INTERNATIONAL ORGANISATION FOR STANDARDISATION
ORGANISATION INTERNATIONALE DE NORMALISATION
ISO/IEC JTC 1/SC 29/WG 11
CODING OF MOVING PICTURES AND AUDIO

ISO/IEC JTC 1/SC 29/WG 11 M39176
Chengdu, China – October 2016

Source: **CNAG-CRG, Stanford University, DMAG-UPC, Silesian University of Technology, BSC**
Status: **Proposal**
Title: **A proposal for a compression technology based in ORCOM and QVZ**
Response to the Joint Call for Proposals for Genomic Information Compression and Storage
Authors: **Łukasz Roguski (Centro Nacional de Análisis Genómico – Centre de Regulació Genòmica (CNAG – CRG)),**
Mikel Hernáez, Idoia Ochoa (Stanford University),
Jaime Delgado, Silvia Llorente (Distributed Multimedia Applications Group – Universitat Politècnica de Catalunya),
Sebastian Deorowicz (Institute of Informatics, Silesian University of Technology),
Josep Lluís Gelpí, Dmitry Repchevsky (Barcelona Supercomputing Center)

Index

1 Introduction

We propose an efficient reference-free compression algorithm which can be applied for compressing data present both in FASTQ and SAM formats. In this proposal, however, we focus on the FASTQ format use-case. The main compressor workflow and DNA sequence compression algorithm is based on the ORCOM [1] publication with minor modifications. The quality scores compression algorithm is based on the QVZ [2] publication.

The description of ORCOM algorithm, with minor changes, comes from the official publication [1]. Similarly, the description of QVZ comes from the official publication [2].

2 Motivation

The technology that we present takes advantage of the high sequence redundancy present in the reads generated by next-generation sequencing platforms. This problem becomes especially visible in high coverage datasets, which are "the de-facto standard" when performing whole genome or exome sequencing. In addition, it has been proven that quality scores are more difficult to compress [3], due to their higher entropy (which can be partially explained by the noisy nature of the quality scores).

Therefore, the proposed technology offers the option to lossily compress the quality scores while preserving the nucleotide sequences (i.e. the reads), which allows for a flexible adaptation of the compression method to the user requirements. The technology offers several degrees of compression, both for processing the reads and quality scores (e.g. tuning the distortion level of the reconstructed quality scores from lossless to almost zero bits per quality score).

3 General information

3.1 Compression workflow

To exploit the redundancy present in high-coverage sequencing data, the compression workflow is divided into two phases. The reads from FASTQ file(s) are firstly distributed into independent bins following a defined similarity criterion. Specifically, in the proposed method we use shared signatures as the similarity criterion. Next, the reads inside the bins are compressed on a bin-by-bin basis. The decompression algorithm consists on the reverse of the second phase. Note that the decompressor does not need to perform any post-processing of the reads. Since the bins are independent, the compression and decompression algorithms are highly parallelizable.

Such approach could be also applied to SAM file(s). However, note that the resulting alignments in the reconstructed file would be potentially reshuffled, and therefore not sorted by their genomic position. Some applications may require alignments being sorted by their genomic position. Thus, in the following, we primarily focus on the FASTQ format.

3.2 Definitions

3.2.1 Minimizer (ORCOM)

Let s = $s[0]s[1]...s[n-1]$ be a string of length n over a finite alphabet Σ of size σ.

We use the following notation, assuming $0 \leqslant i \leqslant j < n$: $s[i]$ denotes the $(i+1)$th symbol of s, $s[i...j]$ denotes the substring $s[i]s[i+1]...s[j]$ and $s \circ t$ the concatenation of strings s and t.

3.2.2 Canonical minimizer (ORCOM)

We assume that the alphabet size is equal to σ = 5 (ACGTN) and denotes the length of the minimizer as *p*. A reasonable value of *p* is about 10 (the default parameter). These values of σ and *p* would yield 5^{10} = 9.77M bins, which would be too much for efficient processing and available resources to store or represent the data (memory and disk). Thus, we reduce the alphabet size σ' = 4 (ACGT), which yields 4^{10}+ 1 bins (all minimizers containing at least one unsupported symbol N are mapped to a single bin, labelled N).

As DNA sequences can be read in two different directions, forward and reverse (with complements of each nucleotide), we also process each read twice in its given and reverse-complemented form. Additionally, we introduce a 'skip zone' *z*, that is, do not look for minimizers in the read suffixes of a given length (default: *z* = 12 symbols). Therefore, the minimizers are sought over $2(r - z - p + 1)$ resulting *p*-mers, where *r* is the read length. We call them *canonical minimizers*.

