Evaluation of compression methods for re-sequencing genomic data

JingruWang
School of Computer Science and Technology
Beijing Institute of Technology, Beijing, China
Email: wangjignru@yeah.net
Lin Dai[1] and Zhang Zhang[2]
[1]School of Computer Science and Technology
Beijing Institute of Technology, Beijing, China
dailiu@bit.edu.cn
[2]CAS Key Laboratory of Genome Sciences and Information,
Beijing Institute of Genomics, Chinese Academy of Sciences, Beijing, China
Email: zhangzhang@big.ac.cn

Abstract—**Along with the development of the sequencing technology, the volume of the genome datasets increase greatly in a fast speed. The excessive surging of genome data causes storage issues to public or private databases as well to upload or transmit genome data via Internet. Data compression is an effective method to solve these problems. However, various genome compression methods adopting different strategies have been presented during previous years, making it challenging to choose the optimal method for practical use. In this paper, we firstly review state of the art on genome compression, then evaluate three excellent algorithms (GReEn, GDC and DELIMINATE) on real data and compare their performance with popular general-purpose compression algorithms, i.e., gizp, bzip2, xz and their parallel version. Instead of declaring the best method, we give advices to choose appropriate methods for specific genome dataset.**

Key words: **Genome Compression, evaluation, GReEn, GDC, DELIMINATE**

I. Introduction

As a core technology of Genomics, DNA sequencing has a constant development since the massively parallel sequencing arose [1, 2]. The reducing cost of sequencing greatly broadens the sequencing scope to address the biological questions at a genome-wide scale. For instance, 1000 Genomes Project (1000GP) has already sequenced 1092 individuals aiming to establish the most detailed catalogue of human genetic variation , and there are other projects that are ongoing to detect Human Disease genes' complex traits by employing the sequencing into clinical diagnostics [3, 4] or to research other species' genetic information. These extensive sequencing would generate huge genome data volume, which intensifies the pressure for storage device. The falling sequencing expense cannot meet the cost to store, transfer and analysis these massive pouring genome datasets, which puzzles the biologists greatly. Data compression is an effective method to relax the conflict.

At the beginning of the study, numerous text compression algorithms have been tried to compress the genetic sequences. Unfortunately, the result is far from satisfying, which worse leads to negative compression rates [5]. Accordingly the compression algorithms have been studied based on the particular intrinsic properties of DNA sequences. Grumbach et al. [5] can be the first study in DNA sequence compression and they first propose horizontal and vertical compression modes in biological sequence compression. Giancarlo et al. [6, 7] survey data compression techniques in computational biology and explain data redundancy of the two modes succinctly: simple repeat fragments and reverse complement structure in a sequence as horizontal direct repeat mode; In addition, identical fragments or approximate duplicates between multiple sequences as vertical repeat mode. Grumbach et al.'s BioCompress [5] uses LZ-like method to reduce DNA sequence size by detecting the direct repeat and reverse complement repeats in the sequence. After that, a series of classic specialized algorithms have been proposed, like GenCompress [8], CTW+LZ [9], DNAcompress [10], GeNML [11], XM [12] etc., laying substantial theoretical foundation for recent study. These early-stage algorithms use the bacterial gene as the benchmark data, which are not suitable for the recent huge bio-data compression. Recent algorithms are compensating this restriction based on the traditional data compression algorithms: Dictionary-based and Statistic compression. A new kind algorithm emerges to squeeze the redundancy of multiple related sequences in vertical mode, e.g. compression numerous individuals within species. This compression mode needs extra reference information to generate the variations (SNPs and indels) between the compression target and reference. Based on that, the recent algorithms can be roughly classified into two categories: reference based and non-reference based algorithms.

