DOI 10.29042/2017-1498-1502

*Helix Vol. 8: 1498-1502*

# DNA Sequence Compression using ERGC and NRGC for Better Compression

**[*1] Raju Bhukya, [2] Hanuman Ram, [3] Nitish Kumar, [4] Avishek Kumar Roy**
Department of Computer Science and Engineering, NIT Warangal
**Email: drrajunitw@gmail.com*

**Received: 20th April 2017, Accepted: 28th April 2017, Published: 1st May 2017**

**Abstract**
Current improvements in computing technology makes it possible and very easy to build genomic sequences from the billions of reads within a minimal cost and time. In this article, we are proposing an algorithm which is analyzing two compression algorithms Efficient Referential Genome Compression algorithm and Novel Referential Genomic Compression algorithm. Both are effective genomic compression algorithms. The proposed algorithm is analyzing their behaviour for different types of DNA sequences as reference and target genome to produce a result which can predict in which circumstances one algorithm outper-forms other and vice versa. The proposed algorithm analyses the inherent structure of target and reference genomes and predicts the better compression algorithm. Then it compresses the target sequence using predicted algorithm. Proposed algorithm's experimental results prove that it works better than all existing algorithms in this field.

## Introduction

Huge sized data leads to many problems during data transfer over Internet. If the data size is huge it creates a lot of burden over the Internet during transmission. It also occupies a huge space in the local systems. One major side effect of huge data is network congestion. Huge transmission cost is another factor which happens due to large size of data. To remediate these problems, data compression is a very essential tool. For the compression of general purpose data, various tools are available in the market. These general-purpose compression algorithms run very effectively on general purpose data which we use in day to day life. But they are not suitable to compress special data such as DNA sequences which has their own specific structures. Due to this, such type of general purpose compression algorithm is a poor choice to compress DNA sequences. To compress the data such as DNA sequences, some compression algorithms which makes use of the specific structures of DNA sequences to compress them more efficiently. The algorithms ERGC and NRGC achieves compression ratios which is better than the current best algorithms. Compression ratio is the ratio of the uncompressed data size to the compressed data size.

## Related Work

Next generation sequencing technique is a revolutionary step in human genome sequencing. Next generation sequencing is a massively parallel deep sequencing technique which was initially developed in 1990. Next generation techniques can process millions of sequences in a parallel process to generate gigabytes of data. It can generate millions or billions of base pairs in a very reasonable time. These billions of base pairs occupies huge space. The average data size generated by Next generation technique remains in gigabytes. Due to the generation of large amount of DNA sequences, it was possible to map human genomes very effectively. This has a major role in the completion of Human Genome Project [32]

But because of the generation of large genomic data, it is become very hard to transmit it over the Internet. Transmission of this much huge amount of data is very costly and leads to Network congestion. Also, to store these data, large local storage is required. To remediate all these problems, a very effective compression algorithm became a need of the hour. Various DNA compression algorithms have been proposed to ful ll that need. We will now brie y discuss some of the proposed genomic compression algorithm which uses a reference from the same species.

The basic idea of referential genome compression is, we first choose a reference sequence R which can be chosen randomly or algorithmically. All the other sequences are then com-pressed with respect to R. The target sequence T is first aligned to reference R and the mismatches between two are identified and encoded.

The most recent algorithm ERGC [17] Divides both the target and the reference sequences into components of identical size and finds one-to-one maps of similar regions from every part. It then outputs identical maps in conjunction with numerous areas of the goal series. Delta encoding and PPMD [28] lossless compression algorithm is used to compress the variations between the reference and the target genomes. If the variations between the reference and the target are small, it outperforms all the best-known algorithms. But its performance degrades when the variations are high. GRS [20] is a reference-primarily based genome compression device exclusively dependent on the Unix application diff. GDC [30] is a LZ77-fashion

 

(Ziv and Lempel, 1977) [21] compression algorithm carefully related to RLZopt (Shanika etal., 2011) [18] in which GDC[30] performs a non-greedy parsing of the target into the reference by means of hashing. On the opposite, RLZopt [18] uses a suffix array.

