RC25449 (WAT1402-065) February 21, 2014
Other

IBM Research Report

A Method for Improving Lossless Compression of Aligned DNA Sequence

Hangu Yeo, Vadim Sheinin
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598
USA

A Method for Improving Lossless Compression of Aligned DNA Sequence

Hangu Yeo and Vadim Sheinin

IBM T. J. Watson Research Center
Yorktown Heights, NY 10598 U. S. A.
{hangu,vadims}@us.ibm.com

Abstract

The huge volumes of genome sequencing data generated with Next Generation Sequencing technology demands compression tools that can handle increasing costs associated to storing and transmitting those data. We present a method for efficient lossless compression of the DNA sequences stored in SAM file format. The compression method compresses alignment information, read sequences and quality values separately using customized compression algorithms. We concentrated our efforts on read base sequence compression optimization, and the key feature of read sequence compression is run-length encoding based on classified and sorted read sequences. The quality values are compressed adaptively with run-length coding and predictive coding followed by Huffman coding technique. The proposed method generates a mapped difference from the reference to the target DNA sequences, classifies the mapped differences using number of mismatched read bases within a sequence into number of groups. The read sequences and quality values within each group are compressed losslessly with different coding strategies, and the coding schemes are decided by the analysis of the histogram created with the indexes of mismatched read bases within the sequence and neighboring quality values around the quality value to be compressed. We focused on the compression of sequences classified into first two groups, perfectly matched sequences and sequences with a single mismatched base, as those sequences account for majority of sequences to be compressed, and the proposed algorithm is a clear winner when compared with well known compression tools.

1. Introduction

The human genome project started in 1990 to map three billion nucleotides of human genome. In 2001, after more than a decade, the first draft of the human genome was published. Advances in genomics during the past ten years have increased the generation of deoxyribonucleic acid (DNA) sequence data and reduced the cost to generate the data especially after the first commercial second (next) generation platform was introduced in 2004. With the advent of Next Generation Sequencing (NGS) technologies, the cost to generate a genome has decreased from about $100,000,000 in 2001 to $10,000 in 2012, and can be less than $1,000 within a couple of years (http://www.genome.gov/sequencingcosts/). As a result, the amount of genomics data is increasing exponentially. For example, accessed in May 2012, New York Genome Center (NYGC) had a plan to produce 9,000 TB of genomics data within the next several years.

The NGS instruments produce tens and hundreds of millions of oversampled (by from thirty times to forty times) short read sequences to ensure each base of three billion base pairs of human genome is sequenced. Each short sequence is a string of characters over the nucleotide alphabets A, C, G and T. This means that 3 billion human genome requires at least 100 GB of disk space to store the samples if we assume one byte for each base. Hence, it created a need to store and transfer very large volumes of data efficiently, and efficient represent of sequencing data is very important.

Although general purpose compression algorithms for text data such as GZIP, BZIP2 OR 7ZIP compress the DNA sequence reasonably well, their compression ratio is only around 3:1 and is likely to be far from optimal. Taking into consideration of properties of genomic data can produce better compression. For example, while the human genome consists of about three billion base pairs, any two human genomes are more than 99% identical. Compression of DNA sequences has been a very active research since the introduction of NGS technology. The compression of genomic sequences can be divided into three categories, assembly based compression, SAM (Sequence Alignment/Map Format) file format compression and FASTQ file format compression.

There have been numerous implementations of a variety of DNA sequence compression algorithms. Jones et al. [1] presented an assembly based compression algorithm. This is the first reference based compression technique reference to its own reference assembled with de novo assembler not the reference human genome. It is noted that the quality of assembly affects compression efficiency. The CRAM compression tool [2] developed by EBI (European Bioinformatics Institute) is a very well-known tool to compress the DNA sequence developed to compress aligned DNA sequences in SAM file format [3]. Popitsch et al. [4] also developed a compression tool that compresses mapped short read data stored in SAM file format, and it encodes only the difference to the reference column-wise. Mahoney [5] developed an algorithm that compresses DNA sequence in FASTQ file format. This is a context model based algorithm using libzpaq public domain compression library. Quality value compression is more difficult to compress than read bases because it covers wider range of values. Quality value data takes up majority of the total compressed file size, and makes the lossless compression tool less efficient. There have been efforts on lossy compression of quality values as a solution to reduce the compressed file size of quality values [2][4][6][7].

