information

MDPI

Article

A Survey on Data Compression Methods for Biological Sequences

Morteza Hosseini *, Diogo Pratas and Armando J. Pinho

Institute of Electronics and Informatics Engineering of Aveiro/Department of Electronics, Telecommunications and Informatics (IEETA/DETI), University of Aveiro, 3810-193 Aveiro, Portugal; pratas@ua.pt (D.P.); ap@ua.pt (A.J.P.)

* Correspondence: seyedmorteza@ua.pt; Tel.: +351-234-370520; Fax: +351-234-370545

Academic Editor: Khalid Sayood
Received: 27 June 2016; Accepted: 29 September 2016; Published: 14 October 2016

Abstract: The ever increasing growth of the production of high-throughput sequencing data poses a serious challenge to the storage, processing and transmission of these data. As frequently stated, it is a data deluge. Compression is essential to address this challenge—it reduces storage space and processing costs, along with speeding up data transmission. In this paper, we provide a comprehensive survey of existing compression approaches, that are specialized for biological data, including protein and DNA sequences. Also, we devote an important part of the paper to the approaches proposed for the compression of different file formats, such as FASTA, as well as FASTQ and SAM/BAM, which contain quality scores and metadata, in addition to the biological sequences. Then, we present a comparison of the performance of several methods, in terms of compression ratio, memory usage and compression/decompression time. Finally, we present some suggestions for future research on biological data compression.

Keywords: protein sequence; DNA sequence; reference-free compression; reference-based compression; FASTA; Multi-FASTA; FASTQ; SAM; BAM

1. Introduction

With the development of high-throughput sequencing (HTS) technologies, huge volumes of sequencing data are produced everyday, fast and cheaply, that can lead to upgrading the scientific knowledge of genome sequence information as well as facilitating the diagnosis and therapy. Along with these advantages, three challenges exist to deal with this data deluge: storage, processing and transmission. Also, the costs associated with the storage, processing and transmission of HTS data are higher compared to sequence generation, that makes the situation more complicated [1]. Compression is a solution that is able to overcome these challenges, by reducing the storage size and processing costs, such as I/O bandwidth, as well as increasing transmission speed [2–4].

In addition to storage space, processing costs and transmission speed reduction, genomic data compression has two other main applications. First, the performance of *de novo* assembly, for example employing de Bruijn graphs [5] to create (parts of) a genome from raw sequence reads without the aid of a reference genome, can be improved by decreasing the memory usage by one order of magnitude [6,7]; Second, through compression, expert models can be trained on a specific sequence and later be exploited in a similar sequence, to produce higher quality aligners [3,8].

Several genomic data compression methods have been presented in the last two decades. In general, each one favours one or some of the parameters usually associated with the compression, such as compression ratio, compression time, decompression time, compression memory usage and decompression memory usage. In this paper, we review a broad range of these approaches. They were

tested on various test datasets, take into consideration various compression parameters and work on various file formats.

The rest of this paper is organized as follows: In Section 2, we review the methods proposed for the compression of protein sequences. Since these sequences are shown to be difficult to compress, there is not a large number of algorithms for compressing them. In Section 3, we discuss the contributions to the compression of genome sequences, that are divided into two main categories: (i) reference-free compression methods, that exploit structural and statistical properties of the sequences; and (ii) reference-based methods, that compress the sequences with the aid of a (set of) reference sequence(s). In Section 4, we review the methods proposed for the compression of the most widely used file formats, which are FASTA/Multi-FASTA, FASTQ and SAM/BAM. In Section 5, we test several genomic data compression algorithms with various datasets and compare their performance in terms of compression ratio, compression/decompression time and memory usage. Finally, in Sections 6 and 7 we draw conclusions and give some directions for future research on the compression of biological data.

2. Protein Sequence Compression

Proteins play a fundamental role in all organisms, due to their involvement in the control of their development [9–11]. Protein sequences are represented by a 20-symbol alphabet of amino acids. Since there is a high number of different amino acids in these sequences, with an intrinsically disordered nature, they are considered to be complex in terms of compression [12–14].

Protein and DNA sequences have different natures. In a biological context, according to the central dogma, protein sequences are outcomes of genomic sequences, with increasing complexity. This means that from DNA sequences we can retrieve protein sequences, although the opposite is not valid. The importance of protein sequences has increased in the last years, namely with the introduction of the Human Proteome Project [15], having a substantial impact in personalized medicine. Although reference-free compression of proteins is known to be hard, it is not the case for reference-based compression. In the past, almost-complete proteomes were very scarce. Nowadays, this has changed. For example, the three sequenced Neanderthal genomes [16] also include the protein sequences, where each one has 24 MB of data. If compressed individually, each Neanderthal protein sequence, with a `7z` compressor, would need 15.9 MB to be stored. However, when compressing the three files together, only 8.6 MB are needed. Therefore, with the increasing number of sequenced genomes and availability of multiple-proteomes, reference-based compression of proteins will play a key role. In the following paragraphs, several reference-free protein compression methods are described.

In [9,13,17], the probability of contexts and statistical models are exploited. In [17], the `CP` (Compress Protein) scheme is proposed, as the first protein compression algorithm, which employs probability to weight all contexts with the maximum of a certain length, based on their similarity to the current context. The `XM` (eXpert Model) method, presented in [9], encodes each symbol by estimating the probability based on previous symbols. When a symbol is recognized as part of a repeat, the previous occurrences are used to find that symbol's probability distribution; then, it is encoded using arithmetic coding. In [13], three statistical models are proposed based on analysing the correlations between amino acids at a short and medium range.

"Approximate repeats" are utilized in the methods presented in [18,19]. In [18], an algorithm is proposed which considers palindromes (reverse complements) and approximate repeats as two characteristic structures. In this method, an `LZ77`-type algorithm is used to encode long approximate repeats and the `CTW` algorithm [20] is used to encode short approximate repeats. Heuristics are also used to improve compression ratios. The `ProtComp` algorithm, presented in [19], exploits approximate repeats and mutation probability for amino-acids to build an optimal substitution probability matrix (including the mutation probabilities).

Exploiting the "dictionary" is taken into consideration in the methods proposed in [10,21,22]. In [21], the `ProtCompSecS` method is presented, which considers the encoding of protein sequences in connection with their annotated secondary structure information. This method is the combination of

the ProtComp algorithm [19] and a dictionary based method, which uses the DSSP (Dictionary of Protein Secondary Structure) database [23]. The CaBLASTP algorithm, proposed in [22], introduces a course database to store unique data that comes from the original protein sequence database. In this method, sequence segments that align to previously seen sequences are added to a link index. CaBLASTP is the combination of a dictionary-based compression algorithm and a sequence alignment algorithm. In [10], a dictionary-based scheme is proposed, which is based on random or repeated protein sequence reduction and ASCII replacement.

A heuristic method is introduced in [24], that exploits protein domain compositions. In this method, a hyper-graph is built for the proteome, based on evolutionary mechanisms of gene duplication and fusion. Then, on the basis of a minimum spanning tree, a cost function is minimized to compress the proteome.

3. Genomic Sequence Compression

Many studies have been carried out on the topic of genomic sequence compression, taking into account the characteristics of these sequences, such as small alphabet size (i.e., four, namely the nucleotides *A* (adenine), *T* (thymine), *C* (cytosine) and *G* (guanine)), repeats and palindromes [25–28]. In this section, two categories of reference-free methods, that are based only on the characteristics of the target sequences, and reference-based methods, that exploit a (set of) reference sequence(s), are considered for describing these studies.

3.1. Reference-Free Methods

The basic idea of reference-free genomic sequence compression is exploiting structural properties, e.g., palindromes, as well as statistical properties of the sequences [25]. The first algorithm which was specifically proposed for genomic sequences compression is biocompress [29]. In this algorithm, that is based on the Ziv and Lempel compression method [30], factors (repeats) and complementary factors (palindromes) in the target sequence are detected and then they are encoded using the length and the position of their earliest occurrences. The biocompress-2 algorithm [31], an extension of biocompress [29], exploits the same methodology, as well as the use of arithmetic coding of order-2 when it cannot find any significant repetition.

The Cfact algorithm [32] consists of two phases, namely (i) the parsing; and (ii) the encoding. In the parsing phase, the most significant repeats, i.e., the repeats which are able to be encoded for achieving local compression gain, are selected. In order to select such repeats, a suffix tree data structure [33] is exploited to find the longest repeats. In the encoding phase, all non-repeats and first occurrences of repeats are encoded using two bits for each base. Also, the next occurrences of repeats are encoded using pointers to their first occurrences, in the form of $(position, length)$.

The use of "approximate repeats" in the target sequences began with the GenCompress algorithm [34] and was followed by the DNACompress algorithm [35]. In GenCompress [34], the position and the length of non-initial occurrences of repeats are used for encoding them. In DNACompress [35], two phases are considered: (i) finding all approximate repeats containing palindromes, for which the PatternHunter search tool [36] is used; and (ii) encoding approximate repeats and non-repeats, for which the Ziv and Lempel compression method [30] is exploited.

NMLComp [37] and GeNML [38] are two methods which utilize the normalized maximum likelihood (NML) model. The NMLComp algorithm [37] proposes a version of the NML model for discrete regression, for the purpose of encoding the approximate repeats, and then combines it with a first order Markov model. The GeNML algorithm [38] presents the following improvements to the methodology used in the NMLComp method: (i) restricting the approximate repeats matches to reduce the cost of search for the previous matches as well as obtaining a more efficient NML model; (ii) choosing the block sizes which are used in parsing the target sequence; and (iii) introducing scalable forgetting factors for the memory model.

Encoding by word-based tagged code (WBTC) [39] is used in the DNACompact algorithm [40]. In the first phase of this method, the target sequence is converted into words in a way that *A*, *T*, *C* and *G* bases are replaced with *A*, *C*, <*space*>*A* and <*space*>*C*, respectively, i.e., the four symbol space is transformed into a three symbol space. In the second phase, the obtained sequence is encoded by WBTC. The advantage of WBTC is that it does not require saving frequencies or codewords with the compressed stream, since the code of words only rely on ranks.

In [41], considering the statistical characteristics of the genomic sequence, the DNAEnc3 method is proposed, which employs several competing Markov models of different orders to obtain the probability distribution of symbols in the sequences, for the aim of compression. The advantages of this method include: (i) exploiting a flexible programming technique that provides the ability to handle the models of order up to sixteen; (ii) handling the inverted repeats; and (iii) providing probability estimates that cover the broad range of context depths used.

An adaptive particle swarm optimization-based memetic algorithm, POMA, is proposed in [42], which is based on comprehensive learning particle swarm optimization (CLPSO) [43,44] and on an adaptive intelligent single particle optimizer (AdpISPO)-based local search. In this method, an approximate repeat vector (ARV) codebook is designed and then optimized using CLPSO as well as AdpISPO, for compressing the target sequence. The approximate repeats which have the fewest base variations exploit the candidate ARV codebooks encoded as particles to achieve the optimal solution in POMA. Thereafter, the weighted fitness values are used to select the leader particles in the swarm. Finally, an AdpISPO-based local search is exploited to fine-tune the leader particles.

