INVITED PAPER

# An Introduction to MPEG-G: The First Open ISO/IEC Standard for the Compression and Exchange of Genomic Sequencing Data

*The amount of data generated by genomic sequencing machines necessitates the development of an efficient representation format. The article provides an overview of the MPEG-G standard focusing on its core, i.e., the coding of genomic information.*

By Jan Voges, Mikel Hernaez, Marco Mattavelli, *Member IEEE*, and Jörn Ostermann, *Fellow IEEE*

**ABSTRACT** | The development and progress of high-throughput sequencing technologies have transformed the sequencing of DNA from a scientific research challenge to practice. With the release of the latest generation of sequencing machines, the cost of sequencing a whole human genome has dropped to less than $600. Such achievements open the door to personalized medicine, where it is expected that genomic information of patients will be analyzed as a standard practice. However, the associated costs, related to storing, transmitting, and processing the large volumes of data, are already comparable to the costs of sequencing. To support the design of new and interoperable solutions for the representation, compression, and management of genomic sequencing data, the Moving Picture Experts Group (MPEG) jointly with working group 5 of ISO/TC276 "Biotechnology" has started to produce the ISO/IEC 23092 series, known as MPEG-G. MPEG-G does not only offer higher levels of compression compared with the state of the art but it also provides new functionalities, such as built-in support for random access in the compressed domain, support for data protection mechanisms, flexible storage, and streaming capabilities. MPEG-G only specifies the decoding syntax of compressed bitstreams, as well as a file format and a transport format. This allows for the development of new encoding solutions with higher degrees of optimization while maintaining compatibility with any existing MPEG-G decoder.

**KEYWORDS** | Bioinformatics; computational biology; data compression; DNA; genomics; standardization.

Manuscript received June 9, 2020; revised March 31, 2021; accepted May 5, 2021. Date of publication June 15, 2021; date of current version August 20, 2021. *(Corresponding author: Jörn Ostermann.)*

**Jan Voges** and **Jörn Ostermann** are with the Institut für Informationsverarbeitung, Leibniz University Hannover, 30167 Hannover, Germany (e-mail: ostermann@tnt.uni-hannover.de).

**Mikel Hernaez** is with the Carl R. Woese Institute for Genomic Biology, University of Illinois at Urbana–Champaign, Champaign, IL 61801 USA, and also with the Center for Applied Medical Research, University of Navarra, 31008 Pamplona, Spain.

**Marco Mattavelli** is with Sciences–Faculté des Sciences et Techniques de l'Ingénieur–Multimedia (SCI-STI-MM), École Polytechnique Fédérale de Lausanne, 1015 Lausanne, Switzerland.

This article has supplementary downloadable material available at https://doi.org/10.1109/JPROC.2021.3082027, provided by the authors.

Digital Object Identifier 10.1109/JPROC.2021.3082027

## I. INTRODUCTION

The development and progress of high-throughput sequencing technologies hold the potential to enable the use of genomic information in many fields. With the release of the latest generation of sequencing machines, the cost of sequencing a whole human genome has dropped to less than $600. In the next few years, such cost is expected to drop further, to about $100. Today, a single sequencing system can deliver the equivalent of 10 000 whole human genomes per year, generating more than 1 PB of data. The potential applications in several fields—such as precision medicine, oncology, and food quality control, just to mention a few—lead

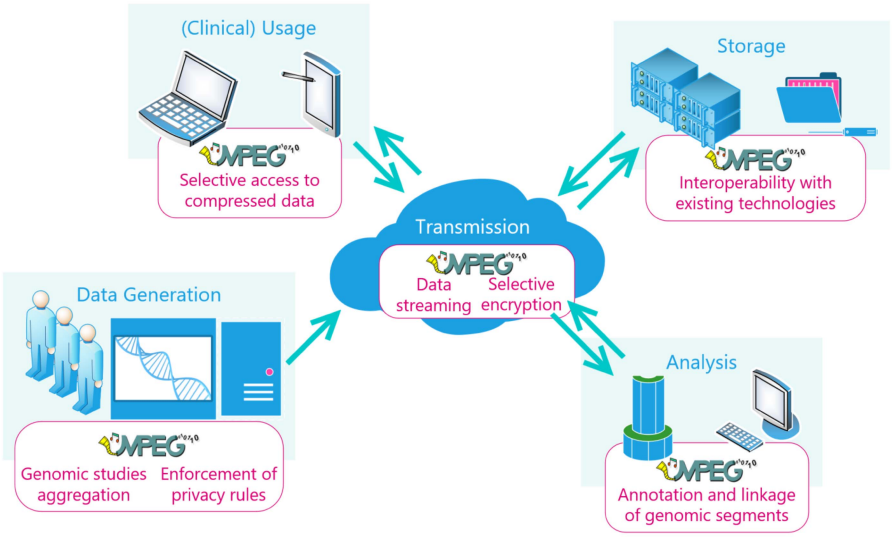


**Fig. 1.** ***Genomic ecosystem made possible by MPEG-G. The availability of a perennial specification will facilitate the creation of an ecosystem comprising interoperable systems and applications.***

to the forecast that the amount of digital genomic data will soon surpass the volume of video data uploaded to YouTube or tweets posted on Twitter [1]. By that point, the costs associated with storing, transmitting, and processing the large volumes of genomic data will largely exceed the costs of sequencing.

Where do such enormous amounts of data come from, and which can be the solutions for handling and processing them? Since high-throughput sequencing technologies are essentially noisy processes, a high degree of "signal redundancy" is required to extract the relevant information. A human genome is composed of about 3.2 billion base pairs, and each could be represented by 2 bits. Hence, 6.4 billion bits, i.e., 800 MB, would suffice to express it. However, in practice, a much larger volume of redundant data must be generated to be able to extract useful information. The essential reason is that, with high-throughput sequencing technologies, it is only possible to sequence, with minimal errors, at most relatively short DNA segments that need to be reassembled to reconstruct, for example, a human genome. Thus, a high level of redundancy is needed to discern reading errors from mutation characteristics of the individual. Moreover, the fragments obtained from the reading process (i.e., reads) are associated with metadata, generated during the sequencing process, which are used to better identify and filter out errors to support more accurate genome reconstructions and further analysis stages. Such support information constitutes a relevant, if not even dominant, portion of the data generated by sequencing machines.

The lack of widespread adoption of appropriate compressed data representation formats is widely recognized as a critical element limiting the potential for genomic data to be used in a wide range of scientific and public health scenarios [2]. This is not due to a lack of specialized compressors for genomic data (see [3] and [4]) but rather to the absence of perennial, fully specified, and reliable solutions able to offer a complete framework for the compressed representation of genomic information in the whole chain from the sequencing machine to the efficient browsing of secondary and tertiary analysis results.

In an effort to provide a response to such need, to the economic problems of the ever-expanding generation of genomic data, and to ease the effective usage of such new types of information, the Moving Picture Experts Group (MPEG)—a joint working group of the International Standardization Organization (ISO) and the International Electrotechnical Commission (IEC)—jointly with working group 5 of ISO/TC276 "Biotechnology" have started a standardization project to support the design of a new and interoperable solution for the representation, compression, and management of genomic sequencing data. The availability of a long-term and reliable standard, not only in the view of the authors of this article or the contributors of the standardization work, will facilitate the creation of an ecosystem of applications that will eventually democratize and fully exploit the still undiscovered potential of genomic applications, such as personalized medicine (see Fig. 1). This standardization project yielded the ISO/IEC 23092 series, also referred to as MPEG-G. MPEG-G has been designed following the open, but rigorous process developed and adopted by ISO and IEC, and thus MPEG, for all its standards in the last 30 years.

The first step of the standardization process of MPEG-G was the identification of a list of requirements, including those for the efficient transport of and selective access to compressed genomic data. A call for proposals was issued in June 2016 jointly by MPEG and working group 5 of ISO/TC276 "Biotechnology," and ten responses were received in October 2016. The identified technologies were evaluated using several criteria, such as compression performance and random access capabilities. Separate

assessments for different types of genomic data were performed: sequence reads, quality scores, read identifiers, alignment information, and alignment metadata. In addition, preliminary computational complexity was assessed by measuring encoding and decoding speed, as well as memory usage. The most valuable technologies were integrated to provide the compression of unaligned and aligned reads and the definition of a genomic information transport layer, which supports both storage and streaming of compressed genomic information. As a result of this process, MPEG-G supports new features associated with complex use cases, most of which are not supported by currently existing formats (such as FASTQ [5] and SAM/BAM [6]). Notable functionality and use cases addressed by MPEG-G include selective access to compressed data, data streaming, enforcement of privacy rules, and selective encryption of sequencing data and metadata.

While TC276 "Biotechnology" jointly with MPEG is the only ISO organization dealing with compact representation and efficient transport of genomic sequencing data, there exist also industry alliances, e.g., the Global Alliance for Genomics and Health (GA4GH). It was founded in 2013 and works toward frameworks and standards to enable the secure sharing of genomic and health-related data. Typically, the scope of these alliances is much wider, and they have limited experience in developing and maintaining standards in information technology.

In what follows, the essential components of MPEG-G are described in more detail, with an emphasis on its features and functionalities, including critical discussions on the role of MPEG-G in the future of genomic data storage, access, sharing, and processing.

The preprint [7] served as a "white paper" introducing MPEG-G to the wide bioinformatics community. Hence, some details presented in this article were already discussed in it. However, here, we lay the focus on accessibility, by providing more context background information. What is more, we provide new insights, by elaborating on the design of MPEG-G encoders. Also, we compare an MPEG-G encoder to *de facto* industry standards and to the state of the art in nonstandard compression tools that can be found in the literature.

