High-throughput DNA sequence data compression

Zhu, Zexuan; Zhang, Yongpeng; Ji, Zhen; He, Shan; Yang, Xiao

DOI:
10.1093/bib/bbt087

High-Throughput DNA Sequence Data Compression

Zexuan Zhu[1], Yongpeng Zhang[1], Zhen Ji[1], Shan He[2] and Xiao Yang[3]

[1]College of Computer Science and Software Engineering, Shenzhen University, China 518060.

[2]School of Computer Science, University of Birmingham, B15 2TT, UK.

[3]Corresponding author. The Broad Institute, 7 Cambridge Center, Cambridge, MA 02142, USA. Tel: 617 714 7919; E-mail: xiaoyang@broadinstitute.org

Running title: High-Throughput DNA Sequence Data Compression

Keywords: next-generation sequencing; data compression, reference-based compression; reference-free compression

---

Zexuan Zhu is an Associate Professor at the Colloge of Computer Science and Software Engineering, Shenzhen University. His research interests lie in bioinformatics and computational intelligence.
Yongpeng Zhang is a M.Sc student at Shenzhen University. His research interests include data compression technique and evolutionary algorithm.
Zhen Ji is a Professor at the Colloge of Computer Science and Software Engineering, Shenzhen University. His research interests include bioinformatics and signal processing.
Shan He is a Lecturer at the School of Computer Science, the University of Birmingham. His research interests include optimisation, data mining, complex network analysis and their applications to biomedical problems.
Xiao Yang is a computational biologist at the Broad Institute. His current research is focused on high throughput sequencing data analysis with applications on genomics and metagenomics.

Abstract

The exponential growth of high-throughput DNA sequence data has posed great challenges to genomic data storage, retrieval and transmission. Compression is a critical tool to address these challenges, where many methods have been developed to reduce the storage size of the genomes and sequencing data (reads, quality scores and metadata). However, genomic data are being generated faster than they could be meaningfully analysed, leaving a large scope for developing novel compression algorithms that could directly facilitate data analysis beyond data transfer and storage. In this article, we categorize and provide a comprehensive review of the existing compression methods specialized for genomic data and present experimental results on compression ratio, time for compression and de-compression. We further present the remaining challenges and potential directions for future research.

Keywords: next-generation sequencing; compression; reference-based compression; reference-free compression

INTRODUCTION

DNA sequencing has been one of the major driving forces in biomedical research. Sanger sequencing [1], the dominant technology for over three decades, has been gradually replaced by cheaper and higher throughput next-generation sequencing (NGS) technologies in recent years. Examples of such sequencing technologies include pyrosequencing, sequencing by synthesis, and single-molecule sequencing [2-5]. As a result, the growth rate of genomic data in the past few years outpaced the rate predicted by Moore's law and data are generated faster than can be meaningfully analyzed. For example, the NGS data obtained by the 1000 Genomes Project in the first six months exceeded the sequence data in NCBI Genbank database accumulated in the past 21 years.

General-purpose compression algorithms like gzip (http://www.gzip.org/) and bzip2 (http://www.bzip.org), frequently used for efficient text storage and transferring, are directly applicable to genomic sequences. Only when longer genomic sequences were obtained and the storage and transferring of these data becomes routine, specialized compression algorithms [8-19] were developed. Compared to general-purpose text compression methods, these

specialized algorithms better utilize repetitive subsequences inherent to genomic sequences to achieve higher compression ratio. Evolutionarily association among genomic sequences have also been exploited such that a known genomic sequence can serve as a template for efficient compression of another homologous sequence. Before NGS techniques were introduced, compression algorithms were intended for relatively short sequences with an upper bound of tens of Megabases and compression of raw sequencing data were typically not required. NGS techniques have introduced major changes in data analysis needs. More scalable compression algorithms [20, 21, 25, 29-30, 32-38, 47-57, 59, 61] have been introduced to handle much longer genomic sequences as well as raw sequencing data. The latter may consist of millions of short sequences (reads) and quality scores indicating the probabilities of corresponding bases being correctly determined. The distinct properties of genomic sequences and raw sequencing data, such as the differences in number, length and alphabet size, lead to distinct algorithmic strategies for data compression. Nonetheless, these methods can be generally categorized as reference-based or reference-free depending on whether known genomic sequences were used to facilitate compression.

In this article, we discuss the compression of *genomic sequences* -- DNA fragments that correspond to partially or fully assembled chromosomes or genomes, and *raw sequencing data* – reads, quality scores and metadata produced by sequencer. For completeness, we include a brief discussion of traditional genomic sequence compression methods that were developed before NGS techniques were introduced, then focus on more recent methods developed for NGS data and present their performance on read datasets. Finally, we discuss remaining challenges and future perspectives.

TRADITIONAL GENOMIC SEQUENCE COMPRESSION

Although general-purpose compression methods such as gzip and bzip2are widely applied to genomic sequences, they do not take into account biological characteristics such as small alphabet size, rich in repeats and palindromes. Compression algorithms tailored for such properties are more likely to achieve better performance, which we outline in this section methods intended for compressing genomic sequences with relatively small size (e.g. 10s of Megabases) when Sanger was the dominant sequencing method. In general, these methods can be grouped as substitutional or dictionary based and statistical based [8].