A. Reference based Algorithms

For reference-based algorithms, how to generate and encode the difference is key technique. Christley et al. [13], Brandon et al. [14] give a series of related coding strategies to encode the variation positions values without giving the method how to generate the variations. Fixed codes (Golomb and Elias codes etc.), variable codes (Huffman) and Delta encode are suggested. Inspired by Christley et al. This inspires Pavlichin et al. [15]and Deorowicz et al. [16]implement directly compression of the variants of a collection of genomes of same species (eg.VCF files in 1000GP) and achieve highly successful. For random search and access to the compressed data without decompression, RLZ [17], RLZ-opt [18] apply the self-index [19] into the genomic data compression and use LZ77 encoding. In addition, GRS [20] ,GDC [21] and GReEn [22] compresses the re-sequencing data referenced the whole genome of an individual with different strategy to generate and encode the variations. While the latter two algorithms show more considerable compression performance. Chern et al. [23] apply end-to-end mapping of target genome to a given reference genome in compression scheme. ABRC [24] uses compressed suffix tree to index the reference for adaptively finding the longest prefix-suffix matches with the target genome.

B. *Non-reference based Algorithms*

Here we conclude that the non-reference algorithms include the methods which are based on the dictionary-based and statistical compression algorithms and some boosting transformation methods. For example, COMRAD [25] is a disk-based compression algorithm based on substitution strategy of dictionary-based method to compress large related DNA datasets. DELIMINATE [26] is a combination of delta encoding [27] and 7z-archiver to progressive elimination of nucleotide characters. Different with DELIMINATE, Bind [28] is novel 'block-length encoded' with combination of unary coding and lzma algorithm which is available as an implementation within the 7-Zip compression package. In addition, transformation algorithms, e.g. Burrows-Wheeler transform (BWT) [29] can permute data redundancy more intensive than the raw data, can greatly boost the compression gain. Cox et al. [30] [31] present a methodology for computing the BWT in a lightweight fashion for large genomic datasets, BEETL library [32] is an implementation of that. The above algorithms are about genomes compression, while there is another hot spot in bio-specialized compression researches, NGS reads compression[33-35], which shares the common strategy with genomes compression. More information about the bio-specialized compression methods can refer the reviews[36, 37].

Although methods have been focusing on pursuit of the high compression performance, there is no specialized and mature program like gzip which is still adopted as the most popular file compression tool during biological data storage process. As the present methods adopt different strategies to compress the DNA sequences, it makes challenging to choose the optimal method for actual use process. Therefore, it is of substantial significance to evaluate compression performance of the specialized compression algorithms with real datasets. To address this challenge, we evaluate genome data compression methods (GReEn, GDC and DELIMINATE) on a number of real datasets and compare them against several general purpose compression algorithms like gzip,bzip2 and xz, aiming to examine their performances for different species in aid of method selection for practice. We also present future directions and possible improvements for better genome data compression based on our comparative results.

II. Materials and Methods

GReEn is inspired by the XM by devising an adaptive model and a static model to provide a good probability estimate for the target genome based on extra reference genome. It was an outstanding representative of reference-based algorithm based on arithmetic encoding. Slightly different with GReEn, GDC compresses multiple genomes of the same species by exploiting the inter-genomic redundancy of the target genomes themselves. GDC used LZ-parsing and Huffman encoding. DELIMINATE algorithm uses delta coding and binary coding first, then it uses 7-zip archiver to compress the various intermediate files. DELIMINATE can be comparable to general-purpose compression algorithm practicably no matter in time and space efficiency. These three algorithms can greatly represent the state-of-art of biological data compression algorithms because of their performance that they not only achieve handsome compression ratio but also take account of the trade-off of time and memory consumption.

So we chose the three specialized algorithms (GReEn, GDC and DELIMINATE) as the representatives of biological data compression methods and compared the compression performance of three specialized algorithm with three excellent general-purpose compression tools (gzip,bzip2 and xz), three parallel version of bzip2 and gzip (pbzip2, mgzip and pigz). We gave a comprehensive evaluation of the compression methods from the compression performance (compression rate, compression / decompression time and memory usage) to the applicability for special of the compression target, aiming at providing the biologists a clearly understanding of the performance of the existing specialized DNA sequence compression algorithm.

For comparing the algorithms' performance for different species and different individuals of the same species, we chose a set of genomes from different species as the test datasets corresponding to the methods' requirement. The datasets include 59 strains of *Escherichia coli*, 11 complete genomics of the *Oryza sativa* and 11 complete genomics of *Homo sapiens*. We listed the parameters for selected methods (Table Ⅰ) and detail information of datasets (Table Ⅱ). Here we generally used the default parameters for the methods, which were the optimal performance of the methods with the best trade-off in run-time and storage space.

