A sorting tool for improving FASTA data compression tools

Tiago Fonseca[1,2], Maria J. P. Sousa[1,2], Armando J. Pinho[1,2], and Diogo Pratas[1,2,3]

[1]IEETA/LASI - Institute of Electronics and Informatics Engineering of Aveiro
[2]DETI - Department of Electronics, Telecommunications and Informatics
University of Aveiro, 3810-193 Aveiro, Portugal,
[3]DoV - Department of Virology, University of Helsinki, 00014 Helsinki, Finland
Email: tiagorfonseca@ua.pt

***Abstract*—The exponential growth in genomic data sequenced has led to a greater need for efficient methods to reduce data volume without loss of information. Data compression emerges as a pivotal strategy to tackle this challenge, given the inherent redundancy and substantial storage requirements of genomic data. In this paper, we introduce a novel method designed to enhance existing data compression techniques by leveraging the sorting of collections of FASTA sequences based on various criteria. This approach capitalizes on clustering similar blocks together, thereby optimizing compression efficiency. We conducted comprehensive benchmarking of our proposed tool, evaluating the impact of sorting based on diverse characteristics of the sequences, including the length and ratio of Guanine-Cytosine present. Our evaluations involved testing the tool across a spectrum of data compression tools using both nearly-synthetic and real datasets. Our findings reveal insights into the interplay between compressed size, the employed compressor, file composition, and sorting technique, elucidating specific scenarios where compression outcomes are influenced. The proposed tool, coded in C++, is made freely available under the GPLv3 license, accessible at https://github.com/cobilab/FASTA-ANALYSIS.**

***Index Terms*—genomic sequences, data compression, FASTA format, sorting techniques**

I. Introduction

The recent advancements in the field of bioinformatics have resulted in a rapid increase in the amount of sequenced data available. However, this data needs to be stored and the computational storage available is limited. To keep up with the vast amounts of new data generated every day, more efficient ways to store it are needed. As a result, several compression algorithms and implementations have been developed in the past few decades to address this issue.

The methods developed for genomic data compression fall into two main categories: lossy and lossless compression. Lossless compression algorithms aim to reduce storage space while preserving all original data, making them essential for genomic data, where any loss of information is unacceptable.

In the realm of lossless compression for genomic data, tools can be broadly categorized into two groups: general-purpose compressors and DNA-specific compressors.

General-purpose compressors offer versatility in handling various data types, enhancing accessibility and ease of use. However, their lack of specialization in the unique characteristics of genomic data often results in suboptimal compression compared to DNA-specific compressors tailored for FASTA data. Among the commonly used general-purpose compressors are gzip [1], zstandard [2], bzip2 [3] and LZMA [4].

DNA-specific compressors are compressors developed to compress genome sequences and therefore, often achieve better results when compressing this type of data in relation to general-purpose compressors. On the other hand, specific genomic compressors take into considerations specificity's of the data, such as inverted repeats, rearrangements, and higher regions of mutational substitutions.

The first DNA-specific compressor created was Biocompress [5], developed in 1993 and based on dictionary-based algorithms. This tool was improved by its successor, Biocompress-2 [6]. After, many other tools have been developed, such as GenCompress [7], DNA-Compress [8], followed by many other tools, including [9]–[15].

The ubiquitous presence of complete genomes has necessitated the development of the FASTA format, enabling the co-existence of genomic sequences and annotations. Specialized compression algorithms, paired with straightforward header coding, have been tailored to this format. Prominent examples include Deliminate [16], MFCompress [17], LEON [18], NAF [19], MBGC [20], and AGC [21].

For downstream analysis, the sequential order of FASTA reads is typically irrelevant. Hence, there's potential to explore sorting these reads based on similarity. Our hypothesis suggests that grouping together FASTA reads with shared similarity could lead to significant reductions in data representability, thus enhancing the effectiveness of existing data compressors for genomic data.