Section II gives a short introduction into the technology of sequencing of DNA molecules. An overview of the MPEG-G specification is provided in Section III. The main parts of the MPEG-G file and transport formats and compression technology are detailed in Sections IV and V, respectively. Some aspects of metadata representation and software interfaces to MPEG-G are highlighted in Section VI. To check implementations against the MPEG-G specifications, MPEG-G provides reference software and tools for testing conformance, as described in Section VII. In Section VIII, the performance of an MPEG-G encoder and other tools is evaluated. The overview of MPEG-G is concluded and future developments are highlighted in Section IX.

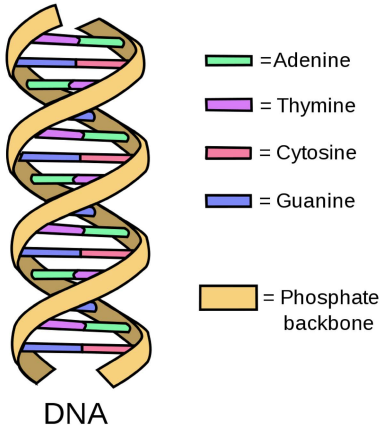


**Fig. 2.** *DNA double helix. Two polynucleotides coiled around each other form a double helix. The polynucleotide strands are connected by bonds between the four nucleobases.*

## II. GENOMIC SEQUENCING DATA

DNA is composed of a double helix formed by two polynucleotides coiled around each other (see Fig. 2). Each nucleotide contains one of four nucleobases: cytosine (C), guanine (G), adenine (A), or thymine (T). For humans, the genome is composed of 23 pairs of chromosomes and can be represented with about 3.2 billion bases associated with one strand of the helix. The complementary strand can be inferred by applying the base-pairing rules (A with T and C with G). Each base can be represented by 2 bits; thus, the human genome can be stored in approximately 800 MB. This is a data size easily manageable by nowadays information technology systems. Thus, where does the need for compression come from?

State-of-the-art sequencing technologies are based on reading fragments of DNA. These read-out fragments are commonly called reads. Technologies reading short reads, consisting of 75 to about 300 bases, in single fragments or pairs, work quite precisely with up to about 99.9% of bases identified correctly [8]. Other technologies reading much longer fragments of up to about 50 000 bases work with much lower accuracy, with 60% up to 88% of bases identified correctly [9]. Such reads can be reassembled, yielding an estimate of the underlying DNA. This process is schematically illustrated in Fig. 3. In the case of humans and many other organisms, reference genomes have already been assembled. In these cases, the assembly stage can be skipped since reads can be directly aligned to the reference. However, even if, only for a small fraction (for humans about 0.1%), each pair of genomes exhibits differences, redundant reading of the DNA is needed to solve ambiguities or to precisely reconstruct regions that differ from the reference. The average read-out redundancy is called coverage and expressed as, e.g., 30×, which means that each locus of the underlying DNA was read out 30 times on average. Thus, on average, 30 reads overlap at each locus. Moreover, mutations, which may or may not

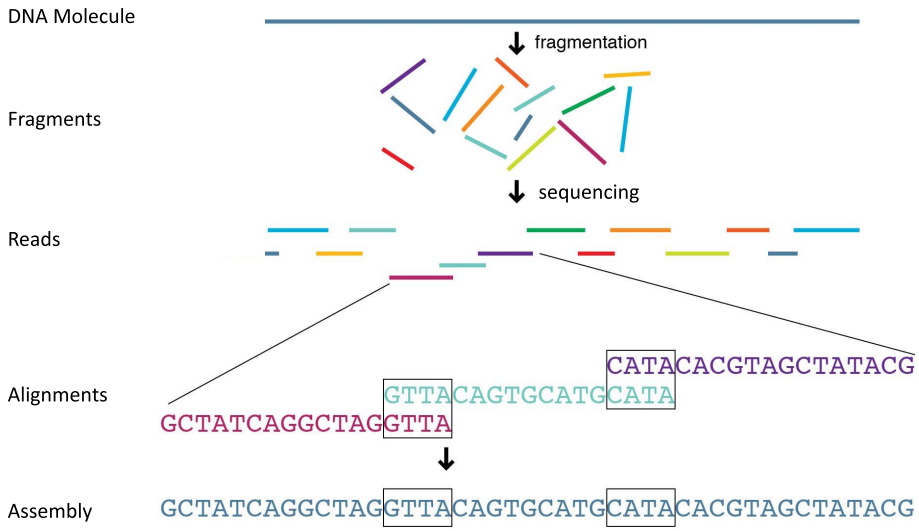


**Fig. 3.** ***Schematic visualization of sequencing process. A DNA molecule is fragmented. Fragments are read by a sequencing device. Reads are aligned and finally merged into an assembly, which is an estimate—as the sequencing is error-prone—of the underlying DNA molecule.***

indicate pathogenic evidence, and sequencing errors can be identified more confidently at higher levels of coverage.

Depending on the employed sequencing technology, the reading process is mostly based on either detecting the intensity of light emitted by fluorescent molecules attached at the end of DNA fragments or detecting variations of electrical signals generated by a DNA strand passing through a molecular pore. The reliability of such indirect and possibly noisy measurements is usually also reported in the form of quality scores. They are causally related to the probability of a base being detected correctly. Quality scores are, e.g., used in the alignment processes to make appropriate decisions.

Fig. 4 shows a visualization of read alignments, depicting some of the discussed terms. At the top of the figure, the reference sequence used for alignment is shown. Below the figure, aligned reads are shown in gray. Multiple reads overlap at each locus of the reference sequence.

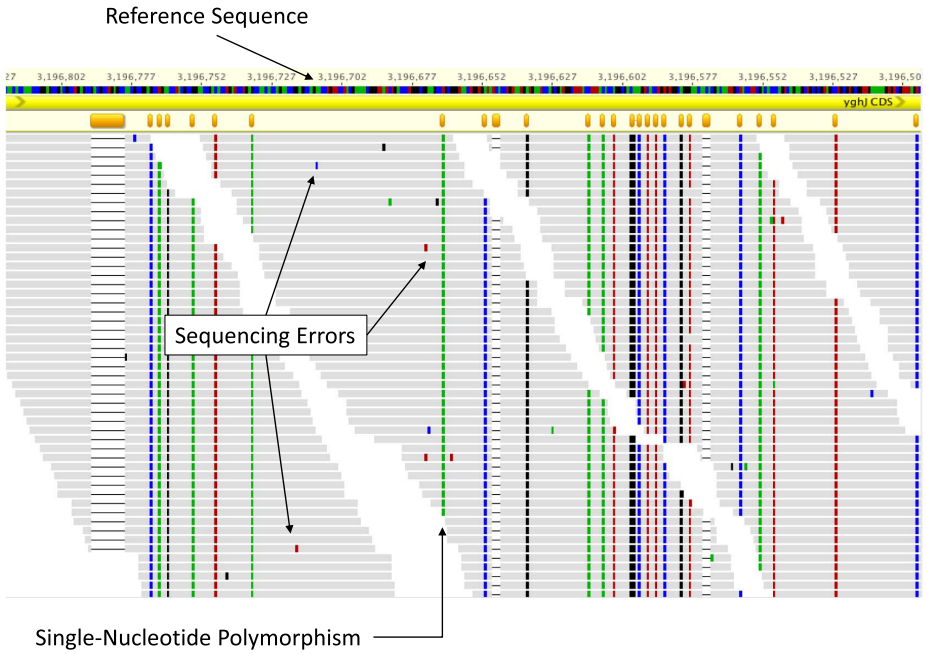


**Fig. 4.** ***Visualization of read alignments. Image produced with a modified version of IGV [10].***

Differences with respect to the reference sequence are highlighted in color. Some differences only occur in single specific reads and can safely be interpreted as sequencing errors. Other differences occur in virtually every read, implying the presence of variants, such as single-nucleotide polymorphisms.

In addition to quality scores, each read is also associated with a so-called read identifier, which carries various information specific to the sequencing process. Therefore, besides the reads themselves, the volume of associated metadata constitutes an important, albeit not the dominant, part of the overall sequencing data volume.

Considering that, for clinical usage, it may be required to sequence-specific genome areas with a coverage larger than 200×, it becomes clear how important compression and selective access to data are to ease and support all stages of acquisition, processing, and analysis of genomic sequencing data.

## III. OVERVIEW OF THE MPEG-G SPECIFICATION

Today, everybody manipulates media information (images, sounds, video, and various metadata) using a variety of interoperable applications that deal with compressed and multiplexed streams making browsing and exchanging of, and selective access to, digital media fast and economical. All the ecosystems of interoperable applications can seamlessly work together by relying on generations of MPEG standards covering compression, carriage, storage, and application programming interfaces (APIs) for the manipulation of these data. The economic success with products (digital cameras, media servers, MP3 players, and so on) and services (broadcast, streaming, and so on) based on MPEG technology was only possible because MPEG standards were developed following a process focused on tight decoder specifications and verification, compatibility assessment, long-term maintenance, and continuous analysis of industry requirements. Within this environment, MPEG-G was developed reusing or revisiting established MPEG technologies, in addition to expanding new compression technologies that better deal with the specificities of sequencing data and metadata.

In this context, the first element of MPEG-G is the specification of a file format and a transport format, which includes: 1) the capabilities of internal indexing; 2) selective access to data; and 3) conversion from/to a streaming format that dynamically supports indexing capabilities.

Regarding the source models used for compression, a multiplicity of coding modes has been developed for the specific genomic information sources. These coding modes facilitate the efficient exploitation of the variety of statistical properties of different sequencing technologies. In the case of reads, this concept corresponds to the use of different tailored predictive coding modes. MPEG-G specifies coding modes for both unaligned reads, for which predictions can be made on common patterns built around clusters of reads, and aligned reads, for which "external" or built-in reference sequences may be used for predictive coding. To increase coding efficiency, an additional step has been done by defining a classification system for reads. They can be classified according to their matching properties with respect to the reference sequences. These developments facilitated the definition of the concept of "descriptors," structured in the form of "descriptor streams," which represents the genomic information in a form that is beneficial for applying entropy coding to them. Such an approach not only turned out to be more efficient in terms of compression but also made it possible to provide multiple dimensions of random data access in the compressed domain.

