algorithms

MDPI

Article

Compression of Next-Generation Sequencing Data and of DNA Digital Files †

Bruno Carpentieri

Dipartimento di Informatica, Università di Salerno; Via Giovanni Paolo II, 132-84084 Fisciano (SA), Italy; bcarpentieri@unisa.it

† This paper is an extended version of our paper published in the Proceedings of the 5th World Multidisciplinary Earth Sciences Symposium, WMESS 2019; (Prague; Czech Republic; 9–13 September 2019).

Received: 9 May 2020; Accepted: 17 June 2020; Published: 24 June 2020

Abstract: The increase in memory and in network traffic used and caused by new sequenced biological data has recently deeply grown. Genomic projects such as HapMap and 1000 Genomes have contributed to the very large rise of databases and network traffic related to genomic data and to the development of new efficient technologies. The large-scale sequencing of samples of DNA has brought new attention and produced new research, and thus the interest in the scientific community for genomic data has greatly increased. In a very short time, researchers have developed hardware tools, analysis software, algorithms, private databases, and infrastructures to support the research in genomics. In this paper, we analyze different approaches for compressing digital files generated by Next-Generation Sequencing tools containing nucleotide sequences, and we discuss and evaluate the compression performance of generic compression algorithms by confronting them with a specific system designed by Jones et al. specifically for genomic file compression: *Quip*. Moreover, we present a simple but effective technique for the compression of DNA sequences in which we only consider the relevant DNA data and experimentally evaluate its performances.

Keywords: data compression; Next-Generation Sequencing data; DNA; genomes

1. Introduction

Next-Generation Sequencing technologies (NGS for short) have finally enabled DNA sequencing at a surprising high speed and low costs.

Recently, the increase in memory and in network traffic used and caused by new sequenced biological data has deeply grown. Genomic projects such as HapMap and 1000 Genomes collected and described the genomes of 2504 individuals from 26 populations, and they contributed to the very large increase in memory and in network traffic occupied and used today by genomics data (see [1,2]).

There is an increasing attention on the new research (and on the important results already obtained) in these areas. The scientific community has been thrilled by the large-scale sequencing of samples of DNA that has been gained in the last eight years. New hardware tools, analysis software, algorithms, private databases, and infrastructures have been recently built to support genomics (see [3,4]).

In this paper, we analyze different approaches for compressing FASTQ, Sequence Alignment/Map (SAM), and binary SAM (BAM) files generated by Next-Generation Sequencing tools containing nucleotide sequences, extending and improving the research we presented in [5].

Moreover, we present a simple but effective technique for the lossless compression of DNA sequences and experimentally evaluate its performances.

This paper is organized as follows:

Section 2 reviews some fundamental concepts: the DNA sequencing workflow and the file formats generated by the machines that are currently used for genomic sequencing and a review of recent data compression tools targeted to genomic compression.

In Section 3, we discuss and evaluate the compression performances of two generic data compression tools, *gzip* and *bzip2*, by confronting them with a specific data compression system designed specifically for FASTA genomic file compression: *Quip* (developed by Jones et al. in [6]).

Section 4 presents a compression algorithm designed for one-dimensional textual files, which code a DNA sequence, and it experimentally assesses its performances. Section 5 discusses our conclusions and future research.

2. NGS

Next-Generation Sequencing (NGS) indicates all the sequencing platforms and the related technologies that were developed after 2005. These platforms have revolutionized the sequencing process by allowing its parallelization, lower costs, and higher performances.

Because of the new hardware and software tools recently developed, there have been frequent changes in the data formats that describe the sequencing results.

In recent years, one of the most important results has been the development of a well-defined and automated workflow for the analysis of genetic data [7].

2.1. Workflow

The sequencing process is not an easy process. It starts from the raw readings of DNA sequences and then digitally maps the genomes.

Its complexity is mainly due to possible errors in the reading or in the alignment phase, and the new NGS instruments have a higher probability of errors with respect to the pre-NGS sequencers.

FASTQ is the textual format that is widely used to represent the output of the NGS platforms. There is not a single version of FASTQ: Sanger FASTQ is the commonly acknowledged standard because it is the format accepted by the American National Center for Biotechnology Information (NCBI) Sequence Read Archive for sending genomic data.