While various sorting tools have been successfully employed to enhance the compression of FASTQ data formats, such as those referenced in the literature [22]–[27], it is notable that there is a lack of similar tools specifically tailored for FASTA format. This presents an untapped opportunity for the development of sorting techniques aimed at optimizing the compression of FASTA files, potentially yielding significant improvements in storage efficiency.

In this article, we introduce a novel tool designed to sort FASTA reads based on specific features, filling the gap in the field of genomic data compression for FASTA files. We then

investigate the most effective features for enhancing compression efficiency by benchmarking them on both synthetic and real datasets. To facilitate benchmarking on synthetic FASTA data, we develop a simulation pipeline, which we also make publicly available for use by the research community. This comprehensive approach allows us to systematically evaluate the impact of various sorting features on compression performance, providing valuable insights into optimizing genomic data compression strategies.

II. Methods

Consider a set of FASTA sequences X constituted by $x_1, x_2, ..., x_n$. The goal is to sort the X set in order to approximate similar sequences according to specific features for improving the performance of a data compressor after this transformation.

To carry out this transformation, we introduce FASTA ANALYSIS (abbreviated as FAN). The methodology of FAN is illustrated in Figure 1, delineating its three primary execution phases. Initially, the input FASTA file is parsed, and the start and end positions of each DNA sequence are recorded in a vector. In the second step, the input file undergoes another reading process, during which its content is analyzed to guide the proper sorting of the vector containing sequence boundaries, based on the selected sorting strategy. Finally, in the third reading iteration of the input file, sequences are directly written to the output file according to the defined order determined by the sorted positions vector.

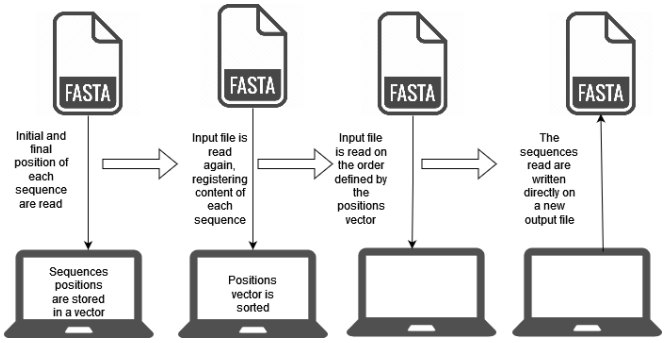

Fig. 1. The four main phases of the FAN tool.

Notice that the first and second phases could be merged, but we deliberately chose to separate them to facilitate additional experiments and allow for flexibility in integrating external tools for the extraction of other features. This approach enhances the versatility and extensibility of the FAN methodology, accommodating potential future enhancements and modifications with ease.

Regarding the features, FAN currently extracts the following five features:

1) genomic sequence length;
2) absolute GC (Guanine and Cytosine) content;
3) absolute AT (Adenine and Thymine) content;
4) GC percentage;
5) AT percentage.

The rationale behind utilizing these characteristics is rooted in the observation that similar genome sequences typically exhibit comparable sizes and proportions of GC/AT nucleotides. However, instances of closely related species may experience rearrangements, leading to duplications or deletions that can alter the values of these features. Hence, to account for variations in genome size, we employ relative measures such as percentages (%) alongside absolute measures. This approach ensures robustness in assessing sequence similarity while accommodating potential variations arising from genomic rearrangements.

Indeed, the FAN framework is not restricted to the features mentioned earlier. More sophisticated features can be incorporated, including measures like Kolmogorov complexity, which has been demonstrated to be a significant feature for distinguishing or relating genomic sequences [28]. This flexibility allows for the integration of advanced features to further enhance the capabilities of FAN in analyzing genomic data and potentially boot data compressors.

After extracting features, the X sequences are sorted using the quicksort algorithm, which can consider either a single feature or a combination of multiple features. If a primary feature is present, it takes precedence in the sorting process. However, in cases where multiple features are involved and ambiguity arises, the remaining features are utilized to determine the sorting order. This approach ensures efficient and accurate sorting of sequences, leveraging their distinctive characteristics while accommodating different priorities.

