GenoDedup: Similarity-Based Deduplication and Delta-Encoding for Genome Sequencing Data (Supplementary Material)

Vinicius Cogo, João Paulo, and Alysson Bessani

Abstract—This is the supplementary material of the article "GenoDedup: Similarity-Based Deduplication and Delta-Encoding for Genome Sequencing Data".

Index Terms—Storage, Deduplication, Compression, Genome Sequencing Data

1 Hamming and Levenshtein Encodings

In this section, we present the encoding details for the Hamming and the Levenshtein algorithms. In the Hamming algorithm, the list of edit operations between two strings with the same size considers only `UNMODIFIED` and `SUBSTITUTION` operations. One may encode the result using 1 bit per edit operation (e.g., 0 for `UNMODIFIED` or 1 for `SUBSTITUTION`), plus the symbol encoding when characters do not match. We employ the mentioned Hamming encoding with the use of Huffman codes [1] to describe the divergent characters. The resulting size (in bits) is given by Equation (1).

$$SizeH = M + C_0 + \ell + (S * huf(\bullet)) \quad (1)$$

The size of the candidate pointer M (in bits) corresponds to $M = log_2(N)$, where N is the expected number of entries in the deduplication index. C_0 describes the first character in the original sequence, which allows one to initiate a chain of conversions from circular delta values to the original quality score sequence. C_0 is unnecessary for DNA sequences since these do not use delta values. It can be a fixed-size value or a Huffman code based on the distribution of the first character observed from several FASTQ files. ℓ is the length of the entry sequence, S is the number of characters that differ from the base chunk, and $huf(\bullet)$ is the function that returns the size of the Huffman code for each divergent character. The best-case scenario for the Hamming encoding results in ℓ bits per sequence, an exact match, which leads to a compression ratio upper bounded by $8\times$.

The Levenshtein algorithm verifies four different edit operations (`UNMODIFIED`, `SUBSTITUTION`, `DELETE`, and `INSERT`), which requires 2 bits per operation. Similarly to the previous encoding, we use Huffman codes to describe characters in `SUBSTITUTION` and `INSERT` operations. The resulting size (in bits) is given by Equation (2).

$$SizeL = M + C_0 + 2 * \ell + ((S + I) * huf(\bullet)) \quad (2)$$

S and I denote the number of `SUBSTITUTION` and `INSERT` operations. In this Levenshtein encoding, the best-case scenario results in $2 * \ell$ bits when all characters match, which is the equivalent to an upper bound of $4\times$ in compression ratio.

We also evaluated a *Delta-Levenshtein* encoding, similar to the proposed *Delta-Hamming*. However, it has a limited impact because it requires more bits to represent its additional operations and it reduces very little the total number of edit operations of deduplicated sequences.

2 Datasets

Our analyses use five representative FASTQ files of human genomes from the 1000 Genomes Project [2]: SRR400039, SRR618664, SRR618666, SRR618669, and SRR622458. They are human genomes sequenced with the Illumina HiSeq 2000 platform [3]. To the best of our knowledge, this machine was the most used NGS machine in sequencing laboratories around the world when we started this work [4]. Additionally, some of the selected genomes were also used in other papers on FASTQ compression (e.g., SRR400039 in Quip's paper [5]). Only the FASTQ file from the first end of these genomes are considered in our analyses, but they sum up 265GB of data and result in almost one billion FASTQ entries.

Table S1 provides additional details about the selected datasets. It contains the size of the selected genomes (in B, MB, and GB), the size (in GB) of the comment, DNA, and QS portions separately, the number of FASTQ entries, the length of each DNA and QS sequence within an entry, and the sequencing coverage [6].

3 Tools

We evaluated several generic and specialized compression tools available in the literature [13], [16]. Ten tools were selected due to their completeness, correctness, compression ratio, and performance: GZIP [7], pigz [8], BSC [9], ZPAQ [10], SeqDB [11], DSRC2 [12], Quip [5], Fqzcomp [13], FaStore [14],

- *VC and AB are with LASIGE, Faculdade de Ciências, Universidade de Lisboa, Lisboa, Portugal. JP is with HASLab—High-Assurance Software Lab, INESC TEC & U. Minho, Portugal. Authors e-mails: vielmo@lasige.di.fc.ul.pt, jtpaulo@inesctec.pt, and anbessani@ciencias.ulisboa.pt.*
- *This work was supported by the European Commission, through SUPERCLOUD project (H2020-ICT-643964), and by National Funds through the Portuguese funding agency, FCT—Fundação para a Ciência e a Tecnologia, within project IRCoC (PTDC/EEISCR/6970/2014) and research units LASIGE (UIDB/00408/2020 and UIDP/00408/2020) and INESC TEC (UIDB/50014/2020).*

Table S1
The datasets used in this work, their size (in B, MB, and GB), the size (in GB) of the comment, DNA, and QS portions separately, the number of FASTQ entries, the length of each DNA and QS sequence within an entry, and the sequencing coverage.