GReEn[16] is also a reference-primarily based genome compression device. It encodes the goal series by use of an arithmetic encoder. GReEn [16] assumes that the sequences are already aligned and may be outstanding through SNPs [33]

The simple functioning of iDoComp [13][29] may be summarized in three fundamental steps: (i) mapping era: in this stage, the target genome is expressed in terms of the reference genome. It makes use of suffix arrays to parse the target into the reference; (ii) post-processing: the submit-processing seems for consecutive suits that may be merged collectively and converted into an approximate in shape and (iii) entropy encoding: entropy encoder compresses the mapping and generates the compressed le.

Subrata Saha and Sanguthevar Rajasekaran (Department of Computer Science and En-gineering, University of Connecticut) have proposed two algorithms for efficient genomic data compression, namely ERGC [17] and NRGC[22]. Other compression algorithms such as GDC[20] , iDoComp[29] etc. are also present in the field but ERGC and NRGC clearly outsmart them in terms of data compression ratio. NRGC is mostly introduced as an improvement over ERGC by the authors. For some cases, it works better than ERGC but not for all cases. NRGC uses preprocessing for better alignment of reference and target genome before applying the compression algorithm. The complexity for the preprocessing stage in NRGC is O (n) where n is the size of reference and/or target genome. For some datasets, this preprocessing actually reduces the overall cost of compression and in those cases NRGC [22] works better than ERGC [17]. But for few cases, this pre-processing increases the overall cost and hence NRGC [22] works poorly as compared to ERGC [17]. Authors did not significantly point out for which type of datasets one algorithm will outperform another algorithm. So, it remains to find out the nature of the datasets for which ERGC will outperform NRGC [22] and for other datasets for which NRGC [22] will outperform ERGC [17]. In this project, we are trying to tackle this problem and are trying to figure out the types of datasets (reference + target pair) for which one algorithm is better than another algorithm.

**Proposed work:**

**Comparison Algorithm for ERGC and NRGC**

Our algorithm is as follow. Let T is a target sequence to be compressed and R is the reference sequence. Initially ERGC breaks the complete reference and target genomes into identical-sized parts and processes every pair of these elements sequentially. If the components in R and T are r1; r2; . . .; rq and t1; t2; . . .; tq, respectively, then r1 and t1 are processed subsequent and so forth.

Let (r',t') be the pair processed at some point inside the set of rules (wherein r0 comes from the reference genomic series R and t0 comes from the target genomic series T). To locate the similarities among r0 and t0, we want to align t0 onto r0. Similar areas among the sequences may be located globally aligning t0 onto r0 the use of greedy alignment set of rules to discover similar areas among sequences with immoderate self-belief (it's applicable whilst the sequences are similar, e.g. Genomic sequences of the same species). Greedy algorithm will be described next.

Greedy alignment set of rules is primarily based on hashing. At first, the algorithm generates all the k-mers from ri and hashes them right into a hash table H (for some appropriate values of k). k-mers are being generated from ti one at a time and we hash them into H until this type of k-mers collides with an access in H. If k-mers collision do not occur, then another set of k0-mers are being generated by algorithm (where k0 < k) from ri and they are hashed into a hash table Hi. It then generates k0-mers from ti separately and hashes them to Hi until one of these k0-mers collides with an access in Hi then it counts the k-mers as mapped k-mers. So, by this way, mapped k-mers is counted in target sequence segment ti then If number of mapped k-mers in the segment ti is greater than 500, then segment ti is counted as mapped segment. By this similar way mapped segment from target sequence t to reference sequence can be counted. To compare ERGC and NRGC, the count of mapped segment is compared against the threshold Segment count which is calculated by following eqn

thresholdSegmentCount ←(length(r)*4)/1000000;

Algorithm 1 Comparison Algorithm for ERGC and NRGC

```
1:  Input: k-mers size K, target sequence T, Reference sequence R,
2:  Output: Better approach for compression between ERGC and NRGC
3:  procedure Comparison Algorithm
4:      (r1; r2; :::; rq) ← R;
5:      (t1; t2; :::::; tq) ← T ;
6:      threshouldSegmentCount ← (length(r) *4)/1000000;
7:      mappedSegmentCoun←0;
8:      for i ← 1; q do
9:          Divide ri into K-mers of size K;
10:         Insert ri k-mers into table H;
11:         count  0;
12:         for j  1; NumberOfKMers(ti) do
13:             if hj has collision in Hash Table H then then
14:                 count  count + 1;
15:             else
16:                 continue;
17:         if count > 500 then
18:             mappedSegmentCount  mappedSegmentCount + 1;
19:     if mappedSegmentCount >= threshouldSegmentCount then
20:         return ERGC
21:     else
22:         return N RGC
```