The FAN algorithm concludes by producing the output of sorted FASTA reads in a new file. This sorted file is then ready for use by a data compressor, enabling optimized compression of genomic data. Notably, the decompression of this file won't necessitate any additional computation, as the preservation of the original order of the reads is deemed unimportant. This streamlined process ensures efficient compression and decompression workflows for genomic data analysis.

The FAN method is implemented in the C++ programming language and is entirely self-contained, with no external dependencies. The complete source code is freely available, under the GNU Public License v3, in the repository hosted at https://github.io/cobilab/FASTA-ANALYSIS.

III. Benchmark

A. Computer characteristics

All results presented here are automatically generated and fully reproducible, encompassing the installation of all benchmark tools, performance comparisons, and plot generation. Reproducing these outcomes is straightforward under a Linux operating system, facilitated by the provided scripts within the repository. In our study, a computer running Linux Ubuntu equipped with 8 Intel® Core™ i7-6700 CPU cores operating at 3.40GHz and 64 GB of RAM was used to execute the benchmark.

B. Datasets

To evaluate the effectiveness of the developed sorting tool and assess the performance of the considered compressors, a combination of real and nearly-synthetic datasets were utilized.

1) Nearly-synthetic sequences: By nearly-synthetic sequences, we refer to real sequences that have been altered according to predefined mutation and transformation rates for testing purposes. The nearly-synthetic sequences integrated into this project were generated using the AlcoR toolkit [29] through a specific process described below. The creation process involves utilizing real sequences, such as a specific virus sequence, as a reference to generate progressively mutated sequences. These mutated sequences are concatenated and merged into a single multi-FASTA file. Due to space constraints, results are provided for a mutation rate of one percent. Nonetheless, the method is made available as open-source, enabling the systematic generation of mutated sequences for analysis and testing purposes.

2) Real (genomic) sequences: The real sequences considered in the benchmark can be divided into two groups. In the first group, each FASTA file represents the sequenced genome from a single organism. These genomes were retrieved from the CNSG Sequence Archive [30] and represent the species Tricitum aestivum [31] and Hellostoma-temmincki [32]. In the second group, collections of related genomes retrieved from the NCBI [33] database were considered for the benchmark. These collections contained a selection of viral and bacterial genomes. For the benchmark, four variations of the genomes retrieved were considered. Table I contains the name, constitution and size of each of the collection datasets considered.

TABLE I
DEFINITION AND COMPOSITION OF THE DATASETS.

Dataset Name	Constitution	Size
Bacterial [34]	Bacteria	3.7 GB
Viral [35]	Viruses	5.7 GB
Viral-Bacterial	Bacteria and Viruses	9.5 GB
Shuffled Viral-Bacterial	Bacteria and Viruses (shuffled)	9.5 GB

Additionally, to harness the full potential impact of sorting for enhancing file compression, we developed a module called "shuffle". This module pseudo-randomly rearranges the order of reads to achieve a more uniform distribution. By doing so, we gain insights into discerning the differences in aggregating similar reads from the most complex cases.

C. Compression benchmark

The benchmark aimed to evaluate the impact of sorting a multi-FASTA file on its compressibility and compare the performance of different sorting techniques and data compressors (general- and specific-purpose). In this benchmark, we considered both general-purpose and DNA-specific compressors. The compressors assessed were gzip, bzip2, LZMA, NAF, MFCompress, JARVIS3, and MBGC. All datasets except the one containing the genome for *Tricitum aestivum* were benchmarked using versions sorted by sequence size, number of AT/CG bases on a sequence, and percentage of AT/CG bases on a sequence.

Table III-C presents some of the results obtained, highlighting the best and worst results in terms of compression gain for each compressor, providing some insights regarding the impact of these factors on the results.

TABLE II
BEST AND WORST PERFORMANCES OBTAINED ON THE BENCHMARK. GIVEN SPACE CONSTRAINS WE HAVE ONLY INCLUDED SOME OF THE RESULTS THAT ARE MARKED WITH THE RANK LABEL (RANKING).