The *alignment* process is the first bioinformatics step in DNA sequence analysis—this is the mapping between the sequences read and the sequence of a reference genome.

The next step is the *assembly* phase in which the sequences that have been read and aligned are now composed together to form the original genetic sequence of the input samples.

The aligned sequence is stored in SAM (Sequence Alignment/Map) format. At the end of the alignment pipeline, the SAM data is converted into BAM (binary SAM), which is a much more compressed format.

The step that follows the alignment phase is the trimming of the obtained sequence. In this phase, we proceed to the extraction of the exon sequences that represent a small fraction of the entire genome and that are believed to code most of the information.

2.2. File Formats

In bioinformatics today, we have to deal with many different file formats often ambiguously or poorly defined.

Flat files contain unstructured records. The interpretation of the content of a flat file depends on the software that will read that file.

The *FASTA* file format is for DNA and amino acid sequences. It was proposed by Pearson and Lipman in the FASTA suite [8].

A FASTA file begins with a line describing the sequence. The description lines (define) are separated from the sequence lines with a 'greater than' symbol (>) at the beginning of the line.

The *FASTQ* file format has been developed as an extension of FASTA, including information on the trustworthiness of the readings.

FASTQ is extremely simple and it is widely used, but this file format is not clearly or unambiguously defined. Therefore, there are many incompatible FASTQ variants—about one for each NGS sequencing machine.

The *SAM* format is the file format used for intermediate and final outputs of alignment software. SAM stands for Sequence Alignment/Map.

The *BAM* format is a compressed format for SAM files that uses the *BGZF* (Blocked GNU Zip Format) format—a block compression format implemented through the gzip standard.

Finally, the *VCF* format [9] is a file format used for storing genetic variants paired with free annotations.

2.3. Compression of Next-Generation Sequencing Data

Large-scale sequencing of DNA samples has brought new interest and it has produced new research in genomic data. The huge increase in the amount of genomic data that is communicated over the internet today has made the study of data compression algorithms specifically targeted to this new type of data necessary.

The NGS workflow we have seen in Section 2.1 implies a pipeline: first of all, sequencing produces a FASTQ file with the readings of the genome. This file is then the input of an alignment program that produces a SAM (or a BAM) file that is subsequently analyzed by a variant caller that stores all the variants in a VCF file.

Potentially, this VCF file has now all the information that could make a reconstruction of the genome possible.

Once all these pipeline steps are completed, we might theoretically store only the VCF file that contains all the information that the actual state-of-the-art knowledge on genetic data about the sequenced genome considers relevant and not the previous and much heavier FASTQ and SAM files.

Generally, this is not a good idea. For example, the alignment and the variant calling tools could be amended over time and the knowledge of what is relevant in genetic data might change after new researches. Re-analyzing the raw data (FASTQ) in the future might permit the finding of new variants.

For the implementation of specific genomic compression tools, two approaches are commonly used: (i) compression of raw NGS data (i.e., FASTQ and SAM files) and (ii) compression of assembled genomes.

The compression of the raw sequencing data, i.e., the content of the FASTQ and SAM files, is mainly devoted to the compression of identifiers, readings, and quality scores. It has been proved that compressing the quality scores is a more difficult task than compressing the readings, due to the higher entropy rate and due to the use of a much larger alphabet. If compressed in a lossless way, the quality score size can be up to 70% smaller than in the original file [10].

2.4. Data Compression Tools

FASTQ, SAM, and BAM files can be compressed by using generic lossless compression tools. This compression software tool is based on the Lempel-Ziv 77 algorithm (widely used in many domains, see, for example, [11,12]).

There are general-purpose lossless compression tools, such as *gzip* [13] and *bzip2* [14], which generally can significantly deflate the size of the genomic file. Their performance is supposed to be good on average on almost all genomic files.

Gzip is based on the Lempel-Ziv 77 algorithm (widely used in many domains, see, for example, [15,16]) and it is a lossless codec that replaces an uncompressed file with a compressed one that has a *.gz* extension.

bzip2 is a general purpose lossless compressor, often more efficient than gzip. It is based on the Burrows–Wheeler transform and on Huffman coding.