In substitutional compression, highly repetitive subsequences are identified and stored in a codebook or dictionary. Compression is achieved by replacing each repetitive subsequence in the target genomic sequence by the

corresponding encoded subsequence in the codebook, the representation of which has a much smaller size. The performances of different algorithms differ mainly based on how many repeated subsequences can be identified and how efficiently they can be encoded.

Both BioCompress [9] and BioCompress-2 [10] methods use Lempel-Ziv (LZ) style substitutional algorithm [17] to compress the exact repeats and palindromes, where the latter also utilizes an order-2 context-based arithmetic encoding scheme [11] to store the non-repetitive regions. Alternatively, GenCompress [12] and DNACompress [13] identify approximate repeats and palindromes so that a larger fraction of the target sequence can be captured. Therefore, a higher compression ratio is achieved. Along the same line, GeNML [14] divides a sequence into blocks of fixed size. Then, blocks rich in approximate repeats are encoded with an efficient normalized maximum likelihood (NML) model; otherwise, plain or order-1 context based arithmetic encoding [11] is used.

A typical statistical compression method consists of two steps. In the first step, a prediction model is established, where a conditional probability is assigned to a base given bases preceding it. For instance, such a prediction model can be a second order Markov chain. Then, an encoding scheme such as arithmetic coding is used to carry out efficient compression using the probability distribution of each base provided by the prediction model. The accuracy of the prediction model is the dominant factor that determines the overall performance of the compression.

XM [18] was recognized as one of the most efficient traditional DNA sequence compression methods in statistical mode [8, 33]. XM uses the weighted summation of a panel of experts' prediction to calculate the probability distribution of bases before compressing the symbols by means of arithmetic coding. The experts consist of second-order Markov model, first-order context Markov model, whose probability distribution is not based on the entire history of the sequence but on a limited preceding context, and repeat expert, that considers the next base predicted is a repeat or complement of a pervious base in a particular offset. CTW-LZ [19] adopts both substitutional and statistical compression. It first uses hash and dynamic programming to search for palindromes and approximate repeats. Then, it utilizes context tree weighting (CTW) [16] prediction model, a method weighing on multiple models to predict succeeding base probabilities, to compress short repeats and uses LZ77 [17] to compress long repeats.

As pointed out in [20, 21], although these traditional genomic sequence compression methods have shown encouraging performance, their performance in terms of compute time, memory usage, and compression rate limit

their applicability to NGS data. We refer readers to a more thorough review [8] of these traditional methods and invest our effort to more recently developed methods for high-throughput sequencing data in the following sections.

ADVANCES IN GENOMIC SEQUENCE COMPRESSION FOR THE NEXT GENERATION SEQUENCING

The cost per raw Megabase of DNA sequence reduced from over 5,000 USD to under 0.1 USD over the past 10 years (http://www.genome.gov/27541954). As a result, sequencing is no longer limited to a handful selected model organisms but is being used as a genomic material surveying method [22]. Meanwhile, resequencing has been applied to study Human populations [23], clinical sequencing of virulent bacterial strains [24] and beyond. The compression strategies of newly assembled sequences vary depending on whether any similar genomic sequence has been previously deposited into the database. Based on this, compression methods can be categorized as reference-based and reference-free.

Reference-based Compression

When a large portion of a newly assembled genomic sequence, i.e. the target, has been captured by a similar, previously documented sequence, i.e. the reference, the target can be fully identified given the reference as well as the differences between the target and the reference. This is the basic idea of reference-based compression, where a typical procedure is given in Figure 1: the target is first aligned to the reference. Then mismatches between the target and the reference are identified and encoded. Each record of a mismatch may consist of the position with respect to the reference, the type (insertion, deletion, or substitution) of mismatch, and the value. Compression is achieved if the space required to encode the mismatches is less compared to store the target itself.

Brandon *et al.* [25] used various coders like Golomb [26], Elias [27], Huffman [28] coding to encode the mismatches and achieved a best 345-fold compression rate when applied to Human mitochondrial genome sequences. Christley *et al.* [29] proposed DNAzip algorithm that relies on the human population variation database, where a variant can be a single nucleotide polymorphism (SNP) or an indel (an insertion or a deletion). All variants of a newly obtained sequence can be identified by comparing the sequence against the database. Because the number of variants in the human population is substantially smaller than the size of human genome, DNAzip was able to

compress the James Watson genome using merely 4MB, i.e. a compression ratio of 750-fold. Different from DNAzip, which is dependent upon the knowledge of variation data, Wang *et al.* [30] presented a *de novo* compression program, GRS, that directly obtains variant information via a modified UNIX diff program [31]. Follow the motivation of *de novo* compression of GRS, GReEn [32] proposed a probabilistic copy model that calculates target base probabilities based on the reference. Given base probabilities as input, an arithmetic coder was then used to encode the target. GReEn was shown to be more effective compared to GRS when the target and the reference have less similarity.