All methods are tested on a PC running 64-bit Ubuntu 12.04.1 LTS with 4x Intel(R) Core(TM) i5-2400 CPU @ 3.1GHz 8GB memory.

TABLE I. OVERVIEW OF CHOSE METHODS

Method	Descriptions	Command	Source file
GReEn	Copy model and Arithmetic coding	GReEnC -v -o GReEnD -v -o	ftp://ftp.ieeta.pt/~ap/codecs/GReEn1.tar.gz
GDC	LZ-like, Delta encoding and Huffman coding	gdc c -ma1000000000 -mp20,4,20 -rn1 -hs8 -rm0 (compression for H. sapiens) gdc c -ma1000000000 -rn1 (compression for rice and E.coli) gdc d	http://sun.aei.polsl.pl/gdc
DELIMINATE	Delta encoding and 7z	delim a delim e	http://metagenomics.atc.tcs.com/compression/DELIMINATE/
gzip	Huffman coding and LZ77 Version 1.4	gzip -6 default compression mode	http://www.gzip.org/
bzip2	BWT, move-to-front transform and Huffman coding.Version 1.0.6	bzip2 -9 default compression mode	http://www.bzip.org/
xz	LZMA2 algorithm Version 5.1.0alpha	xz -6 default compression mode	http://tukaani.org/xz/
pbzip2	Parallel BZIP2 uses libbzip2.Version v1.1.8	Default mode bzip2 -9 with 4 processors	http://compression.ca/pbzip2/
mgzip	a multi-processor capable .gz file creator.Version 1.2c	Default mode gzip -6 and 4 worker threads	https://github.com/jerodsanto/mgzip
pigz	parallel implementation of gzip.Version 2.3.1	Default mode gzip -8 and 4 worker threads	http://www.zlib.net/pigz

TABLE II. OVERVIEW OF THE TESTING DATASETS

Target Genome	Number	File size (Byte)	Reference	Data source
Escherichia_coli	59	299917564	MG1655	NCBI
Oryza sativa	11	4165271984	Build 4.0	NCBI and Plant Biology Michigan State university
Homo sapiens	11	32993445836	GRCh37.p13	NCBI and Korean Bioinformation Center

III. RESULTS AND DISSCUSSION

To evaluate the comprehensive compression ability, we classify the selected methods into three groups: biological specialized compression (BSC), general-purpose compression (GPC) and parallel general-purpose compression (PGPC) and use the datasets test their compression ability. We compare the algorithms' compression results from four concrete aspects: compressed data accumulation trend, data compression ratio, compression /decompression time and memory usage.

For *Homo sapiens* datasets, because the volume of an individual's whole genome reaches nearly three gigabyte around, taking the whole genome as reference to match in the memory is not only time-consuming but also resource-intensive, which is impractical for the ordinary PC users. So, during the evaluation, each chromosome of the dataset was compressed against the respective chromosome of reference genome, which could highlight the methods' advantages greatly. We present storage space growth curve for the compressed data as the number of individuals increasing (Fig.1), average compression rate (Fig.2), total compression time/decompression time (Fig.3) and peak memory usage (Fig. 4).

A. *Compressed Data Accumulation Trend*

As the number of individuals increasing, the slower speed the data volume grows, the less storage space the compressed data occupies, that is the better compression performance for testing datasets. (Fig. 1). There exists apparent otherness among the methods. The BSC methods have a gentle growth trend relative to GPC and PGPC methods, which means that they have a better compression rate. The PGPC has basic consistent trend with the GPC excluding the inappreciable expansion in data size. In addition, PGPC have an outstanding process speed, which we will show later. For the three genomic compression methods of BSC, they have different character because of their different inherent essence. As GReEn and GDC are the reference-based method, they show more sensitive to the differences between the target and the reference. We could notice that, there is an increase of the slope among the gentle trend line, which is due to that the corresponding individual has a larger difference with the reference than the previous individual. GDC took the first individual as the reference for other individuals. Consequently, there is an obvious break point at the position of the first individual due to its non-reference compression. While DELIMINATE not only shows obviously better performance than GPC methods, but also has common trait that is not insensitive to the datasets. For instance, GReEn squeezed the size of 11 Homo sapiens individuals to nearly 570MB, which could be shared to each individual a mean value about 52MB. Because of inter-genome compression tactics, GDC got a slightly high mean value about 84MB compared with GReEn, while it do not need an extra reference in decompression process. DELIMINATE got an average value about 587MB, which was almost 12.8% shrink compared to the best GPC xz. The situation is similar for *Escherichia coli* and *Oryza sativa*. To summarize, the BSC methods have much less disk usage for the final compressed data storage and show thoroughly superiority against GPC methods.