The data compressors in use today that are specifically targeted to genomic data are often reference-based compression methods that align reads to a known reference genomic sequence and store only relative genomic positions. The reference genomic sequence is somehow considered as a reference dictionary and needs to be available at both ends of communication: the sender (compressor)

and the receiver (decompressor). For example, *Scramble* (implementing the CRAM file format) [15] and *DeeZ* [16] are reference-based methods.

The authors in [17,18] present the key ideas for compressive genomics and introduce algorithms that allow direct genetic computation on the compressed data. For a review of compressive genomics, see also [4].

Quip [6] is a special purpose lossless compressor targeted to the compression of genetic data. Quip was presented in 2012. It has been widely used and studied because of its open source distribution that is available on the internet at [19] and it is still updated with new distributions. The version of Quip we have tested is not reference-based (it does not need a reference genomic sequence to compress the data) and it uses arithmetic coding, which is another compression strategy used in many domains (see, for example, [20–22]). There is also a reference-based version of Quip.

Quip is a command line application and generates files that have *.qp* as extension. It uses arithmetic encoding to encode all the components of the FASTQ file, but the arithmetic encoder uses different statistical models—one for each different component.

Moreover, Quip uses adaptive modeling by updating the statistics at run time on which arithmetic coding is based in order to upgrade the compression rate on large files.

Quip improves its performances by applying coding strategies specifically targeted to genomic data. For example, we have a unique reading identifier for each reading, and Quip uses delta encoding to remove the redundancy in the reading identifiers (see [6]).

Quip embeds a Markov chains model to compress nucleotide sequences. The prediction of the nucleotide in a given position in the reading is computed by using the 12 previous positions, and a similar approach is used for quality scores too (see [6]).

Recently, a number of effective compression tools that deal specifically with FASTQ or FASTA data have been developed and they are currently tested on real data: *fastqz* [10], *MFCompress* [23], *ALAPY* [24], and *NAF* [25], just to name a few. Krizukov and others in [26] benchmarked 26 genomic compressors (including Quip, fastqz, MFCompress, ALAPY, NAF, and others), together with 18 general-purpose compressors, on a FASTA files data set.

By taking into consideration both the compression performance and the compression/decompression speed, [26] shows that the recent NAF codec demonstrates the best results on that data set.

3. Quip, gzip, and bzip2: Experimental Results

One interesting question that we have decided to answer in this paper is how well Quip performs with respect to the standard lossless compressors that are generally used with genomic data. Of course, this question is addressed also in [6], where the authors compare Quip with the state–of-the-art general-purpose compressors on a test data set of six files from the Sequence Read Archive (SRA [27]). SRA provides data in its own SRA compression format.

The test data set in [6] is composed by six large files, with file sizes ranging from 8.9 GB to 67.6 GB. We have chosen a different data set composed by smaller files (file sizes ranging from 8.8 MB to 5.31 GB) and we have included in our test data set also files produced by the Repli-Seq Illumina (see, for example, [28]).

Therefore, our tests are different and complementary to the experiments conducted in [6].

The files in our test data set could be downloaded, at the moment of writing, from http://genome.crg.es/encode_RNA_dashboard/hg19 and ftp://ftp.ncbi.nlm.nih.gov/hapmap.

Our test dataset is composed by files in FASTQ or SAM/BAM formats. Table 1 summarizes the specifications of our test data set. Figure 1 shows the results of applying these compression tools to our test data set.

Table 1. Test data set.