3.2.3 Signature (ORCOM)

The problem with strictly defined minimizers is uneven bin distribution, which influences the efficiency of compression and computational resources. To possibly mitigate these problems, we forbid some canonical minimizers, namely those that contain any triple AAA (the most frequent), CCC, GGG or TTT or at least one N. The allowed canonical minimizers are further called *signatures*.

3.2.4 Lloyd-Max Quantizer (QVZ)

Given a random variable X governed by the probability mass function P(·) over the alphabet of size *K*, let $D \in \mathbb{R}^{K \times K}$ be a distortion matrix where each entry $D_{x,y}=d(x,y)$ is the penalty for reconstructing symbol x as y. We further define $\mathcal{Y}$ to be the alphabet of the quantized values of size $M \leq K$.

Thus, a Lloyd–Max quantizer, denoted hereafter as LM(·), is a mapping $\mathcal{X} \to \mathcal{Y}$ that minimizes an expected distortion. Specifically, the Lloyd–Max quantizer seeks to find a collection of boundary points $b_k \in \mathcal{X}$ and reconstruction points $y_k \in \mathcal{Y}$, where $k \in \{1,2,\ldots,M\}$, such that the quantized value of symbol $x \in \mathcal{X}$ is given by the reconstruction point of the region to which it belongs. For region k, any $x \in \{b_{k-1},\ldots,b_k-1\}$ is mapped to y_k, with b_0 being the lowest score in the quality alphabet and b_M the highest score plus one. Thus, the Lloyd–Max quantizer aims to minimize the expected distortion by solving

$$\{b_k, y_k\}_{k=1}^{M} = \operatorname{argmin}_{b_k, y_k} \sum_{j=1}^{M} \sum_{x=b_{j-1}}^{b_j-1} P(x) d(x, y_j)$$

To approximately solve the previous equation, which is an integer programming problem, we employ an algorithm which is initialized with uniformly spaced boundary values and reconstruction points taken at the midpoint of these bins. For an arbitrary *D* and *P*(·), this problem requires an exhaustive search. We assume that the distortion measure *d(x, y)* is quasi-convex over y with a minimum at $y = x$, i.e. when $x \leq y1 \leq y2$ or $y2 \leq y1 \leq x, d(x,y1) \leq d(x,y2)$. If the distortion measure is quasi-convex, an exchange argument suffices to show the optimality of contiguous quantization bins and a reconstruction point within the bin.

Given a distortion matrix D, the defined Lloyd–Max quantizer depends on the number of regions M and the input probability mass function $P(\cdot)$. Therefore, we denote the Lloyd–Max quantizer with M regions as $\mathrm{LM}_{PM}(\cdot)$ and the quantized value of a symbol $x \in \mathcal{X}$ as $\mathrm{LM}_{PM}(x)$.

An ideal lossless compressor applied to the quantized values can achieve a rate equal to the entropy of $\mathrm{LM}_{PM}(X)$, which we denote by $H(\mathrm{LM}_{PM}(X))$. For a fixed probability mass function $P(\cdot)$, the only varying parameter is the number of regions M. Since M needs to be an integer, not all rates are achievable. Because we are interested in achieving an arbitrary rate R, we define an extended version of the LM quantizer, denoted as LME. The extended quantizer consists of two LM quantizers with the numbers of regions given by ρ and $\rho+1$, each of them used with probability $1-r$ and r, respectively (where $0 \leq r \leq 1$). Specifically, ρ is given by the maximum number of regions such that $H(\mathrm{LM}_{P\rho}(X)) < R$ (which implies $H(\mathrm{LM}_{P\rho+1}(X)) > R$). Then, the probability r is chosen such that the average entropy (and hence the rate) is equal to R, the desired rate.

4 Phase I – records binning

4.1 Reads distribution

Each bin corresponds to a signature. For each read present in the FASTQ file(s), its signature is computed. The reads are distributed into bins according to their computed signature. If a valid signature is not found for a read, the read is placed in the special N bin.

4.2 Reads sorting

Once the bins are filled with records, the records within each bin are reordered such that overlapping reads are possibly close to each other. Specifically, we sort the reads s_i, for all i, according to the lexicographical order of the string $s_i[j \ldots r\text{-}1] \circ s_i[0 \ldots j\text{-}1]$, where j is the beginning position of the signature for the read s_i. Such reordering has a major positive impact on the compression, since the overlapping reads are with high probability close to each other in the reordered array.

4.3 QVZ statistics calculation

QVZ assumes the quality scores are generated from a temporal Markov model of order 1. That is, it assumes the probability of having a quality score X at position i depends on the quality score at position i-1, within the same read. Note that the transition probabilities at each position may differ, due to the temporal nature of the Markov model.

Thus the corresponding transition probabilities are given by $P(X_i|X_{i-1})$, for i=1, 2, …, r, where r is the length of the read and $P(X_1|X_0) = P(X_1)$. These probabilities are computed empirically from the data.