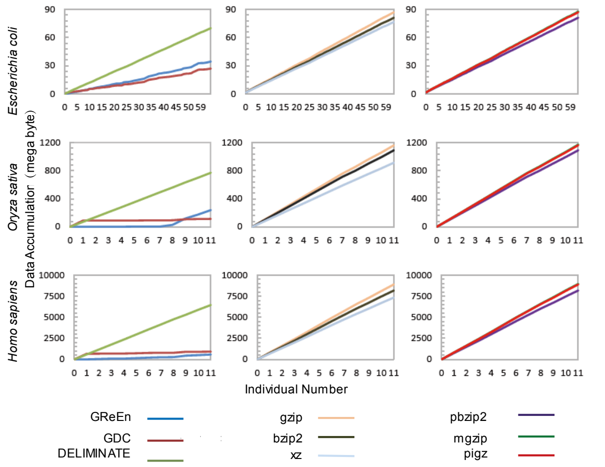

Figure 1. Accumulation trend of compressed data taking up storage space. X-axis represents the individual number, the order of the individuals has no significance, y-axis represents the compressed size in megabyte. For GReEn, the reference size does not be included.

B. *Compression Rate*

To compare the compression rate of three kinds of compression methods, we calculated the compression rate with the equation as in (1) to represent the percent of saving space.

$$\textit{Compression rate} = (1 - \textit{compression\_size}/\textit{raw\_size}) * 100\% \quad (1)$$

We calculated the total compression rate of the three species (Fig. 2). The BSC methods achieved amazing high compression rate ranging from 75.53% to 98.18%, which were nearly 5-30% better than the GPC methods with the compression rate range from 69.55% to 76.99%. The PGPC kept almost consistent pace with GPC for the compression rate. The reference-based methods gain the compression rate no less than 87%, while DELIMINATE achieves the peak value 80.61%. For this aspect, the reference-based methods show insuperable compression ability.

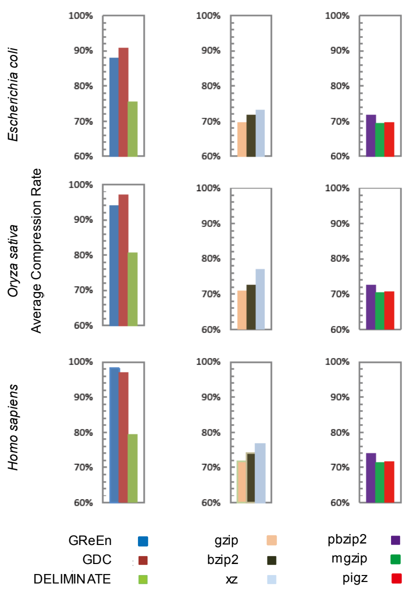

Figure 2. Compression rate. We calculate the compression rate with the equation (1), com_size represents the size of the compressed data, raw_size represents the size of the initial data.The higher the bar the better compression rate.

C. *Compression Time and Decompression Time*

Overhead time is another key point to measure the performance of compression methods. We presented the time cost for compression and decompression (Fig. 3). The BSC methods didn't maintain their predominance any more. BSC methods took more time than gzip and bzip2 because of preprocessing data before compression. Xz had the best compression rate among the GPC methods and yet took the most time consuming. Relative to xz, BSC showed reversely economy. The compression time of GReEn and DELIMINATE is directly proportional to their decompression time, while DELIMINATE shows more time-saving. GReEn' average process speed was about 1.9MB/s~2.5MB/s and a peak range from 12MB/s to 14MB/s (extremely similar individuals). DELIMINATE's speed is about 5.7~13.8MB/s. GDC is asymmetric in time consumption. The time taking in decompression is distinct less than the compression time, which is advantageous in the once-compression and multi-decompression situation. GDC' average speed in compression is about 1.8~6.7MB/s and in decompression is about 71~85MB/s. Gzip and bzip2 have average speed about 7.5~9.5MB/s and a decompression speed 14.2~36.6MB/s (gzip faster). In contrast, PGPC methods got nearly three times improvement in compression speed and slightly faster decompression speed (bzip2 almost two times faster). Therefore, the parallel version methods show amazing fast with just little loss in compression rate, which can be borrowed in the BSC methods.