Sequences	Description	Format	Size
GM-12878-CYTOSOL	Short RNA	FASTQ	5.31 GB
GM-12878-CYTOSOL	Short RNA	BAM	29.1 MB
GM-12801S4	Repli-Seq Illumina	FASTQ	1.01 GB
BG02-1-DS9028-FC20EM7-1	Repli-Seq Illumina	FASTQ	739.6 MB
BG02-1-DS9028-FC20EMB-6	Repli-Seq Illumina	FASTQ	797.3 MB
NA12156 (No graph)	CEPH/UTAH PED. 1408	BAM	8.8 MB
GM-06990S3	Repli-Seq Illumina	BAM	288.3 MB

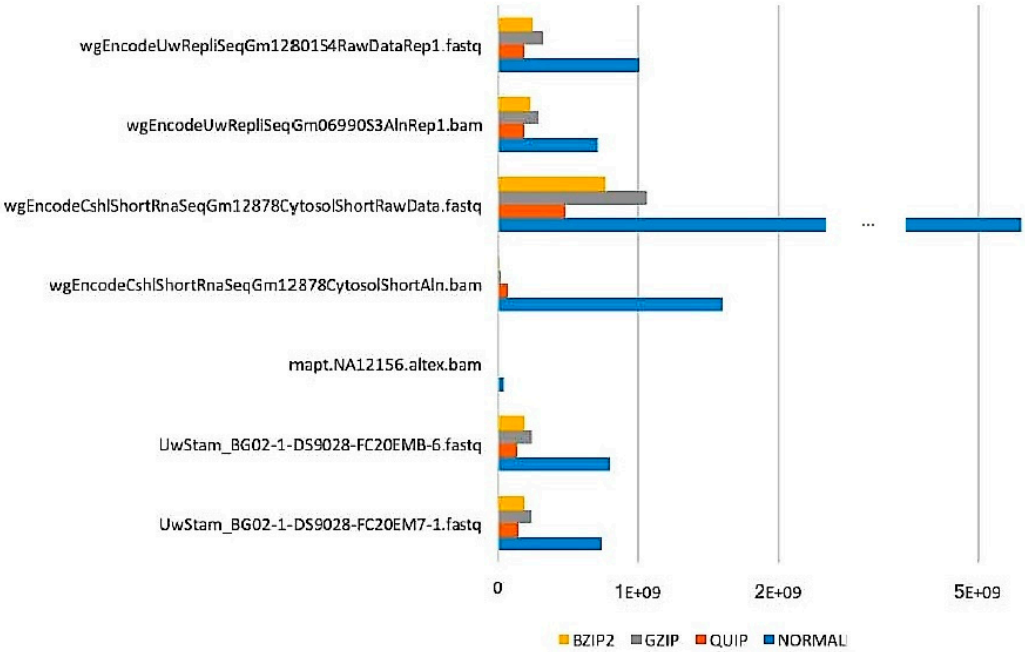

Figure 1. Quip, gzip, and bzip2: experimental results.

In Figure 1, the lossless compression results obtained by bzip2 (i.e., the size of the lossless compressed files) are outlined by yellow lines, the gzip results by gray line, and the quip results by orange lines.

The blue line represents the uncompressed file size.

All these tools achieve a compression ratio that in average is close to 70%.

Gzip has the worst performance and bzip2 is close, but almost always outperformed, to quip.

The compression performances do not seem to depend on the file format and the compression results seem stable independently on the fact that the compressed file is a FASTQ or a BAM. Quip has good performances, but gzip and bzip defend themselves very well.

This confirms the testing results that were presented in [6] but it also shows that the compression performance of Quip on shorter genomic files is closer to bzip2.

4. DNA Compression

There are four nitrogen bases that make up DNA. They are Adenine (A), Cytosine (C), Guanine (G), and Thymine (T). In double-stranded DNA, each of these can only be linked to a single nitrogen base: Guanine can only bind with Cytosine (and vice versa) while Adenine can only bind with Thymine (and vice versa).

Therefore, to memorize a DNA sequence, only one base of each couple (for example, the first element of each couple) need to be coded since the other can be easily deducted [4].

DNA sequencing is normally obtained from a single strand, and the genomic files in FASTA or FASTQ format almost always concern a single-stranded sequence from which the double-stranded DNA can be easily retrieved by coupling each base in the single strand with the other, complementary, base (see [4]).

Even by doing this (and saving about half of the coding), DNA files are very heavy and need several Gigabytes of memory. This of course makes it difficult to store them or to transmit them efficiently.

Here we present an approach to DNA compression that will only consider the relevant DNA data: the combination of the nitrogen bases. By working on such files, we can easily compare and study the DNA of living organisms—even better than with FASTA or FASTQ files.