  ISSN 2319– 5592 (Online)

If the mapped segment count from target sequence to reference sequence is greater than threshold segment count then target sequence t and reference sequence are more horizontally aligned (i.e. More similar) so ERGC algorithm will be used to compress the target sequence t otherwise NRGC algorithm will be used to compress the target sequence because in former case both sequence (target sequence and reference sequence) have less horizontal alignment.

**Experimental Results**

Several experiments were done by us using original DNA sequences. We have used hg19, hg18 and these are retrieved from the UCSC Genome Browser. Korean Genome [1][11] KOREF 20090131 (KOR131 for short) and KOREF 20090224 (KOR224 for short) are retrieved from koreangenome.org, and the genome of a Han Chinese known as YH [Levy et al. (2008)][11] is retrieved from yh.genomics.org.cn. From these data sets, each data set act as reference to evaluate the accuracy of our algorithm. We have taken chromosome 1 for comparison purpose. The implementation accepts only DNA sequence input format FASTA. Fasta layout is a textual content based layout for representing either nucleotide sequences or peptide sequences wherein nucleotides or amino acids are represented the usage of single letter [22] code. The format also permits for series names and remarks to precede the se-quences.

The format originates from FASTA software package deal, but has a preferred in discipline of bioinformatics. The system requirement is at least 4GB of RAM and processor Intel i5 2.3GHz or above. Java version 8 is accepted currently. Next, we present performance evaluation details of our proposed comparison algorithm with respect to both compression algorithm. We have compared ERGC with NRGC using standard benchmark 16 datasets. Given a reference sequence, our algorithm analyzes the target sequence by exploiting the inherent structure of the reference sequence. We use the target and reference pairs of sequences. Our algorithm is designed in such way that it can work with all types of sequences. Now consider the dataset hg19 as reference sequence and hg18, KO131, Ko224 and Yh are as target sequences. The compressed size of hg18 is 3.5MB by ERGC but NRGC compresses it to 3.9MB. Out of two algorithms our algorithm predicts NRGC as better algorithm but it is not the case because both the algorithms work better on this pair with very less difference in compressed sizes.

Now KO131 is compressed using ERGC. It compresses KO131 to 223MB from 251MB but NRGC compressed it to 12.49MB. In this ERGC is working worst while NRGC is giving us better results. The proposed algorithm utilizes the similarity between the reference genomic sequence and target genomic sequence.

In the first case, the proposed algorithm outputs NRGC as better algorithm while ERGC was working better. This happened because the resultant compressed size of target sequence was almost like each other. In that case, proposed algorithm fails to determine better compression algorithm. In the second case, the output generated by both the algorithm is very different. ERGC compressed the target sequence far worse than NRGC.

The pro-posed algorithm outputs NRGC as better compression algorithm. In this case, the proposed algorithm outputs the better working algorithm very efficiently Similarly, with KO224 as target sequence, the performance of ERGC is far worse than the performance of NRGC. In this case also, proposed algorithm correctly predicts NRGC as better compression algorithm. With YH as target sequence, the compressed le size generated by ERGC is 3.3MB while NRGC is 11.4 MB. Since the sizes are comparable to each other, the proposed algorithm outputs NRGC as better algorithm. This is not correct. Similar pattern can be found in other sequence target pairs. It shows that, if the resultant le size is comparable in size, proposed algorithm fails to predict the correct result. But it hardly matters, because if both of the algorithms are performing similarly, then we can choose any one of them to compress the datasets.