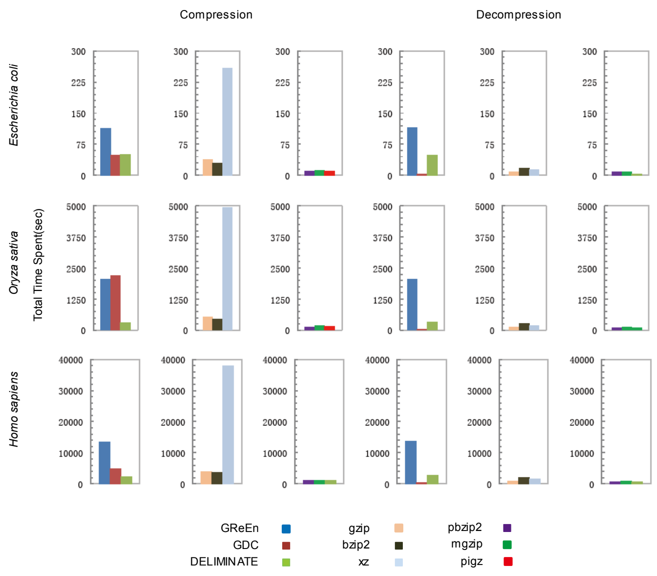

Figure 3. Total compression and decompression time methods took to process the test datasets.

D. *Memory Usage*

Memory usage information is an indicator to measure whether methods are practical on ordinary PC. Generally, the GPC methods have default constant of memory usage, which is nothing to do with the size of input data. Therefore, we just give the peak value of BSC methods' memory usage during the process (Fig. 4). While GReEn and GDC need much more memory to match the reference for data compression compared with DELIMINATE, which uses the 7z archiver taking little memory as the GPC methods. GDC' compression memory usage is proportional to the reference size. For instance, if we use the whole human genome(3GB around) as the reference, its memory usage will increase to 7~8GB quickly and sustain in whole process, which is inadvisable. GDC's decompression memory usage is quite less than the compression process. While GReEn's compression memory usage depends on not only the reference size but also the difference between the target and the reference.

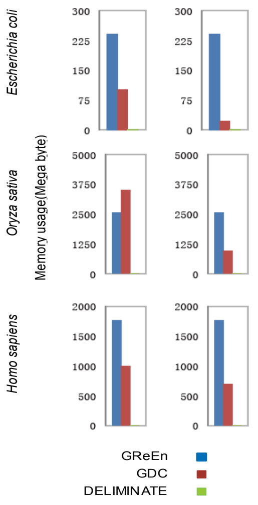

Figure 4. Memory usage of the BSC methods. The peak value of memory usage corresponds to the BSC methods.

Through the analysis above, we conclude that the reference-based BSC methods show overwhelming superiority on the compression rate compared with GPC methods. Although their running-time is somewhat longer, this kind method is the best choose for related genomic data compression, which can great reduce local storage space and alleviate bandwidth pressure of transmission in the network. For multiple decompression scenes, GDC's outstanding decompression speed should be more attractive. DELIMINATE shows better effectiveness both in compression rate and runtime, which can substitute the GPC methods in unrelated genomic sequences compression. In addition, we also observed that PGPC methods show amazing process speed respected to GPC methods. For the reference-based BSC, the pressure of more running-time and memory usage is the bottleneck for practical application, which is because of the preprocessing of the sequences. In summarize, in spite of the high compression rate of BSC methods, there is great latent potential to improve the running speed aiming at being a mature algorithm in practical use.

