GOCE DELCEV UNIVERSITY - STIP
FACULTY OF COMPUTER SCIENCE

ISSN 2545-479X print
ISSN 2545-4803 on line

BALKAN JOURNAL OF APPLIED MATHEMATICS AND INFORMATICS (BJAMI)

BALKAN JOURNAL
OF APPLIED MATHEMATICS AND INFORMATICS

YEAR 2018

VOLUME I, Number 2

GOCE DELCEV UNIVERSITY - STIP, REPUBLIC OF MACEDONIA
FACULTY OF COMPUTER SCIENCE

ISSN 2545-479X print
ISSN 2545-4803 on line

BALKAN JOURNAL OF APPLIED MATHEMATICS AND INFORMATICS

(BJAMI)

YEAR 2018 VOLUME I, Number 2

AIMS AND SCOPE:
BJAMI publishes original research articles in the areas of applied mathematics and informatics.

Topics:
1. Computer science;
2. Computer and software engineering;
3. Information technology;
4. Computer security;
5. Electrical engineering;
6. Telecommunication;
7. Mathematics and its applications;
8. Articles of interdisciplinary of computer and information sciences with education, economics, environmental, health, and engineering.

Address of the editorial office
Goce Delcev University – Stip
Faculty of philology
Krste Misirkov 10-A
PO box 201, 2000 Štip,
R. of Macedonia

BALKAN JOURNAL
OF APPLIED MATHEMATICS AND INFORMATICS (BJAMI), Vol 1

ISSN 2545-479X print
ISSN 2545-4803 on line
Vol. 1, No. 2, Year 2018

EDITORIAL BOARD

CONTENT

UDC: 004.62.052.4:575.111

FIXING GENOMIC DATA DECOMPRESSION ERRORS

Done Stojanov

Faculty of Computer Science, Goce Delcev University, Stip, Macedonia
done.stojanov@ugd.edu.mk

Abstract: The problem of genomic data compression/decompression is considered in this paper. Unlike current research that depends on reference sequence or template, a new hash-based methodology that does not depend on reference sequence is proposed. By applying the hashing formula that was proposed by Reneker and Shyu, 9 bytes of compression gain per zipped read is possible. However, the main emphasis in this paper is put on the correction of errors in genomic data decompression that happen if Renker-Shyu formula is applied.

Keywords: DNA, data, errors, correction.

1. Introduction

Current sequencing projects have revealed a variety of genomic data. Genomic data is tracked, annotated, classified and offered though public DNA databases. Since new data is sequenced and submitted on daily basis, it is very likely that DNA databases will face up storage deficiency in near future. This conclusion has motivated the scientific community to develop efficient genomic data compression algorithms.

The general approach to genomic data compression relies on reference sequence. This means that the compression is done by tracking the positions of difference between the sample that has to be compressed and the reference sequence or the template. Further compression gain is also possible if entropy coding is applied to the positions of difference, based on HUFFMAN [1] or GOLOMB [2] code. However, we must admit that this approach proposed in 2009 by Brandon [3] has limited application since it can be applied only to samples that do not significantly differ. The solution to this problem relies on using more than one template.

SNPs (single-nucleotide polymorphisms) and mutation history are provided as additional information in some works for better compression gain. Utilizing SNPs information, Christley [4] and Pavlichin [5] managed to compress the James-Watson's genome small enough to be sent by mail. Christley [4] compressed the genome down to 4 MB, while further compression down to 2.5 MB was reported by Pavlichin [5].

GRS [6] was the first known software tool that was able to compress genomic data without additional data. However, this application runs slow, it cannot be applied to all samples and the compression gain is not high at all. All these drawbacks are overcame by GreEn [7]. GreEn runs at a higher speed than GRS, has better compression rate and can be applied to any sequence.

Straightforward application of HUFFMAN code was reported by Tembe [8]. While more than 65% of compression gain was measured in that research, some researchers such as Deorowicz and Grabowski [9] proposed a limitation upon the compression pattern to be exclusively applied to samples that come from the same species.

All these methods rely on reference sequence and, without it, data compression would not be possible. Therefore, in this paper we consider the application of hash function for genomic data compression that makes the compression process independent of reference sequence.