Reference-based compression has also been applied to multiple homogeneous genomic sequences. Given a set of genomic sequences, RLZ method [33] arbitrarily selects one to be the reference. Then, each of the remaining target sequence is partitioned into a list of substrings in a greedy manner, where the next substring is selected to be the longest common substring between the reference and the suffix of the target sequence that has not yet considered. Therefore, every substring in any target sequence can be encoded by a substring in the reference in a LZ77 style [17]. RLZ-opt [34] improved upon RLZ by replacing the greedy partitioning method by the lookahead-by-*h* algorithm [35], where a shorter common substring can be selected between the target and the reference to achieve an overall better compression. GDC [36] further improved upon RLZ-opt in the following aspects (i) non-random selection of the reference, (ii) substrings in a target can be part of a non-reference sequence, (iii) approximate string matching is used to identify a substring. Together, these strategies lead to the identification of more repeats among input sequences. An equally important strategy used by GDC is to divide each input sequence into blocks of approximately equal size and encoded with Huffman coding to allow the random access of any specific sections of the compressed data without decompressing the whole data set.

Reference-Free Compression

Reference-based compression methods can achieve excellent compression results when a highly similar reference can be identified or variation data is available for methods dependent upon such information. Nonetheless, reference-free methods are required such as in *de novo* sequencing, metagenomic sequencing and epigenomic sequencing [6], where target genomes share weaker similarities with known ones and variation information is not readily available.

BIND [21] algorithm encodes each target sequence using two binary strings to achieve compression. In the first

string, base A or T is assigned with bit 0 and base G or C with bit 1; whereas in the second string, T or C is assigned with bit 0 and A or G with bit 1. These two strings were subsequently stored by recording the length of alternating blocks of 1's and 0's using unary coding scheme. In DELIMINATE algorithm [37], the positions of the two most dominant types of bases are delta encoded [38] and eliminated from the sequence. The remaining bases are then represented with binary code. BIND and DELIMINATE were shown to achieve higher compression efficiency than the general-purpose compression algorithms such as gzip, bzip2 and lzma when applied to the FNA, FFN, eukaryotic and NGS genomic datasets. Kuruppu *et al*. [20] proposed a substitutional approach COMRAD for compressing a set of highly redundant sequences, i.e. homologous sequences or sequences with multiple identical copies. COMRAD first identifies repetitive substrings with length ranging from hundreds to thousands of base pairs in the input data and includes them in a corpus-wide codebook. Each type of repeat in the codebook is then encoded into a short word. Compression is accomplished by replacing all repeats in the input sequences with the corresponding words from the codebook. Since COMRAD constructs the codebook from the input sequences, no external reference is required during decoding process. By storing the codebook and the compressed sequences with additional space proportional to the length of each sequence, COMRAD also enables random access of the compressed sequences.

Experimental Results

We select a set of genomes that have different species origin, length, and repeat content to benchmark a subset of representative methods with implementations publicly accessible. The details of selected data are given in Table 1 and the compression methods in Table 2.

All methods are tested on a cluster running 64-bit Red Hat 4.4.4-13 with 32-core 3.1GHz Intel(R) Xeon(R) CPU E31220. The performance of each method is evaluated based on compression time, decompression time, and compression ratio (Table 3). For completeness, the compressed file sizes are also given. It can be seen that both reference-based and reference-free methods outperform the general-purpose methods in terms of compression ratio. Whereas on these datasets, the reference-free methods generally achieved a 15%--40% better compression ratio over general-purpose compression programs, the availability of homologous reference has made a striking difference, where the reference-based programs achieved a 40-fold to over a 10,000-fold better compression ratio. However, to recover the target sequence, reference-based methods require the presence of the same reference genome as used for compression. The reference-based methods could take much longer time for compression and decompression for

genomes with large size (when reaching a scale of 100MB in the current experiment) because the compress/decompress process takes longer query time for bases/substrings for longer reference. Thus, when an appropriate reference is available and the compression gain on the target sequences can compensate the extract space requirement for the reference genome (such as in the case of multiple targets), reference-based methods should be favorable. As more genomic sequences have been uniquely documented in public databases, we expect a good use of reference-based methods in cases that require the transferring of large target sequences over the Internet. On the other hand, reference-free methods are self-contained and have a better balance between time efficiency and compression ratio compared to reference-based methods. Also, as shown in the current experiment, the total time used for compression and decompression is comparable between reference-free and general-purpose methods, making the former a more favorable choice.

NEXT GENERATION SEQUENCING DATA COMPRESSION

Each record of a NGS sequencing dataset typically consists of metadata (including read name, platform, project id, etc.), read (sequence) and the quality score sequence, stored either as FASTQ [39] or SAM/BAM format [40], where BAM is a compressed binary version of SAM. SAM or BAM format was designed for storing read alignment against a set of chosen references but has also been used to store unaligned data. Similar to genomic sequence compression, compression methods for reads can also be categorized as reference-based and reference-free. Nonetheless, metadata and quality score sequences consititute a large part of sequencing data. Because their alphabet size is much larger compared to reads, compression of these data fall back on strategies used for general-purpose compression.

