2009). Data structures and compression algorithms for genomic sequence data. Bioinformatics, 25(14), 1731-1738.

[16] Sato, H., Yoshioka, T., Konagaya, A., and Toyoda, T. (2001). DNA data compression in the post genome era. Genome Informatics, 12, 512-514.

[17] Fritz, M. H. Y., Leinonen, R., Cochrane, G., and Birney, E. (2011). Efficient storage of high throughput DNA sequencing data using reference-based compression. Genome research, 21(5), 734-740.

[18] Salomon, D. (2004). Data compression: the complete reference. Springer Science and Business Media.

[19] Christley, S., Lu, Y., Li, C., and Xie, X. (2009). Human genomes as email attachments. Bioinformatics, 25(2), 274-275.

[20] Wang, C., and Zhang, D. (2011). A novel compression tool for efficient storage of genome resequencing data. Nucleic acids research, 39(7), e45-e45.

[21] Grumbach, S., and Tahi, F. (1994). A new challenge for compression algorithms: genetic sequences. Information Processing and Management, 30(6), 875-886.

[22] Grumbach, S., and Tahi, F. (1994). Compression of DNA sequences.

[23] Horowitz, P., and Hill, W. (1989). The art of electronics. Cambridge Univ. Press.

[24] Strogatz, S. H. (2014). Nonlinear dynamics and chaos: with applications to physics, biology, chemistry, and engineering. Westview press.

[25] Thompson, J. M. T., and Stewart, H. B. (2002). Nonlinear dynamics and chaos. John Wiley and Sons.

[26] Gilmore, R., and Lefranc, M. (2012). The topology of chaos: Alice in stretch and squeezeland. John Wiley and Sons.

[27] Hilborn, R. C. (2004). Sea gulls, butterflies, and grasshoppers: A brief history of the butterfly effect in nonlinear dynamics. American Journal of Physics, 72(4), 425-427.

[28] James, R. G., Burke, K., and Crutchfield, J. P. (2014). Chaos forgets and remembers: Measuring information creation, destruction, and storage. Physics Letters A, 378(30), 2124-2127.

[29] Rosenstein, M. T., Collins, J. J., and De Luca, C. J. (1993). A practical method for calculating largest Lyapunov exponents from small data sets. Physica D: Nonlinear Phenomena, 65(1), 117-134.

[30] Grassberger, P., and Procaccia, I. (1983). Estimation of the Kolmogorov entropy from a chaotic signal. Physical review A, 28(4), 2591.

[31] Zhang, W. (Ed.). (2002). Computational and statistical approaches to genomics. Kluwer Academic Publishers.

[32] Sayood, K. (2012). Introduction to data compression. Newnes.

[33] Pruitt, K. D., Tatusova, T., Brown, G. R., and Maglott, D. R. (2012). NCBI Reference Sequences (RefSeq): current status, new features and genome annotation policy. Nucleic acids research, 40(D1), D130-D135.

[34] Wells, L. K., and Travis, J. (1996). LabVIEW for everyone: graphical programming made even easier. Prentice-Hall, Inc..