Compressing data applying Renker and Shyu hashing formula [10] can be done without any problem, but when it comes to decompression, errors in terms of accuracy are likely to happen. We found that these errors happen in hashed **C(X)** (C-cytosine, X-random nucleotide) pairs and **C…C** cytosine tracts (uninterrupted chains of cytosine) and they happen because one of the nucleotides is hashed to the same value as the radix in the hashing formula. Appropriate solutions are proposed and that resulted in 0 errors in genomic data decompression.

2. Materials and methods

In 2005, Reneker and Shyu proposed a unique hashing formula based on equations (1) and (2) that translates genomic string over the alphabet $\sum = \{A, C, T, G\}$ into number. Equation (1) translates nucleotides into positive numbers, which are afterwards used to compute the hash of random genomic read R according to equation (2).

$$f(A) \to 1, f(T) \to 2, f(G) \to 3, f(C) \to 4 \quad \textbf{(1)}$$ (A: adenine, T: thymine, G: guanine, C: cytosine)

$$H(R) = f(R: a_0 a_1 \dots a_{n-2} a_{n-1}) = \sum_{i=0}^{n-1} f(a_i) \times 4^i = f(a_0) \times 4^0 + f(a_1) \times 4^1 + \cdots + f(a_{n-2}) \times 4^{n-2} + f(a_{n-1}) \times 4^{n-1} \quad \textbf{(2)}$$

This concept is suitable for genomic data compression and here we are going to explain why.

Storing DNA sequence as an array of characters requires 1 B (byte) per nucleotide (character). If the sequence contained n nucleotides, n B (bytes) would be required. In terms of the request for storage, the previous is not a problem when it comes to short reads or partial mRNA, but it may be a problem when we deal with human chromosome or even the entire human genome of 3.000.000.000 bp (base pairs).

Applying equations (1) and (2) we can compress short genomic reads of 15 base pairs into a single integer of 4 B (bytes) that results in 11 bytes of compression gain per read. Rather than random, the length of the reads was chosen upon the hash of the read of 15 cytosines which equals 1.431.655.764, which is the maximum that can be stored into a single variable of inter type without overflow. According to equation (1), all other nucleotide translations (f(A), f(T), f(G)) are less than 4 (f(C)) what grants that any other read of 15 base pairs can be also zipped into a single integer of 4 bytes without overflow.

After read's compression, we must know how to decompress/decrypt a part of the read or even the entire read (decompression/decryption means transforming the hash into a string). To do that, we have to know how to decompress nucleotide a_k given the hash of the read H(R) and the position of decryption k as input.

Perhaps one can say that this can be easily done by applying equation (3), but errors in decryption are likely to happen if only this equation is applied and, as we said before, these errors happen because cytosine (C) is hashed to the same value as the radix in equation (2) which equals 4.

$$\mathbf{f(a_k) mod 4} = \left(\mathbf{H(R)}/_{\mathbf{4^k}}\right) \mathbf{mod 4} \quad \textbf{(3)}$$

To prove equation (3) we can write equation (2) into equation (4) in terms of 4^k as a common factor. From equation (4) we get that $H(R)/_{4^k}$ equals equation (5). If we rewrite equation (5) in terms of 4 as a common factor, we get equation (6), wherefrom we get equation (7) that equals equation (3).

$$H(R) = f(a_0) \times 4^0 + \cdots + f(a_{k-1}) \times 4^{k-1} + 4^k \times \left(f(a_k) + f(a_{k+1}) \times 4^1 + \cdots + f(a_{n-1}) \times 4^{n-k-1}\right) \quad \textbf{(4)}$$

$$H(R)/_{4^k} = f(a_k) + f(a_{k+1}) \times 4^1 + \cdots + f(a_{n-1}) \times 4^{n-k-1} \quad \textbf{(5)}$$

$$H(R)/_{4^k} = f(a_k) + 4 \times (f(a_{k+1}) + f(a_{k+2}) \times 4 \dots + f(a_{n-1}) \times 4^{n-k-2}) \quad \textbf{(6)}$$

$$\left(H(R)/_{4^k}\right) \text{mod} 4 = f(a_k) \text{mod} 4 \quad \textbf{(7)}$$

So, we get that if $\left(H(R)/_{4^k}\right) \text{mod} 4 = 1$ then $a_k = 'A'$(Adenine), if $\left(H(R)/_{4^k}\right) \text{mod} 4 = 2$ then $a_k = 'T'$(Thymine), if $\left(H(R)/_{4^k}\right) \text{mod} 4 = 3$ then $a_k = 'G'$(Guanine) and if $\left(H(R)/_{4^k}\right) \text{mod} 4 = 0$ then $a_k = 'C'$(Cytosine).