Rank	Dataset	Sort. Type	Tool	Gain (%)
1	Shuffled Viral-Bacterial	CG	JARVIS3	23.90%
5	Bacterial	AT%	MBGC	20.35%
7	Shuffled Viral-Bacterial	size	NAF	16.75%
10	Shuffled Viral-Bacterial	size	LZMA	15.45%
11	Shuffled Viral-Bacterial	size	MFCompress 8	14.85%
16	Shuffled Viral-Bacterial	CG	MFCompress 4	12.83%
26	Shuffled Viral-Bacterial	AT	bzip2	5.05%
56	Shuffled Viral-Bacterial	size	gzip	0.85%
149	Viral	CG	gzip	-0.10%
156	Viral-Bacterial	AT	LZMA	-0.20%
180	Viral	AT	bzip2	-1.80%
185	Viral	AT%	MFCompress 8	-2.80%
187	Viral	CG%	JARVIS3	-2.90%
188	Viral	AT	NAF	-5.83%
190	Viral	AT%	MFCompress 4	-6%
200	Viral-Bacterial	CG%	MBGC	-28.78%

Notice that the compression gain of a given dataset is defined as the difference between the compression ratio in a sorted dataset and the compression ratio in an unsorted dataset. The instance examples on the table are ranked according to their position in the complete benchmark results.

The results shown in Table III-C, with some cases of interest depicted in Figure 2, indicate that there can be significant differences between the performance of the compression tools depending on the input file considered and the sorting method considered.

The best gains for each compressor used were obtained when compressing the "Shuffled Viral-Bacterial" dataset, except for MBGC which obtained higher gains compressing the "Bacterial" dataset. This may be because this tool is specially designed to compress this type of genomes.

The compressor JARVIS3 has obtained the best overall gains in compression when compressing the "Shuffled Viral-Bacterial" dataset and sorting the sequences contained by the number of CG. These gains are especially significant as JARVIS3 is a tool designed and optimized to compress DNA sequences and it run with smaller k-mer size which are usually associated to lower memory capacities.

It's worth noting that the most significant enhancements in the performance of the general-purpose tools were observed during compression of the "Shuffled Viral-Bacterial" dataset and when sequences were sorted by their size. This observation may stem from the fact that these compressors aren't tailored for genome sequence-specific patterns, yet sorting by size facilitates the efficient clustering of similar sequences and its further compression.

Regarding the worse results obtained, general-purpose compressors have obtained smaller losses than DNA-specific ones.