In summary, MPEG-G currently consists of five standards (ISO/IEC 23092-1–5), also referred to as parts 1–5.

### A. Part 1: Transport and Storage of Genomic Information (see Section IV)

This part specifies how genomic data is structured to facilitate transport, including streaming of (or parts of) files and storage. The main elements are: 1) the specification of a hierarchical structure capable of containing various possible logical organizations of sequencing data (dataset groups and datasets); 2) the structure of metadata attached to each dataset group or dataset; and 3) a so-called master index table (MIT) for the support of random access inside each dataset. Datasets are composed of various access units that constitute the minimal elements that a decoder (specified in part 2) needs to access to be able to fully decode subsets of reads and attached metadata. Concerning the reversibility between transport and file format, a nonnormative reference process is provided as support for implementations.

### B. Part 2: Coding of Genomic Information (see Section V)

This part specifies the decoding syntax used to represent unaligned and aligned reads and the associated quality scores and read identifiers—and, in the case of aligned reads, alignment information and alignment metadata—and reference sequences, if any. This is the part that deals with compression by describing the normative behavior of a compliant decoder. Only the decoding process is specified, while any encoding algorithm can be used, which produces a bitstream compliant with this part.

### C. Part 3: Metadata and Application Programming Interfaces (see Section VI)

This part specifies how information metadata, providing general information, and protection metadata is attached to a dataset, to dataset groups, or to the entire file. The term "metadata," when referring to part 3, must not be confused with the quality scores, read identifiers, alignment information, or alignment metadata, which are instead attached to reads and contained in access units,

as specified in part 2. Other functionalities covered by part 3 include the specification of an API for the access to MPEG-G data from applications built on top of normative decoders, the specification of mechanisms to implement access control, integrity verification, as well as authentication and authorization mechanisms. This part also includes a section devoted to the mapping between the current *de facto* format SAM and MPEG-G data structures.

### D. Part 4: Reference Software (see Section VII)

To support and guide potential implementers of MPEG-G technology, MPEG-G includes reference decoder software. The reference decoder software is normative in the sense that any conforming implementation of the decoding process, taking the same compressed bitstreams and using the same output data structures, must output the same data.

### E. Part 5: Conformance (see Section VII)

This part is fundamental in providing means to test and validate the correct implementation of the MPEG-G technology in different devices and applications to ensure full interoperability among all systems. Conformance testing specifies a normative procedure to assess decoder conformity on an exhaustive set of compressed data.

The standard development work within MPEG is done not only jointly with ISO/TC276/WG5, as mentioned above, but also in liaison with other ISO committees. A close collaboration was established with ISO/TC276/WG3, which deals with analytical methods for genomic data. More recently, to avoid superpositions and strengthen synergies, a liaison with ISO/TC215/SC1, dealing with genomics informatics, has been established. Finally, MPEG shares some of its goals with GA4GH. GA4GH tackled the challenge of the expanding generation of genomic data by publishing the CRAM [11] specification for compression of genomic information, as well as a specification for remote access to genomic information [12]. MPEG, in addition to publishing the specifications for transport, storage, and coding of genomic information, as well as metadata and APIs (parts 1–3), has also published a reference software (part 4) and a conformance testing specification (part 5). These last two parts are critical to ensure interoperability among different implementations. In addition, ISO/IEC has a long-standing practice of continuous verification and maintenance of its standards. Maintenance includes the extension of a standard to meet current industry needs. For example, MPEG-2, the first standard used for storing digital audio and video on DVDs was initially specified in 1994. Its latest amendment is dated 2019. MPEG-2 video decoders of today can decode all MPEG-2 files created since 1994. Furthermore, MPEG-2 decoders of 1994 can still decode MPEG-2 video encoded with modern MPEG-2 codecs although progress in encoder technology now only requires about 50% of the bits that were required at the time of defining MPEG-2.

## IV. TRANSPORT AND STORAGE OF GENOMIC INFORMATION

MPEG-G specifies in its part 1 the file and transport formats for the processing, manipulation, and storage of genomic data compressed according to part 2. These formats were developed considering the various requirements that were collected during the development of MPEG-G. Among the most relevant requirements that highlighted unmet needs are: 1) the unambiguous identification of reference sequences; 2) the assessment of integrity; 3) the enabling of data protection; and 4) the support for compressed data streaming.

More precisely, the format that is intended to be used for the transport of packetized data on a telecommunication network is referred to as the transport format. The format used for storage on a physical medium is referred to as file format. These two formats are based on the ISO/IEC base media file format (ISO/IEC 14496-12—MPEG-4 Part 12), and they are fully reversible: an MPEG-G file can be converted into an MPEG-G transport stream, which, in turn, can be converted into an MPEG-G file. Because of the similarity of transport and file format, we focus, in the following, on the file format.

As shown in Fig. 5, an MPEG-G file is a hierarchical arrangement of data structures that may contain logically organized sequencing data. At the topmost level, an MPEG-G file is organized in a file header and one or more dataset groups. Each dataset group contains a dataset group header and optional containers (for, e.g., metadata and protection information), and it encapsulates one or more datasets. Each dataset is composed of a dataset header, a dataset parameter set, and optional containers (for metadata and protection information). It can also optionally contain an MIT, which facilitates random access inside each dataset. Each dataset carries one or more access units. The access unit is the actual data structure that contains the compressed sequencing data. It constitutes the smallest data structure that can be decoded by a decoder, which is compliant with part 2. Hence, the access unit data structure represents the link between part 1 and part 2. Besides an access unit header and other optional containers, an access unit contains a collection of blocks. Each block is a portion of a descriptor stream, and it can be decoded independently using information from the dataset parameter set and from other access units, such as access units containing (fragments of) reference sequences.

Each data structure in an MPEG-G file is also associated with optional metadata ("information metadata") that provides general information about the data, such as the origin of the biological sample, a log of the operations performed on the data, and information associated with the preparation of the samples and the sequencing process. In addition, protection information ("protection metadata") can be associated with each data structure, providing the support for different selective protection approaches of the

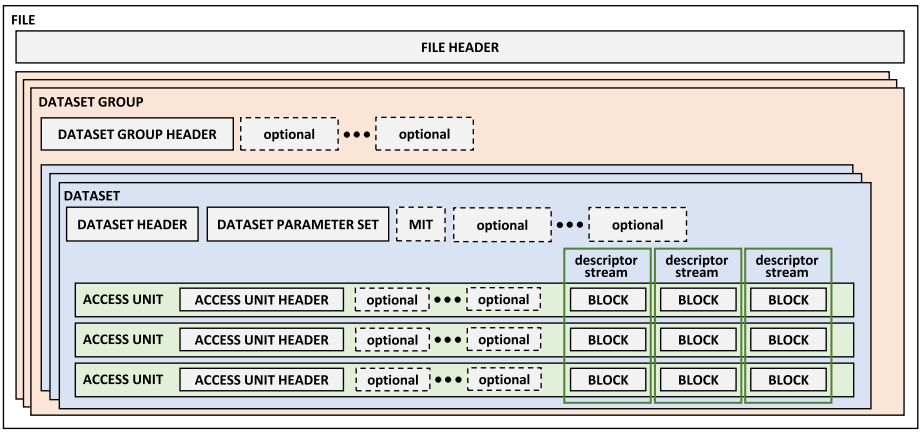


**Fig. 5.** *Key elements of the MPEG-G file format. Multiple dataset groups contain multiple datasets of sequencing data. Each dataset is composed of access units. Each access unit contains blocks of coded genomic information.*

data. The make-up of information metadata and protection metadata is specific in part 3.

The hierarchical design of an MPEG-G file facilitates an abundance of use cases. For instance, the file can simply contain the data from a single sequencing run of a portion of a human chromosome. In another scenario, with regard to whole genome sequencing (WGS) experiments, an MPEG-G file could be used to structure the storage of the sequencing data of a trio of individuals (father, mother, and child) as follows: there would be three distinct dataset groups, one for each individual in the trio. Then, each dataset group would contain datasets related to sequencing runs for the same individual, performed for example at different moments in time. This example shows how MPEG-G files can be employed in a variety of use cases.

As mentioned above, part 1 of MPEG-G also provides a transport format for the efficient packetization of compressed data. The transport format provides an extra set of structures allowing the conversion of the file format structures into data streams, which consists of multiple packets. These can be dynamically adapted to channel characteristics. Moreover, the transport format facilitates features, such as error detection, out-of-order delivery, and retransmission of erroneous data at the protocol level.

## V. CODING OF GENOMIC INFORMATION

Fig. 6 shows the block diagram of the general encoding process. The encoding process consists of three main stages: preprocessing, descriptor generation, and transformation and entropy coding. Each stage will be elaborated separately in what follows.