The first time when the accuracy of decryption is undermined is when a_k is preceded by Cytosine or $a_{k-1} = 'C'$. In such a case, instead of equation (3), equation (8) must be applied to decrypt correctly a_k.

$$\mathbf{f(a_k) mod 4} = \left(\mathbf{H(R)}/_{\mathbf{4^k}} - \mathbf{1}\right) \mathbf{mod 4} \quad \mathbf{(8)}$$

To prove equation (8) we shall consider once again equation (2) under $a_{k-1} = 'C'$, $f(a_{k-1}) = 4$. In such a case, equation (2) transforms into equations (9) and (10).

$$H(R) = f(a_0) \times 4^0 + \cdots + f(a_{k-2}) \times 4^{k-2} + 4 \times 4^{k-1} + f(a_k) \times 4^k + f(a_{k+1}) \times 4^{k+1} + \cdots + f(a_{n-1}) \times 4^{n-1}$$

(9)

$$H(R) = f(a_0) \times 4^0 + \cdots + f(a_{k-2}) \times 4^{k-2} + 4^k + f(a_k) \times 4^k + f(a_{k+1}) \times 4^{k+1} + \cdots + f(a_{n-1}) \times 4^{n-1} \quad \mathbf{(10)}$$

If we take 4^k out of parenthesis in equation (10), this equation becomes equation (11).

$$H(R) = f(a_0) \times 4^0 + \cdots + f(a_{k-2}) \times 4^{k-2} + 4^k \times (1 + f(a_k) + f(a_{k+1}) \times 4 + \cdots + f(a_{n-1}) \times 4^{n-k-1}) \quad \mathbf{(11)}$$

According to equation (11) $H(R)/_{4^k}$ equals $1 + f(a_k) + f(a_{k+1}) \times 4 + \cdots + f(a_{n-1}) \times 4^{n-k-1}$. The further transformation of $H(R)/_{4^k}$ in terms of 4 as a common factor results in equations (12) and (13).

$$H(R)/_{4^k} = 1 + f(a_k) + 4 \times (f(a_{k+1}) + \cdots + f(a_{n-1}) \times 4^{n-k-2}) \quad \mathbf{(12)}$$

$$H(R)/_{4^k} - 1 = f(a_k) + 4 \times (f(a_{k+1}) + \cdots + f(a_{n-1}) \times 4^{n-k-2}) \quad \mathbf{(13)}$$

From equation (13) we get that $\left(H(R)/_{4^k} - 1\right) \text{mod} 4 = f(a_k) \text{mod} 4$.

Since we cannot know in advance if a_k is preceded by cytosine or not, an additional test (14) must be conducted. Given that test (14) is true, equation (8) must be applied to decrypt correctly a_k and if false equation (3) is applied instead. In fact, test (14) checks if a_k is preceded by cytosine or not upon the hash of the read **H(R)** and the position of decryption **k** as input.

$$IF(H(R)\ mod\ 4^{k-1} = H(R)\ mod\ 4^k)\ then\ a_{k-1} = '\ C': Equation\ (8)\ is\ applied \quad \mathbf{(14)}$$

Test (14) can be also proved. For that purpose, we consider once again equation (9), which is a special case of equation (2) given that $a_{k-1} = 'C'$. If we take 4^{k-1}out of parenthesis in equation (9), we get equation (15).

$$H(R) = f(a_0) \times 4^0 + \cdots + f(a_{k-2}) \times 4^{k-2} + 4^{k-1} \times (4 + f(a_k) \times 4 + f(a_{k+1}) \times 4^2 + \cdots + f(a_{n-1}) \times 4^{n-k}) \quad \mathbf{(15)}$$

According to equation (15) $H(R) \text{mod}\ 4^{k-1} = f(a_0) \times 4^0 + \cdots + f(a_{k-2}) \times 4^{k-2}$ and if we look back at equation (11), we get that $H(R) \text{mod}\ 4^k$ also equals $f(a_0) \times 4^0 + \cdots + f(a_{k-2}) \times 4^{k-2}$ and this happens only if a_k is preceded by Cytosine.