Reference-based Read Compression

Resequencing is a cost effective way to identify mutations present in a target genomic region, where mutations are identified by aligning reads against a reference sequence [41]. The complete read data need not be stored but the variant positions with respect to the reference.

The typical reference-based read compression involves two major steps: read mapping and encoding of the mapping results, which is illustrated in Figure 2. First, reads are aligned to an appropriately selected reference genome (typically a reference with a high similarity to the target) using one of the NGS aligners such as Bowtie [42],

BWA [43] and Novoalign (http://www.novocraft.com). The choice of aligner depends on the type of NGS data and the specific application, where we refer the readers to this recent review [44]. Then, the mapping results (Table 4) are encoded via schemes such as arithmetic coding and Huffman coding.

GenCompress [45] uses Bowtie as the read aligner and then Golomb [26], Elias Gamma [27], MOV [46] or Huffman coding is used to encode the mapping results. Differently, SlimGene [47] carries out alignment using CASAVA software toolkit (http://www.illumina.com/pages.ilmn?ID=314). Then during encoding step, two binary vectors are generated. The first vector has the length equivalent to the reference, where a bit '1' indicates that at least one read is mapped to the corresponding position. The second vector is a concatenation of binary encoding of read mapping results. Given that the chosen reference is highly similar to the target, the space required for the two binary vectors is substantially less compared to storing reads themselves. In addition to the mapping and encoding procedure, CRAM [41] incorporates de Bruijn graph [48] based assembly approach to reads that failed to be mapped to the chosen reference, where the assembled contigs are used as references. Similarly, in addition to the standard reference-based compression component (enabled with parameter –r), Quip [49] also has the de Bruijn graph based *de novo* assembly component (enabled with parameter -a) when the references are not available. Different from the aforementioned methods that treat each read separately, NGC [50] traverses each column of read alignment to exploit the common features among multiple reads mapped to the same genomic positions and represents the mapped read bases in a per-column way using run-length encoding [51]. It is worth noting that as an alternative to SAM/BAM format, a multi-tier data organization Goby file format (http://campagnelab.org/software/goby/) also explores the idea of reference-base compression for storing read alignment, sequences of aligned reads are not explicitly stored but looked up in a reference genome when necessary.

Reference-Free Read Compression

In applications such as *de novo* sequencing, metagenomic sequencing, and epigenomic sequencing, where there is no clear choice of a reference, reference-free compression algorithms are needed.

[Are you able to insert some general description or summary of how reference-free methods described in this paragraph work? Like what we did for reference-based in the previous section]. The reference-free compression algorithms handle the redundancy of reads mainly with losslessly statistical encoding method, e.g., Huffman coding

and arithmetic coding, or substitutional method e.g. LZ77. Data transformation like BWT or reorganization like clustering are also imposed to aggregate similar reads to facilitate compression coding. Tembe *et al*. [52] proposed G-SQZ method. First, the frequency of each unique <base, quality> tuple is calculated given the input. Then, Huffman code is generated for each tuple where ones with higher frequency are encoded with less number of bits. Lastly, the encoded tuples along with a header containing meta-information such as the platform and the number of reads are written to a binary file to fulfill the compression. In DSRC algorithm [53], the input is divided into blocks of 32 records and every 512 blocks is grouped into a superblock. Each superblock is indexed and compressed independently, such as via LZ77 encoding, to enable fast access to individual record. Bhola *et al*. [54] first divided each read into substrings of a fixed length, where each substring is queried against a dictionary for a highly similar word (string) and in the meanwhile, first order Markov encoding [55] is applied to the substring. The query substring is encoded by registering the differences compared to the similar word identified in the dictionary or Markov encoded if less space is required this way. Quip [49] program, in addition to the reference-based compression mode as discussed previously, also includes an implementation of reference-free statistical compression using arithmetic coding based on high order Markov chains. Bonfield and Mahoney [56] presented two methods Fqzcomp and Fastqz, both using *order-N* context model and arithmetic coder. Differently, Fastqz contains an additional preprocessing step before the context modeling to reduce the data volume and it can be applied to reference-based compression.

Several algorithms have been introduced that explore similarities among input reads to achieve compression. ReCoil [57] algorithm first constructs a undirected graph with vertices denoting reads and edges carrying weights equivalent to the number of common *k*mers between the end nodes. Then a maximum spanning tree (MST) [58] is identified and traversed starting from an arbitrarily selected root node. The read corresponding to the root is stored directly whereas each remaining read is encoded via the set of maximal longest common substrings shared from its parent. 7-Zip compressor is further used to compress the encoded reads. Cox *et al.* [59] presented BEETL, which utilizes Burrows Wheeler Transformation (BWT) [60] to indirectly explore repeats among input reads. The transformed data are then further compressed via general-purpose compressors like gzip, bzip2, or 7-Zip. SCALCE algorithm [61] clusters input reads sharing the same longest ‘core’ substring identified by locally consistent parsing algorithm [62], then reads within a cluster are compressed by gzip or LZ77-like compression methods.