### A. Preprocessing

Unaligned sequencing data consist of the actual sequence reads, the associated quality scores, and read

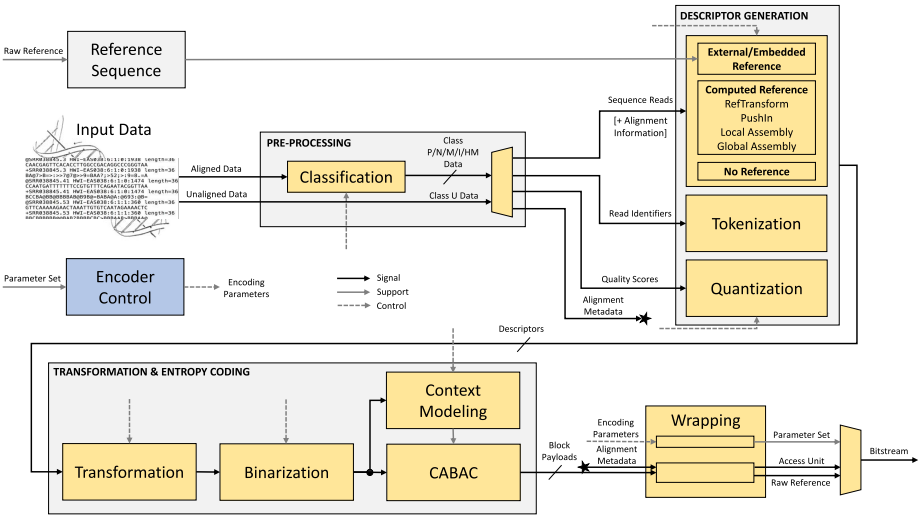


**Fig. 6.** *Block diagram of the general encoding process. The encoding process consists of three main stages: preprocessing, descriptor generation, and transformation and entropy coding. Note that a parameter set can be reused by feeding it back into the encoder control, and a raw reference can also be reused by initializing a reference sequence from it.*

**Table 1** Data Classes

| Class Name | Semantics |
|---|---|
| P | Reads perfectly matching to a reference sequence |
| N | Reads containing mismatches w.r.t. a reference sequence, which are unknown bases only |
| M | Reads containing, w.r.t. a reference sequence, at least one substitution, and possibly unknown bases, but no insertions, no deletions, and no clipped bases |
| I | Reads containing, w.r.t. a reference sequence, at least one insertion, deletion or clipped base, and possibly unknown bases or substitutions |
| HM | Half-mapped read pairs where only one read is mapped |
| U | Unmapped reads |

identifiers. The triplet consisting of one read, its identifier, and associated quality scores is referred to as an unaligned genomic record. Aligned sequencing data, in addition, contain alignment information and optional alignment metadata. Before encoding, any implementation is free to classify genomic records into six classes according to the result of the alignment of their reads against one or more reference sequences. While unaligned genomic records have a dedicated class (U), aligned genomic records can be assigned to one of the other classes summarized in Table 1.

After the classification, as the final step of the preprocessing stage, genomic records are split into their constituents before further processing: sequence reads, quality scores and read identifiers, and alignment information and alignment metadata in case of aligned data.

## B. Descriptor Generation

The input of the descriptor generation stage is sequence reads, quality scores, and read identifiers, as well as alignment information and alignment metadata in case of aligned genomic records. These different types of data are processed separately. In what follows, the coding of sequence reads (and alignment information, in the case of aligned data), quality scores, and read identifiers is elaborated in more detail. The alignment metadata is further processed in the wrapping stage (see Fig. 6), which is specified in part 3.

*1) Coding Modes for Sequence Reads:* For the encoding of sequence reads (and alignment information, in the case of aligned data), an encoder is free to choose between four approaches.

1) *External Reference:* A reference sequence is available as an external resource (locally or remotely).
2) *Embedded Reference:* The reference sequence is embedded as a dataset in the same file.
3) *Computed Reference:* A reference sequence is computed from the sequence reads already processed.
4) *No Reference:* Sequence reads are, in principle, forwarded verbatim to the transformation and entropy coding stage (see Fig. 6).

Each of these approaches yields a set of descriptor streams, which contain all information necessary to fully reconstruct the sequence reads (and the alignment information, in the case of aligned data). The goal in designing the descriptor streams was to ensure uncorrelation among them, if possible. This ensures maximum compression efficiency once the descriptor streams are fed through the transformation and entropy coding stage.

When using a computed reference, the encoder can choose between four reference computation algorithms.

1) *Reference Transformation ("RefTransform"):* To improve compression efficiency, an available external reference is modified before encoding sequence reads.
2) *Read Concatenation ("PushIn"):* The reference is created by simple concatenation of already encoded sequence reads.
3) *Local Assembly:* A local assembly of the underlying sequence is built per group of reads.
4) *Global Assembly:* Common patterns shared among several reads are identified, which are encoded only once along with the bases specific to each read.

The algorithms "PushIn" and "global assembly" can also be used to encode unaligned data. As will be shown in the following, the selection of specific encoding techniques depends on the specific scenario requirements.

From an application point of view, unaligned data can, for example, be encoded according to different approaches, depending on the actual scenario at hand. Here, we illustrate two such application scenarios: a "low latency" scenario, in which the "no reference" approach is used, and a "high compression ratio" scenario, in which the "computed reference" approach with the "global assembly" reference computation algorithm is used.

*a) Low latency* In a streaming setting, when low latency has higher priority than compression ratio, a "high throughput" compression approach is desirable. In such a case, the descriptor generation stage is virtually bypassed, forwarding the split genomic record information almost directly to the transformation and entropy coding stage. This approach enables streaming, such as in a setting where genomic data needs to be transmitted to a remote device "on-the-fly" already during the sequencing process.

*b) High compression ratio* A high compression ratio is reached by leveraging the high redundancy in the sequencing data by, for example, applying the global assembly reference computation algorithm. This approach achieves a maximum compression ratio but requires the availability of the entire data and a few preprocessing stages, which may impact the compression latency. Here, the coding of reads relies on the identification of common patterns (i.e., "signatures") shared among several reads. These common patterns are encoded only once along with the bases specific to each read (i.e., the "residuals"). The presence of such signatures enables the implementation of indexing schemes with which the compressed data can be searched by means of pattern matching algorithms. This mode is suitable, for example, for long-term storage of unaligned reads. Examples of preprocessing technologies

suited for this compression mode include those presented in ORCOM [13], HARC [14], SPRING [15], FaStore [16], and, in general, all (future) preprocessing technologies that cluster reads based on common patterns.

From an application point of view, aligned reads can also be encoded according to the actual application scenario at hand. Here, we illustrate two such scenarios: a "clinical study" scenario, in which a reference-based approach (i.e., "external reference" or "embedded reference") is used, and a "reference-free" scenario, in which the "computed reference" approach with the "local assembly" reference computation algorithm is used.

*c) Clinical study* In this scenario, a multitude of human WGS experiments is performed. Here, it is beneficial to represent reads originating from different experiments by their differences with respect to a single set of reference sequences (i.e., one sequence per chromosome). In the case that the data of the sequencing of multiple genomes are stored in multiple datasets within a single MPEG-G file, the reference sequences can be embedded as additional datasets within the same MPEG-G file. This means that reference sequences can be shared among datasets. Also, external reference sequences can be used. MPEG-G specifies how external reference sequences can be identified unambiguously using a uniform resource identifier, checksums, and so on.

*d) Reference-free* In this approach, reads are compressed without referring to any reference sequence by applying the local assembly reference computation algorithm. Here, a local assembly of the underlying sequence is built per group of reads, and reference-based compression with respect to the computed local assembly is then applied [17], [18]. In this case, access to any reference sequences is needed at neither the encoder nor the decoder side.

*2) Coding Modes for Quality Scores:* Due to their higher entropy and larger alphabet, quality scores have proven more difficult to compress than reads [19], [20]. In addition, there is evidence that quality scores are inherently noisy, and downstream applications that use them do so in varying heuristic manners. As a result, quantization of quality scores can not only significantly alleviate storage requirements but also provide variant calling performance comparable, and sometimes superior, to the performance achieved using the uncompressed data. Therefore, in MPEG-G, quality scores can be encoded either in a lossless manner or a quantized manner. When encoding quality scores losslessly, several transformations can be applied to the quality scores (see Section V-C). Quantization of quality scores, however, can lead to a dramatic reduction of the file size after entropy coding. To minimize any quantization effects, MPEG-G provides several mechanisms to allow the fine-grained selection of quantization schemes at the encoder.

In the case of unaligned reads, an MPEG-G compliant encoder is free to choose any beneficial scalar quantization scheme. This includes quantization schemes of recently published research, such as [21]–[25]. The used representative values are signaled to the decoder by the means of a codebook. The quantized quality scores are signaled to a decoder as indexes into this codebook.

In the case of aligned reads, MPEG-G introduces an additional dimension to fine-tune quality score quantization: codebooks can be chosen per genomic position, i.e., per locus. As an illustrative example, an encoder could choose to select codebooks per genomic position using a simple genotyping model, such as in [26]. The MPEG-G specification allows the use of up to 16 codebooks per access unit.

The locus-based inference of codebooks using a simple genotyping model is schematically depicted in Fig. 7. At any locus $l$ in the sequenced genome, the genotype is represented by a discrete random variable $G$. The genotype is the set of alleles found at a locus across all reads covering it. Consider a set of reads that are aligned to a reference sequence, and assume that the reads have been sorted by their mapping positions, as shown in the figure. Given such a set of reads, we denote with $N$ the number of reads covering locus $l$. Let $n_i$ be the symbol from read $i$ covering the locus $l$ and $q_i$ the value of the corresponding quality score. The goal is to compute the posterior distribution of the genotype $G$, given the observable nucleotides $\mathbf{n} = \{n_i\}_{(i=1)}^{N}$, parameterized by the observable quality scores $\mathbf{q} = \{q_i\}_{(i=1)}^{N}$. The posterior probability is proportional to the likelihood times the prior

$$P(G|\mathbf{n};\mathbf{q}) \propto P(\mathbf{n}|G;\mathbf{q}) \cdot P(G).$$

The likelihood is given by

$$P(\mathbf{n}|G;\mathbf{q}) = \prod_{i=1}^{N} P(n_i|G;q_i)$$

where $P(n_i|G;q_i)$ is the likelihood of having observed $n_i$ given the assumption that the genotype was $G$, parameterized by $q_i$. This likelihood can be computed without further assumptions from the observable nucleotides and quality scores. For further information, we refer the reader to [26]. Given the genotype likelihood $P(\mathbf{n}|G;\mathbf{q})$, a codebook can be selected by applying a metric over it. Therefore, one codebook identifier per genomic position is sent to a decoder along with the indexes into the relevant codebooks. Before entropy coding, the quality score indexes are split into separate streams per codebook. Note that an additional stream for the locus-based codebook selection is also required. Finally, an MPEG-G encoder is also allowed to tune the quantization by selecting different codebooks per class and per access unit.