Until now, we know that if test (14) returns true a_k is preceded by cytosine and equation (8) must be applied to decrypt correctly a_k, on the other hand equation (3) has to be applied. However, there is one more case that is critical when test (14) returns false but instead of equation (3), equation (8) must be applied. This situation happens if cytosine tract (**C…C**) before a_k.

To discuss this situation, we consider the case when a_k is preceded by cytosine being also preceded by cytosine(... $\mathbf{CCa_k}$ ...). In this situation, H(R) equal equations (15), (16) and (17). Equations (16) and

(17) equal equation (15), but they are written in terms of 4^{k-1} and 4^{k} as common factors. From equations (16) and (17) we get that condition (14) is not satisfied because $H(R) \bmod 4^{k-1}$ and $H(R) \bmod 4^{k}$ mutually differ, i.e. $H(R) \bmod 4^{k-1} = \cdots + f(a_{k-3}) \times 4^{k-3}$ and $H(R) \bmod 4^{k} = \cdots + f(a_{k-3}) \times 4^{k-3} + 4^{k-1}$ but since a_k is preceded by Cytosine, equation (8) must be applied to decrypt correctly a_k.

$$H(R) = \cdots + f(a_{k-3}) \times 4^{k-3} + f(C) \times 4^{k-2} + f(C) \times 4^{k-1} + f(a_k) \times 4^{k} + \cdots$$

$$= \cdots + f(a_{k-3}) \times 4^{k-3} + 4 \times 4^{k-2} + 4 \times 4^{k-1} + f(a_k) \times 4^{k} + \cdots$$

$$= \cdots + f(a_{k-3}) \times 4^{k-3} + 4^{k-1} + 4^{k} + f(a_k) \times 4^{k} + \cdots \quad \textbf{(15)}$$

$$H(R) = \cdots + f(a_{k-3}) \times 4^{k-3} + 4^{k-1} \times (1 + 4 + f(a_k) \times 4 + \cdots) \quad \textbf{(16)}$$

$$H(R) = \cdots + f(a_{k-3}) \times 4^{k-3} + 4^{k-1} + 4^{k} \times (1 + f(a_k) + \cdots) \quad \textbf{(17)}$$

If test (14) returns false, we conduct an additional test (18), for which if it is true, equation (8) must be applied. This test (18) we derived from equations (16) and (17) and it covers the case when a_k is preceded by cytsoine tract (**C…C**). From equations (16) and (17) we get that $H(R) \bmod 4^{k-1} = \cdots + f(a_{k-3}) \times 4^{k-3}$ and $H(R) \bmod 4^{k} = \cdots + f(a_{k-3}) \times 4^{k-3} + 4^{k-1}$, i.e. $H(R) \bmod 4^{k} = H(R) \bmod 4^{k-1} + 4^{k-1}$.

$$IF(H(R) mod\ 4^{k} = H(R) mod\ 4^{k-1} + 4^{k-1}) : Equation\ (8)\ is\ applied \quad \textbf{(18)}$$

This discussion was summarized into algorithm that is presented below.

output: *char* $\boldsymbol{a_k}$ algorithm: ***Nucleotide Decompression*** (**input**: *int* $\boldsymbol{H(R)}$, $\boldsymbol{k}$)