The DNA-COMPACT (DNA COMpression based on a Pattern-Aware Contextual modeling Technique) algorithm [45] exploits complementary contextual models and consists of two phases. In the first phase, the exact repeats and palindromes are searched and then represented by a compact quadruplet. In the second phase, the non-sequential contextual models are introduced in order to exploit the features of DNA sequences; then, the predictions of these models are synthesized using the logistic regression model. In this method, logistic regression is shown to lead to less biased results, rather than Bayesian averaging. It is worth noting that DNA-COMPACT can handle both reference-free and reference-based genome compression.

Transforming genomic sequences into images, where the one-dimensional space is substituted by a two-dimensional space, is an approach that is taken into consideration by [46,47]. In [46], firstly, the Hilbert space-filling curve is exploited to map the target sequence into an image. Secondly, a context weighting model is used for encoding the image. In [47], the CoGI (Compressing Genomes as an Image) algorithm is proposed, which initially transforms the genomic sequence into a binary image (or bitmap); then, it uses a rectangular partition coding method [48] to compress that image. Finally, it exploits entropy coding for further compression of the encoded image as well as the mismatches.

An optimized context weighting method is proposed in [49]. In this method, it is initially shown that the weighting of context models is the same as the weighting of their description lengths; then, on the basis of the minimum description length, the weight optimization algorithm is proposed. Finally, the least squares algorithm is employed for weight optimization.

The GeCo algorithm [50], derived from [51,52], exploits a combination of context models of several orders for reference-free, as well as for reference-based genomic sequence compression. In this method, extended finite-context models (XFCMs), that are tolerant against substitution errors, are introduced. Furthermore, cache-hashes are employed in high order models to make the implementation of GeCo more flexible. The cache-hash uses a fixed hash function to simulate a specific structure, which is a middle point between a dictionary and a probabilistic model. In order to make GeCo more flexible, in terms of memory optimization, the cache-hash takes into consideration only the last hashed entries in memory. This way, the quantification of memory that is required to run in any sequence will be flexible and predictable.

It is worth pointing out that two of the protein compression methods mentioned before, XM [9] and the method proposed in [18], are able to be employed as reference-free genome sequence compressors.

3.2. Reference-Based Methods

The key idea of reference-based genome sequence compression is exploiting the similarity between a target sequence and a (set of) reference sequence(s). For this purpose, the target is aligned to the reference and the mismatches between these sequences are encoded. Since the decompressor has access to the reference sequence(s), the reference-based methods can obtain very high compression rates [26,28,53,54].

The DNAzip algorithm [55] was presented for compressing James Watson's genome, considering the high similarity between human genomes, implying that only the variation is required to be saved. In this method, variation data are considered in three parts: (i) SNPs (single nucleotide polymorphisms), which are saved as a position value on a chromosome and a single nucleotide letter of the polymorphism; (ii) insertions of multiple nucleotides, which are saved as position values and the sequence of nucleotide letters to be inserted; and (iii) deletions of multiple nucleotides, which are saved as position values and the length of the deletion. Finally, the following techniques are used to further compress the target genome: variable integers for positions (VINT), delta positions (DELTA), SNP mapping (DBSNP) and *k*-mer partitioning (KMER).

In the RLZ algorithm [56], each genome is greedily parsed into factors, using the LZ77 approach [30]. Then, through a single pass over the collection, the compression is done relative to the reference sequence, which is used as a dictionary. It is worth pointing out that RLZ supports random access to arbitrary substrings. Later, in [57], the authors presented an improved version of the RLZ method, which firstly, uses non-greedy parsing; and secondly, exploits the correlation between the starting points of long factors and their positions in the reference sequence, to improve the compression performance.

The GRS algorithm [58], which is based on the diff utility in Unix, finds the longest subsequences that are identical in the reference and the target sequences. In this method, a similarity measure is calculated and if it has a larger value than an identified threshold, the difference between the reference and the target sequences are compressed with Huffman encoding [59]; otherwise, the reference and the target sequences are divided into smaller pieces and the same strategy, i.e., comparing the value of the similarity measure with an identified threshold, is used to compress these pieces.

A statistical compression method, GReEn (Genome Re-sequencing Encoding), is proposed in [60], which exploits a probabilistic copy model. In this method, the probability distribution of each symbol in the target sequence is estimated using an adaptive model (the copy model), which assumes that the symbols of the target sequence are a copy of the reference sequence, or a static model, relying on the frequencies of the symbols of the target sequence. These probability distributions are then sent to an arithmetic encoder [61]. GReEn includes two parameters that can be used to alter the performance: (i) the size of the *k*-mer used for searching copy models; and (ii) the number of prediction failures that the copy model can tolerate, before it is restarted.

The GDC (Genome Differential Compressor) method [62], that is based on the LZ77 approach [30], considers multiple genomes of the same species. In this method, a number of extra reference phrases, that are extracted from other sequences, along with an LZ-parsing for detection of approximate repeats are used. The GDC has several variations including *normal*, *fast*, *advanced*, *advanced-R* and *ultra*. In the GDC-*ultra*, that provides the maximum compression ratio among other variations, multiple references are exploited. The GDC 2 method [63], that is an improved version of the GDC, exploits two-level LZSS factoring [64] to encode the sequences as a list of matches and literals. The advantage of this method in comparison to GDC is considering short matches after some longer ones. It is worth pointing out that both the GDC and GDC 2 methods support random access to an individual sequence.

In [53], an adaptive method is proposed, which first divides the reference sequence into fixed length blocks; then, it exploits the compressed suffix tree [65], that have already been computed for each block, to find the longest prefix-suffix matches between the reference and the target sequences of the same species. It is worth pointing out that for the purpose of implementing this method, it is possible to use parallelization on multiple cores.

COMRAD (COMpression using RedundAncy of DNA) [66] is a dictionary construction method, in which a disk-based technique, RAY [67], is exploited to identify exact repeats in large DNA datasets. In this method, the collection is frequently scanned to identify repeated substrings, and then use them to construct a corpus-wide dictionary. An advantage of COMRAD is supporting random access to individual sequences and subsequences, which resolves the need for decoding the whole data set.

The fragments of nucleotides with the hierarchical tree structure are exploited in [68]. In this method, the difference between the reference and the target sequences is taken into consideration, in the sense that the sizes of sub-fragments and matching offsets are flexible to the stepped size structure. Finally, the difference sequence, the sub-fragment size and the matching offset, are compressed using a coding scheme, which is based on the PPM method [69].

The DNA-COMPACT method [45], which can be used for both reference-free and reference-based genome compression, includes two phases. In the first phase, an adaptive mechanism, rLZ, is presented which firstly, considers a sliding window for the subsequences of the reference sequence; and secondly, searches for the longest exact repeats in the current fragment of the target sequence, in a bi-directional manner. In the second phase, the non-sequential contextual models are introduced and then their predictions are synthesized exploiting the logistic regression model.

In the FRESCO (Framework for REferential Sequence COmpression) method [70], a compressed suffix tree [65] of the reference sequence is exploited to match the prefixes of an input string with substrings of the reference sequence. In addition, three techniques are used to improve the compression performance: (i) selecting a good reference; (ii) rewriting the reference sequence; and (iii) second-order compression, that is reference-based compressing of an already referentially compressed sequence.

Taking miniaturized devices into consideration, a genome compression method, based on Distributed Source Coding, is proposed in [71]. In this method, a low-complexity Slepian-Wolf coding [72] is used for the non-repeats, and the adaptive length hash coding is used for the exact repeats. Then, a factor graph model is introduced to detect insertion, deletion and substitution between the reference sequence and the target sequence.

The ERGC (Efficient Referential Genome Compressor) method, based on a divide and conquer strategy, is proposed in [73]. In this method, first, the reference sequence and the target sequence are divided into some parts with equal sizes; Then, a greedy algorithm, which is based on hashing, is used to find one-to-one maps of similar regions between the two sequences; Finally, the identical maps, as well as dissimilar regions of the target sequence, are fed into the PPMD compressor [74] for further compression of the results.

The iDoComp algorithm [75] consists of three phases. In the first phase, named mapping generation, the suffix arrays are exploited to parse the target sequence into the reference sequence. In the second phase, named post-processing of the mapping, the consecutive matches, which can be merged together to form an approximate match, are found and used for reducing the size of mapping. Finally, in the third phase, named entropy encoding, an adaptive arithmetic encoder is used for further compression of the mapping and then generation of the compressed file.

The CoGI method [47] can be used for reference-free as well as for reference-based genome compression. Considering the latter case, at first, a proper reference genome is selected using two techniques based on base co-occurrence entropy and multi-scale entropy [76–78]. Next, the sequences are converted to bit-streams, using a static coding scheme, and then the XOR (exclusive or) operation is applied between each target and the reference bit-streams. In the next step, the whole bit-streams are transformed into bit matrices like binary images (or bitmaps). Finally, the images are compressed exploiting a rectangular partition coding method [48].

4. File Formats

The methods described in this section have been designed for compressing files in one or more of the FASTA/Multi-FASTA, FASTQ and SAM/BAM formats. In these file formats, other information, such as identifiers and quality scores, are added to the raw genomic or protein sequences. It is worth

noting that there are other formats for storing biological data, such as Illumina Export format, but they were developed for targeting a specific technology [4].

Also, the Multiple Alignment Format (MAF) is a file type containing alignments between entire genomes of several species, represented in a two-dimensional style. Some algorithms have been proposed for their lossy and lossless compression [79–81]. Moreover, the Variant Call Format (VCF) is a generic format for storing DNA polymorphism data such as SNPs, insertions, deletions and structural variants, together with rich annotations [82]. Its compression has been addressed recently in [83].

4.1. FASTA/Multi-FASTA

"FASTA" is a text-based format which represents nucleotides or amino acids with single-letter codes [84]. A sequence in this format has two parts: (i) the description, that starts with the ">" symbol; and (ii) the sequence data. The "Multi-FASTA" format, as shown in Figure 1, is the concatenation of different single sequence FASTA files. In this section, several methods for compressing FASTA/Multi-FASTA files are described.

Figure 1. A sample of the Multi-FASTA file.

The `BIND` algorithm [85] employs `7-Zip` for compressing the sequence headers. Also, for the `DNA` reads, it uses bit-level encoding for the four nucleotide bases (*A*, *T*, *C* and *G*) and then splits the codes into two data streams. Next, it uses a unary coding scheme to represent the repeat patterns in these two data streams in a "block-length encoded" format. Finally, `BIND` uses the `LZMA` [86] algorithm to compress the block-length files.

The `DELIMINATE` (a combination of Delta encoding and progressive ELIMINATion of nuclEotide characters) method, proposed in [87], performs the compression in two phases. In the first phase, all non-*ATGC*, as well as all lower-case characters, are recorded, and then a file without these characters is transferred to the next step. In the second phase, the positions of the two most frequent nucleotide bases are delta encoded; then, the two other bases are represented by a binary code. Finally, all generated files are further compressed with the `7-Zip` archiver.