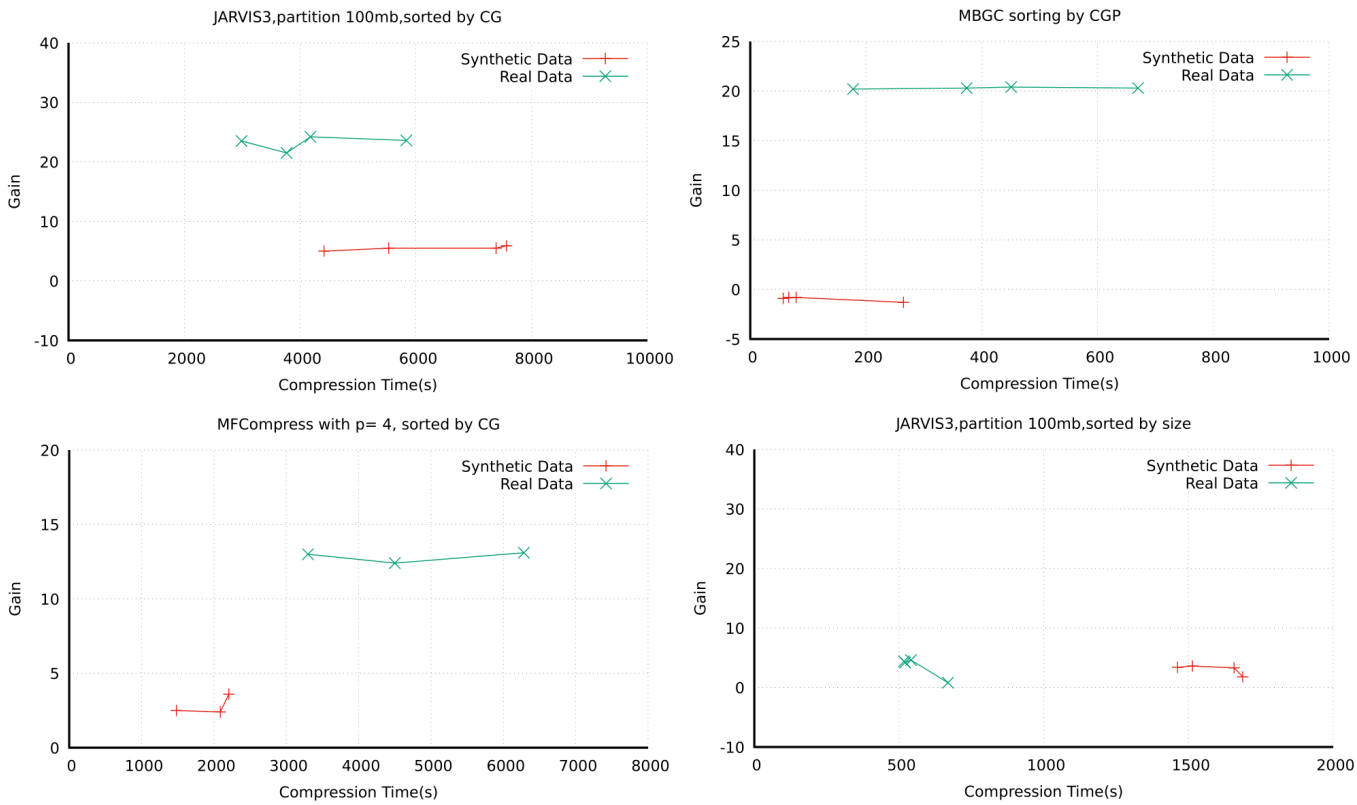

Fig. 2. Depiction of the two best compression gain cases (the figures on top), and two average cases (the figures on the bottom). The green and red line points stand for the synthetic and real data, respectively. In each line points, the multiple points represent different parameters (compression levels) of each compressor.

Additionally, MBGC has registered the most variance in performance between best and worst results when using sorting, which may indicate that this tool is particularly sensitive to the order of the input sequences.

Figure 2 depicts some cases of interest, specifically the two scenarios in which the compression gains are higher and the two scenarios depicting two average cases on which gains are obtained. Each figure contains two distinct executions: the green line represents the execution using one of the real datasets detailed previously and the red line represents the execution using a nearly-synthetic dataset. For each case represented, there is a set of points that represent the specific level of execution of the compressor used. The values on the x-axis represent the time used to perform a compression command in seconds. The values on the y-axis represent the difference between the compression ratio on the unsorted version and the sorted version, designated as the compression gain.

In the top two plots represented in Figure 2, it is possible to see that there is a substantial difference between the gains obtained using real data in relation to the gains obtained using nearly-synthetic data. The compressors considered, JARVIS3 and MBGC, however, have relatively stable results throughout the different compressor levels tested.

The two cases represented in the bottom two plots demonstrate smaller gains in performance. Nevertheless, sorting still provided the best compression gains on real datasets compared to the nearly-synthetic datasets considered.

Although the examples presented here are relatively straightforward to analyze it is not always the case. In some instances, the variance of values throughout the different levels does not allow for definitive conclusions to be drawn in terms of the average value of gain.

We recommend employing FAN with sequence length as the primary feature due to its consistent performance across compression tools and datasets. However, optimizing sorting is viable, especially considering real-world scenarios involving single uploads for multiple downloads (hundreds of thousands or millions), as decompression users typically do not need to handle the sorting task.

IV. Conclusions

We have developed a tool that can sort FASTA files based on various criteria. We have also demonstrated the potential of this approach for optimizing currently available compression methods.

The compressor benchmark developed was executed and it was shown that sorting the FASTA files can lead to improvements to the compression process of up to 23.9%. These gains are particularly significant when considering DNA-specific

tools, as these tools are already optimized for this type of data. Moreover, the sorting feature that add the most consistent improvement was the sequence length, although the best case was given by the number of CG's.