4.4 QVZ codebook computation

Since we assume the data follows a Markov-1 model, for a given position i, where i goes from 1 to r, the length of the read, we design as many quantizers Q^i_q as there were unique possible quantized values q in the previous context i – 1. This collection of quantizers forms the codebook for QVZ. For an unquantized quality score X_i, we denote the quantized version as Q_i, so $[Q_1, Q_2, \ldots, Q_r]$ is the random vector representing a quantized sequence. The quantizers are defined as

$$\mathcal{Q}^1 = LME^{P(X_1)}_{\alpha H(X_1)}$$

$$\mathcal{Q}_q^i = LME_{\alpha H(X_i|Q_{i-1}=q)}^{P(X_i|Q_{i-1}=q)}, \text{ for } i = 2, \ldots, r$$

where $\alpha \in [0, 1]$ is the desired compression factor. $\alpha = 0$ corresponds to 0 rate encoding, $\alpha = 1$ to lossless compression, and any value in between scales the input file size by that amount. Note that the entropies can be directly computed from the corresponding empirical probabilities.

4.5 QVZ Computation of the probability P

Next we show how the probabilities needed for the LMEs are computed.

In order to compute the quantizers defined above, we require $P(X_{i+1} \mid Q_i)$, which must be computed from the empirical statistics $P(X_{i+1} \mid X_i)$ found earlier. The first step is to calculate $P(Q_i \mid X_i)$ recursively, and then to apply Bayes rule and the Markov Chain property to find the desired probability. Once $P(Q_i \mid X_i)$ is computed, we can compute the desired probability $P(X_{i+1} \mid Q_i)$ (we refer to [2] for details on the specific computations).

4.6 QVZ Codebook generation

For the generation of the codebook, we need to calculate empirical probabilities $P(X_{i+1} \mid Q_i)$ and the value of the parameter α provided by the user. With this information we can compute the quantizers $\mathcal{Q}^1$and $\mathcal{Q}_q^i$, for $i \in [2, \ldots, r]$.

5 Phase II - records compression

5.1 DNA sequence compression

We maintain a sequence buffer (sliding window) of *m* previous reads, storing also the position of the signature in each read. For each read, we seek the read from the buffer that maximizes the overlap. The distance between a pair of considered reads depends on the number of elementary operations transforming one into another. For example, if the pair of reads is:

```
AAC<b>G</b>TXXXXCGGCAT
  C<b>C</b>TXXXXCGGCAT<b>CC</b>
```

to find that they differ in one mismatch (G versus C) and 2 end symbols of the second read have to be inserted, hence the distance is $c_m \times 1 + c_i \times 2$, where c_m and c_i are the mismatch and insert cost, respectively. The default values for the parameters are $c_m = 2$ and $c_i = 1$. Thus, in the above example, the final distance is $2 \times 1 + 1 \times 2 = 4$. The read among the *m* previous ones that minimizes such distance, and is not greater than max_{dist} (defined as half of the read length) is considered a reference for the current read.

Next, the referential matching data are sent into a few buffers:

- Flags

Values from $\{f_{copy}, f_{diss}, f_{ex}, f_{oth}\}$ with the following meaning:

- f_{copy} - the current read is identical to the previous one
- f_{diss} - the read is not similar to any read from the buffer; more precisely, the similarity distance exceeds a specified threshold max_{dist}

- f_{ex} - the read overlaps with some read from the buffer without mismatches
- f_{oth} - the read overlaps with some read from the buffer, but not in a way corresponding to flag f_{ex}

- Lengths
Read lengths are stored here

- The five letter buffers: N, A, C, G, T
(used only if 'Flag' is set to f_{ex} or f_{oth})
'N' stores (i) all mismatching symbols from the current read where at the corresponding position of the references read there is a symbol N, and (ii) all trailing symbols from the current read beyond the match (i.e. CC in the example above).

'X' , for $X \in \{A,C,G,T\}$, stores all mismatching symbols from the current read, where at the corresponding position of the referenced read there is a symbol X (in our running example, the mismatching C would be encoded in the letter buffer 'G'. Note that the alphabet size for any 'X' letter buffer is 4, that is, $\{A,C,G,T,N\} \setminus \{X\}$.

- Shift
(Used only if "Flag" is set to f_{ex} or f_{oth})
Stores the offsets of the current reads against their referenced read. The offset may be negative. For our running example, the offset is +2.

- Matches
(Used only if "Flag" is set to f_{oth})
Stores information on mismatch positions. For our running example, the matching area has 13 symbols, but 4 of them belong to the signature and can thus be omitted (as the signature's position in the current read is known from the corresponding value in the buffer "Shift").