IV. CONCLUSIONS

Based on all the evaluation above, we conclude that the reference-based methods show superb compression gain to compress the homologous genomes. For the requirement to compress a large amount of the genomes with high similarity the reference-based methods is a good choice, but the key is to choose a good reference. The availability of a reference genome for mapping can highly improve the compression rate achieved by standard Lempel-Ziv techniques. While, mapping to the reference to find the difference is quite memory and CPU cycle consuming. The transformation methods, reordering the input first use the BWT to gather the same base together will enhance the compression gain heavily. For large amount genome transformation, the consumption of time and computation resource can't be ignored. Compared with the universal compression method the time-consuming and memory usage have the potential to improve for the biological compression method. Parallel realization is a handsome strategy to reduce the run-time. In addition, the genomic dataset always has a series of follow-up processes, it will be more practical that if it permits the random access and retrieval on compressed datasets directly, which is the ultimate goal of our research in the future.

ACKNOWLEDGMENT

The authors gratefully acknowledge Tony Wu for editing this manuscript and give pertinent suggestion.

REFERENCES

[1] J. Shendure and H. Ji, "Next-generation DNA sequencing," Nature biotechnology, vol. 26, pp. 1135-1145, 2008.
[2] D. C. Koboldt, K. M. Steinberg, D. E. Larson, R. K. Wilson, and E. R. Mardis, "The next-generation sequencing revolution and its impact on genomics," Cell, vol. 155, pp. 27-38, 2013.

[3] E. R. Mardis, "A decade/'s perspective on DNA sequencing technology," Nature, vol. 470, pp. 198-203, 2011.

[4] H. L. Rehm, "Disease-targeted sequencing: a cornerstone in the clinic," Nature Reviews Genetics, vol. 14, pp. 295-300, 2013.

[5] S. Grumbach and F. Tahi, "Compression of DNA sequences," 1994.

[6] R. Giancarlo, D. Scaturro, and F. Utro, "Textual data compression in computational biology: a synopsis," Bioinformatics, vol. 25, pp. 1575-1586, 2009.

[7] R. Giancarlo, D. Scaturro, and F. Utro, "Textual data compression in computational biology: Algorithmic techniques," Computer Science Review, vol. 6, pp. 1-25, 2012.

[8] X. Chen, S. Kwong, and M. Li, "A compression algorithm for DNA sequences," IEEE engineering in medicine and Biology, vol. 20, pp. 61-66, 2001.

[9] T. Matsumoto, K. Sadakane, and H. Imai, "Biological sequence compression algorithms," GENOME INFORMATICS SERIES, pp. 43-52, 2000.

[10] X. Chen, M. Li, B. Ma, and J. Tromp, "DNACompress: fast and effective DNA sequence compression," Bioinformatics, vol. 18, pp. 1696-1698, 2002.

[11] G. Korodi and I. Tabus, "An efficient normalized maximum likelihood algorithm for DNA sequence compression," ACM Transactions on Information Systems (TOIS), vol. 23, pp. 3-34, 2005.

[12] M. D. Cao, T. I. Dix, L. Allison, and C. Mears, "A simple statistical algorithm for biological sequence compression," in Data Compression Conference, 2007. DCC'07, 2007, pp. 43-52.

[13] S. Christley, Y. Lu, C. Li, and X. Xie, "Human genomes as email attachments," Bioinformatics, vol. 25, pp. 274-275, 2008.

[14] M. C. Brandon, D. C. Wallace, and P. Baldi, "Data structures and compression algorithms for genomic sequence data," Bioinformatics, vol. 25, pp. 1731-8, Jul 15 2009.

[15] D. S. Pavlichin, T. Weissman, and G. Yona, "The human genome contracts again," Bioinformatics, vol. 29, pp. 2199-202, Sep 1 2013.

[16] S. Deorowicz, A. Danek, and S. Grabowski, "Genome compression: a novel approach for large collections," Bioinformatics, vol. 29, pp. 2572-8, Oct 15 2013.

[17] S. Kuruppu, S. J. Puglisi, and J. Zobel, "Relative Lempel-Ziv compression of genomes for large-scale storage and retrieval," in String Processing and Information Retrieval, 2010, pp. 201-206.