In this paper we present a lossless DNA compression algorithm that compresses DNA sequences in SAM file format given a known reference genome. We presume that the reference genome is available to both the encoding unit which compresses the sequence and the decoding unit which decompresses the compressed data without any additional external information. The proposed algorithm first generates a mapped difference from the reference to the target DNA sequences, and compresses the mapped difference and quality values losslessly with adaptive encoding strategies. The proposed algorithm works fairly well against general purpose compression algorithm such as GZIP, and is compared against SAM and FASTQ file format compression tools.

2. Compression of Aligned DNA Sequence

Public repositories such as SRA (Sequence Read Archive) normally stores DNA sequences in human-readable file format called FASTQ. In FASTQ file format, each sequence consists of a sequence of DNA bases and a string of quality values for the read. The quality value reflects the level of confidence in the readout of each base. The sequence of DNA bases is composed of four bases (A, T, G and C), and the set of quality values which ranges from '!' to '~' (ASCII code from 33 to 126) is far larger than the four DNA nucleotides. This makes quality value compression more difficult than read base compression. The inclusion of quality values is optional, and the FASTA format does not include quality value.

The SAM format is a generic format to store read sequences and their alignment to a reference sequence or assembly, and contains associated metadata for the alignment information to a reference sequence as well as read sequences and corresponding quality values. Short read sequence alignment tools such as BWA (Burrows-Wheeler Aligner, bio-bwa.sourceforge.net) and BOWTIE2 (bowtie-bio.sourceforge.net/bowtie2) have received the most attention as short read alignment algorithms that map files in FASTQ file format to a reference genome, and produces output files in SAM format. Figure 1 depicts the block diagram of overall compression scheme. As depicted in the figure, parsed metadata, parsed read sequence and parsed quality values are compressed independently and the compressed files, in turn, are further compressed using GZIP or LZMA compression. The SAM file aligned for FASTA file format does not include quality information.

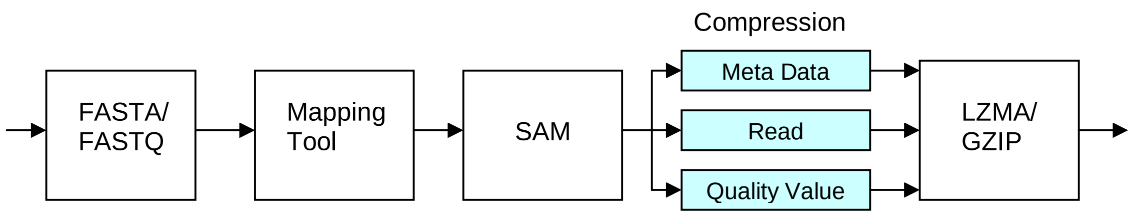


Figure 1. Block diagram (loss compression of DNA sequence).

2.1 Compression of Metadata

The metadata in SAM file format consists of nine mandatory fields that contain alignment information for each read sequence. Since the data types (either string or integer data types respectively) of nine mandatory fields are not identical it is not likely that exactly one encoding strategy will be optimal for different types of fields. Figure 2 depicts metadata compression schemes. The metadata compression is straight forward and simple. Each of nine mandatory fields is parsed and compressed separately using standard compression techniques such as, for example, but not limited to, run-length coding, differential coding and table index coding.