As already mentioned, the quantization of quality scores can not only significantly alleviate storage requirements but also provide variant calling performance comparable,

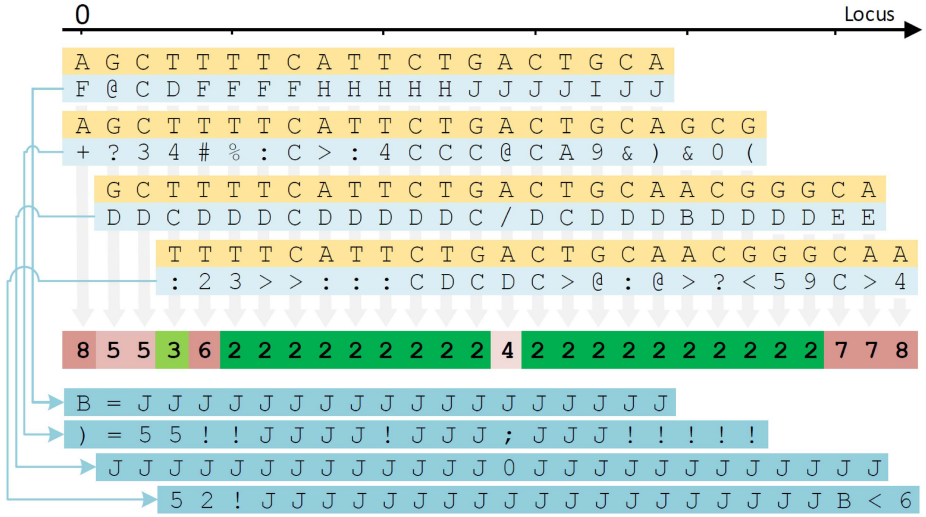


**Fig. 7.** ***Locus-based quantizer selection. Top: alignment of four reads. The colored bar in the middle represents the uncertainty about the genotype at each locus, where dark green represents low uncertainty, while red represents high uncertainty. The numbers shown in the middle bar are the number of representative values chosen for each locus. Bottom: quantized quality scores for the four reads. This figure was previously published in [26].***

and sometimes superior, to the performance achieved using the uncompressed data. Variant calling performance can be measured by running different variant calling pipelines on the original data and the data containing quantized quality scores. Afterward, each set of variants is compared against a consensus set of variants. This process yields the following values: true positives: variants that are both in the consensus set and in the set of called variants; false positives: variants that are in the called set of variants, but not in the consensus set; and false negatives: variants that are in the consensus set, but not in the set of called variants. These values are used to compute the recall/sensitivity (the proportion of called variants that are included in the consensus set) and the precision (the proportion of consensus variants that are called by the variant calling pipeline). Recent works, such as [26], have shown that quantizing quality scores has a negligible or positive impact on these measures, while, at the same time, quality scores are compressed down to well below 0.5 bits per quality score compared to the approximately 3 bits needed for losslessly compressed quality scores.

*3) Coding Modes for Read Identifiers:* Read identifiers are broken down into a series of tokens, which can be of three main types: strings, digits, and single characters. A read identifier is represented as a set of differences and matches with respect to one of the previously coded read identifiers. This approach is not tailored to any specific implementation of a sequencing manufacturer and only assumes that, within the same sequencing run, the structure of read identifiers is mostly constant. This method (or variants of it) has been previously employed in compressors such as Samcomp [27], DeeZ [18], FaStore [16], and AliCo [28].

## C. Transformation and Entropy Coding

Storing different types of data in separate descriptor streams allows for significantly higher compression effectiveness.

To compress the heterogeneous set of descriptors, MPEG-G specifies the use of context-adaptive binary arithmetic coding (CABAC) [29], which is used in popular video coding standards, such as H.264/AVC [30] and H.265/HEVC [31], and the genomic data compression solutions, AFRESh [32] and AQUa [33]. CABAC is also used in GABAC, the first implementation of an entropy codec compliant to part 2 [34]. By selecting CABAC, the implementation of compliant codecs is simplified significantly, as a wide range of implementations, both in hardware and in software, is currently available.

As shown in Fig. 8, given an input descriptor stream (named "descriptor subsequence" in part 2), the compression process consists of two main stages: transformation and entropy coding. Each step of the transformation stage is optional (except for the sign extraction as part of the subsymbol transformation, in case that a transformed subsequence contains signed symbols). The transformation stage comprises the following pipeline: subsequence transformation, subsymbol split (including sign flag extraction), and subsymbol transformation. The (mandatory) entropy coding stage consists of binarization, context modeling, and CABAC.

*1) Transformation:* In the subsequence transformation, the input symbols are processed using one of the following three transformations (or no transformation, i.e., pass-through): 1) equality transformation, where a symbol is replaced by a flag indicating equality with its predecessor and a correction symbol, if required; 2) run-length

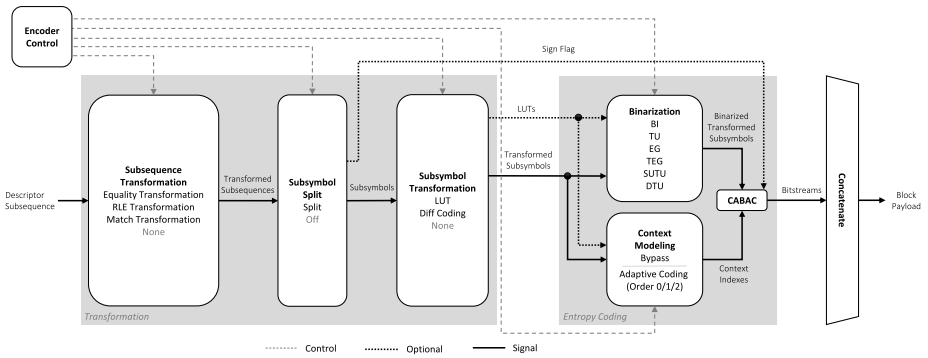


**Fig. 8.** ***Block diagram of a possible implementation of transformation and entropy coding. This figure is an adapted version of a figure published in [34].***

transformation, where a repeated symbol is replaced by the symbol itself and its run length; and 3) match transformation, an LZ77-style transformation [35]. Each subsequence transformation will create two or three transformed subsequences. In the further steps of the compression process, the transformed subsequences are processed separately.

The symbols comprising a transformed subsequence can be further split into subsymbols, where, for example, one subsymbol contains the upper half of (most significant) bits and the other subsymbol contains the lower half of (least significant) bits. The subsymbol split also extracts, if required, the sign of each transformed subsequence symbol. This sign flag is separately sent to the CABAC engine. It is encoded with a special context index along with the "normal" bins that make up the transformed subsymbols.

In the subsymbol transformation, a lookup table (LUT) transformation can be applied where each subsymbol is replaced by a symbol from an LUT. As an alternative, a differential transformation ("diff coding") can be applied. Here, each symbol is replaced by the arithmetic difference to its predecessor.

The goal of this transformation process is to facilitate more effective compression when using CABAC. To achieve this goal, the transformation process yields transformed subsymbols that typically exhibit a reduced number of symbols (by exploiting redundancy through, e.g., a run-length transformation), smaller alphabets (by converting symbols into homogeneous substreams), adapted distributions (e.g., by replacing symbols through an LUT and, as such, allowing for binarizations that are shorter and/or offer higher predictability), or a combination hereof.

*2) Entropy Coding:* CABAC is used for the entropy coding of transformed subsymbols. This implies that, before the binary arithmetic coding engine is invoked, the transformed subsymbols must be converted into a binary representation. Probability (i.e., context) modeling is then performed per binarization bit (i.e., bin). In the binarization stage, the transformed subsymbols are converted into a set of bins. To generate this set of bins, part 2 specifies ten binarization processes, such as the truncated unary (TU) or exponential Golomb (EG) binarizations, as shown in Fig. 8.

In parallel with the binarization stage, context modeling is performed. A context is a number between 0 (representing a probability of 0) and 127 (representing a probability of 1) used to encode a specific bin with CABAC. Contexts are grouped into context sets. Each context set contains the contexts that are needed to support the encoding of one transformed subsymbol in its binarized representation.

The MPEG-G specification provides four different modes of context modeling. In the first mode (bypass), no context modeling is performed, and all bins are processed using contexts that assume equiprobability. Here, contexts are not adapted. In the second mode (order-0 adaptive coding), there is one context set for each symbol. Here, the contexts are adapted to the probability of each symbol. In the third (order-1 adaptive coding) and fourth modes (order-2 adaptive coding), context sets are selected based on the previously encoded symbol and based on the two previous encoded symbols, respectively. This way, conditional probabilities between consecutive symbols are modeled. Also, here, the contexts are adapted.

## D. Decoding Process

In addition to the syntax and semantics of the compressed sequencing data, part 2 also defines the decoding process.

The normative input of an MPEG-G decoding process is a concatenation of data structures called data units. Data units can be of three types. A data unit named "raw reference" encapsulates the coded representation of one or more reference sequences. A data unit can also contain parameters used during the decoding process; it is then referred to as "parameter set." A data unit containing the coded representation of actual reads and associated read identifiers, quality scores, and so on is named "access unit." The block diagram of the general encoding process shown in Fig. 6 depicts, in its bottom right corner, how these three types of data units can be merged into a single bitstream.