In [88], a method is presented in which the information of mismatch between the reference and the target genome sequences is employed for the compression. This information includes the distance to the next mismatch, the mismatch type (insertion, deletion or replacement) and the difference nucleotides, that are combined in the form of triplets (*differential length*, *mismatch type*, *nucleotide*). It is worth pointing out that in this method Fibonacci codes [89] are exploited in order to encode each "differential length".

The `MFCompress` method [90] exploits finite-context models, which are probabilistic models that select the probability distribution by estimating the probability of the next symbol in the sequence based on the *k* previous symbols (order-*k* context). `MFCompress` encodes the sequence headers using single finite-context models and encodes the `DNA` sequences using multiple competing finite-context models [41], along with arithmetic coding.

"Assembly principles" are utilized in the `LEON` method [91], that is based on a reference probabilistic de Bruijn Graph. This graph is built *de novo* from the reads and saved in a `Bloom` filter [92].

In this method, the reads are considered as the paths in the de Bruijn Graph and compressed by memorizing the anchoring k-mers and the list of bifurcations. Since the LEON scheme can exploit the same probabilistic de Bruijn Graph for encoding the quality scores, this method is able to be used for FASTQ files, too.

In [93], a *de novo* parallelized software, MetaCRAM, is proposed. It integrates taxonomy identification, alignment, assembly and compression. For the taxonomy identification, it exploits the Kraken method [94] to identify the mixture of species included in the sample. Also, for the alignment and assembly, at first, MetaCRAM uses Bowtie2 [95] to align reads to the reference genomes, and then it uses IDBA-UD [96] to find reference genomes for unaligned reads. Finally, MetaCRAM exploits Huffman [59], Golomb [97], and Extended Golomb encodings [98] for compressing the reads, and QualComp [99] for compressing the quality scores. Unaligned reads are compressed with the reference-free MFCompress compression method [90]. The MetaCRAM method is also able to support the FASTQ file format.

4.2. FASTQ

"FASTQ" is a text-based format which represents biological sequences and their corresponding quality scores with ASCII characters. This file format is considered as the *de facto* standard for storing the output of high-throughput sequencing instruments [100]. A sequence in FASTQ format, as shown in Figure 2, has four parts: (i) the sequence identifier; (ii) the raw sequence letters; (iii) a "+" character, optionally followed by the same sequence identifier; and (iv) the quality scores. The following methods support the compression of FASTQ files.

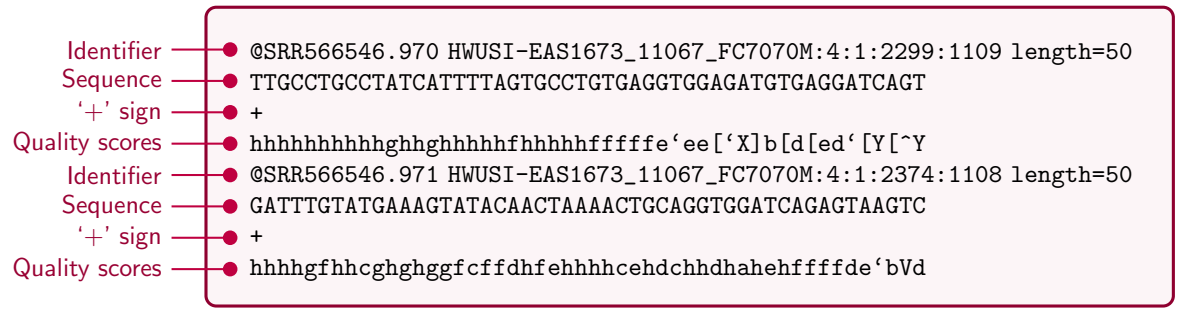

Figure 2. A sample of the FASTQ file.

The GenCompress method [101], first, maps short sequences to a reference genome; then, it encodes the addresses of the short sequences, their length and their probable substitutions, using entropy coding algorithms, e.g., Golomb [97], Elias Gamma [102], MOV (Monotone Value) [103] or Huffman coding [59]. Similar to GenCompress, the G-SQZ scheme [104] employs Huffman coding; however, it does compression without altering the relative order. Also, G-SQZ supports random access to the compressed data.

Dividing the input file into blocks is an approach used in the DSRC [54] and DSRC 2 [105] methods. The DSRC method [54] groups blocks into superblocks; then, using the statistics of blocks inside each superblock, it encodes the superblocks, independently. In this method, LZ77-style encoding, order-0 as well as order-1 Huffman encodings are used for compressing the identifiers, the DNA reads, and the quality scores, respectively. DSRC 2 [105] is an improved version of DSRC, implemented using multi-threading. In this approach, Huffman encoding, as well as arithmetic encoding [106], were added to the existing algorithm used for compressing athe DNA reads in DSRC. Also, in this method, the quality scores can be compressed arithmetically.

In [107], a method is proposed which at first parses the identifiers into repeating and variable components and then uses adaptive arithmetic coding for the compression of the variable components.

In the next step, it exploits Markov encoding as well as the methods based on repeat finding to compress the DNA reads. Finally, it uses run length-based methods for compressing the quality scores.

In the Quip scheme [108], arithmetic coding associated with models based on order-3 and high-order Markov chains are exploited for compressing the read identifiers, the nucleotide sequences and the quality scores. This method has some variations, including "Quip -r" and "Quip -a", that perform reference-based and assembly-based compression, respectively. In the assembly-based compression, the first 2.5 million reads are employed to assemble contigs that can be used instead of a reference sequence database. It is worth pointing out that Quip also supports the SAM/BAM file format.

A boosting scheme, SCALCE, is proposed in [109], which re-organizes the reads to obtain higher compression ratio and speed. In order to perform this re-organization, the long "core" substrings, derived from the locally consistent parsing (LCP) method [110–112], that are shared between the reads, are considered. Finally, these reads are "bucketed" and compressed together, using Lempel-Ziv variants in each bucket.

The SeqDB scheme [113] performs the compression in two phases. In the first phase, a byte-packing method, named SeqPack, is proposed, which interleaves DNA reads with their corresponding quality scores by exploiting a 2D array. Using this array, only one byte is required to store the information of reads and quality scores, instead of the two bytes used by FASTQ. In the second phase, the Blosc method [114] is exploited to compress the interleaved data. This method benefits from two optimization approaches, namely multi-threading and cache-aware blocking, and parallelization through SIMD vectorization [115].

In [116], two methods, named Fqzcomp and Fastqz, are proposed for compressing the files in FASTQ format. The Fqzcomp method exploits a byte-wise arithmetic coder [117] as well as context models that are tuned to the type of data. The Fastqz method employs the "libzpaq" compression library to specify the context models in ZPAQ format [118]. The ZPAQ is based on PAQ [119], which combines the bit-wise predictions of several context models, adaptively.

The ORCOM (Overlapping Reads COmpression with Minimizers) method [120], utilizes the concept of distributing the data into disk bins and compressing each bin. Also, it uses "minimizers" [121] to pass similar (overlapping) DNA reads into the same bin [94,122–125]. In order to compress the streams of data, on bin-by-bin basis, ORCOM employs a context-based compressor, PPMd [126], or an arithmetic coder [106].

A light-weight compression method, named LW-FQZip, is presented in [127], which eliminates redundancy from a FASTQ file at first step. Then, it employs incremental and run-length-limited methods for compressing the identifier and the quality scores, respectively. Next, in order to encode the reads, a light-weight mapping model is introduced which maps them against a reference sequence. Finally, the LZMA [86] algorithm is utilized to compress all the processed data.

4.3. SAM/BAM

"SAM" (Sequence Alignment/Map) is a text-based format which represents biological sequences that are aligned to a reference sequence. A sequence in SAM format, as shown in Figure 3, contains two parts: (i) the header, that starts with an "@" character; and (ii) the alignment section, that has 11 fields, such as QNAME, CIGAR, SEQ and QUAL. It is worth pointing out that a "BAM" file is the binary version of a SAM file, that is compressed with the BGZF (Blocked GNU Zip Format) tool [3,128,129]. In this section, several methods proposed for compressing the SAM/BAM files are described.

In [130], a method is proposed in which the starting position of a read, with respect to the reference sequence, is encoded with the Golomb algorithm [97]. Also, in the case of a non-perfect match between the read and the reference, a list of variations, including position on the read and type (substitution, insertion and deletion), is stored and a part of it is then encoded using the Golomb and Gamma coding algorithms [97,102]. In this method, the amount of quality information that is stored can be changed, in order to increase the efficiency.

```
@HD VN:1.5 SO:coordinate
@SQ SN:ref LN:45
r001   99 ref  7 30 8M2I4M1D3M = 37  39 TTAGATAAAGGAT *
r002    0 ref  9 30 3S6M1P1I4M *  0   0 AAAAGATAAGG   *
r003    0 ref  9 30 5S6M       *  0   0 GCCTAAGCT     * SA:Z:ref,29,-,6H5M,17,0;
r004    0 ref 16 30 6M14N5M    *  0   0 ATAGCTTCA     *
r003 2064 ref 29 17 6H5M       *  0   0 TAGGC         * SA:Z:ref,9,+,5S6M,30,1;
r001  147 ref 37 30 9M         =  7 -39 CAGCGGC       * NM:i:1
```

Figure 3. A sample of the SAM file.

The Goby approach [131] introduces multiple techniques for compressing structured high-throughput sequencing (HTS) data. These techniques include: (i) field reordering; (ii) field modeling; (iii) template compression; and (iv) domain modeling. Also, Goby employs a number of engineering techniques, such as: the protocol buffers [132] middleware, a multi-tier data organization, and a storage protocol for structured messages.

The NGC method [133] introduces a lossless approach for compressing read bases, namely vertical difference run-length encoding (VDRLE), which utilizes the features of multiple mapped reads instead of considering each read individually. As a result, the number of required code words is reduced. Also, NGC introduces a lossy approach for compressing quality values, which maps all quality values within an interval to a single value. In addition, for the compression of read names (stream QNAME) NGC uses the longest common prefix of the last n reads.

The Samcomp method [116] has two variations, Samcomp1 and Samcomp2. In Samcomp1, the same method as Fqzcomp [116] is used to compress the identifier and the quality scores. Also, for compressing the reads, a per-coordinate model is used. In Samcomp2, that can read SAM files in any order, instead of the per-position model a reference difference model is exploited in which a history of previous bases matching the context is used.

In [134], the DeeZ method is proposed, which exploits "tokenization" of read names and compresses them with delta encoding. This method also employs an order-2 arithmetic coding, in a block-by-block manner, for compressing the quality scores and the mapping locations, and thus providing the random-access ability. It is worth pointing out that gzip [135] is used in DeeZ, to compress the other fields.