Furthermore, although it was found that not all sorting options represented gains in performance for all datasets considered, it is important to note that the top gains obtained have often surpassed the worst losses for each compressor considered.

Overall, it is also important to note that these gains in performance are very significant given the massive amounts of data that needs to be stored.

ACKNOWLEDGEMENTS

The authors thank the Finnish Computing Competence Infrastructure (FCCI) for supporting this project with computational and data storage resources.

FUNDING

This work was partially funded by National Funds through the FCT (Foundation for Science and Technology), in the context of the project UIDB/00127/2020. M.S. has received funding from the FCT - reference UI/BD/154658/2023. D.P. is funded by national funds through FCT, I.P., under the Scientific Employment Stimulus - Institutional Call - reference CEECINST/00026/2018.

REFERENCES

[1] J.-l. Gailly and M. Adler, "Gnu gzip," *GNU Operating System*, 1992.

[2] Facebook. (2015) Zstandard - real-time data compression algorithm. [Online]. Available: https://facebook.github.io/zstd/

[3] J. Seward. (1996) bzip2 and libbzip2. [Online]. Available: https://www.sourceware.org/bzip2/

[4] Tukaani.org. (2024) Lzma utils. [Online]. Available: https://tukaani.org/lzma/

[5] S. Grumbach and F. Tahi, "Compression of dna sequences," in *[Proceedings] DCC93: Data Compression Conference*. IEEE, 1993, pp. 340–350.

[6] ——, "A new challenge for compression algorithms: genetic sequences," *Information processing & management*, vol. 30, no. 6, pp. 875–886, 1994.

[7] X. Chen, S. Kwong, and M. Li, "A compression algorithm for dna sequences," *IEEE Engineering in Medicine and biology Magazine*, vol. 20, no. 4, pp. 61–66, 2001.

[8] ——, "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.

[9] G. Manzini and M. Rastero, "A simple and fast dna compressor," *Software: Practice and Experience*, vol. 34, no. 14, pp. 1397–1411, 2004.

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

[11] T. Bose, M. H. Mohammed, A. Dutta, and S. S. Mande, "Bind–an algorithm for loss-less compression of nucleotide sequence data," *Journal of biosciences*, vol. 37, pp. 785–789, 2012.

[12] D. Pratas, M. Hosseini, and A. J. Pinho, "GeCo2: An optimized tool for lossless compression and analysis of DNA sequences," in *Practical Applications of Computational Biology and Bioinformatics, 13th International Conference*. Springer, 2020, pp. 137–145.

[13] M. Silva, D. Pratas, and A. J. Pinho, "Efficient DNA sequence compression with neural networks," *GigaScience*, vol. 9, no. 11, p. giaa119, 2020.

[14] D. Pratas, M. Hosseini, J. M. Silva, and A. J. Pinho, "A reference-free lossless compression algorithm for dna sequences using a competitive prediction of two classes of weighted models," *Entropy*, vol. 21, no. 11, p. 1074, 2019.

[15] D. Pratas and A. J. Pinho, "JARVIS2: a data compressor for large genome sequences," in *2023 Data Compression Conference (DCC)*. IEEE, 2023, pp. 288–297.

[16] 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, no. 19, pp. 2527–2529, 2012.

[17] A. J. Pinho and D. Pratas, "Mfcompress: a compression tool for fasta and multi-fasta data," *Bioinformatics*, vol. 30, no. 1, pp. 117–118, 2014.

[18] G. Benoit, C. Lemaitre, D. Lavenier, E. Drezen, T. Dayris, R. Uricaru, and G. Rizk, "Reference-free compression of high throughput sequencing data with a probabilistic de bruijn graph," *BMC bioinformatics*, vol. 16, pp. 1–14, 2015.

[19] K. Kryukov, M. T. Ueda, S. Nakagawa, and T. Imanishi, "Nucleotide archival format (naf) enables efficient lossless reference-free compression of dna sequences," *Bioinformatics*, vol. 35, no. 19, pp. 3826–3828, 2019.