Compression of Metadata and Quality Scores

The metadata, such as read name, platform, and project identifiers, normally contain numeric and non-numeric text characters. Typically, they can be extracted from sequencing data and compressed separately using general text compression methods like Delta, Huffman, arithmetic, run-length, and LZ encoding. Sometimes, part of metadata can even be discarded [49, 61] when they do not affect downstream analysis.

Quality score sequences, on the other hand, are typically more important and required to be distributed along with reads. A quality score is denoted by an ASCII character, which can be converted to the probability of the corresponding base been correctly determined during sequencing. The probability information is directly utilized in many NGS analysis programs. As the alphabet size is large e.g. 40, general-purpose compression schemes have been directly used to losslessly preserve the quality score information. For instance, as discussed previously, G-SQZ [52] uses Huffman coding to encode <base, quality> tuples based on frequency. SlimGene [47] uses Markov encoding taking account of the position-dependent distribution of the quality scores and the higher order Markovian property between adjacent quality values. In DSRC algorithm [53], if the variation of quality score values are small within a sequence, run-length encoding was used to encode such a sequence, otherwise, Huffman encoding was used. In both [50] and [61], arithmetic encoding approach has been used to compress quality scores but achieved no obviously superior performance compared to general-purpose compression algorithms.

Because quality scores may not precisely determine base accuracy and adjacent bases with similar ASCII values of quality scores are likely to have similar sequencing accuracy, a series of algorithms have been introduced to cluster quality scores with similar values to achieve more aggressive compression. As the original quality score values are not exactly preserved, such methods are termed lossy compression. For example, SlimGene [47] employs randomized rounding strategy, a coarse quantization, to reduce the alphabet size of the quality scores and the resulting score sequences are compressed with Markov encoding. The experimental results show that this lossy scheme causes no significant loss of accuracy in downstream analysis such as variant calling. The reference-based method CRAM [41] stores quality scores for read bases that differ from the reference, and a user-specified percent of quality scores for read bases that are identical to the reference. Quip [49] uses the third order Markov chain based arithmetic coding to exploit the correlation of neighboring quality scores, where coarse binning is applied to read positions. QScores-Archiver [66] uses three lossy transformations based on coarse binning (UniBinning, Truncating,

and LogBinning). The transformed quality scores are further compressed using Huffman encoding, gzip, or bzip2. NGC algorithm [50] uses a non-uniform quantization to bin quality scores, where smaller quantization intervals are used for quality scores of bases that differ from the reference than bases that match the reference. The size of interval sets can be adjusted to control the loss of information, which positively correlates with the accuracy of the downstream analyses. QualComp [67] utilizes rate distortion theory [68] to minimize the distortion between input quality sequence and the compressed sequence post compression using the maximum allowable word size for storing each quality sequence as specified by the user. The word size controls the balance between the compression ratio and the accuracy for downstream analysis. Janin et al. [69] proposed a method on the premise that if a base in a read can, with high probability, be predicted by the context of bases surrounding it, then the base is less informative and its quality score can be discarded or aggressively compressed. The majority of bases can be predicted by aggregating reads into a compressed index via BWT and longest common prefix array [70], the corresponding quality scores could then be converted to an average value and effectively compressed.

Lossy compression on quality scores has obtained substantial reduction on the storage space of NGS data. Yet challenges and opportunities remain to apply lossy compression to improve the efficiency of NGS data storage and transmission. For example, to develop lossy compression methods to minimizing the loss of accuracy in storing read mapping results and in SNP calling [67].

Experimental Results

In this section, the performance of the compression methods for raw sequencing data is tested using ten NGS data sets. Similar to our test for genomic sequence compression, we also choose datasets that correspond to difference species (Human, plant, worm, fungus, and bacteria) and have different sizes (ranging from 2.7G to 12.8G). In addition, we select data produced by several popular NGS sequencing platforms including Illumina, 454, SOLiD, Ion Torrent. For the Human genomic data, we selected data from all four types NGS platforms. All sequencing data and the reference genomic sequences (Table 5) are obtained from NCBI/Sequence Read Archive (SRA) as FASTQ and FASTA format, respectively. We select ‘Quip –a’, ‘Quip –r’, and CRAM to represent the reference-based methods; DSRC, Quip, and Fqzcomp for reference-free methods, and the general-purpose compression methods bzip2 and gzip to be used as the baseline (Table 6). Default parameters are used for all methods. For ‘Quip –r’ and CRAM, the FASTQ files are converted to BAM format following [41], i.e., the reads in FASTQ are mapped to the reference

based on BWA 0.5.9 [43] and the mapping results are then converted to BAM format using samtools-0.1.18 [40]. All methods were executed on the same compute-cluster as previously mentioned. Note that although we intentionally discussed the compression of reads and quality scores separately from algorithmic point of view, all methods apply compression to both in an integrated fashion, hence each program is applied to the complete sequencing data.