{

if $(H(R) mod\ 4^{k-1} == H(R)\ mod\ 4^{k})$

$f(a_k) \leftarrow \left({}^{H(R)}/_{4^k} - 1\right) mod 4$

else{

if $(H(R) mod\ 4^{k} == H(R) mod\ 4^{k-1} + 4^{k-1})$

$f(a_k) \leftarrow \left({}^{H(R)}/_{4^k} - 1\right) mod 4$

else

$f(a_k) \leftarrow \left({}^{H(R)}/_{4^k}\right) mod 4$

}

if$(f(a_k) = =1)$

a_k = 'A'

else if $(f(a_k) = =2)$

a_k = 'T'

else if $(f(a_k) = =3)$

a_k = 'G'

else

a_k = 'C'

return a_k

}

3. Results and discussion

A short reads compression/decompression program was written in C#, Figure 1. This program accepts short reads as input (upper text control Figure 1), computes its hash (middle text control Figure 1) and upon the hash decrypts the read, either by applying only equation (3) (middle text control Figure 1) or/and by applying tests (14) and (18) and the additional correction equation (8) (lower text control Figure 1). Short reads decompression is made by the previous algorithm for all positions in the reads.

We made five tests on 11-b.p (base pairs) reads on Acer Aspire 5570Z computer with Genuine Intel T2080 @ 1.73 GHz and 2 GB RAM, Table 1. Four of the reads contain critical CT-tandem repeats (record 1-record 4 Table 1) while the last sample contained Cytosine tract before Thymine, Table 1.

Applying only equation (3) Thymine nucleotides in CT-tandem repeats were not accurately decrypted (Table 1, red-labeled nucleotides, record 1- record 4). In addition, five base pairs were not accurately decrypted near/in the Cytosine tract (Table 1, red-labeled nucleotides record 5). The total number of errors equaled 15, Figure 2. Applying proposed tests (14) and (18) and the additional correction equation (8), all these 15 errors were solved that resulted in 0 errors in genomic data decompression (Table 1, blue-labeled nucleotides).

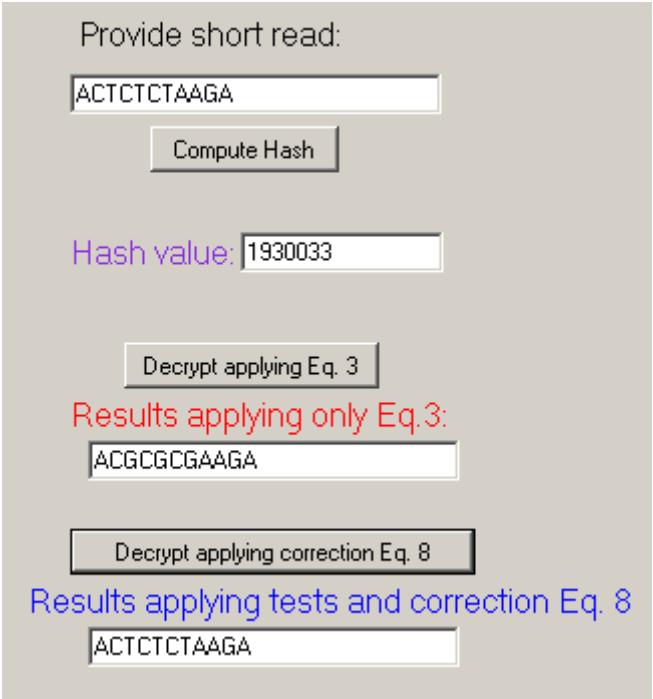

Figure 1. Screenshot of the application

Table 1: Decryption results

Input	Hash Value	Applying eq. 3	Applying tests (14) and (18) and correction eq. (8)	Errors
ACTGTATAAGA	1926897	ACGGTATAAGA	ACTGTATAAGA	1
ACTCTATAAGA	1926961	ACGCGATAAGA	ACTCTATAAGA	2
ACTCTCTAAGA	1930033	ACGCGCGAAGA	ACTCTCTAAGA	3
ACTCTCTCTGA	2044721	ACGCGCGCGGA	ACTCTCTCTGA	4
ACCCCCTAAGA	1930577	ACAAAAGAAGA	ACCCCCTAAGA	5