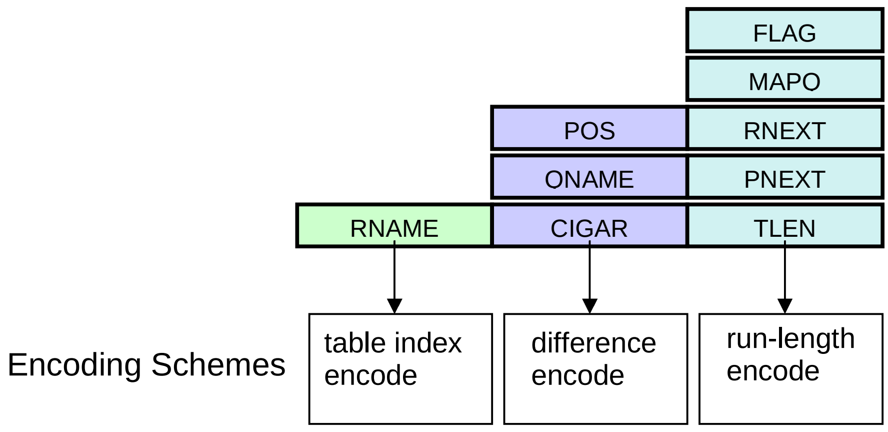


Figure 2. Meta data compression

2.2 Compression of Read Sequence

The read sequence is composed of four bases (A, T, G and C). A key feature of read sequence compression is run-length encoding based on classified and sorted read sequences. Run-length coding is a simple encoding scheme and works well when there are many strings of the same alphabets within the sequences. Generally, run-length coding is not effective to compress the read bases because the four bases randomly occur within the sequence without having any pattern. To improve the efficiency of the simple run-length coding technique, the read sequences are preprocessed as depicted in Figure 3. The processed read sequences include long stretches of identical characters, and the run-length coding efficiency is improved dramatically.

Referring to Figure 3, in the first step, the coding/compression portions of the method take read sequences and a reference genome as input files, and align read sequences to the reference genome using metadata fields such as rname, cigar, pos, etc. In the next step, the read sequences are partitioned into groups based on the number of mismatched bases within the sequence. Since most of the read sequences are perfectly or near-perfectly matched to the reference genome using the mapping position information, most of the read sequences belong to a few groups, and the outputs of the classification are Group0, Group1 and Group 2. Group 0 corresponds only to the read sequences which are perfectly matched. Group 1 corresponds to read sequences which include only one mismatched base in the sequence. Group 2 corresponds to read sequences which include more than one mismatched base in the sequence. Figure 4 shows exemplary read sequences for Group 0, Group 1 and Group 2. The matched bases are replaced with a character ‘=’.

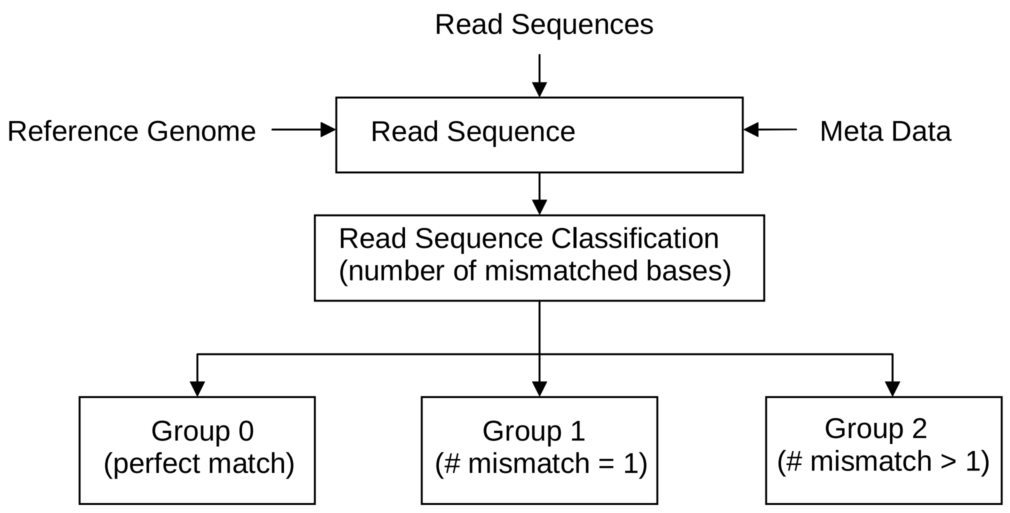


Figure 3. Flowchart (read sequence compression)

Group 0	Group 1	Group 2
===============	========A======	====A=========C
===============	==============C	========T=====G
===============	===========G===	===========GA==
===============	==============G	====C====G=====
===============	===========T===	==========C====
===============	==========C====	C=======A======

Figure 4. Read sequences in each group after alignment and classification.