The compression results are reported in Table 7. Note that the time spent on read mapping is included in the time of compression. In the current experiment, both reference-based and reference-free methods outperform the general-purpose methods in terms of compression ratio. The superiority of reference-based methods to the reference-free methods is not as obvious as that in the previous experiment of genomic sequence. For instance, the reference-based method 'Quip –r' shows slightly better compression ratio and compression time than the other methods. In the reference-free methods, Quip and Fqzcomp are competitive in terms of both compression time and compression ratio. DSRC is observed to obtain lower compression ratio than the other two methods. The general-purpose compression program gzip is typically faster during decompression whereas when combined with the compression time, it has no clear advantage over other methods. Although CRAM utilized lossy compression strategy, in the current experiments, it doesn't have superior performance over or reference-free compression, indicating a scope of improvement. Note that parameter settings of the tested methods could be optimized to improve the compression efficiency, yet it is not the focus of this article. We point readers to an interesting competition on NGS data competition by Pistoia Alliance at http://www.sequencesqueeze.org.

CHALLENGES AND FUTURE PROSPECTS

Despite advances in high-throughput DNA sequencing data compression, challenges remain. We list several aspects that may benefit from further development of compression algorithms for NGS data analysis.

Toward compressive genomics: Most of the DNA sequence data compression methods have been used for improving the efficiency in data storage and transfer. However, algorithms that can directly work with compressed data may potentially reduce both memory and run time. Loh et al. [71] presented the compressive version of alignment algorithms BLAST and BLAT to operate on compressed non-redundant large genomic data sets and

achieved a similar accuracy but substantial reduction in run time. The self-index data structures such as FM-index [72], compressed suffix arrays [73, 74], and LZ-index [75], enable the query, extract, retrieval and random access on compressed datasets directly. They have shown great promises in read mapping applications [44] and are expected to be the important support for designing compressive techniques. Routine analysis on compressed sequence data remains to be a challenge.

High-performance DNA Sequence Compression: High-performance computing is a natural direct remedy for processing larger data. Although distributed computing on MapReduce cloud [76, 77] has been utilized, such solution is typically limited to jobs with inherent embarrassed parallelism and has been applied to process uncompressed data. Parallel compression methods are yet to be developed and more complicated tasks requiring synchronization among different processors require more sophisticated parallel algorithms, in which case, parallel programming model like Message Passing Interface (MPI) can be explored.

Encrypted Compression: High-throughput sequencing has greatly promoted personalized genomics. As more people have their own genomes sequenced for the purpose of disease prevention, diagnosis, and treatment, transmission of the relevant compressed data over the Internet raises privacy concerns over information access and data security [6]. One solution for this issue could be the design of encrypted compression method.

Redundancy Elimination of DNA Sequence Database: Currently, most of the public available high-throughput sequencing data are stored in the databases of some big data centers like NCBI, EBI, and KEGG. The data in these public databases suffer from high redundancy as many projects, like the 1000 Genomes Project, are set up to investigate genomes with high similarity. The redundancy adds cost of storage and reduces query efficiency. Compressed data structure could be utilized to eliminate the redundancy in the databases. Although efforts have been made [20, 78], developing scalable algorithms to large databases such as NCBI remains open.

FUNDING

This work is supported in part by the NSFC funding, under grants 61001185 and 61171125, in part by the NSFC-RS joint project under grants IE111069 and 61211130120, and in part by Science Foundation of Shenzhen City under grant KQC201108300045A.

Key Points

1) The compression of high-throughput DNA sequence data is important to the efficient storage, transfer, and many downstream analyses of the data.

2) This review paper provides a comprehensive categorization and assessment of many high-throughput DNA sequence compression algorithms.

3) The future development of NGS data analysis can benefit from the compressive genomics, high-performance compression, encrypted compression, and redundancy elimination of DNA sequence databases. .

References

1. Sanger F, Nicklen S, Coulson AR. DNA sequencing with chain-terminating inhibitors, Proceedings of the National Academy of Sciences of the United States of America 1977;74:5463-5467.
2. Margulies M, Egholm M, Altman WE et al. Genome sequencing in microfabricated high-density picolitre reactors, Nature 2005;437:376-380.
3. Shendure J, Ji H. Next-generation DNA sequencing, Nature Biotechnology 2008;26:1135-1145.
4. Metzker ML. Sequencing technologies - the next generation, Nature Reviews Genetics 2010;11:31-46.
5. Branton D, Deamer DW, Marziali A et al. The potential and challenges of nanopore sequencing, Nature Biotechnology 2008;26:1146-1153.
6. Kahn SD. On the Future of Genomic Data, Science 2011;331:728-729.
7. Pennisi E. Will Computers Crash Genomics?, Science 2011;331:666-668.
8. Giancarlo R, Scaturro D, Utro F. Textual data compression in computational biology: a synopsis, Bioinformatics 2009;25:1575-1586.
9. Grumbach S, Tahi F. Compression of DNA sequences, In: Data Compression Conference 1993; 340- 350.
10. Grumbach S, Tahi F. A New Challenge for Compression Algorithms, Genetic Sequences, Information Processing & Management 1994;30:875-886.
11. Rissanen, JJ, Langdon, GG, Jr. Arithmetic coding. IBM Journal of Research and Development , 179; 23: 149–162.
12. Chen X, Kwong S, Li M. A compression algorithm for DNA sequences and its applications in genome comparison, In: Proceedings of the fourth annual international conference on Computational molecular biology. 2000, p. 107.
13. Chen X, Li M, Ma B et al. DNACompress: fast and effective DNA sequence compression, Bioinformatics 2002;18:1696-1698.
14. Korodi G, Tabus I, Rissanen J et al. DNA sequence compression - Based on the normalized maximum likelihood model 2007; 24: 53.
16. Willems FMJ, Shtarkov YM, Tjalkens TJ. The context-tree weighting method: Basic properties. IEEE Transactions on Information Theory, 1995; 41: 653-664.
17. Ziv J, Lempel A. A universal algorithm for sequential data compression, Information Theory, IEEE Transactions on 1977;23:337-343.
18. Cao MD, Dix TI, Allison L et al. A simple statistical algorithm for biological sequence compression, In: Data Compression Conference 2007;43-52.
19. Matsumoto T, Sadakane K, Imai H. Biological sequence compression algorithms, Genome informatics. Workshop on Genome Informatics 2000;11:43-52.