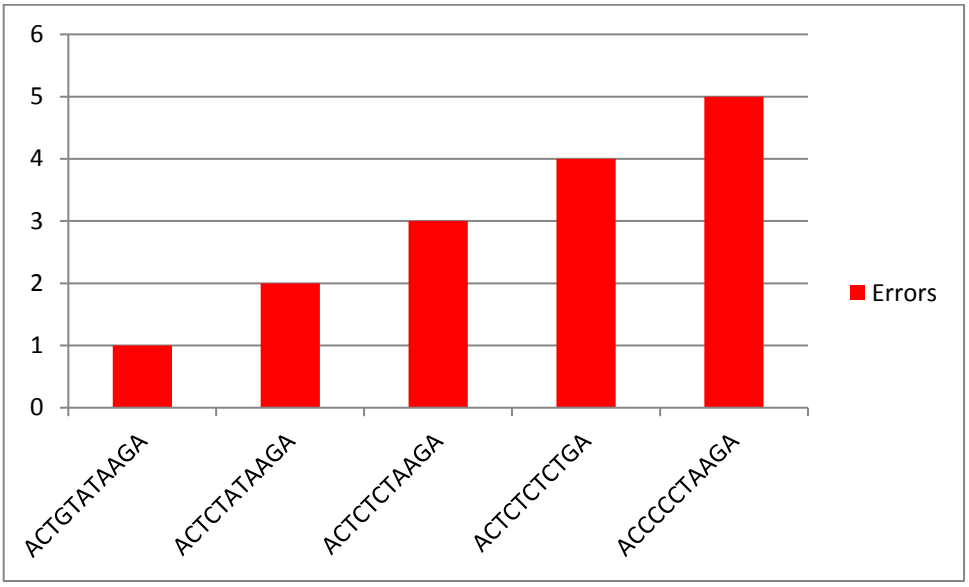

Figure 2. Number of base pairs errors per short read

Concluding remarks

We considered the problem of genomic data compression/decompression by applying the hashing formula that was proposed by Reneker and Shyu. We showed that using this formula short reads of 15 base pairs can be zipped into a single integer of 4 bytes that results in 11 bytes or around 73 % of compression gain per read. This compression is possible without having to use reference sequence, which is a common practice in all current research of genomic data compression. However, we should be very careful when applying this formula for data decompression since errors are likely to happen due to the fact that one of the nucleotides is mapped to the same value as the radix. We noticed this case and we proposed a solution that works based on two tests and one additional equation that eliminated the possibility of errors in genomic data decompression.

References

1. Huffman, D.A. (1952). *"A method for the construction of minimum-redundancy codes"*: IEEE, In: Proceedings of the IRE. 1098–1101.
2. Golomb, S. (1966). *"Run-length encodings"*: IEEE Information Theory Society, IEEE transactions on information theory. 12(3): 399–401.
3. Brandon, M.C., Wallace, D.C., Baldi, P. (2009). *"Data structures and compression algorithms for genomic sequence data"*: Oxford University Press, Bioinformatics. 25(14): 1731–1738.
4. Christley, S., Lu, Y., Li, C., Xie, X. (2008). *"Human genomes as email attachments"*: Oxford University Press, Bioinformatics. 25(2): 274–275.
5. Pavlichin, D.S., Weissman, T., Yona, G. (2013). *"The human genome contracts again"*: Oxford University Press, Bioinformatics. 29(17): 2199–2202.
6. Wang, C., Zhang, D. (2011). *"A novel compression tool for efficient storage of genome resequencing data"*: Oxford University Press, Nucleic acids research. 39(7): e45–e45.
7. Pinho, A.J., Pratas, D., Garcia, S.P. (2011). *"GReEn: a tool for efficient compression of genome resequencing data"*: Oxford University Press, Nucleic acids research. 40(4): e27–e27.
8. Tembe, W., Lowey, J., Suh, E. (2010). *"G-SQZ: Compact encoding of genomic sequence and quality data"*: Oxford University Press, Bioinformatics. 26(17): 2192–2194.
9. Deorowicz, S., Grabowski, S. (2011). *"Robust relative compression of genomes with random access"*: Oxford University Press, Bioinformatics. 27(21): 2979–2986.
10. Reneker, J., Shyu, C.R. (2005). *"Refined repetitive sequence searches utilizing a fast hash function and cross species information retrievals"*: BioMed Central, BMC bioinformatics. 6(1): 111.