Raw references and parameter sets are used during the decoding process of access units but do not produce output. Fig. 6 shows analogously how raw references and parameter sets are fed into the encoder as supplementary

signals. It is the decoding process of access units that produce a normative output either in the form of so-called MPEG-G records, for access units containing reads and so on, or in the form of a raw reference structure, for access units containing a (part of a) compressed reference sequence. An MPEG-G record can be regarded as an improved SAM record: in an MPEG-G record, read pairs are typically coded in the same record unless certain conditions are met, such as the pairing distance is above a user-defined threshold, or the mate is mapped to a different reference sequence.

## VI. METADATA AND APPLICATION PROGRAMMING INTERFACES

Part 3 of MPEG-G has essentially two goals: 1) to specify the syntax and semantics of the metadata that can be attached to datasets and dataset groups or to the entire file and 2) to specify APIs that provide interoperability between applications that are built on top of normative decoders.

Two main types of metadata are specified: protection metadata and information metadata. Protection metadata are related to the protection technology applied to single access units, entire datasets, or dataset groups. Information metadata are related to metadata as specified in data repositories, such as the European Genome-Phenome Archive (EGA) or the Sequence Read Archive (SRA) [36].

Regarding the protection metadata, genomic data contained in an MPEG-G file can be linked to multiple owner-defined privacy rules, which implement restrictions on data access and usage. The privacy rules and the description of the protection technology are specified using the eXtensible Access Control Markup Language (XACML) Version 3.0 [37] and are carried in a specific metadata protection structure as specified in part 1. Part 3 specifies the syntax and semantic of the XACML description and how it can be decompressed.

These protection description containers provide—in addition to the privacy rules to be applied to the information they refer to—mechanisms to manage the confidentiality and integrity of the information. By using protection techniques implemented at the application level and combined with privacy rules, the genomic data are efficiently protected from unauthorized access. Therefore, only users authorized by the rules contained in the MPEG-G file can perform operations over such protected regions.

As mentioned above, MPEG-G supports the protection of genomic information at different levels in its hierarchy of logical data structures. The protection information specifies how the data structures at the same level, as well as the protection information containers of a layer immediately below, are encrypted. Specific protection information of dataset group metadata and dataset metadata is represented using the W3C recommendation XML Encryption Syntax and Processing Version 1.1. Other protection information, such as for the protection of access units, is specified using explicit XML schemas. Also, authentication and integrity may be provided by means of electronic signatures using the W3C recommendation XML Signature Syntax and Processing Version 1.1.

**Table 2** Example of Dataset Group Metadata

| Element Name | Element Type | Mandatory |
|---|---|---|
| Title | String | Yes |
| Type | Controlled vocabulary | Yes |
| Abstract | String | No |
| Project center name | Custom type | No |
| Description | String | No |
| Samples | Custom type | Yes |
| Extensions | Custom type | No |

The second type of metadata, information metadata, corresponds to well-known metadata sets, such as those in EGA or SRA specifications. This allows easy interoperability in converting metadata to and from already existing databases. Normative metadata "profiles" that include a subset or all core elements to ease interoperability can be specified out of the large set of core elements specified in part 3. In addition, part 3 provides a normative extension mechanism to be able to include new elements not already specified in the set of core elements. In such a case, any decoder can correctly decode the syntax, but the extended semantics can only be provided by an external uniform resource identifier.

Metadata can apply to any element of an MPEG-G file, and the linking mechanism covers the entire hierarchical structure. Metadata present at file header apply to all elements of the file that inherits their value to all lower elements of the hierarchy, unless new or different metadata elements appear in a header structure at a lower hierarchy level to overwrite a subset of elements or provide new elements absent in the higher hierarchy elements. Such a mechanism of hierarchical inheritance provides the maximum level of flexibility to attach metadata to file elements but also avoids repetitions and the possibility of inconsistencies. An example of dataset group metadata is shown in Table 2.

Another important functionality provided in part 3 is the specification of APIs enabling standardized access to MPEG-G files. In case the information is protected, operations are controlled by privacy and protection rules, as described above. Whenever the caller of API methods is not authorized to access the full content, only the content for which the caller is authorized is returned. APIs are applied to data structures that are organized in hierarchy levels, so the context of a hierarchy level defines the scope of an operation. The considered hierarchy levels are dataset group, dataset, and access unit. How the APIs get access to the file or bitstream is left open to implementations of part 3. APIs are logically partitioned into five main groups: 1) "Genomic Information Functions" used for querying the structure of and retrieving, the genomic information coded in a bitstream; 2) "Metadata Functions" used

for querying the structure of, and retrieving, the metadata associated with the coded genomic data; 3) "Protection Functions" used for retrieving the protection metadata associated with the coded genomic data; 4) "Reference Functions" used to retrieve the reference associated with a dataset; and 5) "Statistics Functions" used for retrieving various types of statistics associated with a dataset.

Finally, part 3 also provides a normative specification for the conversion to and from the SAM file format. The specification supports a full round trip for well-specified fields and provides conversion processes to and from the MPEG-G representation for ambiguous or not-well-defined optional fields of the SAM file format.

## VII. REFERENCE SOFTWARE AND CONFORMANCE

To support and guide the implementation of MPEG-G, part 4 provides a normative reference software. The reference software is normative in the sense that any conforming implementation of the decoding process, taking the same conformant compressed bitstreams and using the same normative output data structures, will output the same data. That being said, complying implementations are not expected to follow the algorithms or even the programming techniques used by the reference software. Such software is solely intended as a support to the process of developing implementations of an ecosystem of compliant devices and applications. Hence, the availability of a normative implementation is only additional support to the textual specification. It should also be underlined that the reference software is not intended as an optimized implementation. As such, the reference software should not be used as a benchmark of computational performance.

MPEG also plans to soon provide an informative reference encoder software. The reference encoder software is not intended to provide an exhaustive implementation of all possible coding options; that is, it does not aim at implementing all features, as well as possible preprocessing and optimization approaches. However, it will act as a guide for implementing higher performance encoders.

Conformance (part 5) is fundamental in providing means to test and validate the correct implementation of the MPEG-G technology in different devices and applications and to ensure interoperability among all systems. Conformance testing specifies a normative procedure to assess conformity to parts 1 and 2. For this purpose, an exhaustive set of bitstreams was generated. This set exercises all decoder functionalities without explicitly considering encoder or decoder efficiency. Every decoder claiming MPEG-G conformance will have to demonstrate the correct decoding of the complete set of bitstreams.

## VIII. RESULTS AND DISCUSSION

During the development of MPEG-G, the best-performing compression technologies, according to the results of the call for proposals and continuous improvements during the development process, were selected for integration. However, only the decoding process is normative and specified. This guarantees the interoperability of applications implementing MPEG-G, while the encoding process is open to algorithmic and implementation-specific innovations. As such, the compression performance achievable by implementations of MPEG-G will vary from encoder to encoder and will most likely improve over time. Nevertheless, to give the reader a sense of the compression capabilities achievable by MPEG-G, we present the compression performance of a specific MPEG-G encoder. Also, we compare its compression performance to the state of the art in *de facto* industry standards, i.e., gzip [38] as *de facto* industry standard for the compression of unaligned data in the FASTQ format, and BAM [6] and CRAM 3.0 [11] as *de facto* industry standards for the compression of aligned data in the SAM format. What is more, we also compare the compression performance of the MPEG-G encoder to the state of the art in nonstandard compression tools that can be found in the literature, namely, DSRC 2 [39] and Quip [40] for the compression of unaligned data and DeeZ [18] for the compression of aligned data.

We show results for ten data items from the MPEG-G Genomic Information Database [41]. The database includes sequencing data generated for different experiment types, such as WGS, cancer genome sequencing, metagenomics sequencing, and RNA sequencing. Also, the data originate from different species, such as *D. melanogaster*, *H. sapiens*, *T. cacao*, *S. cerevisiae*, *E. coli*, *P. aeruginosa*, and ΦX174. The data were generated with various sequencing technologies, such as sequencing by synthesis, single-molecule real time sequencing, nanopore sequencing, and ion semiconductor sequencing. MPEG-G was developed with the goal that it should "generalize" on the entire database. Hence, an MPEG-G encoder can select, based on, e.g., experiment type, species, and sequencing technology, the best combination of encoding parameters.

Seven of the ten data items used here originate from human WGS experiments, and special attention was paid to use data produced by different sequencing technologies. Specifically, the unaligned human WGS data were produced with Illumina HiSeq 2000, Ion Torrent PGM, and Illumina NovaSeq 6000 systems (items 01-1 & 01-2, 11, and 32, respectively). The aligned human WGS data were produced with Illumina HiSeq 2000, PacBio, and Illumina NovaSeq 6000 systems (items 02, 03, and 37, respectively). The three remaining data items cover additional use cases, such as ultrahigh depth virus (ΦX174) sequencing (item 33), human tumor sequencing (item 22), and (human) RNA-Seq (item 30). We refer the reader to the Supplemental Material for exhaustive results, which are also presented in [42]. A part of the results has already been presented in [43].