20. Kuruppu S, Beresford-Smith B, Conway T et al. Iterative Dictionary Construction for Compression of Large DNA Data Sets, IEEE-ACM Transactions on Computational Biology and Bioinformatics 2012;9:137-149.
21. Bose T, Mohammed MH, Dutta A et al. BIND - An algorithm for loss-less compression of nucleotide sequence data, Journal of Biosciences 2012;37:785-789.
22. Peterson J, Garges S, Giovanni M et al. The NIH Human Microbiome Project, Genome Research 2009;19:2317-2323.
23. Altshuler DM, Durbin RM, Abecasis GR et al. An integrated map of genetic variation from 1,092 human genomes, Nature 2012;491:56-65.
24. Didelot X, Bowden R, Wilson DJ et al. Transforming clinical microbiology with bacterial genome sequencing, Nature Reviews Genetics 2012;13:601-612.
25. Brandon MC, Wallace DC, Baldi P. Data structures and compression algorithms for genomic sequence data, Bioinformatics 2009;25:1731-1738.
26. Golomb S. Run-Length Encodings, IEEE Transactions on Information Theory 1965;12:399-401.
27. Elias P. Universal codeword sets and representations of the integers, Information Theory, IEEE Transactions on 1975;21:194-203.
28. Huffman DA. A method for the construction of minimum-redundancy codes, Proceedings of the IRE 1952;40:1098-1101.
29. Christley S, Lu Y, Li C et al. Human genomes as email attachments, Bioinformatics 2009;25:274-275.
30. Wang C, Zhang D. A novel compression tool for efficient storage of genome resequencing data, Nucleic Acids Research 2011;39:E45-U74.
31. Miller W, Myers EW. A file comparison program, Software: Practice and Experience 1985;15:1025-1040.
32. Pinho AJ, Pratas D, Garcia SP. GReEn: a tool for efficient compression of genome resequencing data, Nucleic Acids Research 2012;40.
33. Kuruppu S, Puglisi SJ, Zobel J. Relative Lempel-Ziv Compression of Genomes for Large-Scale Storage and Retrieval. String Processing and Information Retrieval. 2010, 201-206.
34. Kuruppu S, Puglisi SJ, Zobel J. Optimized relative Lempel-Ziv compression of genomes. In: Proceedings of Australasian Computer Science Conference 2011;91-98.
35. Horspool RN. The effect of non-greedy parsing in Ziv-Lempel compression methods. In: Data Compression Conference 1995;302-311.
36. Deorowicz S, Grabowski S. Robust relative compression of genomes with random access, Bioinformatics 2011;27:2979-2986.
37. Mohammed MH, Dutta A, Bose T et al. DELIMINATE-a fast and efficient method for loss-less compression of genomic sequences, Bioinformatics 2012;28:2527-2529.
38. Hunt JJ, Vo K-P, Tichy WF. Delta algorithms: An empirical analysis, ACM Transactions on Software Engineering and Methodology (TOSEM) 1998;7:192-214.
39. Cock PJA, Fields CJ, Goto N et al. The Sanger FASTQ file format for sequences with quality scores, and the Solexa/Illumina FASTQ variants, Nucleic Acids Research 2010;38:1767-1771.
40. Li H, Handsaker B, Wysoker A et al. The Sequence Alignment/Map format and SAMtools, Bioinformatics 2009;25:2078-2079.
41. Fritz MH-Y, Leinonen R, Cochrane G et al. Efficient storage of high throughput DNA sequencing data using reference-based compression, Genome Research 2011;21:734-740.
42. Langmead B, Trapnell C, Pop M et al. Ultrafast and memory-efficient alignment of short DNA sequences to the human genome, Genome Biology 2009;10.
43. Li H, Durbin R. Fast and accurate short read alignment with Burrows-Wheeler transform, Bioinformatics 2009;25:1754-1760.
44. Li H, Homer N. A survey of sequence alignment algorithms for next-generation sequencing, Briefings in Bioinformatics 2010;11:473-483.