5. Results

In order to test different genomic data compression methods, we employed several genomes with different species origins and lengths, from ∼70 Megabytes to ∼200 Gigabytes, which are shown in Tables 1 and 2. As it is shown in these tables, we have used 11 genomes for the reference-free compression methods, and six target and six reference genomes for the reference-based compression methods. These genomes are in three different formats, namely FASTA/Multi-FASTA, FASTQ and BAM. The results presented in this paper can be replicated using the software available at [136].

The human genome was chosen to incorporate the dataset, because it is one of the most sequenced, a trend that will certainly continue, due to the personalized medicine paradigm. We choose the chimpanzee and gorilla, because they are very similar and (not so) similar to the human genome, respectively. As it will be shown in the experimental results, this variety is important in the evaluation of reference-based methods. The particular chromosomes used (8, 11 and 16), were chosen randomly. We have also included the rice genome, because it is not a primate genome and it is commonly used. Regarding the camera dataset, it is a compilation of multiple organisms, namely prokaryotic. This compilation has more than 40 GB of data.

Table 1. Genomic sequence datasets used by reference-free compression methods.

File Format	Dataset	File Size (MB)
FASTA/Multi-FASTA	Human (GRC) [137]	2,987
	Chimpanzee [138]	3,179
	Rice5 [139]	166
	CAMERA Prokaryotic Nucleotide [140]	43,945
FASTQ	ERR174310_1 [141]	51,373
	ERR174310_2 [142]	51,373
	ERR194146_1 [143]	209,316
	ERR194146_2 [144]	209,316
BAM	NA12877_S1 [145]	74,455
	NA12878_S1 [146]	108,290
	NA12882_S1 [147]	88,318

Table 2. Genomic sequence datasets used by reference-based compression methods.

File Format	Target Genome	File Size (MB)	Reference Genome	File Size (MB)
FASTA/Multi-FASTA	HS8 [148]	138	HSCHM8 [149]	136
	HS11 [150]	128	HSCHM11 [151]	125
	HS11 [150]	128	PT11 [152]	126
	HS11 [150]	128	PA11 [153]	127
	HSK16 [154]	71	HS16 [155]	78
	Rice5 [156]	166	Rice7 [157]	164

For making a comparison between different compression methods, all of them were executed on a server running 64-bit Ubuntu with 16-core 2.13 GHz Intel® Xeon® CPU E7320 and with 256 GB of RAM; then, their performance was evaluated based on compression ratio, i.e., the ratio of the size of original sequence to the size of compressed sequence, compression/decompression time and compression/decompression memory usage. It is worth pointing out that some of the methods addressed in this paper are missing from the tables of results, due to several reasons. These reasons are described in [158].

The results of compressing genomic data sequences using different reference-free as well as general-purpose methods for FASTA/Multi-FASTA, FASTQ and BAM file formats are reported in Tables 3–5, respectively. Also, the compression results for the reference-based methods are shown in Table 6. In all of these tables, C-Ratio, C-Time (min), D-Time (min), C-Mem (MB) and D-Mem (MB) stand for compression ratio, compression and decompression times in minutes of CPU time, compression and decompression memory usage in Megabytes, respectively. For the compression and decompression times, we opted for reporting CPU time instead of real-time, to give a more accurate idea of the cost in terms of computational resources needed by each of the methods. Of course, methods that use multi-threading usually require less real-time to run rather than the corresponding CPU time. Also, for some of the methods, the total memory required depends on the number of threads used and hence on the particular machine that those methods are running. For all of the methods, the default parameters were used. It should be noted that in Tables 3–6, the best results in terms of different compression metrics are shown with bold font.

The results of FASTA/Multi-FASTA file compression with `DELIMINATE` [87], `MFCompress` [90], `gzip` [135] and `LZMA` [86] methods, reported in Table 3, show that `MFCompress` outperforms the other methods in terms of compression ratio, except for the Rice dataset, which has the smallest size among the datasets. In terms of other performance parameters, such as compression/decompression time and compression/decompression memory, `gzip` outperforms the others.

Table 3. Reference-free compression results of genomic sequence data for FASTA/Multi-FASTA file format.

Dataset	Method	Metric				
		C-Ratio	C-Time (min)	D-Time (min)	C-Mem (MB)	D-Mem (MB)
Human (GRC) (2.92 GB)	DELIMINATE [87]	4.95	15.08	5.90	685.55	73.54
	MFCompress [90]	**4.96**	34.47	29.48	1,240.45	1,266.49
	gzip [135]	3.59	**8.85**	**0.50**	**6.87**	**6.87**
	LZMA [86]	4.39	117.70	1.26	199.52	31.58
Chimpanzee (3.10 GB)	DELIMINATE	5.36	16.64	6.62	683.31	73.55
	MFCompress	**5.43**	33.93	28.55	1,377.85	1,374.18
	gzip	3.85	**8.79**	**0.52**	**6.87**	**6.87**
	LZMA	4.68	118.31	1.26	199.52	31.58
Rice5 (0.16 GB)	DELIMINATE	**5.22**	0.59	0.33	368.11	43.41
	MFCompress	5.15	2.15	1.72	1,169.64	1,082.49
	gzip	3.28	**0.53**	**0.03**	**6.87**	**6.87**
	LZMA	4.46	6.71	0.07	199.71	25.79
CAMERA (42.91 GB)	DELIMINATE	5.07	198.24	85.38	684.66	687.44
	MFCompress	**5.39**	571.74	223.61	2,764.16	2,668.16
	gzip	3.52	**141.33**	**7.66**	**6.87**	**6.87**
	LZMA	4.51	1,720.09	16.48	199.71	31.58

Table 4. Reference-free compression results of genomic sequence data for FASTQ file format.

Dataset	Method	Metric				
		C-Ratio	C-Time (min)	D-Time (min)	C-Mem (MB)	D-Mem (MB)
ERR174310_1 (50.17 GB)	Fqzcomp [116]	4.76	**49.58**	64.79	77.21	67.31
	Quip [108]	4.76	95.36	140.09	395.16	389.22
	DSRC [54]	4.64	140.67	177.37	10,608.25	12,227.26
	SCALCE [109]	**4.97**	259.72	75.12	5,270.25	1,053.52
	gzip [135]	2.90	119.32	**9.48**	**6.87**	**6.87**
	LZMA [86]	3.61	1,731.28	28.59	199.52	31.58
ERR174310_2 (50.17 GB)	Fqzcomp	4.93	**51.79**	65.47	77.27	67.24
	Quip	4.93	94.76	123.29	394.69	391.01
	DSRC	4.82	150.67	159.58	11,109.85	11,867.62
	SCALCE	**5.16**	233.46	73.51	5,320.82	1,047.57
	gzip	2.98	117.65	**9.12**	**6.87**	**6.87**
	LZMA	3.74	1,716.42	27.74	199.71	31.58
ERR194146_1 (204.41 GB)	Fqzcomp	4.87	**186.94**	234.54	79.36	67.21
	Quip	4.78	326.87	438.08	397.96	392.16
	DSRC	4.72	490.42	637.80	11,288.26	11,965.20
	SCALCE	**6.25**	956.79	292.31	5,288.01	1,053.46
	gzip	3.94	273.20	**33.28**	**6.87**	**6.87**
	LZMA	4.84	6,369.49	88.22	199.52	31.58
ERR194146_2 (204.41 GB)	Fqzcomp	4.85	**166.93**	221.19	61.49	65.18
	Quip	4.77	296.14	426.60	396.16	392.41
	DSRC	4.70	600.75	672.58	11,306.27	14,600.30
	SCALCE	**6.18**	950.32	291.13	5,336.13	1,058.48
	gzip	3.90	272.37	**33.69**	**6.87**	**6.87**
	LZMA	4.80	6,295.37	89.15	199.52	31.58

Table 5. Compression results of genomic sequence data for BAM file format.

Dataset	Method	Metric				
		C-Ratio	C-Time (min)	D-Time (min)	C-Mem (MB)	D-Mem (MB)
NA12877_S1 (72.71 GB)	Deez [134]	**1.72**	440.80	505.27	2,368.35	5,803.17
	Quip [108]	1.41	548.97	584.99	464.84	459.89
	gzip [135]	1.00	**72.85**	**12.11**	**6.87**	**6.87**
	LZMA [86]	0.99	865.02	133.36	199.45	31.58
NA12878_S1 (105.75 GB)	Deez	**1.73**	668.89	752.77	2,586.70	5,584.94
	Quip	1.38	802.84	893.90	460.81	459.61
	gzip	1.00	**109.88**	**17.68**	**6.87**	**6.87**
	LZMA	0.99	1,261.44	193.97	199.52	31.58
NA12882_S1 (86.25 GB)	Deez	**1.79**	553.17	641.44	2,699.24	5,773.42
	Quip	1.38	665.75	720.99	463.47	459.09
	gzip	1.00	**86.44**	**14.40**	**6.87**	**6.87**
	LZMA	0.99	1,024.85	158.24	199.45	31.58

Table 6. Compression results of reference-based methods.

Dataset	Method	Metric				
		C-Ratio [a]	C-Time (min)	D-Time (min)	C-Mem (MB)	D-Mem (MB)
HS8 (138 MB) **Ref** [b]**:** HSCHM8	iDoComp [75]	**241.77**	9.95	28.48	**787.58**	787.58
	GReEn [60]	102.98	242.99	247.32	1,124.38	1,110.83
	GeCo [50]	70.65	693.67	579.50	4,897.60	4,897.65
	GDC 2 [63]	239.72	**2.89**	**0.07**	1,513.25	**166.84**
	ERGC [73]	4.47	3.44	0.22	12,256.34	12,256.34
HS11 (128 MB) **Ref:** HSCHM11	iDoComp	104.08	12.36	17.72	**758.23**	693.89
	GReEn	60.74	200.41	233.40	1,061.21	1,042.00
	GeCo	67.81	517.44	509.67	4,897.79	4,897.73
	GDC 2	**110.97**	**2.78**	**0.06**	1,415.80	**111.89**
	ERGC	4.30	4.16	0.22	11,855.81	11,855.81
HS11 (128 MB) **Ref:** PT11	iDoComp	**29.19**	29.06	19.15	**781.27**	781.27
	GReEn	11.99	252.55	279.45	1,078.43	1,062.30
	GeCo	24.77	507.39	520.43	4 897.84	4,897.77
	GDC 2	27.49	4.65	**0.08**	1,434.61	**118.70**
	ERGC	4.30	**4.35**	0.22	11,853.75	11,853.75
HS11 (128 MB) **Ref:** PA11	iDoComp	6.44	130.77	96.27	1,128.72	1,128.72
	GReEn	4.21	323.82	278.12	**1,065.21**	1,060.59
	GeCo	**12.34**	517.65	524.30	4,897.84	4,897.77
	GDC 2	6.21	18.45	**0.16**	1,459.12	**310.48**
	ERGC	4.30	**4.16**	0.22	11,854.96	11,854.96
HSK16 (71 MB) **Ref:** HS16	iDoComp	173.93	5.56	10.66	**461.78**	461.78
	GReEn	45.88	104.35	113.94	765.93	778.14
	GeCo	45.37	362.98	296.28	4,897.79	4,897.73
	GDC 2	**193.34**	2.23	**0.04**	963.97	**83.57**
	ERGC	9.16	**1.60**	0.06	11,304.26	11,304.26
RICE5 (166 MB) **Ref:** RICE7	iDoComp	**204.38**	28.22	34.50	**950.05**	950.05
	GReEn	169.16	204.66	221.81	1,286.32	1,285.14
	GeCo	91.68	616.70	628.95	4,897.79	4,897.73
	GDC 2	199.28	**1.73**	**0.08**	1,712.41	**116.05**
	ERGC	4.44	5.36	0.27	11,789.49	11,789.49

[a] The size of the reference sequences has not been included in calculation of the compression ratio (C-Ratio); [b] **Ref:** Reference sequence.