- HReads
(Used only if "Flag" is set to f_{diss})
Here the "hard" reads (not similar enough to any read from the buffer) are dispatched. They are stored almost verbatim: the only change in the representation is to replace the signature with an extra control symbol not present in the sequence alphabet.

- Rev
Contains binary flags that specify if each read is processed directly or first reverse-complemented.

Finally, buffers are compressed. 'Flags' and 'Rev' are compressed with a simple range coder (RC) of order-4. Other remaining buffers are compressed with a general-purpose compressor PPMd also using order-4 context models.

5.2 Quality scores compression

The encoding process is summarized in Algorithm 1 in Figure 1. First, we generate the codebook and quantizers. For each read, we quantize all scores sequentially, with each value forming the left context for the next value. As they are quantized, scores are passed to an adaptive arithmetic encoder, which uses a separate model for each position and context.

Algorithm 1 Encoding of quality scores

Input: Set of N reads $\{\mathbf{X}_j\}_{j=1}^{N}$
Output: Set of quantizers $\{\mathcal{Q}_q^l\}$ (codebook) and compressed representation of reads
Compute empirical statistics of input reads
Compute codebook $\{\mathcal{Q}_q^l\}$ according to Algorithm 1
for all $j = 1$ to N **do**
  $[X_1, \ldots, X\ ] \leftarrow \mathbf{X}_j$
  $Q_1 \leftarrow \mathcal{Q}^1(X_1)$
  for all $i = 2$ to L **do**
    $Q_i \leftarrow \mathcal{Q}^i_{Q_{i-1}}(X_i)$
  end for
  Pass $[Q_1, \ldots, Q_L]$ to arithmetic encoder
end for

Figure 1. Algorithm 1

6 Covered MPEG requirements

Note that the proposed technology is not a stand-alone FASTQ format compressor. However, it can be used to build one. Alternatively, technology can be used to compress the sequence and information streams independently, not being strictly limited to fixed data file format. Therefore, the FASTQ format requirements specified by MPEG are covered partially [4] and are described in the table below.

Req ID	Requirement	Rationale/Notes
1.1	The solution shall support lossless compression of reads headers.	Does not cover
1.2	The solution shall support lossy compression of reads headers by preserving at least a unique read identifier and pairing information when available.	Does not cover
1.3	The solution shall support preservation of pairing information	Does not cover
1.4	The solution shall support lossless compression of nucleotides sequences supporting a minimum of 5 symbols (A, C, G, T, N)	Yes
1.5	The solution shall support lossless compression of quality scores.	Yes
1.6	The solution shall support lossy compression of	Yes

	quality scores.	
1.7.1	The solution shall structure compressed data so that parallel processing is enabled and compression efficiency is preserved	Yes
1.7.2	The solution should structure compressed data so that efficient querying of data is enabled and compression efficiency is preserved	Possible decompression of bins by specifying their signature
1.8	The solution shall preserve the association among headers, nucleotides and quality scores	Solution does not cover read identifiers compression

7 Test results

The tests have been done using the following files, which are fully described in document [5]:

01 (HG)
07 (HG)
14 (BC)
18 (BC)
20 (MT)
24 (PL)
25 (PL)

11, 12 and 26 have been used for QUALs validation (QVZ).

The complete numbers are described in the Excel file accompanying this document.

8 Acknowledgements

The work done in this proposal has been partially supported by the Spanish Government under the project: Secure Genomic Information Compression (GenCom, TEC2015-67774-C2-1-R and TEC2015-67774-C2-2-R), the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement No. 305444 (RD-Connect), a fellowship from the Basque Government, a Stanford Graduate Fellowships Program in Science and Engineering, the Stanford Data Science Initiative (SDSI) and an NIH grant with number 1 U01 CA198943-01.

9 References

[1] Szymon Grabowski, Sebastian Deorowicz, Lukasz Roguski, Disk-based compression of data from genome sequencing. Bioinformatics. 2015 May 1;31(9):1389-95
[2] Greg Malysa, Mikel Hernaez, Idoia Ochoa, Milind Rao, Karthik Ganesan and Tsachy Weissman, QVZ: lossy compression of quality values. Bioinformatics. 2015 Oct 1;31(19):3122-9.
[3] Bonfield,J.K. and Mahoney,M.V. (2013) Compression of FASTQ and SAM format sequencing data. PloS One, 8, e59190.

[4] ISO/IEC JTC 1/SC 29/WG 11 - ISO/TC 276/WG 5 MPEG2016/N16323 - Requirements on Genomic Information Compression and Storage, Geneva, June 2016.
[5] ISO/IEC JTC 1/SC 29/WG 11 - ISO/TC 276/WG 5 N16322/N96 - Database for Evaluation of Genome Compression and Storage, Geneva, 2016.