[18] S. Kuruppu, S. J. Puglisi, and J. Zobel, "Optimized relative Lempel-Ziv compression of genomes," in Proceedings of the Thirty-Fourth Australasian Computer Science Conference-Volume 113, 2011, pp. 91-98.

[19] G. Navarro and V. Mäkinen, "Compressed full-text indexes," ACM Computing Surveys (CSUR), vol. 39, p. 2, 2007.

[20] C. Wang and D. Zhang, "A novel compression tool for efficient storage of genome resequencing data," Nucleic Acids Research, vol. 39, pp. e45-e45, 2011.

[21] S. Deorowicz and S. Grabowski, "Robust relative compression of genomes with random access," Bioinformatics (Oxford, England), vol. 27, p. 2979, 2011.

[22] A. J. Pinho, D. Pratas, and S. P. Garcia, "GReEn: a tool for efficient compression of genome resequencing data," Nucleic Acids Research, vol. 40, pp. e27-e27, 2011.

[23] B. G. Chern, I. Ochoa, A. Manolakos, A. No, K. Venkat, and T. Weissman, "Reference Based Genome Compression," IEEE Information Theory Workshop, 2012.

[24] S. Wandelt and U. Leser, "Adaptive efficient compression of genomes," Algorithms Mol Biol, vol. 7, p. 30, 2012.

[25] S. Kuruppu, B. Beresford-Smith, T. Conway, and J. Zobel, "Iterative Dictionary Construction for Compression of Large DNA Data Sets," IEEE/ACM Transactions on Computational Biology and Bioinformatics, vol. 9, pp. 137-149, 2012.

[26] M. H. Mohammed, A. Dutta, T. Bose, S. Chadaram, and S. S. Mande, "DELIMINATE--a fast and efficient method for loss-less compression of genomic sequences: sequence analysis," Bioinformatics, vol. 28, pp. 2527-9, Oct 1 2012.

[27] J. J. Hunt, K.-P. Vo, and W. F. Tichy, "Delta algorithms: An empirical analysis," ACM Transactions on Software Engineering and Methodology (TOSEM), vol. 7, pp. 192-214, 1998.

[28] T. Bose, M. H. Mohammed, A. Dutta, and S. S. Mande, "BIND - an algorithm for loss-less compression of nucleotide sequence data," J Biosci, vol. 37, pp. 785-9, Sep 2012.

[29] M. Burrows and D. J. Wheeler, "A block-sorting lossless data compression algorithm," 1994.

[30] S. Mantaci, A. Restivo, G. Rosone, and M. Sciortino, "An extension of the Burrows Wheeler transform and applications to sequence comparison and data compression," in Combinatorial Pattern Matching, 2005, pp. 178-189.

[31] M. J. Bauer, A. J. Cox, and G. Rosone, "Lightweight algorithms for constructing and inverting the BWT of string collections," Theoretical Computer Science, vol. 483, pp. 134-148, 2013.

[32] A. J. Cox, M. J. Bauer, T. Jakobi, and G. Rosone, "Large-scale compression of genomic sequence databases with the Burrows-Wheeler transform," Bioinformatics, vol. 28, pp. 1415-9, Jun 1 2012.

[33] K. Daily, P. Rigor, S. Christley, X. Xie, and P. Baldi, "Data structures and compression algorithms for high-throughput sequencing technologies," BMC Bioinformatics, vol. 11, p. 514, 2010.

[34] M. Hsi-Yang Fritz, R. Leinonen, G. Cochrane, and E. Birney, "Efficient storage of high throughput DNA sequencing data using reference-based compression," Genome Res, vol. 21, pp. 734-40, May 2011.

[35] D. C. Jones, W. L. Ruzzo, X. Peng, and M. G. Katze, "Compression of next-generation sequencing reads aided by highly efficient de novo assembly," Nucleic Acids Research, vol. 40, pp. e171-e171, 2012.

[36] R. Giancarlo, S. E. Rombo, and F. Utro, "Compressive biological sequence analysis and archival in the era of high-throughput sequencing technologies," Brief Bioinform, Dec 17 2013.

[37] Z. Zhu, Y. Zhang, Z. Ji, S. He, and X. Yang, "High-throughput DNA sequence data compression," Brief Bioinform, Dec 3 2013.