Suppose we have a file that contains only combinations of the four nitrogen bases. For example, if we begin with a file in the FASTA format, we first need to preprocess the file by considering only one strand (in case the file reports both strands of a double-stranded DNA) and by removing all the other information that do not refer directly to DNA—leaving in the file with all the sequences of nitrogen bases.

We can code each of the four nitrogen bases with only two bits. The way DNA is made and the fact that only four nitrogen bases are present in the file often brings repeated sequences of the same character in the file. Therefore, we can implement a compression algorithm that is theoretically based on run length encoding.

The coding of a DNA sequence will be done by coding each nitrogen base by using two bits. Given that the bases are often consecutively repeated in the text, after coding a base, we will code the number of consecutive occurrences of that base.

This number of occurrences can be potentially unlimited and, therefore, we need a strategy to represent it in a decodable way.

For each base we choose to code this number by using first a fixed number of bits that depends on the base and on the file to be coded.

Algorithm 1 and Algorithm 2 present the basic encoding and decoding steps.

Suppose we have associated to each base x a function $maxrep\ (x)$, where $2^{maxrep\ (x)} - 2$ indicates the maximum number of expected consecutive repetitions for that base in the input file.

$maxrep\ (x)$, for each base x, can be computed by preprocessing the input file and by sending the maxrep numbers to the decoder. Another possibility for managing the maxrep numbers is to initialize these values to a common value and subsequently to use a wise management of the decoded information that leads to a run time adjustment done consistently by encoder and decoder.

In Algorithm 1, we read a base from the input file, we output two bits to code the base and then we count how many consecutive repetitions of that base are present. If there are exactly or more than $2^{maxrep\ (x)} - 1$ repetitions, we will need to send another one or more groups of $maxrep\ (x)$ bits to code that number.

For example, suppose that our file derives from a common FASTQ file (i.e., a file that considers only one of the two strands) and that now we need to code CCAAACCCCCAAA.

If we have allowed two bits for the coding of the repetitions of both C and A, then we will code up to three repetitions with the first two bits. When we code "three" with these two bits, the next four bits will still refer to the same base and will be used to code zero (no more), one (five total), two (six total), or three (seven total) repetitions, and so on.

In our example, we will need two bits to code a C, then two bits to code one more repetition of C. Two bits for A, two bits to code two more repetitions of A. Two bits for C, two bits to code three more repetitions of C, two bits to code one more repetition of C (for a total of five Cs). The previous two bits shall also code the stop of the repetitions of C. Then two bits for A and two bits to code two more repetitions of A.

The total number of bits needed by our example is 18 bits. By coding two bits for character, 26 bits would have been necessary.

If the compressor can see the file to be compressed in advance, it can establish the optimal number of bits to code the repetitions of each base and it can send this number, together with the compressed file, to the decompressor. This will improve the compression, and the amount of data needed for this communication will be negligible with respect to the size of the compressed file.

However, if the compressor and decompressor work in lockstep, (i.e., on-line compression), then it is possible for both compressor and decompressor to estimate at run time, adaptively, the number of bits needed to code the repetitions for each base.

The price we pay, in this case, is sometimes a very small increment in the compressed file dimension. Other times, for longer files, to have the possibility of a variable length encoding of the repetitions can bring advantages.

Algorithm 1: Coding

- x = readnextbase (F); **do**
- output 2 bits coding x;
- *count* = 0;
- **while** (y = readnextbase (F) == x) *count*++;
- **while** (*count* >= $2^{maxrep\ (x)} - 1$) **do begin**
    - output *maxrep* (x) bits, all set to 1;
    - *count* = *count* − $2^{maxrep\ (x)} - 1$;
    - **end**
- output *maxrep* (x) bits coding in decimal *count*;
- $x = y$; **while** (*not EOF*(F));

Algorithm 2: Decoding

do

- receive 2 bits and decode next base x;
- *count* = 1; **do**
    - receive *maxrep* (x) bits in y;
    - *count* = *count* + decimalvalue (y);

  while (decimalvalue (y) = $2^{maxrep\ (x)} - 1$);
- output *count* occurrences of x; **while** (*therearemorebitstodecode* ());

Experimental Results: Compressing DNA Files

This new representation of the DNA data is shorter than the original text file and it is easier to compress.

