OXFORD

Sequence analysis

GABAC: an arithmetic coding solution for genomic data

Jan Voges ⬤ [1],*,[†], Tom Paridaens ⬤ [2],*,[†], Fabian Müntefering[1], Liudmila S. Mainzer[3,4], Brian Bliss[3], Mingyu Yang[5], Idoia Ochoa[5], Jan Fostier ⬤ [2], Jörn Ostermann[1] and Mikel Hernaez ⬤ [4],*

[1]Institut für Informationsverarbeitung (TNT), Leibniz University Hannover, 30167 Hannover, Germany, [2]IDLab, Ghent University—imec, 9050 Ghent, Belgium, [3]National Center for Supercomputing Applications, [4]Carl R. Woese Institute for Genomic Biology and [5]Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA

*To whom correspondence should be addressed.

[†]The authors wish it to be known that, in their opinion, the first two authors should be regarded as Joint First Authors.

Associate Editor: John Hancock

Abstract

Motivation: In an effort to provide a response to the ever-expanding generation of genomic data, the International Organization for Standardization (ISO) is designing a new solution for the representation, compression and management of genomic sequencing data: the Moving Picture Experts Group (MPEG)-G standard. This paper discusses the first implementation of an MPEG-G compliant entropy codec: GABAC. GABAC combines proven coding technologies, such as context-adaptive binary arithmetic coding, binarization schemes and transformations, into a straightforward solution for the compression of sequencing data.

Results: We demonstrate that GABAC outperforms well-established (entropy) codecs in a significant set of cases and thus can serve as an extension for existing genomic compression solutions, such as CRAM.

Availability and implementation: The GABAC library is written in C++. We also provide a command line application which exercises all features provided by the library. GABAC can be downloaded from https://github.com/mitogen/gabac.

Contact: voges@tnt.uni-hannover.de or tom.paridaens@ugent.be or mhernaez@illinois.edu

Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

The Moving Picture Experts Group (MPEG or ISO/IEC JTC1/SC29/WG11) is designing a new solution for the representation, compression and management of genomic sequencing data: the MPEG-G standard. The standard in itself consists of a set of decoding specifications and does not provide, *per se*, actual encoding implementations.

This paper discusses the first implementation of an MPEG-G compliant entropy codec: GABAC. GABAC combines proven coding technologies, such as context-adaptive binary arithmetic coding (CABAC) (Marpe *et al.*, 2003), binarization schemes and transformations, into a straightforward solution for the compression of sequencing data. Although GABAC has been designed following the MPEG-G specifications, in what follows we will show that integrating GABAC into other genomic compression solutions, such as CRAM (Bonfield, 2014; Fritz *et al.*, 2011) or DeeZ (Hach *et al.*, 2014) can lead to significant improvements.

2 Materials and methods

The first step of typical state-of-the-art genomic data compressors such as CRAM, DeeZ or implementations of the MPEG-G standard is to represent genomic information by splitting the data into a set of homogeneous descriptor streams, each containing one specific type of data (e.g. mapping positions, quality scores, mismatch information, etc.). Each stream is more likely to contain stationary data, enabling a more effective data compression, as redundancies are more probable and value distributions are more predictable.

Given an input stream, the compression process implemented by GABAC (and specified in the MPEG-G standard) consists of a five-stage pipeline: input parsing, (optional) 3-step transformation, 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 stage that converts the symbol stream into one or more transformed substreams. In the first transformation step, the input symbols are processed using one of the following three transformations (or no
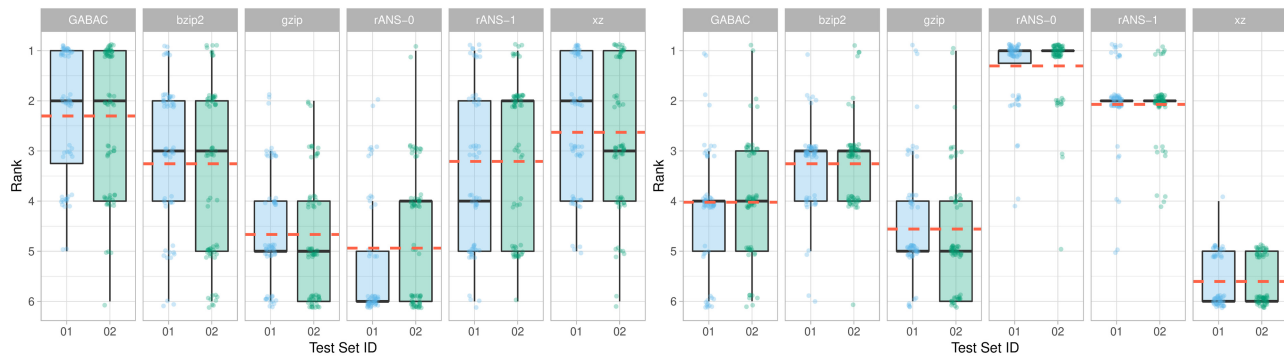
Fig. 1. Rank of compression performance (L) and speed (R). Dots were jittered for clarity. The *x*-axes show the test set ID (01 and 02) from which the descriptor stream files were generated. The *y*-axes denote the actual ranks. Each dot depicts the ranking a codec achieved on one specific descriptor stream file. The red lines denote the mean ranks, averaged over both test items. (Color version of this figure is available at *Bioinformatics* online.)