The read sequences in Group 0 account for 70% of the total sequences, but this group needs zero bit to encode because it is perfectly matched and the decoder does not need any information other than metadata to decode. The Group 1 encoding is depicted in Figure 5. The read sequences classified in Group 1 are sorted in the first step based on the indexes of mismatched base, and if there are multiple sequences with same mismatched base index, the sequences are lexicographically ordered. Figure 6 shows an exemplary sorting for Group 1 based on the indexes. In the next step, a histogram is built using the presorted read sequences classified into the Group 1 (read sequences with a single mismatched base). This step is performed by, for example, histogram builder as depicted in Figure 5. The distribution of indexes decides how the read sequences will be encoded. If the indexes are uniformly distributed along one hundred possible positions (Figure 7), the values of mismatched bases and positions of the mismatched bases are encoded using run-length coding independently into two separate bit streams. In the first string, the mismatched bases are run-length coded column-wise, and the indexes of the mismatched based coded in the first string are run-length coded in the second string. If the distribution of indexes is not uniformly distributed with a couple of peaks as depicted in Figure 7, it means that there are many sequences having same mismatched positions, and column-wise run-length coding works well. The two coding schemes are illustrated in Figure 8.

Read sequences classified into Group 2 (more than one mismatched base) accounts for less than 15% of the total input read sequences, and are sorted and run-length coded column-wise, too. Thus, only metadata is encoded for sequences in Group 0, and both metadata and read sequence data are encoded for sequences in Group 1 and Group 2. The sorting process in Figure 5 is to maximize the length of stretches of '=' characters. Since run-length is basically a pair consisting of a single character and a positive integer number that indicates the run length of the character, the entireties of the sequences within the perfectly matched group can be encoded with zero bit spending. The efficiency of run-length coding is improved when number of mismatched base is more than one by sorting the sequences in the group to have the length of column-wise stretch of '=' longer.

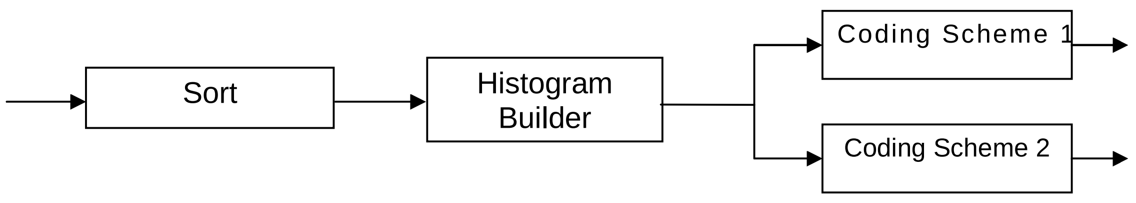
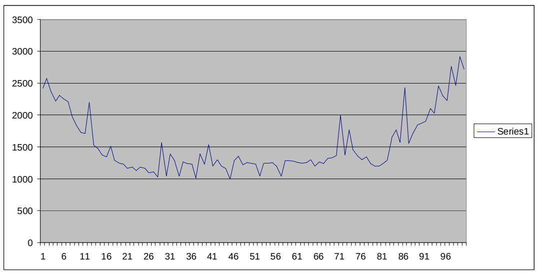


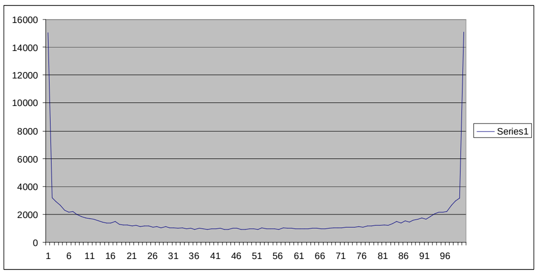
Figure 5. Flowchart (Group 1 read sequence coding schemes)

```
========A======             ==============C
==============C             ==============G
===========G===    ---->    ===========G===
==============G    sort     ===========G===
===========T===             ===========T===
===========G===             ========A======
```

Figure 6. Read sequences in Group 1 are sorted using mismatched indexes as a key value.



Coding Scheme 1



Coding Scheme 2

Figure 7. Histograms using indexes of mismatched bases.

```
==============C
==============G
===========G===
===========G===
===========T===
========A======
```

Coding Scheme 1

Construct two strings, one composed of unmatched base from each sequence, and the second string using the

Sequence: C (G,3) T A
Index: (0,2) (3,3) 6

Coding Scheme 2

Column-wise run-length coding in vertical direction

(=,53) A (=,14) (G,2) T (=,7) C G

Figure 8. Coding example of read sequences in Group 0.