To experimentally test the algorithm, we considered a test data set composed by six commonly used DNA files: *Lambda Virus* (file size: 48,502 bytes), *Homo sapiens.GRCh38.dna* (file size: 3,072,712,323 bytes), *SRR741411_2* (file size: 7,982,945,875 bytes), *Mais* (file size: 2,104,355,422 bytes), *Cricetus* (file size: 2,320,022,665 bytes), *Pinus* (file size: 20,547,720,415 bytes). All those files have been preprocessed from the original FASTA format to include only the nitrogen bases.

Our coding strategy was used in combination with standard compression tools and we compared the compression results obtained, with or without this coding.

The coding parameters were adaptively estimated on-line and, for double-stranded FASTQ files, the coding of successive, identical, couples of nitrogen bases was optimized by considering the repetitions of the first base in the couple.

The experimental results obtained are summarized in Tables 2 and 3. Each file in the test data set was compressed in nine different ways: by using the standard, lossless, one-dimensional, compression utilities *zip*, *gzip*, *bzip2*, by using our coding strategy, our coding in combination with the *BWT* transform, our coding strategy in combination with *zip*, our coding strategy in combination with both the *BWT* transform and *zip*.

Table 2. Experimental results (1).

File	Algorithm	Compressed size	Compression Ratio
LambdaVirus.fa (48,502 bytes)	zip	14,465	3.35
	gzip	14,316	3.39
	bzip2	13,249	3.66
	Our coding	12,220	3.97
	BWT + our coding	12,218	3.97
	Our coding + zip	12,236	3.96
	BWT + our coding + zip	13,325	3.63
Homo_sapiens.GRCh38dna_sm (3,072,712,323 bytes)	zip	847,980,783	3.62
	gzip	847,980,647	3.62
	bzip2	775,099,444	3.96
	Our coding	748,228,083	4.11
	BWT + our coding	707,234,325	4.34
	Our coding + zip	726,112,596	4.23
	BWT + our coding + zip	646,265,050	4.75
SRR741411_2 (7,982,945,875 bytes)	zip	2,254,125,397	3.54
	gzip	2,254,125,141	3.54
	bzip2	2,040,455,140	3.91
	Our coding	1,987,213,134	4.02
	BWT + our coding	1,804,446,210	4.42
	Our coding + zip	1,926,300,241	4.14
	BWT + our coding + zip	1,764,562,905	4.52

We have not used *Quip* in our experiments or other FASTA compressors because they are not targeted to this kind of input.

The two tables show that the best standard utility to compress this type of data is, as expected, *bzip2*. In fact, *bzip2* outperforms *zip* and *gzip* on all the files but not on the first file, which is the smallest.

Our coding strategy compresses the files with a compression ratio that is in general close in the range from *4 to 1*, outperforming by itself *zip*, *gzip*, and *bzip2*.

There is still space for compression improvements.

In fact, if we pair our strategy to a standard compressor like *zip*, or if we order the data by using the Burrow Wheeler Transform, or if we do both things, we constantly get an improvement in compression.

This improvement, on the other side, corresponds to a higher computational cost (in our experiments, using the *BWT* slowed down the compression time even up to four times than using only our coding strategy).

Table 3. Experimental results (2).

File	Algorithm	Compressed Size	Compression Ratio
Mais (2,104,355,422 bytes)	zip	569,168,389	3.70
	gzip	569,168,251	3.70
	bzip2	529,468,910	3.97
	Our coding	537,100,232	3.91
	BWT + our coding	496,965,993	4.23
	Our coding + zip	502,914,200	4.18
	BWT + our coding + zip	469,326,805	4.48
Cricetus (2,320,022,665 bytes)	zip	654,051,291	3.55
	gzip	654,051,149	3.55
	bzip2	609,362,970	3.81
	Our coding	579,639,181	4.00
	BWT + our coding	536,234,412	4.33
	Our coding + zip	555,831,866	4.17
	BWT + our coding + zip	534,583,442	4.34
Pinus (20,547,720,415 bytes)	zip	5,763,264,652	3.57
	gzip	5,763,264,397	3.57
	bzip2	5,479,811,899	3.75
	Our coding	5,120,956,852	4.01
	BWT + our coding	4,688,228,900	4.38
	Our coding + zip	4,875,200,876	4.21
	BWT + our coding + zip	4,522,676,308	4.54

5. Conclusions

In this paper, different approaches for the compression of FASTQ, BAM, and SAM files generated by Next-Generation Sequencing tools containing nucleotide sequences have been analyzed.