In Table 4, the FASTQ file compressors, including Fqzcomp [116], Quip [108], DSRC [54], SCALCE [109], gzip [135] and LZMA [86], have been compared. Based on this table, although the SCALCE method outperforms the other methods in terms of compression ratio, making use of a non-reversible reordering of the reads, the Fqzcomp method is the best approach in terms of compression time. Also, gzip, which is a general-purpose method, has better performance rather than the other methods in terms of decompression time as well as compression/decompression memory usage.

The compression results of genomic sequences in BAM format is reported in Table 5. This table shows that the Deez [134] approach provides the best results in terms of compression ratio. However, in terms of other compression metrics, including compression/decompression time and compression/decompression memory usage, gzip performs better in comparison to the other methods. It is worth pointing out that the general-purpose algorithms in this table, i.e., gzip and LZMA, are unable to compress BAM files, since the representation of BAM files is already in binary and the methods cannot extract redundancy from the data.

In Table 6, reference-based genome sequence compressors have been compared. Based on this table, the best result, in terms of compression ratio, is not provided by a single approach. iDoComp [75] outperforms the other methods, including GReEn [60], GeCo [50] and GDC 2 [63] in three cases: compressing HS8 with reference HSCHM8, HS11 with reference PT11, and RICE5 with reference RICE7. Also, GDC 2 provides better results for the compression of HS11 with reference HSCHM11, as well as HSK16 with reference HS16. Finally, GeCo outperforms the others for compressing HS11 with reference PA11. In terms of compression/decompression time as well as decompression memory usage, the GDC 2 approach has the best performance, except for three cases. Also, in terms of compression memory usage, the iDoComp approach always performs better than the other approaches, except for one case. It is worth noting that, as it is clearly shown in the table, GDC 2 has a prominent advantage over the other reference-based methods, that is carrying out the compression and the decompression of sequences in far less time.

6. Discussion

The importance of biological data to the future of mankind is unquestionable. It is now inevitable that huge amounts of data will be produced and will have to flow swiftly, in order to support the deployment of the personalized and precision medicine paradigms. No doubt, data compression is a key enabling technology to achieve this goal. A clear manifestation of the importance that is been given to this subject is the recent interest of the working group of ISO/IEC, MPEG (Moving Picture Experts Group), in starting standardization efforts in genetic information processing, particularly compression [3].

Since its inception in the beginning of the 1990s, many algorithms have been proposed for the compression of the several variants of biological data sequences. However, often we verify that it is straightforward to encounter datasets in which a given method excels, but it is also not too much difficult to find out some other datasets for which the same method struggles to provide reasonable results. Moreover, frequently, datasets less favorable to a certain algorithm are successfully compressed by some other approach. This shows how rich and diverse are the data produced in this domain and also that most techniques are still targeting only subsets of the statistical and algorithmic characteristics of the data. The good news is that, as more different approaches are proposed, the union of these subsets of statistical and algorithmic characteristics gets larger. On the other hand, it also suggests that, in order to have compression tools that are effective in a greater number of datasets, multiple techniques need to be combined appropriately.

The primary goal of most researchers working in this field has been to develop good models for compressing the data, without giving much attention to problems such as fast random access to the compressed data, without considering the possibility of operating on average hardware specs or without taking advantage of relatively accessible massively parallel hardware (e.g., GPUs or FPGAs). Although there is no doubt that it is fundamental to proceed research towards better data compression

models—and there is still much to be done in this front—it is also paramount to address all the other relevant aspects. The effort that the MPEG group is willing to put here may be the glue that is missing. This should be the opportunity to congregate efforts and to build a widely accepted standard that will surely impact the future of genomics. Similarly to what has been happening in the field of video compression, on which the MPEG group has played a major role in producing a standard incorporating many diverse techniques, we foresee that a winning standard for the compression of genomic data will certainly integrate many of the ideas that have been put forward for the last 20 years.

Besides the goal of reducing storage needs, compression-based approaches also play other key roles in the genomics field. Their redundancy extraction characteristic, allows unveiling singularities or working as compact structures. Several application fields have successfully used data compression. For example, genotyping, where the main objective is to detect relative singularity, or better, differences between individuals. For an application, see [159]. Also, retrieving Single Nucleotide Polymorphisms (SNP) data—detecting and localizing one base mutations on genomic data [160]. Other application is aligning DNA sequences using compression [161]; Using data compression to detect large transformations between the DNA of different individuals or species, also known as rearrangement detection, has also been shown to work efficiently [162]; It has also been used for efficient storage of data structures in pan-genome analysis, namely using de Bruijn graphs [163,164]. Here, the problem is to deal with large amounts of information and its fast retrieval. Finally, is has been used for measuring complexity of sequences [165] and searching in string collections [166].

7. Conclusions

To overcome the challenges of storage, processing and transmission of biological data, that arise from the increasing data production of high-throughput sequencing technologies, compression methods need to be exploited. In this paper, methods concerning the compression of protein and DNA sequences were reviewed. Moreover, different approaches proposed for compressing FASTA, FASTQ and SAM/BAM file formats were discussed. Finally, several compression methods were executed on the same server machine and their performance was compared, in terms of compression ratio, compression/decompression CPU time and compression/decompression memory usage.

Based on the obtained results, it can be concluded that currently available methods for biological sequence compression do not allow a one-size-fits-all approach, i.e., a single compression method is not able to provide the best results in terms of all the parameters evaluated, such as compression ratio, compression/decompression time or compression/decompression required memory. Nevertheless, some of the methods are in a level of development that permit already use them as practical and effective tools.

In the case of FASTA and Multi-FASTA data, it is clear that special purpose methods, such as `DELIMINATE` [87] and `MFCompress` [90] provide an effective increase in compression ratio, although, as expected, at the cost of some additional computational resources. Also, although both `DELIMINATE` and `MFCompress` seem to provide equivalent performance—usually, `DELIMINATE` is faster and uses less memory, but does not compress as good as `MFCompress`—the choice of MFCompress to integrate a recent pipeline (`MetaCRAM` [93]) suggests that `MFCompress` is the current best choice for FASTA and Multi-FASTA data compression.

Regarding data in FASTQ format, if the original order of the reads is not important, then `SCALCE` [109] clearly seems to be the choice—read reordering is, in fact, a source of improvement in compression. In the case the original order needs to be preserved, then about $\log_2 n! \approx n \log_2 n - n \log_2(e)$ additional bits are required for recovering the initial ordering of n reads. However, if the aim is to always have the original file after decompression, including read ordering, then `Fqzcomp` [116] is the tool that currently offers the best performance, not only in terms of compression ratio, but also in terms of used resources. As in other cases, `gzip` [135] is the fastest method to decompress and also the one requiring less memory to run, but it falls short in terms of

compression ratio. On the other hand, `LZMA` [86] is competitive in terms of compression ratio, but at the cost of very high compression times.

In what concerns the compression of BAM files, the `Deez` tool seems to be, currently, the best choice. Although it requires some more memory than `Quip`, it provides a considerable better compression rate for equivalent CPU times. Both `gzip` and `LZMA` fall short regarding compression ratio.

In the case of reference-based methods, probably the most striking observation is the dramatic variation of the compression ratio from dataset to dataset. Here, the performance of a method depends both on the target to compress and the reference used. Particularly, the "proximity" of the target to the reference seems to be a key factor. For example, for higher "proximity", both `iDoComp` [75] and `GDC 2` [63] offer similar compression ratios, with `GDC 2` providing very fast decompression. However, when the "proximity" between the target and the reference is lower, we see `GeCo` providing about twice as much compression ratio. One of the challenges here seems to be that it is not always possible to know beforehand the degree of "proximity" between a certain target and reference. In fact, in a certain sense, reference-based compression provides an estimate of the degree of "proximity" between both sequences, which is only available after compression is performed.

Acknowledgments: We would like to thank the FCT—Foundation for Science and Technology in Portugal, for their support of this research, within the Doctoral Programme FCT MAP-i in Computer Science, and also acknowledge european funds through FEDER, under the COMPETE 2020 and Portugal 2020 programs, in the context of the projects UID/CEC/00127/2013 and PTDC/EEI-SII/6608/2014.

Author Contributions: Morteza Hosseini, Diogo Pratas and Armando J. Pinho conceived and designed the experiments; Morteza Hosseini and Diogo Pratas performed the experiments; Morteza Hosseini, Diogo Pratas and Armando J. Pinho analyzed the data; Morteza Hosseini, Diogo Pratas and Armando J. Pinho wrote the paper. All authors have read and approved the final manuscript.

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

References

1. Muir, P.; Li, S.; Lou, S.; Wang, D.; Spakowicz, D.J.; Salichos, L.; Zhang, J.; Weinstock, G.M.; Isaacs, F.; Rozowsky, J.; et al. The real cost of sequencing: Scaling computation to keep pace with data generation. *Genom. Biol.* **2016**, doi:10.1186/s13059-016-0917-0.
2. Kahn, S. On the future of genomic data. *Science* **2011**, *331*, 728–729.
3. Alberti, C.; Mattavelli, M.; Hernandez, A.; Chiariglione, L.; Xenarios, I.; Guex, N.; Stockinger, H.; Schuepbach, T.; Kahlem, P.; Iseli, C.; et al. *Investigation on Genomic Information Compression and Storage*; ISO/IEC JTC 1/SC 29/WG 11 N15346; ISO: Geneva, Switzerland, 2015; pp. 1–28.
4. Giancarlo, R.; Rombo, S.; Utro, F. Compressive biological sequence analysis and archival in the era of high-throughput sequencing technologies. *Brief. Bioinform.* **2014**, *15*, 390–406.
5. De Bruijn, N. A Combinatorial Problem. Available online: https://pure.tue.nl/ws/files/4442708/597473.pdf (accessed on 11 October 2016).
6. Compeau, P.; Pevzner, P.; Tesler, G. How to apply de Bruijn graphs to genome assembly. *Nat. Methods* **2011**, *29*, 987–991.
7. Conway, T.; Bromage, A. Succinct data structures for assembling. *Bioinformatics* **2011**, *27*, 479–486.
8. Cao, M.; Dix, T.; Allison, L. A genome alignment algorithm based on compression. *BMC Bioinform.* **2010**, *11*, doi:10.1186/1471-2105-11-599.
9. Cao, M.; Dix, T.; Allison, L.; Mears, C. A simple statistical algorithm for biological sequence compression. In Proceedings of the DCC '07: Data Compression Conference, Snowbird, UT, USA, 27–29 March 2007; pp. 43–52.
10. Rafizul Haque, S.; Mallick, T.; Kabir, I. A new approach of protein sequence compression using repeat reduction and ASCII replacement. *IOSR J. Comput. Eng. (IOSR-JCE)* **2013**, *10*, 46–51.
11. Ward, M. *Virtual Organisms: The Startling World of Artificial Life*; Macmillan: New York, NY, USA, 2014.
12. Wootton, J. Non-globular domains in protein sequences: Automated segmentation using complexity measures. *Comput. Chem.* **1994**, *18*, 269–285.