Table 1. Compressed sizes (in bytes) for different codec sets applied to the CRAM and DeeZ descriptor streams

		Uncompressed streams size	CRAM	CRAM+GABAC	CRAM+GABAC−gzip−rANS-0
CRAM streams	01	2 196 327 888	412 453 039	**409 337 321**	**409 337 321**
	02	1 991 360 268	406 796 124	**404 059 221**	404 080 825
DeeZ streams	01	2 838 530 295	538 155 557	**535 118 397**	**535 118 397**
	02	2 555 531 164	566 444 432	**564 839 381**	564 864 140

Note: Best results in bold.

transformation, i.e. pass-through): (i) run-length coding, where a repeated symbol is replaced by the symbol itself and its run-length; (ii) match coding, an LZ77-style transformation and (iii) equality coding, where a symbol is replaced by a flag indicating equality with its predecessor and a correction symbol, if required. In the second transformation step, for each sub-stream separately, a look-up table transformation can be applied. Finally, in the third transformation step, one can choose to apply differential coding. For each transformed sub-stream, a binarization algorithm (used for conversion of each symbol into a bit string) is picked together with a context selection algorithm. In the last step, each bit of the binarization is combined with a context and both are processed using CABAC. For more detailed explanations on the GABAC working principles, we refer the reader to the Supplementary Data.

3 Results and discussion

To analyze the performance of the GABAC encoder, we modified the compression tools CRAM and DeeZ to output their internal data representations, such as mapping positions and read pairing information, to separate descriptor stream files. We encode every descriptor stream with the (entropy) codecs used in CRAM (i.e. bzip2, gzip, rANS order-0, rANS order-1 or xz), plus GABAC. This way we can compare the performance of GABAC to the currently used (entropy) codecs.

As input to our modified versions of CRAM and DeeZ, we used chromosome 11 from items 01 and 02 of the MPEG-G Genomic Information Database (https://mpeg-g.org), whose corresponding BAM files occupy 6.9 and 4.2 GiB, respectively. This resulted in a test set of 129 descriptor streams. To further emulate block-wise compression (i.e. random access capabilities), all streams were limited to 200 MiB, which resulted in a test set size of 8.9 GiB. This approach allows for the analysis of a more extensive test set in a random access environment, while preserving a reliable representation of the coding performance for each of the compared codecs. More information on the experimental details are provided as Supplementary Data.

Figure 1 shows, for each codec, the ranking distribution for the different streams along the compression axis (left) and the compression speed axis (right). Speed ranks were computed excluding the times used for the search for the best GABAC configuration.

Note that the rANS-0 and -1 executables were produced without compiler optimizations. The dotted red lines denote the mean rank for each codec, averaged over both test items. To provide a proxy for the spread between the ranks, we computed the average compression ratios (i.e. compressed size divided by uncompressed size) and speeds for the set of codecs that rank first (22% and 49 MiB/s, respectively) and the set of codecs that rank last (34% and 2 MiB/s, respectively). As it can be observed in Fig. 1, GABAC obtains the best compression ratios on average. GABAC is also faster than gzip and xz provided that its optimal configuration [i.e. the optimal combination of transformation(s), binarization(s) etc.] is known (see Supplementary Data).

To further analyze to which extent existing solutions can benefit from the incorporation of GABAC, Table 1 shows the compressed sizes for different codec sets (where for each stream the codec with the highest compression ratio is selected): the set of codecs used in CRAM, the set of codecs used in CRAM plus GABAC and the set of codecs used in CRAM plus GABAC with gzip and rANS order-0 removed. This table shows that adding GABAC to the set of CRAM codecs further improves compression effectiveness. Removing gzip does not affect the compression [Fig. 1 (left) shows that gzip is never ranked first] and removing rANS order-0 has only a minor effect (<0.005%).

Hence, we can conclude that adding GABAC as entropy codec to both CRAM and DeeZ is beneficial both in terms of compression ratio and speed. Additionally, it has been shown that the (entropy) codec set can be limited to bzip2, rANS order-1, GABAC and xz, with no performance degradation. Finally, we can conclude that it would be valuable for implementations of the MPEG-G standard to be able to employ other (entropy) codecs in addition to GABAC.

4 Conclusion

GABAC, the first implementation of an entropy codec as specified in the MPEG-G standard, already outperforms well-established (entropy) codecs, improving in several cases both the compression ratio and the compression speed. In addition, the proposed implementation forms a valid extension to the set of (entropy) codecs used in CRAM. We hope that it will serve as a baseline for future implementations, within and outside of MPEG-G, as the performance of new implementations is expected to improve over time.

Funding

Conflict of Interest: none declared.

References

Bonfield,J.K. (2014) The Scramble conversion tool. *Bioinformatics*, **30**, 2818–2819.

Fritz,M.H.-Y. *et al.* (2011) Efficient storage of high throughput DNA sequencing data using reference-based compression. *Genome Res.*, **21**, 734–740.

Hach,F. *et al.* (2014) DeeZ: reference-based compression by local assembly. *Nat. Methods*, **11**, 1082–1084.

Marpe,D. *et al.* (2003) Context-based adaptive binary arithmetic coding in the H.264/AVC video compression standard. *IEEE Trans. Circuits Syst. Video Technol.*, **13**, 620–636.