[20] S. Grabowski and T. M. Kowalski, "Mbgc: multiple bacteria genome compressor," *GigaScience*, vol. 11, p. giab099, 2022.

[21] S. Deorowicz, A. Danek, and H. Li, "AGC: compact representation of assembled genomes with fast queries and updates," *Bioinformatics*, vol. 39, no. 3, p. btad097, 2023.

[22] R. Wan, V. N. Anh, and K. Asai, "Transformations for the compression of FASTQ quality scores of next-generation sequencing data," *Bioinformatics*, vol. 28, no. 5, pp. 628–635, 2012.

[23] S. Chandak, K. Tatwawadi, and T. Weissman, "Compression of genomic sequencing reads via hash-based reordering: algorithm and analysis," *Bioinformatics*, vol. 34, no. 4, pp. 558–567, 2018.

[24] Ł. Roguski, I. Ochoa, M. Hernaez, and S. Deorowicz, "FaStore: a space-saving solution for raw sequencing data," *Bioinformatics*, vol. 34, no. 16, pp. 2748–2756, 2018.

[25] D. Lee and G. Song, "FastqCLS: a FASTQ compressor for long-read sequencing via read reordering using a novel scoring model," *Bioinformatics*, vol. 38, no. 2, pp. 351–356, 2022.

[26] H. Sun, Y. Zheng, H. Xie, H. Ma, X. Liu, and G. Wang, "PMFFRC: a large-scale genomic short reads compression optimizer via memory modeling and redundant clustering," *BMC bioinformatics*, vol. 24, no. 1, p. 454, 2023.

[27] D. Pratas and A. J. Pinho, "An experimental sorting method for improving metagenomic data encoding," *arXiv preprint arXiv:2401.01786*, 2024.

[28] ——, "On the approximation of the Kolmogorov complexity for DNA sequences," in *Pattern Recognition and Image Analysis: 8th Iberian Conference, IbPRIA 2017, Faro, Portugal, June 20-23, 2017, Proceedings 8*. Springer, 2017, pp. 259–266.

[29] J. M. Silva, W. Qi, A. J. Pinho, and D. Pratas, "AlcoR: alignment-free simulation, mapping, and visualization of low-complexity regions in biological data," *GigaScience*, vol. 12, p. giad101, 2023.

[30] C. N. G. (CNGB). Cngb sequence archive (cnsa). [Online]. Available: https://db.cngb.org/cnsa/

[31] J. Aury, S. Engelen, B. Istace, C. Monat, P. Lasserre-Zuber, C. Belser, C. Cruaud, H. Rimbert, P. Leroy, S. Arribat, I. Dufau, A. Bellec, D. Grimbichler, N. Papon, E. Paux, M. Ranoux, A. Alberti, P. Wincker, and F. Choulet. (2022) Supporting data for "long-read and chromosome-scale assembly of the hexaploid wheat genome achieves high resolution for research and breeding". [Online]. Available: http://dx.doi.org/10.5524/102205

[32] G. Fan, Y. Song, L. Yang, X. Huang, S. Zhang, M. Zhang, X. Yang, Y. Chang, H. Zhang, Y. Li, S. Liu, L. Yu, J. Chu, I. Seim, C. Feng, T. Near, R. Wing, W. Wang, K. Wang, J. Wang, X. Xu, H. Yang, X. Liu, N. Chen, and S. He. (2020) Genomic data of the kissing gourami, helostoma temminkii. [Online]. Available: http://dx.doi.org/10.5524/102190

[33] N. C. for Biotechnology Information NCBI. (1988) National center for biotechnology information. [Online]. Available: http://www.ncbi.nlm.nih.gov/

[34] NCBI. Bacterial genomes. [Online]. Available: https://www.ncbi.nlm.nih.gov/datasets/genome/?taxon=2

[35] ——. Viral genomes. [Online]. Available: https://www.ncbi.nlm.nih.gov/datasets/genome/?taxon=10239