13. Benedetto, D.; Caglioti, E.; Chica, C. Compressing proteomes: The relevance of medium range correlations. *EURASIP J. Bioinform. Syst. Biol.* **2007**, *2007*, doi:10.1155/2007/60723.
14. Yu, J.; Cao, Z.; Yang, Y.; Wang, C.; Su, Z.; Zhao, Y.; Wang, J.; Zhou, Y. Natural protein sequences are more intrinsically disordered than random sequences. *Cell. Mol. Life Sci.* **2016**, *73*, 2949–2957.
15. The Human Proteome Project. Available online: http://www.thehpp.org (accessed on 11 October 2016).
16. Three sequenced Neanderthal genomes. Available online: http://cdna.eva.mpg.de/neandertal (accessed on 11 October 2016).
17. Nevill-Manning, C.; Witten, I. Protein is incompressible. In Proceedings of the DCC '99: Data Compression Conference, Snowbird, UT, USA, 29–31 March 1999; pp. 257–266.
18. Matsumoto, T.; Sadakane, K.; Imai, H. Biological sequence compression algorithms. *Genom. Inform.* **2000**, *11*, 43–52.
19. Hategan, A.; Tabus, I. Protein is compressible. In Proceedings of the 6th Nordic Signal Processing Symposium, Espoo, Finland, 9–11 June 2004; pp. 192–195.
20. Willems, F.; Shtarkov, Y.; Tjalkens, T. The context tree weighting method: Basic properties. *IEEE Trans. Inf. Theory* **1995**, *41*, 653–664.
21. Hategan, A.; Tabus, I. Jointly encoding protein sequences and their secondary structure. In Proceedings of the IEEE International Workshop on Genomic Signal Processing and Statistics (GENSIPS 2007), Tuusula, Finland, 10–12 June 2007.
22. Daniels, N.; Gallant, A.; Peng, J.; Cowen, L.; Baym, M.; Berger, B. Compressive genomics for protein databases. *Bioinformatics* **2013**, *29*, 283–290.
23. Kabsch, W.; Sander, C. Dictionary of protein secondary structure: Pattern recognition of hydrogen-bonded and geometrical features. *Biopolymers* **1983**, *22*, 2577–2637.
24. Hayashida, M.; Ruan, P.; Akutsu, T. Proteome compression via protein domain compositions. *Methods* **2014**, *67*, 380–385.
25. Giancarlo, R.; Scaturro, D.; Utro, F. Textual data compression in computational biology: Algorithmic techniques. *Comput. Sci. Rev.* **2012**, *6*, 1–25.
26. Zhu, Z.; Zhang, Y.; Ji, Z.; He, S.; Yang, X. High-throughput DNA sequence data compression. *Brief. Bioinform.* **2013**, *16*, doi:10.1093/bib/bbt087.
27. Bakr, N.; Sharawi, A. DNA lossless compression algorithms: Review. *Am. J. Bioinform. Res.* **2013**, *3*, 72–81.
28. Wandelt, S.; Bux, M.; Leser, U. Trends in genome compression. *Curr. Bioinform.* **2014**, *9*, 315–326.
29. Grumbach, S.; Tahi, F. Compression of DNA sequences. In Proceedings of the DCC'93: Data Compression Conference, Snowbird, UT, USA, 30 March–2 April 1993; pp. 340–350.
30. Ziv, J.; Lempel, A. A universal algorithm for sequential data compression. *IEEE Trans. Inf. Theory* **1977**, *23*, 337–343.
31. Grumbach, S.; Tahi, F. A new challenge for compression algorithms: Genetic sequences. *Inf. Process. Manag.* **1994**, *30*, 875–886.
32. Rivals, E.; Delahaye, J.; Dauchet, M.; Delgrange, O. A guaranteed compression scheme for repetitive DNA sequences. In Proceedings of the DCC '96: Data Compression Conference, Snowbird, UT, USA, 31 March–3 April 1996; p. 453.
33. Ukkonen, E. On-line construction of suffix trees. *Algorithmica* **1995**, *14*, 249–260.
34. Chen, X.; Kwong, S.; Li, M.; Delgrange, O. A compression algorithm for DNA sequences and its applications in genome comparison. In Proceedings of the 4th Annual International Conference of Research in Computational Molecular Biology (RECOMB '00), Tokyo, Japan, 8–11 April 2000; pp. 107–117.
35. Chen, X.; Li, M.; Ma, B. DNACompress: Fast and effective DNA sequence. *Bioinformatics* **2002**, *18*, 1696–1698.
36. Ma, B.; Tromp, J.; Li, M. PatternHunter: Faster and more sensitive homology search. *Bioinformatics* **2002**, *18*, 440–445.
37. Tabus, I.; Korodi, G.; Rissanen, J. DNA sequence compression using the normalized maximum likelihood model for discrete regression. In Proceedings of the DCC '03: Data Compression Conference, Snowbird, UT, USA, 25–27 March 2003; pp. 253–262.
38. Korodi, G.; Tabus, I. An efficient normalized maximum likelihood algorithm for DNA sequence compression. *ACM Trans. Inf. Syst.* **2005**, *23*, 3–34.
39. Gupta, A.; Agarwal, S. A scheme that facilitates searching and partial decompression of textual documents. *Int. J. Adv. Comput. Eng.* **2008**, *1*, 99–109.

40. Gupta, A.; Agarwal, S. A novel approach for compressing DNA sequences using semi-statistical compressor. *Int. J. Comput. Appl.* **2011**, *33*, 245–251.
41. Pinho, A.; Ferreira, P.; Neves, A.; Bastos, C. On the representability of complete genomes by multiple competing finite-context (Markov) models. *PLoS ONE* **2011**, *6*, e21588.
42. Zhu, Z.; Zhou, J.; Ji, Z.; Shi, Y. DNA sequence compression using adaptive particle swarm optimization-based memetic algorithm. *IEEE Trans. Evolut. Comput.* **2011**, *15*, 643–658.
43. Liang, J.; Suganthan, P.; Deb, K. Novel composition test functions for numerical global optimization. In Proceedings of the IEEE Swarm Intelligence Symposium (SIS 2005), Pasadena, CA, USA, 8–10 June 2005; pp. 68–75.
44. Liang, J.; Qin, A.; Suganthan, P.; Baskar, S. Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. *IEEE Trans. Evolut. Comput.* **2006**, *10*, 281–295.
45. Li, P.; Wang, S.; Kim, J.; Xiong, H.; Ohno-Machado, L.; Jiang, X. DNA-COMPACT: DNA compression based on a pattern-aware contextual modeling technique. *PLoS ONE* **2013**, *8*, e80377.
46. Guo, H.; Chen, M.; Liu, X.; Xie, M. Genome compression based on Hilbert space filling curve. In Proceedings of the 3rd International Conference on Management, Education, Information and Control (MEICI 2015), Shenyang, China, 29–31 May 2015; pp. 1685–1689.
47. Xie, X.; Zhou, S.; Guan, J. CoGI: Towards compressing genomes as an image. *IEEE/ACM Trans. Comput. Biol. Bioinform.* **2015**, *12*, 1275–1285.
48. Mohamed, S.; Fahmy, M. Binary image compression using efficient partitioning into rectangular regions. *IEEE Trans. Commun.* **1995**, *43*, 1888–1893.
49. Chen, M.; Chen, J.; Zhang, Y.; Tang, M. Optimized context weighting based on the least square algorithm. In *Wireless Communications, Networking and Applications*, Proceedings of the 2014 International Conference on Wireless Communications, Networking and Applications (WCNA 2014), Shenzhen, China, 27–28 December 2014; Zeng, Q.-A., Ed.; Lecture Notes in Electrical Engineering; Springer: Berlin/Heidelberg, Germany, 2016; Volume 348, pp. 1037–1045.
50. Pratas, D.; Pinho, A.; Ferreira, P. Efficient compression of genomic sequences. In Proceedings of the DCC '16: Data Compression Conference, Snowbird, UT, USA, 30 March–1 April 2016; pp. 231–240.
51. Pinho, A.J.; Pratas, D.; Ferreira, P.J. Bacteria DNA sequence compression using a mixture of finite-context models. In Proceedings of the 2011 IEEE Statistical Signal Processing Workshop (SSP), Nice, France, 28–30 June 2011; pp. 125–128.
52. Pratas, D.; Pinho, A.J. Exploring deep Markov models in genomic data compression using sequence pre-analysis. In Proceedings of the 2014 22nd European Signal Processing Conference (EUSIPCO), Lisbon, Portugal, 1–5 September 2014; pp. 2395–2399.
53. Wandelt, S.; Leser, U. Adaptive efficient compression of genomes. *Algorithms Mol. Biol.* **2012**, *7*, doi:10.1186/1748-7188-7-30.
54. Deorowicz, S.; Grabowski, S. Compression of DNA sequence reads in FASTQ format. *Bioinformatics* **2011**, *27*, 860–862.
55. Christley, S.; Lu, Y.; Li, C.; Xie, X. Human genomes as email attachments. *Bioinformatics* **2009**, *25*, 274–275.
56. Kuruppu, S.; Puglisi, S.; Zobel, J. Relative Lempel-Ziv compression of genomes for large-scale storage and retrieval. *String Process. Inf. Retr.* **2010**, *6393*, 201–206.
57. Kuruppu, S.; Puglisi, S.; Zobel, J. Optimized relative Lempel-Ziv compression of genomes. *Conf. Res. Pract. Inf. Technol. Ser.* **2011**, *113*, 91–98.
58. Wang, C.; Zhang, D. A novel compression tool for efficient storage of genome resequencing data. *Nucleic Acids Res.* **2011**, *39*, 5–10.
59. Huffman, D. A method for the construction of minimum redundancy codes. *Proc. IRE* **1952**, *40*, 1098–1101.
60. Pinho, A.; Pratas, D.; Garcia, S. GReEn: A tool for efficient compression of genome resequencing data. *Nucleic Acids Res.* **2012**, *40*, doi:10.1093/nar/gkr1124.
61. Rissanen, J. Generalized Kraft inequality and arithmetic coding. *IBM J. Res. Dev.* **1976**, *20*, 198–203.
62. Deorowicz, S.; Grabowski, S. Robust relative compression of genomes with random access. *Bioinformatics* **2011**, *27*, 2979–2986.
63. Deorowicz, S.; Danek, A.; Niemiec, M. GDC 2: Compression of large collections of genomes. *Sci. Rep.* **2015**, *5*, doi:10.1038/srep11565.

