GABAC: an arithmetic coding solution for genomic data

Tom Paridaens, Jan Voges, Mikel Hernaez, Jan Fostier, and Jörn Ostermann

In an effort to provide a response to the ever-expanding generation of genomic data, MPEG (Moving Picture Experts Group), under the auspices of ISO (International Organization for Standardization), is designing a new solution for the representation, compression and management of genomic sequencing data: the MPEG-G standard. This standard specifies an abstract representation of sequencing data and offers support for, among others, multi-dimensional random access, extensive meta-data information, data and privacy protection, data storage and streaming. Part 2 of the MPEG-G standard focuses on specifying the coding of the sequencing data. Note that the standard is a set of specifications (i.e., a book) assembled after an open competition for technologies, and thus, it does not provide, *per se*, actual encoding implementations. This paper discusses the first baseline implementation of an MPEG-G compliance entropy encoder/decoder: GABAC. GABAC combines proven coding technologies, such as context-adaptive binary arithmetic coding (CABAC) [1], binarization schemes, and transformations into one straight-forward solution for the compression of the sequencing data. GABAC will be part of the open-source full MPEG-G encoder/decoder suite Genie, currently under development by the Mitogen initiative[1].

Methods: The MPEG-G standard represents genomic information by splitting the data into a set of descriptor streams. Each descriptor stream has been designed to contain one specific type of data (e.g., mapping positions, quality scores or mismatch information). Thanks to this approach, all data contained in a descriptor stream is homogeneous. Note that other genomic data compressors also employ a similar approach of generating streams containing statistically similar data [2, 3]. This property allows for effective data compression, as redundancies are more probable and value distributions are more predictable.

Given an input stream, the compression process specified in the MPEG-G standard consists of a five-step pipeline: input parsing, (optional) 3-step transformations, symbol binarization, context selection, and CABAC.

In the input parsing step, the descriptor stream is parsed into a stream of symbols. These symbols are then processed by the 3-step transformation step, which converts the symbol stream into one or more transformed sub-streams. Available transformations are as follows. i) run-length encoding, where repetitions of symbols are translated into a symbol sub-stream and a length sub-stream); ii) match coding, an LZ77-style transformation where blocks of symbols are replaced by a pointer and length value, indicating either the position and the length of a perfect match of this block in the previously encoded symbols, or indicating a raw symbol if no match is found; iii) equality coding, where a symbol is replaced by a flag indicating equality of the symbol and its predecessor and a correction symbol, if required; iv) a look-up table transformation; and v) differential coding.

In the first part of the transformation step, the symbols are processed using either run-length encoding, match coding, or equality coding. In the second part of the transformation step, which is applied to each transformed sub-stream separately, a look-up table transformation can be performed. Finally, in the third part of the transformation step, a differential coding can be applied.

Each transformed sub-stream is then processed separately during the rest of the process. For each sub-stream a binarization algorithm, used for conversion of each symbol into a bit string, is chosen together with a context selection algorithm. In the last step, each bit of the binarization (called a bin) is combined with a context (selected using the context selection algorithm) and both are processed using CABAC.

Results: To analyze the performance of the GABAC encoder, a test set of 206 descriptor stream files (cropped to a maximum size of 200 MiB each to emulate random access capabilities) has been selected. These files contain data (such as mapping positions, pairing information, or unmapped reads) generated from items 02, 03, 05, 07, 08, 09, 10, and 20 of the MPEG-G Genomic Information Database[2]. The total uncompressed size is 12.97 GiB.

The performance of GABAC was compared to the codecs that are used in CRAM, i.e. gzip, bzip2, xz, rANS order-0, and rANS order-1. The rANS codecs were extracted from CRAM and are publicly available[3]. The tests were performed in parallel on a set of five servers, each equipped with 2 Intel Xeon E5-2650 v3 CPUs and 128 GiB of RAM, running Ubuntu 14.04.

Table 1 shows the total compressed size and the total encoding and decoding times for each of the tested codecs for all considered streams. Additionally, the table shows the compressed size for the complete test set when, for each file, the codec with the highest compression ratio across the CRAM encoders is selected (CRAM), and across the CRAM encoders and GABAC (CRAM + GABAC). GABAC offers the highest compression ratio across all coding solutions, while being 5.5 times faster than the second best compressor. Additionally, adding GABAC to the CRAM set of encoders offers an additional compression gain of 79 MiB and a speed-up with a factor of 2.4 in compression time.

	Compressed Size (MiB)	Encoding Time	Decoding Time
gzip	3,524	3h 25m 18s	**06m 02s**
bzip2	3,088	33m 55s	20m 00s
xz	2,944	4h 47m 38s	09m 25s
rANS-0	4,143	**06m 01s**	07m 08s
rANS-1	3,400	06m 54s	08m 20s
GABAC	**2,877**	45m 25s	20m 18s
CRAM	2,879	2h 25m 58s	**09m 32s**
CRAM + GABAC	**2,800**	**1h 01m 17s**	20m 08s

Table 1: Total compressed size and decoding & encoding times.

Figure 1 shows the compression ranking for each codec and each descriptor stream file. The x-axis shows the MPEG-G Ge-

[1] https://github.com/mitogen

[2] https://mpeg.chiariglione.org/standards/MPEG-G/genomic-information-representation/MPEG-G-genomic-information-database

[3] https://github.com/voges/rans