2.3 Compression of Quality Values

Base-calling is a central part in genomic sequencing effort, and it is a process to assign bases to chromatogram peaks and produce base sequences as accurate as possible in the presence of noisy environment which may lead to wrong readout of the sequencing signal. The probability for base calling mistakes is represented by scores that reflect the level of confidence of readout of each base, and higher values represent greater confidence. These scores are known as quality values, and the quality values are part of FASTQ file format. In the FASTQ file format, genomic sequences are represented by a set of read bases along with quality values of each base-call.

As already mentioned in the previous section, the range of quality value (from 33 to 126 in ASCII code) is larger than four DNA nucleotides. This makes the quality value compression more difficult than the read sequence compression. Experiments show some properties of quality values. One such property of quality values is that the quality values are non-uniformly distributed [8], i.e., there are certain values which occur very frequently. Another such property of quality values is that for a given quality value, there is a strong correlation between that quality value and a few previous quality values with quality typically reducing along the length of the sequence [5].

A key feature of quality value compression is encoding the quality values adaptively with run-length coding and predictive coding schemes. To improve the efficiency of the coding scheme, the quality value sequences are presorted lexicographically, and the presorted quality value sequences include long stretches of identical characters and the correlation between vertically positioned quality values is improved as well. Only the DNA sequences corresponding to Group 0 are sorted because Group 1 and Group 2 are sorted using mismatched indexes as key values as described in the previous section.

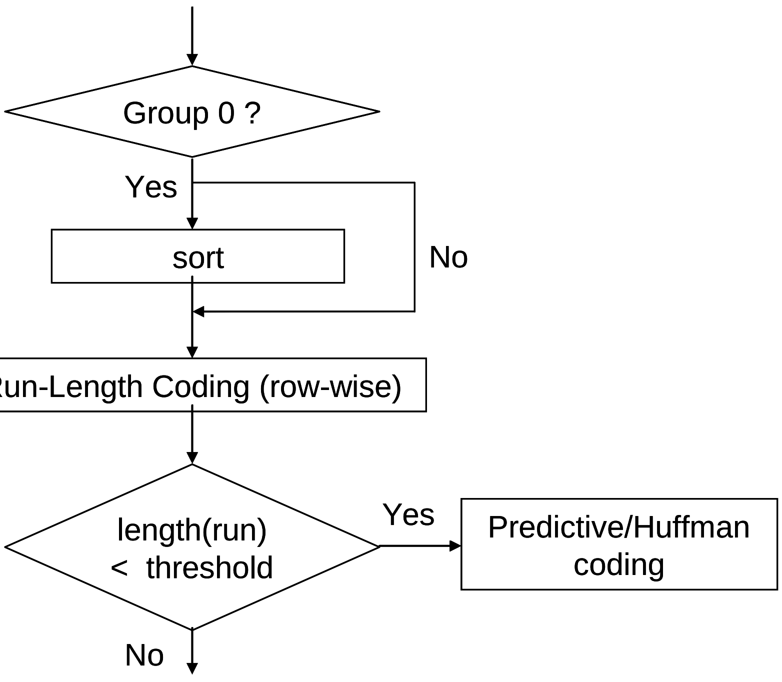


Figure 9. Flowchart (Quality value coding)

Figure 9 shows an exemplary method for quality value compression. In the first step, the DNA sequences are sorted lexicographically followed by row-wise run-length coding on the sorted sequences to obtain a code result. The presorting enhances the correlation between horizontally and vertically positioned quality values further. The length of the run is examined whether or not it is less than a threshold. If so, the quality value is predicted using the neighboring three quality values (one left, one upper and one upper-left positioned) followed by Huffman coding technique.

3. Simulation Results and Comparison

The proposed lossless DNA compression algorithm compresses DNA sequences in SAM file format. The paired-end test FASTQ files were randomly selected and downloaded from ftp://ftp-trace.ncbi.nih.gov/1000genomes, and aligned using public domain software BOWTIE2 and reference human genome hg19 file downloaded from http://genome.ucsc.edu. The test files are compressed using three compression tools, GZIP, CRAM and proposed compression tools, and the compression ratio is calculated by dividing the input file size (paired-end FASATA file size in Table 1 and paired-end FASTQ file size in Table 2) by the compressed output file size.