64. Storer, J.; Szymanski, T. Data compression via text substitution. *J. ACM* **1982**, *29*, 928–951.
65. Grossi, R.; Vitter, J. Compressed suffix arrays and suffix trees with applications to text indexing and string matching. In Proceedings of the 32nd ACM Symposium on Theory of Computing, Portland, OR, USA, 21–23 May 2000; pp. 397–406.
66. Kuruppu, S.; Beresford-Smith, B.; Conway, T.; Zobel, J. Iterative dictionary construction for compression of large DNA data sets. *IEEE/ACM Trans. Comput. Biol. Bioinform.* **2012**, *9*, 137–149.
67. Cannane, A.; Williams, H. General-purpose compression for efficient retrieval. *J. Assoc. Inf. Sci. Technol.* **2001**, *52*, 430–437.
68. Dai, W.; Xiong, H.; Jiang, X.; Ohno-Machado, L. An adaptive difference distribution-based coding with hierarchical tree structure for DNA sequence compression. In Proceedings of the DCC '13: Data Compression Conference, Snowbird, UT, USA, 20–22 March 2013; pp. 371–380.
69. Cleary, J.; Witten, I. Data compression using adaptive coding and partial string matching. *IEEE Trans. Commun.* **1984**, *32*, 396–402.
70. Wandelt, S.; Leser, U. FRESCO: Referential compression of highly-similar sequences. *IEEE/ACM Trans. Comput. Biol. Bioinform.* **2013**, *10*, 1275–1288.
71. Jung, S.; Sohn, I. Streamlined genome sequence compression using distributed source coding. *Cancer Inform.* **2014**, *13*, 35–43.
72. Pradhan, S.; Ramchandran, K. Distributed source coding using syndromes (DISCUS): Design and construction. *IEEE Trans. Inf. Theory* **2003**, *49*, 626–643.
73. Saha, S.; Rajasekaran, S. ERGC: An efficient referential genome compression algorithm. *Bioinformatics* **2015**, *31*, 3468–3475.
74. Moffat, A. Implementing the PPM data compression scheme. *IEEE Trans. Commun.* **1990**, *38*, 1917–1921.
75. Ochoa, I.; Hernaez, M.; Weissman, T. iDoComp: A compression scheme for assembled genomes. *Bioinformatics* **2015**, *31*, 626–633.
76. Costa, M.; Goldberger, A.; Peng, C. Multiscale entropy analysis of complex physiologic time series. *Phys. Rev. Lett.* **2002**, *89*, 068102.
77. Richman, J.; Moorman, J. Physiological time-series analysis using approximate entropy and sample entropy. *Am. J. Physiol. Heart Circ. Physiol.* **2000**, *278*, 2039–2049.
78. Cosic, I. Macromolecular bioactivity: Is it resonant interaction between macromolecules?—Theory and applications. *IEEE Trans. Biomed. Eng.* **1994**, *41*, 1101–1114.
79. Hanus, P.; Dingel, J.; Chalkidis, G.; Hagenauer, J. Compression of whole genome alignments. *IEEE Trans. Inf. Theory* **2010**, *56*, 696–705.
80. Matos, L.; Pratas, D.; Pinho, A. A compression model for DNA multiple sequence alignment blocks. *IEEE Trans. Inf. Theory* **2013**, *59*, 3189–3198.
81. Matos, L.; Neves, A.; Pratas, D.; Pinho, A. MAFCO: A compression tool for MAF files. *PLoS ONE* **2015**, *10*, e0116082.
82. Danecek, P.; Auton, A.; Abecasis, G.; Albers, C.A.; Banks, E.; DePristo, M.A.; Handsaker, R.E.; Lunter, G.; Marth, G.T.; Sherry, S.T.; et al. The variant call format and VCFtools. *Bioinformatics* **2011**, *27*, 2156–2158.
83. Layer, R.M.; Kindlon, N.; Karczewski, K.J.; Quinlan, A.R.; Consortium, E.A.; Exome Aggregation Consortium. Efficient genotype compression and analysis of large genetic-variation data sets. *Nat. Methods* **2016**, *13*, 63–65.
84. Lipman, D.; Pearson, W. Rapid and sensitive protein similarity searches. *Brief. Bioinform.* **1985**, *227*, 1435–1441.
85. Bose, T.; Mohammed, M.; Dutta, A.; Mande, S. BIND—An algorithm for loss-less compression of nucleotide sequence data. *J. Biosci.* **2012**, *37*, 785–789.
86. `LZMA`. Available online: http://www.7-zip.org/sdk.html (accessed on 11 October 2016).
87. Mohammed, M.; Dutta, A.; Bose, T.; Chadaram, S.; Mande, S. DELIMINATE—A fast and efficient method for loss-less compression of genomic sequences: Sequence analysis. *Bioinformatics* **2012**, *28*, 2527–2529.
88. Chen, W.; Lu, Y.; Lai, F.; Chien, Y.; Hwu, W. Integrating human genome database into electronic health record with sequence alignment and compression mechanism. *J. Med. Syst.* **2012**, *36*, 2587–2597.
89. Apostolico, A.; Fraenkel, A. Robust transmission of unbounded strings using Fibonacci representations. *IEEE Trans. Inf. Theory* **1987**, *33*, 238–245.
90. Pinho, A.; Pratas, D. MFCompress: A compression tool for FASTA and multi-FASTA data. *Bioinformatics* **2013**, *30*, 117–118.

91. Benoit, G.; Lemaitre, C.; Lavenier, D.; Drezen, E.; Dayris, T.; Uricaru, R.; Rizk, G. Reference-free compression of high throughput sequencing data with a probabilistic de Bruijn graph. *BMC Bioinform.* **2015**, *16*, doi:10.1186/s12859-015-0709-7.
92. Kirsch, A.; Mitzenmacher, M. Less hashing, same performance: Building a better bloom filter. *J. Random Struct. Algorithms* **2008**, *33*, 187–218.
93. Kim, M.; Zhang, X.; Ligo, J.G.; Farnoud, F.; Veeravalli, V.V.; Milenkovic, O. MetaCRAM: An integrated pipeline for metagenomic taxonomy identification and compression. *BMC Bioinform.* **2016**, *17*, doi:10.1186/s12859-016-0932-x.
94. Wood, D.; Salzberg, S. Kraken: Ultrafast metagenomic sequence classification using exact alignments. *Genom. Biol.* **2014**, *15*, doi:10.1186/gb-2014-15-3-r46.
95. Langmead, B.; Salzberg, S. Fast gapped-read alignment with bowtie 2. *Nat. Methods* **2012**, *9*, 357–359.
96. Peng, Y.; Leung, H.C.; Yiu, S.M.; Chin, F.Y. IDBA-UD: A de novo assembler for single-cell and metagenomic sequencing data with highly uneven depth. *Bioinformatics* **2012**, *28*, 1420–1428.
97. Golomb, S. Run-length encodings. *IEEE Trans. Inf. Theory* **1966**, *12*, 399–401.
98. Somasundaram, K.; Domnic, S. Extended golomb code for integer representation. *IEEE Trans. Multimed.* **2007**, *9*, 239–246.
99. Ochoa, I.; Asnani, H.; Bharadia, D.; Chowdhury, M.; Weissman, T.; Yona, G. Qualcomp: A new lossy compressor for quality scores based on rate distortion theory. *BMC Bioinform.* **2013**, *14*, doi:10.1186/1471-2105-14-187.
100. Cock, P.; Fields, C.; Goto, N.; Heuer, M.; Rice, P. The Sanger FASTQ file format for sequences with quality scores, and the Solexa/Illumina FASTQ variants. *Nucleic Acids Res.* **2009**, *38*, 1767–1771.
101. Daily, K.; Rigor, P.; Christley, S.; Xie, X.; Baldi, P. Data structures and compression algorithms for high-throughput sequencing technologies. *BMC Bioinform.* **2010**, *11*, doi:10.1186/1471-2105-11-514.
102. Elias, P. Universal codeword sets and representations of the integers. *IEEE Trans. Inf. Theory* **1975**, *21*, 194–203.
103. Baldi, P.; Benz, R.; Hirschberg, D.; Swamidass, S. Lossless compression of chemical fingerprints using integer entropy codes improves storage and retrieval. *J. Chem. Inf. Model.* **2007**, *47*, 2098–2109.
104. Tembe, W.; Lowey, J.; Suh, E. G-SQZ: Compact encoding of genomic sequence and quality data. *Bioinformatics* **2010**, *26*, 2192–2194.
105. Roguski, L.; Deorowicz, S. DSRC 2—Industry-oriented compression of FASTQ files. *Bioinformatics* **2014**, *30*, 2213–2215.
106. Salomon, D.; Motta, G. *Handbook of Data Compression*; Springer: Berlin, Germany, 2010.
107. Bhola, V.; Bopardikar, A.; Narayanan, R.; Lee, K.; Ahn, T. No-reference compression of genomic data stored in FASTQ format. In Proceedings of the IEEE International Conference on Bioinformatics and Biomedicine (BIBM 2011), Atlanta, GA, USA, 12–15 November 2011; pp. 147–150.
108. Jones, D.; Ruzzo, W.; Peng, X.; Katze, M. Compression of next-generation sequencing reads aided by highly efficient de novo assembly. *Nucleic Acids Res.* **2012**, *40*, doi:10.1093/nar/gks754.
109. Hach, F.; Numanagic, I.; Alkan, C.; Sahinalp, S. SCALCE: Boosting sequence compression algorithms using locally consistent encoding. *Bioinformatics* **2012**, *28*, 3051–3057.
110. Sahinalp, S.; Vishkin, U. Efficient approximate and dynamic matching of patterns using a labeling paradigm. In Proceedings of the 37th Annual Symposium on Foundations of Computer Science (FOCS), Burlington, VT, USA, 14–16 October 1996; pp. 320–328.
111. Cormode, G.; Paterson, M.; Sahinalp, S.; Vishkin, U. Communication complexity of document exchange. In Proceedings of the 11th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), San Francisco, CA, USA, 9–11 January 2000; pp. 197–206.
112. Batu, T.; Ergun, F.; Sahinalp, S. Oblivious string embeddings and edit distance approximations. In Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithm (SODA), Miami, FL, USA, 22–24 January 2006; pp. 792–801.
113. Howison, M. High-throughput compression of FASTQ data with SeqDB. *IEEE/ACM Trans. Comput. Biol. Bioinform.* **2013**, *10*, 213–218.
114. Alted, F. Available online: http://www.blosc.org (accessed on 11 October 2016).