Fig. 9 shows the results for the unaligned data. Here, in addition to the compression results of the *de facto* industry-standard gzip and of the nonstandard compression tools DSRC 2 and Quip, we show the results of the mentioned MPEG-G encoder in two different

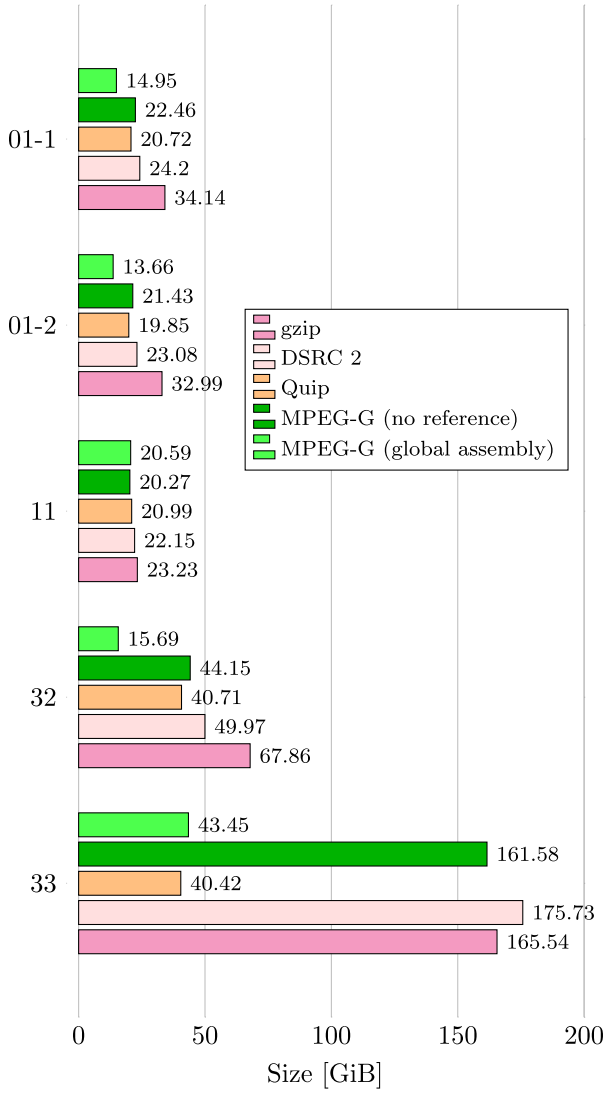


**Fig. 9.** ***Compression results for unaligned data. The results for "MPEG-G (no reference) " correspond to the MPEG-G encoder configured to work in a "low latency" application scenario using the "no reference" encoding approach (see Section V-B1). The results for "MPEG-G (global assembly)" correspond to the MPEG-G encoder configured to work in a "high compression ratio" application scenario using the "computed reference" encoding approach with the "global assembly" reference computation algorithm (see Section V-B1).***

configurations. The dark green bars display the compression results when the MPEG-G encoder is configured to use the "no reference" encoding approach, as it may be used in a "low latency" application scenario (see Section V-B1). As shown in Fig. 6, the descriptor generation stage is virtually bypassed in this case. The light green bars display the compression results when the MPEG-G encoder is configured to use the "computed reference" encoding approach with the "global assembly" reference computation algorithm, as it may be used in a "high compression ratio" application scenario (see Section V-B1).

Fig. 10 displays the results for the aligned data. Here, we compare the compressed sizes achieved by the MPEG-G encoder with the compressed sizes achieved by the *de facto* industry standards, BAM and CRAM 3.0, and the nonstandard compression tool, DeeZ. Here, the MPEG-G encoder is configured to use the "external reference" encoding approach, as it may be used in a "clinical study" application scenario (see Section V-B1).

For unaligned data, as observed from Fig. 9, the MPEG-G encoder produces smaller bitstreams than gzip in all cases. What is more, the MPEG-G encoder configured to use the "global assembly" encoding approach outperforms all other tools, except Quip for item 33. Here, the Quip encoding algorithm seems to exhibit a particularly high efficiency for the ultrahigh depth virus sequencing data.

For aligned data, as observed from Fig. 10, CRAM 3.0, DeeZ, and the MPEG-G encoder achieve significant improvements over BAM. Here, in particular, the MPEG-G encoder outperforms all other tools in all cases. These

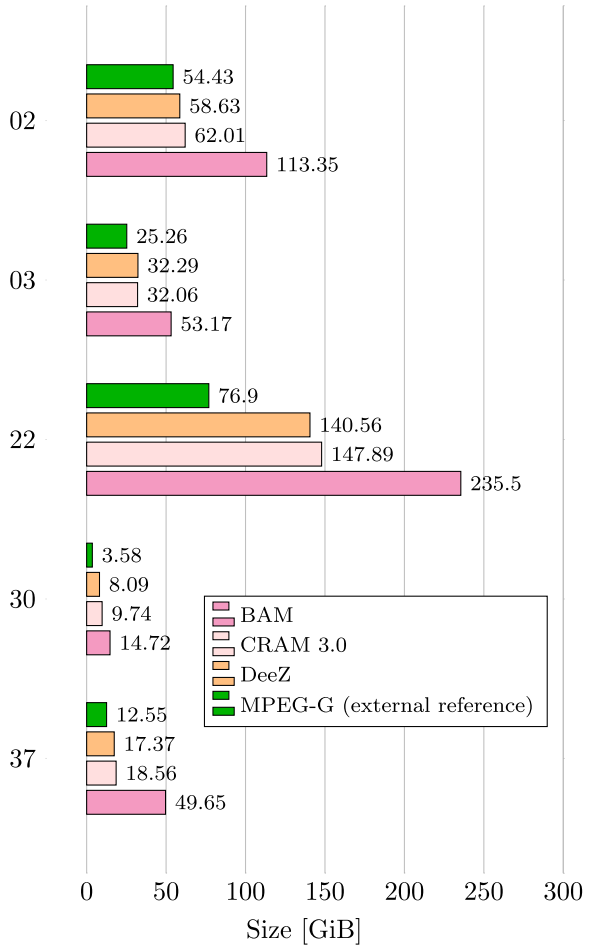


**Fig. 10.** ***Compression results for aligned data. The results for "MPEG-G (external reference)" correspond to the MPEG-G encoder configured to use the "external reference" encoding approach (which corresponds to the "clinical study" application scenario presented in Section V-B1). This approach is used to be able to make a fair comparison between the MPEG-G encoder and BAM, as well as CRAM 3.0, because both BAM and CRAM 3.0 use the concept of an external reference.***

results show that MPEG-G encoders typically create more compact representations than CRAM 3.0.

On top of that, the results obtained by MPEG-G technology can further be improved due to the freedom that the specification of the decoder in part 2 facilitates. For instance, an MPEG-G encoder may select beneficial quantization schemes for quality scores, which reduce the size of compressed quality scores by roughly 80% [26] while, at the same time, retaining variant calling performance.

A thorough evaluation of compression performance entails an evaluation of the computational complexity that is needed to achieve a certain compression. Here, in this regard, we use the encoding and decoding times as proxies for the computational complexity. For instance, on a typical compute server, gzip (running single-threaded) needs roughly 2.5 h to encode item 01-1. DSRC 2 (running with 16 threads) and Quip (running single-threaded) require 0.5 and 1.8 h, respectively. The MPEG-G encoder, using the "no reference" encoding approach, requires 1.7 h when running single-threaded and 20 min when running with eight threads. Also, typically, most codecs exhibit an asymmetric computation complexity, with decoding being usually much less complex. This applies, with the exception of Quip (which needs 153 min for decoding), also to the present example: gzip needs 15 min for decoding, and DSRC 2 needs 27 min. The MPEG-G decoder requires 64 (running single-threaded) and 11 min (running with eight threads), respectively. These examples only show a minor excerpt of our simulations. In summary, the MPEG-G implementation is in all cases faster than DSRC 2, Quip, BAM, CRAM 3.0, and DeeZ and only slightly outperformed in decoding time by gzip. In some cases, the MPEG-G implementation is even significantly faster, for example, in comparison with CRAM 3.0, where the MPEG-G implementation is roughly twice as fast. We refer the reader to the Supplemental Material for the exhaustive results.

Many factors impact the encoding and decoding performance in terms of timing. Most importantly, in most cases, only the decoding of a part of the compressed data is required, which is known as random access. Here, the MPEG-G encoder used to generate the results was configured to use an access unit size of 65 536 MPEG-G records. This has tremendous implications on random data access. For instance, item 01 contains 207 579 467 MPEG-G records, which is equivalent to twice as many FASTQ records, due to the paired-end sequencing protocol. In the case of gzip compression, all records are compressed into a single file with a size of 34 GiB. Hence, if a user wants to access a specific subset of records, the entire 34-GiB file needs to be decompressed. The compressed MPEG-G file ("no reference" encoding approach) has a size of 22 GiB, but, in contrast to the gzip file, every access unit can be accessed independently, where the average access unit size here is roughly 7 MiB. At the same time, MPEG-G offers, due to its advanced transport format, more efficient streaming capabilities than its alternatives.

## IX. CONCLUSION

The widely used formats, FASTQ and SAM/BAM, for representing genomic information were designed when sequencing data were scarce and precious, and the range of applications was limited. The new paradigm introduced by high-throughput sequencing machines—relatively inexpensive high coverage sequencing, with an almost infinite number of derived biological protocols and downstream analysis workflows—strongly encourages the adoption of a more sophisticated way to store, handle, and share genomic data. MPEG-G represents an important step in that direction. It paves the way to novel software solutions that will allow independent groups and organizations around the world to seamlessly communicate and share data, without losing interoperability with existing applications.

MPEG-G technology provides storage and transport capabilities for both unaligned and aligned sequencing data. It further supports the representation of both single reference genomes (assemblies) and collections thereof. Sequencing data and their associated metadata are sets of heterogeneous data, each characterized by its own statistical behaviors. Therefore, MPEG-G provides several strategies for the classification of these data and their representation. Within MPEG-G, the encoder optimization space for compression performance and selective data access is wide and enables many different solutions, which can be optimized for different applications and even for specific sequencing technologies and species. For example, an encoder can optimize the data compression mode for high compression and indexing (archival), or low latency (streaming applications). Furthermore, aligned reads can be compressed either reference-free or reference-based. The used reference sequences can be embedded as datasets within the same MPEG-G file or stored as external reference sequences, using an unambiguous specification of these external reference sequences. Quality scores can also be compressed either lossless or quantized.

MPEG-G comes with the tools to verify that a decoder complies with its different parts. Furthermore, bitstreams can be verified to be MPEG-G compliant. These are essential features that enable the development of independent and yet compatible solutions for clinical usage and analysis of omics data. These tools also ensure that future extensions of MPEG-G will not break any compatibility with existing systems.