Genome	Size (B)	Size (MB)	Size (GB)	Comments	DNA	QS	Entries	Length	Coverage
SRR400039_1	33712200479	33712.2	33.7	8.3	12.7	12.7	124331027	101	3.91×
SRR618664_1	64619170090	64619.2	64.8	16.2	24.3	24.3	239715311	100	7.46×
SRR618666_1	62342666306	62342.7	62.3	15.7	23.4	23.4	231285558	100	7.20×
SRR618669_1	79617500309	79617.5	79.6	20.0	29.8	29.8	295256611	100	9.20×
SRR622458_1†	23617651110	23617.7	23.6	4.0	9.8	9.8	96097046	101	3.02×
Total	**263909188294**	**263909.2**	**264.2**	**64.2**	**100**	**100**	**986685553**	—	—

Table S2
Tools, versions and parameters used in our comparisons. $F1 = original FASTQ, $F2 = compressed file, $F3 = decompressed file.

Tool	Version	Compression	Decompression
GZIP [7]	1.6	$F1	-d $F2
pigz [8]	2.3.1	$F1	-d $F2
BSC [9]	3.1.0	e $F1 $F2	d $F2 $F3
ZPAQ [10]	7.15	a $F2 $F1 -m5 -t24	x $F2 -t24 -force
SeqDB [11]	0.2.1	120 120 $F1 $F2	$F2 > $F3
DSRC2 [12]	2.00	c -t24 $F1 $F2	d -t24 $F2 $F3
Quip [5]	1.1.8	$F1	-d $F2
Fqzcomp [13]	4.6	$F1 $F2	-d -X $F2 $F3
FaStore [14]	0.8.0	--lossless --in $F1 --out $F2 --threads 24	--in $F2 --out $F3 --threads 24
SPRING [15]	1.0	-c -i $F1 -o $F2 -t 24	-d -i $F2 -o $F3 -t 24

and SPRING [15]. Table S2 presents the version of the tool used in our tests and the arguments passed to compress and decompress data. In this table, $F1 is the path to the original FASTQ file (usually passed as input to the compressor), $F2 is the path to the resulting compressed file, and $F3 is the path to the decompressed file (usually different from $F1 to compare their content hashes later).

We experimented with many other tools, but they were discarded from the comparison for several reasons, which we describe in the remaining of this section.

LZMA2 is an improved multi-threaded version of the generic compressor entitled Lempel-Ziv-Markov chain algorithm (LZMA), which was implemented in the 7-Zip compressor [17]. We experimented the FASTQ file of the SRR400039 genome with this tool. However, it has compressed this file 3.26× at a throughput of 1.36MB/s and decompressed it at 14.1MB/s.

PPM (Prediction by partial matching) is another interesting generic compression algorithm to be considered [18]. It is also one of the possible compression methods used by 7-Zip. We experimented with a tool called PPMd with the FASTQ file of the SRR400039 genome, and it has compressed this file 3.95× at a throughput of 17.1MB/s and decompressed it at 3.86MB/s.

These two tools were discarded from our complete comparison because they are slower than GZIP and have a lower compression ratio than ZPAQ. Other tools fall in the same situation since these do not surpass the performance of GZIP nor compress more than ZPAQ, namely: BZIP2 [19], G-SQZ [20], and KIC [21]. ZPAQ achieving a better compression ratio than PPMd is usually justified by the fact that ZPAQ (and similar context mixing algorithms) use many prediction models from different contexts while PPMd uses a single one [22].

The idea of context mixing brings another tool, called LPAQ8 [23], into the discussion. LPAQ8 is a slow compressor that reaches the best compression ratio in many benchmarks and was developed by the same person that developed ZPAQ. Other FASTQ compression tools (e.g., LFQC [24] and LFastqC [22]) separate the different portions of FASTQ files (i.e., comments, DNA, and QS) and compress them separately.

Both LFQC [24] and LFastqC [22] use LPAQ8 to compress the quality score sequences, while for the comments and DNA sequences the former uses ZPAQ and the latter uses Mfcompress [25]. However, LPAQ8 uses signed integers to track the file length, which makes it crash if its input files have more than 2GB. As presented in the seventh column of Table S1, all QS portions of our datasets have more than 2GB, which prevents us from using any tool that employs LPAQ8 for compressing these portions.