We have used 16 datasets to evaluate performance of proposed algorithm. In those 16 datasets, algorithm predicts NRGC as better compression algorithm in 8 cases, while ERGC have better performance in other 8 cases. In 8 data pairs, in which NRGC is predicted as better compression algorithm, 5 cases are predicted correctly by proposed algorithm while 3 cases are predicted incorrectly. These 3 cases have similar output by NRGC and ERGC. In remaining 8 cases, in which ERGC is predicted as better performing, all the 8 cases are predicted correctly, that is proposed algorithm has correctly predicted that ERGC will work better in these cases. Above analysis shows that, for the cases in which both ERGC and NRGC are performing similarly, proposed algorithm outputs the incorrect result but if the performance difference is very high, algorithm predicts correct output. Proposed algorithm has predicted 13 out of 16 results correctly and has a success percent of 81.25 on the given dataset.

 

**Result Interpretation**

| Reference Sequence | Reference Sequence Actual Size(MB) | Target Sequence | Target Sequence Actual Size(MB) | Compressed Sequence Size By ERGC(MB) | Compressed Sequence Size By NRGC(MB) | Comparison Algorithm Output |
|---|---|---|---|---|---|---|
| hg19 | 254 | hg18 | 252 | 3.5 | 3.9 | NRGC |
| hg19 | 254 | KO131 | 251 | 223 | 12.49 | NRGC |
| hg19 | 254 | KO224 | 251 | 225.3 | 12.4 | NRGC |
| hg19 | 254 | YH | 251 | 3.3 | 11.4 | NRGC |
| hg18 | 252 | hg19 | 254 | 222 | 3.9 | NRGC |
| hg18 | 252 | KO131 | 251 | 4.3 | 11.5 | ERGC |
| hg18 | 252 | KO224 | 251 | 3.8 | 10.80 | ERGC |
| hg18 | 252 | YH | 251 | 2.45 | 10.26 | ERGC |
| KO224 | 251 | hg19 | 254 | 225 | 9.70 | NRGC |
| KO224 | 251 | hg18 | 252 | 4.45 | 9.92 | ERGC |
| KO224 | 251 | KO131 | 251 | 1.76 | 1.99 | ERGC |
| KO224 | 251 | YH | 251 | 2.82 | 6.18 | NRGC |
| YH | 251 | hg19 | 254 | 225 | 9.02 | NRGC |
| YH | 251 | hg18 | 252 | 5.03 | 10.41 | ERGC |
| YH | 251 | KO131 | 251 | 3.71 | 7.32 | ERGC |
| YH | 251 | KO224 | 251 | 4.16 | 8.68 | ERGC |

All Sequences Size to be compressed is approx. 4028 MB. Compressed Sequences Size by ERGC is 1161 MB. Compressed Sequences Size by NRGC is 140 MB. Compressed Sequences Size by adding Comparison Algorithm is 96 MB. Compressed Sequences Size is expected 88 MB. Accuracy can be de ned for our algorithm as ratio of compressed sequences size by Manual with compressed sequence size by our algorithm. Accuracy of the algorithm in tested sequences is 91.27. Above analysis shows that our algorithm, if run along with ERGC and NRGC, produces better compression ratio in most of the cases. Time complexity of our comparison algorithm is O(nlogn) which is less than or equal to the overall time complexity of NRGC(O(n*n)) and ERGC(O(nlogn)).

**Conclusion**

Data compression is a most typical problem in biology especially for NGS data. In this article, we have proposed a comparison algorithm to compare ERGC and NRGC and to use reference based compression efficiently and effectively. It is evident from the simulation results that the proposed algorithm is indeed an effective compressor compared with the state-of-the-art algorithms existing in the current literature. Our algorithm works better in terms of time complexity than the already existing compression algorithms.

**References**

[1] Ahn,S.-M. et al. (2009) The first Korean genome sequence and analysis: full genome sequencing for a socio-ethnic group. Genome Res., 19, 1622{1629.

[2] Altschul,S.F. et al. (2004) Basic local alignment search tool. J. Mol. Biol., 215, 403{410.

[3] Brandon,M.C. et al. (2009) Data structures and compression algorithms for genomic sequence data. Bioinformatics, 25, 1731{1738.

[4] Cao,M.D. et al. (2007) A simple statistical algorithm for biological sequence compression. Proceedings of the 2007 IEEE Data Compression Conference (DCC 07), pp. 43{52.