In analogy to the digital media industry MPEG-G aims to make genomic data access, processing, and sharing—either in the cloud or on local storage—as simple as streaming an audio file or watching a movie. One of the main drivers toward this goal has been the open and fair process of technology evaluation and specification under the supervision of international and neutral institutions, such as ISO and IEC. With this objective in mind, MPEG is currently working to extend the scope of MPEG-G to tertiary analysis results and annotations. This work is

expected to be finalized in early 2022 and will provide a unique fully indexable container of compressed genomic information. ■

## Acknowledgment

The development of the MPEG-G specification is a collaborative effort. The following people contributed to the actual MPEG-G development: Junaid J. Ahmad, Claudio Alberti, Simone Casale-Brunet, Patrick Cheung, Jaime Delgado, Jan Fostier, Silvia Llorente, Liudmila S. Mainzer, Fabian Müntefering, Daniel Naro, Ibrahim Numanagić, Idoia Ochoa, Tom Paridaens, Massimo Ravasi, Daniele Renzi, Paolo Ribeca, and Giorgio Zoia. MPEG received additional input from other experts, including Bonnie Berger, Noah Daniels, Nicolas Guex, Christian Iseli, Raymond Krasinski, Christian Rohlfing, S. Cenk Sahinalp, and Ioannis Xenarios.

## REFERENCES

[1] Z. D. Stephens *et al.*, "Big data: Astronomical or genomical?" *PLoS Biol.*, vol. 13, no. 7, Jul. 2015, Art. no. e1002195.

[2] D. Pavlichin, T. Weissman, and G. Mably, "The quest to save genomics: Unless researchers solve the looming data compression problem, biomedical science could stagnate," *IEEE Spectr.*, vol. 55, no. 9, pp. 27–31, Sep. 2018, doi: 10.1109/MSPEC.2018.8449046.

[3] I. Numanagić *et al.*, "Comparison of high-throughput sequencing data compression tools," *Nature Methods*, vol. 13, no. 12, pp. 1005–1008, Dec. 2016.

[4] M. Hernaez, D. Pavlichin, T. Weissman, and I. Ochoa, "Genomic data compression," *Annu. Rev. Biomed. Data Sci.*, vol. 2, no. 1, pp. 19–37, Jul. 2019.

[5] P. J. A. Cock, C. J. Fields, N. Goto, M. L. Heuer, and P. M. Rice, "The Sanger FASTQ file format for sequences with quality scores, and the Solexa/Illumina FASTQ variants," *Nucleic Acids Res.*, vol. 38, no. 6, pp. 1767–1771, Apr. 2010.

[6] H. Li *et al.*, "The sequence alignment/map format and SAMtools," *Bioinformatics*, vol. 25, no. 16, pp. 2078–2079, Aug. 2009.

[7] C. Alberti *et al.*, "An introduction to MPEG-G, the new ISO standard for genomic information representation," *bioRxiv*, 2018, Art. no. 426353. [Online]. Available: https://www.biorxiv.org/content/early/2018/10/08/426353, doi: 10.1101/426353.

[8] S. Goodwin, J. D. McPherson, and W. R. McCombie, "Coming of age: Ten years of next-generation sequencing technologies," *Nature Rev. Genet.*, vol. 17, no. 6, pp. 333–351, Jun. 2016.

[9] J. L. Weirather *et al.*, "Comprehensive comparison of Pacific Biosciences and Oxford Nanopore Technologies and their applications to transcriptome analysis," *F1000Research*, vol. 6, p. 100, Feb. 2017.

[10] J. T. Robinson *et al.*, "Integrative genomics viewer," *Nature Biotechnol.*, vol. 29, no. 1, pp. 24–26, Jan. 2011.

[11] M. H.-Y. Fritz, R. Leinonen, G. Cochrane, and E. Birney, "Efficient storage of high throughput DNA sequencing data using reference-based compression," *Genome Res.*, vol. 21, no. 5, pp. 734–740, May 2011.

[12] J. Kelleher *et al.*, "Htsget: A protocol for securely streaming genomic data," *Bioinformatics*, vol. 35, no. 1, pp. 119–121, Jan. 2019.

[13] S. Grabowski, S. Deorowicz, and Ł. Roguski, "Disk-based compression of data from genome sequencing," *Bioinformatics*, vol. 31, no. 9, pp. 1389–1395, May 2015.

[14] 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, Feb. 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, Aug. 2019.

[16] 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, Aug. 2018.

[17] J. Voges, M. Munderloh, and J. Ostermann, "Predictive coding of aligned next-generation sequencing data," in *Proc. Data Compress. Conf. (DCC)*, Mar. 2016, pp. 241–250.

[18] F. Hach, I. Numanagic, and S. C. Sahinalp, "DeeZ: Reference-based compression by local assembly," *Nature Methods*, vol. 11, no. 11, pp. 1082–1084, Nov. 2014.

[19] I. Ochoa, M. Hernaez, R. Goldfeder, T. Weissman, and E. Ashley, "Effect of lossy compression of quality scores on variant calling," *Briefings Bioinf.*, vol. 18, no. 2, pp. 183–194, 2017.

[20] C. Alberti *et al.*, "An evaluation framework for lossy compression of genome sequencing quality values," in *Proc. Data Compress. Conf. (DCC)*, Mar. 2016, pp. 221–230.

[21] D. L. Greenfield, O. Stegle, and A. Rrustemi, "GeneCodeq: Quality score compression and improved genotyping using a Bayesian framework," *Bioinformatics*, vol. 32, no. 20, pp. 3124–3132, Oct. 2016.

[22] I. Ochoa, H. Asnani, D. Bharadia, M. Chowdhury, T. Weissman, and G. Yona, "QualComp: A new lossy compressor for quality scores based on rate distortion theory," *BMC Bioinf.*, vol. 14, no. 1, p. 187, Dec. 2013.

[23] Y. W. Yu, D. Yorukoglu, J. Peng, and B. Berger, "Quality score compression improves genotyping accuracy," *Nature Biotechnol.*, vol. 33, no. 3, pp. 240–243, Mar. 2015.

[24] G. Malysa, M. Hernaez, I. Ochoa, M. Rao, K. Ganesan, and T. Weissman, "QVZ: Lossy compression of quality values," *Bioinformatics*, vol. 31, no. 19, pp. 3122–3129, Oct. 2015.

[25] M. Hernaez, I. Ochoa, and T. Weissman, "A cluster-based approach to compression of quality scores," in *Proc. Data Compress. Conf. (DCC)*, Mar. 2016, pp. 261–270.

[26] J. Voges, J. Ostermann, and M. Hernaez, "CALQ: Compression of quality values of aligned sequencing data," *Bioinformatics*, vol. 34, no. 10, pp. 1650–1658, May 2018.

[27] J. K. Bonfield and M. V. Mahoney, "Compression of FASTQ and SAM format sequencing data," *PLoS ONE*, vol. 8, no. 3, Mar. 2013, Art. no. e59190.

[28] I. Ochoa, H. Li, F. Baumgarte, C. Hergenrother, J. Voges, and M. Hernaez, "AliCo: A new efficient representation for SAM files," in *Proc. Data Compress. Conf. (DCC)*, Mar. 2019, pp. 93–102.

[29] D. Marpe, H. Schwarz, and T. Wiegand, "Context-based adaptive binary arithmetic coding in the H.264/AVC video compression standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 620–636, Jul. 2003.

[30] T. Wiegand, G. J. Sullivan, G. Bjøntegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 560–576, Jul. 2003.

[31] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand, "Overview of the high efficiency video coding (HEVC) standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1649–1668, Dec. 2012.

[32] T. Paridaens, G. Van Wallendael, W. De Neve, and P. Lambert, "AFRESh: An adaptive framework for compression of reads and assembled sequences with random access functionality," *Bioinformatics*, vol. 33, no. 10, pp. 1464–1472, 2017.

[33] T. Paridaens, G. Van Wallendael, W. De Neve, and P. Lambert, "AQUa: An adaptive framework for compression of sequencing quality scores with random access functionality," *Bioinformatics*, vol. 34, no. 3, pp. 425–433, Feb. 2018.

[34] J. Voges *et al.*, "GABAC: An arithmetic coding solution for genomic data," *Bioinformatics*, vol. 36, no. 7, pp. 2275–2277, Apr. 2020.

[35] J. Ziv and A. Lempel, "A universal algorithm for sequential data compression," *IEEE Trans. Inf. Theory*, vol. IT-23, no. 3, pp. 337–343, May 1977.

[36] R. Leinonen, H. Sugawara, M. Shumway, and On Behalf of the International Nucleotide Sequence Database Collaboration, "The sequence read archive," *Nucleic Acids Res.*, vol. 39, pp. D19–D21, Jan. 2011.

[37] *eXtensible Access Control Markup Language (XACML) Version 3.0*, Org. Adv. Struct. Inf. Standards, Burlington, MA, USA, 2013.

[38] L. P. Deutsch. (1996). *GZIP File Format Specification Version 4.3*. [Online]. Available: https://tools.ietf.org/html/rfc1952

[39] Ł. Roguski and S. Deorowicz, "DSRC 2—Industry-oriented compression of FASTQ files," *Bioinformatics*, vol. 30, no. 15, pp. 2213–2215, Aug. 2014.

[40] 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 Res.*, vol. 40, no. 22, p. e171, Dec. 2012.

[41] *MPEG-G Genomic Information Database*, Standard ISO/IEC JTC1/SC29/WG11, N18963, 2019.

[42] J. Voges, A. Salvucci, C. Alberti, and M. Mattavelli, *Results of MPEG-G Codec Performance and Other State-of-the-Art Compression Algorithms*, Standard ISO/IEC JTC1/SC29/WG8 M56361, 2021.

[43] *MPEG-G Performance Benchmarks*, Standard ISO/IEC JTC1/SC29/WG11, N19559, 2019.