Developers from LFQC suggested, in an issue report[1], to replace LPAQ8 by ZPAQ to compress the QS sequences when they are bigger than 2GB. However, it would make LFQC's results very similar to the ones from ZPAQ, which vouched for discarding it from the complete comparison.

Finally, another tool that separates the portions of FASTQ files and uses the Mfcompress [25] for the DNA sequences is the MZPAQ [26]. As the name suggests, it uses the ZPAQ algorithm for the QS sequences. Unfortunately, we were not able to find a publicly available source-code or executable of MZPAQ, which prevents us from adding it to the complete evaluation. We even considered running its underlying software separately, but Mfcompress has compressed a human genome only 34.3% more than GZIP [25] and using the ZPAQ to compress the QS sequences would result in ZPAQ's low restore throughput, incurring in the same limitations as the previously mentioned generic tools.

1. https://github.com/mariusmni/lfqc/issues/4

Table S3
Compressed size in MB.

Genome	GZIP	pigz	BSC	ZPAQ	SeqDB	DSRC	Quip	Fqzcomp	FaStore	SPRING	GenoDedup
SRR400039_1	12041	12036	8441	7617	16728	8693	7410	7454	7180	**6509**	8202
SRR618664_1	21498	21511	14930	13355	32204	15239	12971	13094	N/A	**10702**	14624
SRR618666_1	21298	21277	14849	13299	31117	15132	12920	13054	N/A	**10674**	14318
SRR618669_1	26304	26302	18253	162958	39580	18572	15833	16025	N/A	**12869**	17627
SRR622458_1	5408	5401	4051	3206	12274	5607	4909	4706	3826	**3438**	7751

Table S4
Compression ratio (i.e., $original\_size/compressed\_size$).

Genome	GZIP	pigz	BSC	ZPAQ	SeqDB	DSRC	Quip	Fqzcomp	FaStore	SPRING	GenoDedup
SRR400039_1	2.800	2.801	3.994	4.426	2.015	3.878	4.550	4.523	4.695	**5.179**	4.110
SRR618664_1	3.006	3.004	4.328	4.839	2.007	4.240	4.982	4.935	N/A	**6.038**	4.419
SRR618666_1	2.927	2.930	4.198	4.688	2.003	4.120	4.825	4.776	N/A	**5.841**	4.354
SRR618669_1	3.027	3.027	4.362	4.886	2.012	4.287	5.029	4.968	N/A	**6.187**	4.517
SRR622458_1	4.367	4.373	5.830	7.367	1.924	4.212	4.811	5.018	6.173	**6.869**	3.047
Average	3.225	3.227	4.543	5.241	1,992	4.148	4.839	4.844	5.434	**6.023**	4.089

Table S5
Compression throughput in MB/s (i.e., $original\_size/compression\_time$).

Genome	GZIP	pigz	BSC	ZPAQ	SeqDB	DSRC	Quip	Fqzcomp	FaStore	SPRING
SRR400039_1	12.804	251.584	161.302	5.416	474.820	**636.079**	28.888	59.457	14.053	35.301
SRR618664_1	13.460	255.412	167.842	5.347	425.126	**1576.077**	28.108	61.250	N/A	41.449
SRR618666_1	13.547	259.761	133.211	5.302	418.407	**1558.567**	29.200	61.060	N/A	40.720
SRR618669_1	13.772	201.563	136.800	5.408	277.413	**1421.741**	28.404	61.671	N/A	35.928
SRR622458_1	24.026	437.364	200.150	5.088	481.993	**1686.975**	28.979	59.044	36.960	62.316
Average	15.522	281.137	159.861	5.312	415.552	**1375.888**	28.716	60.497	25.506	43.143

Table S6
Decompression throughput in MB/s (i.e., $compressed\_size/decompression\_time$).

Genome	GZIP	pigz	BSC	ZPAQ	SeqDB	DSRC	Quip	Fqzcomp	FaStore	SPRING
SRR400039	44.596	69.571	45.380	1.210	**164.009**	140.203	3.604	9.847	47.871	22.522
SRR618664_1	44.601	70.528	43.528	1.112	**82.576**	75.815	3.293	9.578	N/A	19.493
SRR618666_1	44.650	69.762	35.694	1.155	**92.611**	76.041	3.426	9.641	N/A	21.391
SRR618669_1	38.012	58.579	36.727	1.110	**102.276**	54.303	3.235	9.561	N/A	18.953
SRR622458_1	34.891	62.075	69.842	0.688	197.975	**280.337**	3.604	9.228	42.511	22.183
Average	41.350	66.103	46.234	1.055	**127.890**	125.340	3.432	9.571	45.191	20.908

4 Experimental Evaluation

In this section, we present the complete tables comparing the ten selected tools using the five selected datasets in terms of compressed size (Table S3), compression ratio (Table S4), compression throughput (Table S5), and decompression throughput (Table S6).