115. Alted, F. Why modern CPUs are starving and what can be done about it. *Comput. Sci. Eng.* **2010**, *12*, 68–71.
116. Bonfield, J.; Mahoney, M. Compression of FASTQ and SAM format sequencing data. *PLoS ONE* **2013**, *8*, e59190.
117. Shelwien, E. Available online: http://compressionratings.com/i_ctxf.html (accessed on 11 October 2016).
118. Mahoney, M. Available online: http://mattmahoney.net/dc/zpaq.html (accessed on 11 October 2016).
119. Mahoney, M. *Adaptive Weighing of Context Models for Lossless Data Compression*; Technical Report CS-2005–16; Florida Institute of Technology CS Department: Melbourne, FL, USA, 2005.
120. Grabowski, S.; Deorowicz, S.; Roguski, L. Disk-based compression of data from genome sequencing. *Bioinformatics* **2014**, *31*, 1389–1395.
121. Roberts, M.; Hayes, W.; Hunt, B.; Mount, S.; Yorke, J. Reducing storage requirements for biological sequence comparison. *Bioinformatics* **2004**, *20*, 3363–3369.
122. Movahedi, N.; Forouzmand, E.; Chitsaz, H. De novo co-assembly of bacterial genomes from multiple single cells. In Proceedings of the IEEE International Conference on Bioinformatics and Biomedicine (BIBM 2012), Philadelphia, PA, USA, 4–7 October 2012; pp. 1–5.
123. Li, Y.; Kamousi, P.; Han, F.; Yang, S.; Yan, X.; Suri, S. Memory efficient minimum substring partitioning. In Proceedings of the 39th international conference on Very Large Data Bases (VLDB 2013), Trento, Italy, 26–30 August 2013; pp. 169–180.
124. Chikhi, R.; Limasset, A.; Jackman, S.; Simpson, J.; Medvedev, P. On the representation of de Bruijn graphs. In Proceedings of the 18th Annual International Conference on Research in Computational Molecular Biology (RECOMB 2014), Pittsburgh, PA, USA, 2–5 April 2014; pp. 35–55.
125. Deorowicz, S.; Kokot, M.; Grabowski, S.; Debudaj-Grabysz, A. KMC 2: Fast and resource-frugal *k*-mer counting. *Bioinformatics* **2015**, *31*, 1569–1576.
126. Shkarin, D. PPM: One step to practicality. In Proceedings of the DCC '02: Data Compression Conference, Snowbird, UT, USA, 2–4 April 2002; pp. 202–211.
127. Zhang, Y.; Li, L.; Yang, Y.; Yang, X.; He, S. Light-weight reference-based compression of FASTQ data. *BMC Bioinform.* **2015**, *16*, 188.
128. Li, H.; Handsaker, B.; Wysoker, A.; Fennell, T.; Ruan, J.; Homer, N.; Marth, G.; Abecasis, G.; Durbin, R. The Sequence Alignment/Map format and SAMtools. *Bioinformatics* **2009**, *25*, 2078–2079.
129. The SAM/BAM Format Specification Working Group. Sequence Alignment/Map Format Specification. Available online: https://samtools.github.io/hts-specs/SAMv1.pdf (accessed on 11 October 2016).
130. Fritz, M.; Leinonen, R.; Cochrane, G.; Birney, E. Efficient storage of high throughput DNA sequencing data using reference-based compression. *Genom. Res.* **2011**, *21*, 734–740.
131. Campagne, F.; Dorff, K.; Chambwe, N.; Robinson, J.; Mesirov, J. Compression of structured high-throughput sequencing data. *PLoS ONE* **2013**, *8*, e79871.
132. Varda, K. `PB`. Available online: https://github.com/google/protobuf (accessed on 11 October 2016).
133. Popitsch, N.; Von Haeseler, A. NGC: Lossless and lossy compression of aligned high-throughput sequencing data. *Nucleic Acids Res.* **2013**, *41*, doi:10.1093/nar/gks939.
134. Hach, F.; Numanagic, I.; Sahinal, S. DeeZ: Reference-based compression by local assembly. *Nat. Methods* **2014**, *11*, 1081–1082.
135. `gzip`. Available online: http://www.gzip.org (accessed on 11 October 2016).
136. Rebico. Available online: http://bioinformatics.ua.pt/software/rebico (accessed on 11 October 2016).
137. Human (GRC). Available online: ftp://ftp.ncbi.nlm.nih.gov/genomes/Homo_sapiens/Assembled_chromosomes/seq (accessed on 11 October 2016).
138. Chimpanzee. Available online: ftp://ftp.ncbi.nlm.nih.gov/genomes/Pan_troglodytes/Assembled_chromosomes/seq (accessed on 11 October 2016).
139. Rice5. Available online: ftp://ftp.plantbiology.msu.edu/pub/data/Eukaryotic_Projects/o_sativa/annotation_dbs/pseudomolecules/version_5.0 (accessed on 11 October 2016).
140. CAMERA Prokaryotic Nucleotide. Available online: ftp://ftp.imicrobe.us/camera/camera_reference_datasets/10572.V10.fa.gz (accessed on 11 October 2016).
141. ERR174310_1. Available online: ftp://ftp.sra.ebi.ac.uk/vol1/fastq/ERR174/ERR174310/ERR174310_1.fastq.gz (accessed on 11 October 2016).

142. ERR174310_2. Available online: ftp://ftp.sra.ebi.ac.uk/vol1/fastq/ERR174/ERR174310/ERR174310_2.fastq.gz (accessed on 11 October 2016).
143. ERR194146_1. Available online: ftp://ftp.sra.ebi.ac.uk/vol1/fastq/ERR194/ERR194146/ERR194146_1.fastq.gz (accessed on 11 October 2016).
144. ERR194146_2. Available online: ftp://ftp.sra.ebi.ac.uk/vol1/fastq/ERR194/ERR194146/ERR194146_2.fastq.gz (accessed on 11 October 2016).
145. NA12877_S1. Available online: ftp://ftp.sra.ebi.ac.uk/vol1/ERA207/ERA207860/bam/NA12877_S1.bam (accessed on 11 October 2016).
146. NA12878_S1. Available online: ftp://ftp.sra.ebi.ac.uk/vol1/ERA207/ERA207860/bam/NA12878_S1.bam (accessed on 11 October 2016).
147. NA12882_S1. Available online: ftp://ftp.sra.ebi.ac.uk/vol1/ERA207/ERA207860/bam/NA12882_S1.bam (accessed on 11 October 2016).
148. *Homo sapiens*, GRC Reference Assembly—Chromosome 8. Available online: ftp://ftp.ncbi.nlm.nih.gov/genomes/Homo_sapiens/Assembled_chromosomes/seq/hs_ref_GRCh38.p7_chr8.fa.gz (accessed on 11 October 2016).
149. *Homo sapiens*, CHM Reference Assembly—Chromosome 8. Available online: ftp://ftp.ncbi.nlm.nih.gov/genomes/Homo_sapiens/Assembled_chromosomes/seq/hs_alt_CHM1_1.1_chr8.fa.gz (accessed on 11 October 2016).
150. *Homo sapiens*, GRC Reference Assembly—Chromosome 11. Available online: ftp://ftp.ncbi.nlm.nih.gov/genomes/Homo_sapiens/Assembled_chromosomes/seq/hs_ref_GRCh38.p7_chr11.fa.gz (accessed on 11 October 2016).
151. *Homo sapiens*, CHM Reference Assembly—Chromosome 11. Available online: ftp://ftp.ncbi.nlm.nih.gov/genomes/Homo_sapiens/Assembled_chromosomes/seq/hs_alt_CHM1_1.1_chr11.fa.gz (accessed on 11 October 2016).
152. *Pan troglodytes* (Chimpanze) Reference Assembly, v3.0—Chromosome 11. Available online: ftp://ftp.ncbi.nlm.nih.gov/genomes/Pan_troglodytes/Assembled_chromosomes/seq/ptr_ref_Pan_tro_3.0_chr11.fa.gz (accessed on 11 October 2016).
153. *Pongo abelii* (Orangutan) Reference Assembly—Chromosome 11. Available online: ftp://ftp.ncbi.nlm.nih.gov/genomes/Pongo_abelii/Assembled_chromosomes/seq/pab_ref_P_pygmaeus_2.0.2_chr11.fa.gz (accessed on 11 October 2016).
154. *Homo sapiens*, GRC Reference Assembly—Chromosome 16. Available online: ftp://ftp.ncbi.nlm.nih.gov/genomes/Homo_sapiens/Assembled_chromosomes/seq/hs_ref_GRCh38.p7_chr16.fa.gz (accessed on 11 October 2016).
155. *Homo sapiens*, Korean Reference—Chromosome 16. Available online: ftp://ftp.kobic.re.kr/pub/KOBIC-KoreanGenome/fasta/chromosome_16.fa.gz (accessed on 11 October 2016).
156. *Oryza sativa* (Rice), v5.0. Available online: ftp://ftp.plantbiology.msu.edu/pub/data/Eukaryotic_Projects/o_sativa/annotation_dbs/pseudomolecules/version_5.0 (accessed on 11 October 2016).
157. *Oryza sativa* (Rice), v7.0. Available online: ftp://ftp.plantbiology.msu.edu/pub/data/Eukaryotic_Projects/o_sativa/annotation_dbs/pseudomolecules/version_7.0 (accessed on 11 October 2016).
158. Pratas, D. Available online: https://raw.githubusercontent.com/pratas/rebico/master/methods.txt (accessed on 11 October 2016).
159. Li, H. BGT: Efficient and flexible genotype query across many samples. *Bioinformatics* **2015**, doi:10.1093/bioinformatics/btv613.
160. Sambo, F.; Di Camillo, B.; Toffolo, G.; Cobelli, C. Compression and fast retrieval of SNP data. *Bioinformatics* **2014**, *30*, 3078–3085.
161. Cao, M.D.; Dix, T.I.; Allison, L. A genome alignment algorithm based on compression. *BMC bioinform.* **2010**, *11*, doi:10.1186/1471-2105-11-599.
162. Pratas, D.; Silva, R.M.; Pinho, A.J.; Ferreira, P.J. An alignment-free method to find and visualise rearrangements between pairs of DNA sequences. *Sci. Rep.* **2015**, *5*, doi:10.1038/srep10203.
163. Beller, T.; Ohlebusch, E. Efficient construction of a compressed de Bruijn graph for pan-genome analysis. In *Combinatorial Pattern Matching*; Springer: Berlin, Germany, 2015.

164. Baier, U.; Beller, T.; Ohlebusch, E. Graphical pan-genome analysis with compressed suffix trees and the Burrows–Wheeler transform. *Bioinformatics* **2015**, *32*, 497–504.
165. Pinho, A.J.; Garcia, S.P.; Pratas, D.; Ferreira, P.J. DNA sequences at a glance. *PLoS ONE* **2013**, *8*, e79922.
166. Wandelt, S.; Leser, U. MRCSI: Compressing and searching string collections with multiple references. *Proc. VLDB Endow.* **2015**, *8*, 461–472.

© 2016 by the authors; licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC-BY) license (http://creativecommons.org/licenses/by/4.0/).