

Comparative Analysis of Arithmetic Coding Computational Complexity

Amir Said

Hewlett-Packard Laboratories, 1501 Page Mill Rd., ms 1203, Palo Alto, CA, USA

E-mail: *said@ieee.org*

New hardware is making obsolete previous assumptions about the most demanding computations for arithmetic coding. For instance, it is not advantageous to avoid multiplications that can be done in a single CPU clock cycle. To design efficient implementations for the modern processors we need to have a good understanding of the computation costs [1].

The important arithmetic coding tasks include: (a) Interval update and arithmetic operations; (b) Carry propagation and bit moves; (c) Interval renormalization; (d) Interval search for decoding; (e) Probability estimation (source modeling); (f) Support for non-binary symbol alphabets (binary coders only) [1,2]. In this work we profile these computations by comparing the running times of many implementations, changing one part at a time.

We benchmark the performance of binary arithmetic coding implemented with 16, 32 bit integer and 48 bit floating-point arithmetic, compared with the MQ encoder [4]. Next, we evaluate the changes from bit to multi-byte renormalization [3], and carry propagation. We also tested static and adaptive coders, with different ways of updating cumulative distributions or binary probability estimates [1,2].

The results show that arithmetic operations, except division, have now small costs. It was found that even the floating-point version is competitive, and that replacing one division with several multiplications can be advantageous. Source modeling can be quite time-consuming, depending on its strategy and exploitation of the arithmetic coding properties. Requiring an update for each coded symbol implies the need for divisions, or use of binary coders. Results show that significantly faster adaptation, with small coding loss, can be obtained by periodic updates of the probability estimates [1]. This strategy also proved to be faster than tree-based coding and/or cumulative distribution updating [2].

- [1] A. Said, "Arithmetic Coding," in *Lossless Compression Handbook* (K. Sayood, ed.), Academic Press, San Diego, CA, 2003.
- [2] A. Moffat, R.M. Neal, and I.H. Witten, "Arithmetic coding revisited," *ACM Transactions on Information Systems*, vol. 16, no. 3, pp. 256-294, 1998.
- [3] M. Schindler, "A fast renormalisation for arithmetic coding," *Proc. IEEE Data Compression Conference*, 1998.
- [4] D.S. Taubman and M.W. Marcellin, *JPEG 200 Image Compression Fundamentals, Standards and Practice*, Kluwer Academic Publishers, Boston, 2002.