The compression ratio of the proposed method is compared with a general compression algorithm (GZIP) and CRAM lossless compression mode in TABLE 1. Table 1 compares compressed file sizes using input files in FASTA file format. The proposed method significantly outperforms CRAM in compressing header and read sequences, for example 26:1 (CRAM) versus 92:1 (proposed method) with SRR702072 as an input sequence. In Table 2, we compare the three compression tools using input files in FASTQ file format. The proposed algorithm still outperforms the CRAM tool in compressing three components (header, read bases and quality values) of the FASTQ files. While the proposed algorithm obtains up to 92:1 compression ratio for header and read sequence compression, we believe that improving compression ratio of quality value beyond 4:1 is infeasible. For example, the quality value only compression ratio is 3.9:1 (proposed algorithm) and 3.5:1 (CRAM) with SRR702072 as an input sequence. The results in Table 1 and Table 2 indicate that the compressed quality value data accounts for majority of the total compressed file size.

Table 1. Compression ratio comparison (FASTA file format compression)

Sequence	GZIP	CRAM	proposed
SRR062634	3.9	20	43
SRR043384	4.0	16	34
SRR081241	3.9	23	66
SRR077487	3.9	23	65
SRR037780	4,2	17	39
SRR100335	3.7	18	37
SRR702072	3.9	26	92
ERR013151	3.7	19	41

Table 2. Compression ratio comparison (FASTQ file format compression)

Sequence	GZIP	CRAM	proposed
SRR062634	2.9	7.1	7.9
SRR043384	3.9	10.4	11.6
SRR081241	3.1	6.8	7.5
SRR077487	3.1	6.8	7.4
SRR037780	3.6	13.2	14.7
SRR100335	2.9	6.9	7.4
SRR702072	3.0	6.6	7.6
ERR013151	3.4	7.8	9.0

4. Conclusion

The huge volumes of genome sequencing data generated with NGS technology demands compression tools that can handle increasing costs associated to storing and transmitting those data. We have shown methods for efficient lossless compression of the DNA sequences stored in SAM file format. The proposed algorithm provides higher compression ratios when we compare the performance of the proposed algorithm with a general text compression algorithm (GZIP) and one of the very well-known compression tools (CRAM). Especially in the absence of the quality scores, the simulation results show that the proposed algorithm would be a clear winner in compressing header and read sequence data information. The limitation of the proposed algorithm is that it only supports lossless compression of quality values. Given the fact that compressing quality values beyond 4:1 seems feasible, directions for the future work will include developing lossy compression for quality values.

5. References

[1] D. C. Jones, W. L. Ruzzo, X. Peng and M. G. Katze, “Compression of Next-Generation Sequencing Reads Aided by Highly Efficient De Novo Assembly,” Nucleic Acids Research, August 2012.
[2] M. H. Fritz, R. Leinonen, G. Cochrane and E. Birney, “Efficient Storage of High Throughput DNA Sequencing Data using Referenced-Based Compression,” Genome Research, vol. 21, pp. 734-740, 2011.
[3] The SAM Format Specification, The SAM Format Specification Workgroup, September 2011.
[4] N. Popitsch and A. Haeseler, “NGC: Lossless and Lossy Compression of Aligned High-Throughput Sequencing Data,” Nucleic Acids Research, October 2012.
[5] J. K. Bonfield and M. Mahoney, “Compression of FASTQ and SAM Format Sequencing Data,” PLoS ONE, vol. 8, e59190, March 2013.

[6] I. Ochoa, H. Asnani, D. Bharadia, M. Chowdhury, T. Weissman and G. Yona, "QualComp: A New Lossy Compressor for Quality Scores Based on Rate Distortion Theory," BMC Bioinformatics, vol. 14, June 2013.
[7] C. Kozanitis, C. Saunders, S. Kruglyak, V. Bafna and G. Varghese, "Compressing Genomic Sequence Frgments using SlimGene," Journal of Computational Biology, vol. 18, pp. 401-413, 2011.
[8] R. Wan, V. N. Anh and K. Asai, "Transformations for the Compression of FASTQ Quality Scores of Next Generation Sequencing Data," Bioinformatics, vol. 25 pp. 628-635, March 2012.