45. Daily K, Rigor P, Christley S et al. Data structures and compression algorithms for high-throughput sequencing technologies, BMC Bioinformatics 2010;11.
46. Baldi P, Benz RW, Hirschberg DS et al. Lossless compression of chemical fingerprints using integer entropy codes improves storage and retrieval, Journal of chemical information and modeling 2007;47:2098-2109.
47. Kozanitis C, Saunders C, Kruglyak S et al. Compressing Genomic Sequence Fragments Using SlimGene, Journal of Computational Biology 2011;18:401-413.
48. Pevzner PA, Tang HX, Waterman MS. An Eulerian path approach to DNA fragment assembly, In: Proceedings of the National Academy of Sciences of the United States of America 2001;98:9748-9753.
49. Jones DC, Ruzzo WL, Peng X et al. Compression of next-generation sequencing reads aided by highly efficient de novo assembly, Nucleic Acids Research 2012;40.
50. Popitsch N, von Haeseler A. NGC: lossless and lossy compression of aligned high-throughput sequencing data, Nucleic Acids Research 2013;41.
51. Storer JA. Data compression: methods and theory. Computer Science Press, Inc., 1988.
52. Tembe W, Lowey J, Suh E. G-SQZ: compact encoding of genomic sequence and quality data, Bioinformatics 2010;26:2192-2194.
53. Deorowicz S, Grabowski S. Compression of DNA sequence reads in FASTQ format, Bioinformatics 2011;27:860-862.
54. Bhola V, Bopardikar AS, Narayanan R et al. No-Reference Compression of Genomic Data Stored in FASTQ Format 2011;150.
55. Nelson M. Data compression book. IDG Books Worldwide, Inc., 1991.
56. Bonfield JK, Mahoney MV. Compression of FASTQ and SAM Format Sequencing Data, Plos One 2013;8:e59190.
57. Yanovsky V. ReCoil - an algorithm for compression of extremely large datasets of DNA data, Algorithms for Molecular Biology 2011;6.
58. Dementiev R, Sanders P, Schultes D et al. Engineering an external memory minimum spanning tree algorithm. Exploring New Frontiers of Theoretical Informatics. 2004, 195-208.
59. Cox AJ, Bauer MJ, Jakobi T et al. Large-scale compression of genomic sequence databases with the Burrows-Wheeler transform, Bioinformatics 2012;28:1415-1419.
60. Burrows M, Wheeler DJ. A block-sorting lossless data compression algorithm 1994.
61. Hach F, Numanagic I, Alkan C et al. SCALCE: boosting sequence compression algorithms using locally consistent encoding, Bioinformatics 2012;28:3051-3057.
62. Sahinalp SC, Vishkin U. Efficient approximate and dynamic matching of patterns using a labeling paradigm 1996;328.
63. Jeon YJ, Park SH, Ahn SM et al. SOLiDzipper: A High Speed Encoding Method for the Next-Generation Sequencing Data, Evolutionary Bioinformatics 2011;7:1-6.
64. Sakib MN, Tang J, Zheng WJ et al. Improving Transmission Efficiency of Large Sequence Alignment/Map (SAM) Files, Plos One 2011;6.
65. Grassi E, Di Gregorio F, Molineris I. KungFQ: A Simple and Powerful Approach to Compress fastq Files, IEEE-ACM Transactions on Computational Biology and Bioinformatics 2012;9:1837-1842.
66. Wan R, Anh VN, Asai K. Transformations for the compression of FASTQ quality scores of next-generation sequencing data, Bioinformatics 2012;28:628-635.
67. Ochoa I, Asnani H, Bharadia D et al. QualComp: a new lossy compressor for quality scores based on rate distortion theory, BMC Bioinformatics 2013;14:187-187.
68. Cover TM, Thomas JA. Network Information Theory, Elements of Information Theory 1991:374-458.
69. Janin L, Rosone G, Cox AJ. Adaptive reference-free compression of sequence quality scores Bioinformatics 2013.
70. Bauer MJ, Cox AJ, Rosone G et al. Lightweight LCP construction for next-generation sequencing datasets. Algorithms in Bioinformatics. Springer, 2012, 326-337.
71. Loh P-R, Baym M, Berger B. Compressive genomics, Nature Biotechnology 2012;30:627-630.

72. Ferragina P, Manzini G. Indexing compressed text, Journal of the ACM 2005;52:552-581.
73. Makinen V, Navarro G, Siren J et al. Storage and Retrieval of Highly Repetitive Sequence Collections, Journal of Computational Biology 2010;17:281-308.
74. Makinen V, Navarro G, Siren J et al. Storage and Retrieval of Individual Genomes. Research in Computational Molecular Biology, Proceedings. 2009, 121-137.
75. Arroyuelo D, Navarro G, Sadakane K. Stronger Lempel-Ziv Based Compressed Text Indexing, Algorithmica 2012;62:54-101.
76. Afgan E, Baker D, Coraor N et al. Harnessing cloud computing with Galaxy Cloud, Nature Biotechnology 2011;29:972-974.
77. Stein LD. The case for cloud computing in genome informatics, Genome Biology 2010;11.
78. Cochrane G, Cook CE, Birney E. The future of DNA sequence archiving. GigaScience, 2012;1:2.