In the first part of the paper, the basic concepts of the application domain have been presented, for example: DNA sequencing workflow and the file formats generated by the machines used for NGS.

The second part of our research discusses and evaluates the performance of the Quip compressor with respect to state-of–the-art generic compression tools, namely, gzip and bzip2.

Quip is a specific compression system targeted to genetic files and developed in [6].

Our testing results have been performed on a test data set that is very different from the one used in the experimental evaluation in [6], and they confirm the results in [6].

The experimental evidences have shown that Quip is effective and outperforms gzip and bzip2 on almost all the files in our test data set.

We have also presented a coding strategy for raw DNA data. We have experimented this coding strategy and we have compared it with the standard, one-dimensional data compression tools.

The experimental results obtained are promising.

Future research should involve a deeper experimentation of our coding strategy and the design of better compressors specifically based on this strategy.

We are also considering approaches that combine data compression and data security for NGS data, following the approach in [29].

Funding: This research received no external funding.

Acknowledgments: The author thanks the students of Felice D'Avino for conducting preliminary experiments on genomic data compression, and Ferdinando D'Avino, Lino Sarto, and Sergiy Shevchenko for implementing and testing a preliminary version of a DNA compression algorithm related to the algorithm presented in Section 4.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. International HapMap Project. Available online: https://www.genome.gov/10001688/international-hapmap-project (accessed on 2 May 2020).
2. 1000 Genomes: A Deep Catalog of Human Genetic Variation. Available online: https://www.internationalgenome.org/ (accessed on 2 May 2020).
3. Siepel, A. Challenges in funding and developing genomic software: Roots and remedies. *Genome Boil.* **2019**, *20*, 147. [CrossRef] [PubMed]
4. Hernaez, M.; Pavlichin, D.; Weissman, T.; Ochoa, I. Genomic data compression. *Annu. Rev. Biomed. Data Sci.* **2019**, *2*, 19–37. [CrossRef]
5. Carpentieri, B. Next Generation Sequencing Data and its Compression. In *IOP Conference Series, Proceedings of the 5th World Multidisciplinary Earth Sciences Symposium (WMESS 2019), Prague, Czech Republic, 9–13 September 2019*; IOP Publishing: Bristol, UK, 2019; Volume 362, p. 012059.
6. Jones, D.C.; Ruzzo, W.L.; Peng, X.; Katze, M.G. Compression of next-generation sequencing reads aided by highly efficient de novo assembly. *Nucleic Acids Res.* **2012**, *40*, e171. [CrossRef] [PubMed]
7. Koboldt, D.C.; Ding, L.; Mardis, E.R.; Wilson, R.K. Challenges of sequencing human genomes. *Brief. Bioinform* **2010**, *11*, 484–498. [CrossRef]
8. Pearson, W.; Lipman, D.J. Improved tools for biological sequence comparison. *Proc. Natl. Acad. Sci. USA* **1988**, *85*, 2444–2448. [CrossRef] [PubMed]
9. Danecek, P.; Auton, A.; Abecasis, G.; Albers, C.A.; Banks, E.; De Pristo, M.A.; Handsaker, R.E.; Lunter, G.; Marth, G.T.; Sherry, S.T.; et al. The variant call format and VCF tools. *Bioinformatics* **2011**, *27*, 2156–2158. [CrossRef] [PubMed]
10. Bonfield, J.; Mahoney, M.V. Compression of FASTQ and SAM format sequencing data. *PLoS ONE* **2013**, *8*, e59190. [CrossRef]
11. Rizzo, F.; Storer, J.A.; Carpentieri, B. LZ-based image compression. *Inf. Sci.* **2001**, *135*, 107–122. [CrossRef]
12. Pizzolante, R.; Carpentieri, B. Visualization, band ordering and compression of hyperspectral images. *Algorithms* **2012**, *5*, 76–97. [CrossRef]
13. gzip. Available online: https://www.gzip.org/ (accessed on 2 May 2020).
14. bzip2. Available online: http://www.bzip.org/ (accessed on 2 May 2020).
15. Bonfield, J. The Scramble conversion tool. *Bioinformatics* **2014**, *30*, 2818. [CrossRef] [PubMed]
16. Hach, F.; Numanagic, I.; Sahinalp, S.C. DeeZ: Reference-based compression by local assembly. *Nat. Methods* **2014**, *11*, 1082–1084. [CrossRef] [PubMed]
17. Loh, P.-R.; Baym, M.; Berger, B. Compressive genomics. *Nat. Biotechnol.* **2012**, *30*, 627–630. [CrossRef] [PubMed]
18. Daniels, N.M.; Gallant, A.; Peng, J.; Cowen, L.J.; Baym, M.; Berger, B. Compressive genomics for protein databases. *Bioinformatics* **2013**, *29*, i283–i290. [CrossRef] [PubMed]
19. Quip. Available online: https://homes.cs.washington.edu/~{}dcjones/quip/ (accessed on 26 May 2020).
20. Pizzolante, R.; Carpentieri, B. Lossless, low-complexity, compression of three-dimensional volumetric medical images via linear prediction. In Proceedings of the 18th International Conference on Digital Signal Processing (DSP), Fira, Greece, 1–3 July 2013; pp. 1–6.
21. Pizzolante, R.; Castiglione, A.; Carpentieri, B.; De Santis, A.; Castiglione, A. Protection of Microscopy Images through Digital Watermarking Techniques. In Proceedings of the International Conference on Intelligent Networking and Collaborative Systems, Salerno, Italy, 10–12 September 2014; pp. 65–72.
22. Castiglione, A.; Pizzolante, R.; Palmieri, F.; Masucci, B.; Carpentieri, B.; De Santis, A.; Castiglione, A. On-board format-independent security of functional magnetic resonance images. *ACM Trans. Embed. Comput. Syst.* **2017**, *16*, 1–15. [CrossRef]
23. Pinho, A.J.; Pratas, D. MFCompress: A compression tool for FASTA and multi-FASTA data. *Bioinformatics* **2013**, *30*, 117–118. [CrossRef] [PubMed]