[5] Christley, S. et al. (2009) Human genomes as email attachments. Bioinformatics, 25, 274{275.

[6] Deorowicz,S. et al. (2013) Genome compression: a novel approach for large collections. Bioinformatics, 29, 1{7.

[7] Deorowicz,S. and Grabowski,S. (2011) Robust relative compression of genomes with random access. Bioinformatics, 27, 2979{2986.

[8] Golomb,S.W. (1966) Run-length encodings. IEEE Trans. Inform. Theor., 12, 399{401.

[9] Hu man,D. (1952) A method for the construction of minimum-redundancy codes. Proc. Inst. Radio Eng., 1098{1101.

[10] Kurtz,S. et al. (2004) Versatile and open software for comparing large genomes. Genome Biol., 5, R12.

[11] Levy, S. et al. (2008) The diploid genome sequence of an Asian individual. Nature, 456, 60{66.

[12] Mo at, A. (1990) Implementing the PPM data compression scheme. IEEE Trans. Commun., 38, 1917{1921.

]13] Ochoa,I. et al. (2014) iDoComp: a compression scheme for assembled genomes. Bioinformatics, 31, 626{633.

[14] Pavlichin,D. et al. (2013) The human genome contracts again. Bioinformatics, 29, 2199{2202.

[15] Peter,E. (1975) Universal codeword sets and representations of the integers. IEEE Trans. Inform. Theor., 21, 194{203.

[16] Pinho,A.J. et al. (2012) GReEn: a tool for efficient compression of genome resequencing data. Nucleic Acids Res., 40, e27.

[17] Saha,S. and Rajasekaran,R. (2015) ERGC: an efficient referential genome compression algorithm. Bioinformatics, 31, 3468{3475.

[18] Shanika,K. et al. (2011) Optimized relative lempel-ziv compression of genomes. 34thAus-tralasian Computer Science Conference, 113, pp. 91{98.

[19] Stephens,Z.D. et al. (2015) Big data: astronomical or genomical? PLoS Biol., 13, e1002195.

[20] Wang,C. and Zhang,D. (2011) A novel compression tool for efficient storage of genome resequencing data. Nucleic Acids Res., 39, E45{U74.

[21] Ziv, J. and Lempel, A. (1977) A universal algorithm for sequential data compression. IEEE Trans Inform. Theor., 23, 337{343.

 

[22] Subrata Saha, Sanguthevar Rajasekaran Bioinformatics NRGC: a novel referential genome compression algorithm (01 August 2016)

Comment on: `ERGC: an efficient referential genome compression algorithm' Sebastian Deorowicz Szymon Grabowski Idoia Ochoa Mikel Hernaez Tsachy Weissman Bioinformatics(2015) 32 (7): 1115-1117.

[24] Subrata Saha Sanguthevar Rajasekaran Bioinformatics (2015) Authors' response to `Comment on: ERGC: An efficient Referential Genome Compression Algorithm 32 (7): 1118-1119.

[25] Fritz,M.H.-Y. et al. (2011) Efficient storage of high throughput DNA sequencing data using reference-based compression. Genome Res., 21, 734{740.

[26] Sara A.Shehab, Arabi Keshk, Hany Mahgoub (2012) Fast Dynamic Algorithm for Sequence Alignment based on Bioinformatics

[27] Chandrasai Potladurthi, Shyam Perugu, Durga Bhavani S (2014) Deciphering the sequence alignment by Needleman-Wunsch algorithm on to reduce computational time via high performance computing

[28] A. Mo at, Dept. of Computer Sci., Melbourne Univ., Parkville, Vic., Australia, Imple-menting the PPM data compression scheme : 1917 - 1921,Nov 1990

[29] Idoia Ochoa,Mikel Hernaez, Tsachy Weissman(2014) iDoComp: a compression scheme for assembled genomes

[30] Sebastian Deorowicz, Agnieszka Danek, Marcin Niemiec (2015), GDC 2: Compression of large collections of genomes

[31] Tom Madden, The BLAST Sequence Analysis Tool

[32] Trevor Dale, (1999) Human Genome Project Essay

[33] Davis MW, Hammarlund M (2006), Single-nucleotide polymorphism mapping

  ISSN 2319– 5592 (Online)