References

[1] D. A. Huffman, "A method for the construction of minimum-redundancy codes," *Proc. of the Institute of Radio Engineers (IRE)*, vol. 40, no. 9, pp. 1098–1101, 1952.

[2] L. Clarke *et al.*, "The 1000 Genomes Project: data management and community access," *Nature methods*, vol. 9, no. 5, pp. 459–462, 2012.

[3] I. Illumina, "Hiseq 2000 sequencing system," 2010, in: https://www.illumina.com/documents/products/datasheets/datasheet_hiseq2000.pdf. Accessed on Feb. 10, 2020.

[4] J. Hadfield, "NGS mapped," 2020, in: http://enseqlopedia.com/ngs-mapped/. Accessed on Feb. 10, 2020.

[5] 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, no. 22, p. e171, 2012.

[6] I. Illumina, "Estimating sequencing coverage," 2014, available at https://jp.support.illumina.com/content/dam/illumina-marketing/documents/products/technotes/technote_coverage_calculation.pdf. Accessed on Feb. 10, 2020.

[7] P. Deutsch, "GZIP file format specification version 4.3," Internet Requests for Comments, RFC Editor, RFC 1952, May 1996.

[8] M. Adler, "pigz - parallel gzip," 2019, in: https://zlib.net/pigz/. Accessed on Feb. 10, 2020.

[9] I. Grebnov, "BSC: High performance data compression library," 2020, in: http://libbsc.com/. Accessed on Feb. 10, 2020.

[10] M. Mahoney, "The ZPAQ compression algorithm," 2015, in: http://mattmahoney.net/dc/zpaq_compression.pdf. Accessed on Feb. 10, 2020.

[11] M. Howison, "High-throughput compression of FASTQ data with SeqDB," *IEEE/ACM TCBB*, vol. 10, no. 1, pp. 213–218, 2013.

[12] Ł. Roguski and S. Deorowicz, "DSRC 2—industry-oriented compression of FASTQ files," *Bioinformatics*, vol. 30, no. 15, pp. 2213–2215, 2014.
[13] J. K. Bonfield and M. V. Mahoney, "Compression of FASTQ and SAM format sequencing data," *PLOS ONE*, vol. 8, no. 3, p. e59190, 2013.
[14] L. 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.
[15] S. Chandak, K. Tatwawadi, I. Ochoa, M. Hernaez, and T. Weissman, "SPRING: a next-generation compressor for FASTQ data," *Bioinformatics*, vol. 35, no. 15, pp. 2674–2676, 2018.
[16] S. Deorowicz and S. Grabowski, "Compression of DNA sequence reads in FASTQ format," *Bioinformatics*, vol. 27, no. 6, pp. 860–862, 2011.
[17] I. Pavlov, "LZMA," 2019, in: https://www.7-zip.org/. Accessed on Feb. 10, 2020.
[18] A. Moffat, "Implementing the ppm data compression scheme," *IEEE Transactions on Communications*, vol. 38, no. 11, pp. 1917–1921, 1990.
[19] J. Seward, "bzip2 and libbzip2," 2019, in: https://github.com/enthought/bzip2-1.0.6. Accessed on Feb. 10, 2020.
[20] W. Tembe, J. Lowey, and E. Suh, "G-SQZ: compact encoding of genomic sequence and quality data," *Bioinformatics*, vol. 26, no. 17, pp. 2192–2194, 2010.
[21] Y. Zhang, K. Patel, T. Endrawis, A. Bowers, and Y. Sun, "A FASTQ compressor based on integer-mapped k-mer indexing for biologist," *Gene*, vol. 579, no. 1, pp. 75–81, 2016.
[22] S. Al Yami and C.-H. Huang, "LFastqC: A lossless non-reference-based FASTQ compressor," *PLOS ONE*, vol. 14, no. 11, 2019.
[23] M. Mahoney, "The LPAQ compression algorithm," 2007, in: http://mattmahoney.net/dc/#lpaq. Accessed on Feb. 10, 2020.
[24] M. Nicolae, S. Pathak, and S. Rajasekaran, "LFQC: a lossless compression algorithm for FASTQ files," *Bioinformatics*, vol. 31, no. 20, pp. 3276–3281, 2015.
[25] 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.
[26] A. El Allali and M. Arshad, "MZPAQ: a FASTQ data compression tool," *Source code for biology and medicine*, vol. 14, no. 1, p. 3, 2019.