24. ALAPY. Available online: http://alapy.com/services/alapy-compressor/ (accessed on 26 May 2020).
25. Kryukov, K.; Ueda, M.T.; Nakagawa, S.; Imanishi, T. Nucleotide Archival Format (NAF) enables efficient lossless reference-free compression of DNA sequences. *Bioinformatics* **2019**, *35*, 3826–3828. [CrossRef] [PubMed]
26. Kryukov, K.; Ueda, M.T.; Nakagawa, S.; Imanishi, T. Sequence Compression Benchmark (SCB) database—A comprehensive evaluation of reference-free compressors for FASTA-formatted sequences. *bioRxiv* **2019**, arXiv:642553.
27. Leinonen, R.; Sugawara, H.; Shumway, M. International nucleotide sequence database collaboration the sequence read archive. *Nucleic Acids Res.* **2010**, *39*, D19–D21. [CrossRef] [PubMed]
28. Marchal, C.; Sasaki, T.; Vera, D.L.; Wilson, K.; Sima, J.; Rivera-Mulia, J.C.; Trevilla-Garcia, C.; Nogues, C.; Nafie, E.; Gilbert, D.M. Genome-wide analysis of replication timing by next-generation sequencing with E/L Repli-seq. *Nat. Protoc.* **2018**, *13*, 819–839. [CrossRef] [PubMed]
29. Pizzolante, R.; Castiglione, A.; Carpentieri, B.; De Santis, A.; Palmieri, F.; Castiglione, A. On the protection of consumer genomic data in the Internet of Living Things. *Comput. Secur.* **2018**, *74*, 384–400. [CrossRef]