Mixed Huffman codes for on-line and off-line applications

Ryszard Stasinski*, and Grzegorz Ulacha†

*Institute of Multimedia
Telecommunications
Poznan University of Tech.
Poznan, 61-131, Poland
ryszard.stasinski@put.poznan.pl

† Faculty of Computer Science
and Information Technology
West Pomeranian University of Tech.
Szczecin, 71-210, Poland
gulacha@wi.zut.edu.pl

Abstract

In the paper Huffman codes that mix different r-nary code elements in one code, the *mixed Huffman codes*, are analyzed [1]. The Huffman code generalization usually leads to shortening of average codeword length: a statistical test shows that for source alphabets longer than 8-12 elements more than 99% of the best compact codes are mixed Huffman ones. This is also true for practical mixed Huffman codes, which is demonstrated in experiments with data files containing up to milion elements for sources of size 12-17 symbols. The codes are derived in the same way as other Huffman ones: iteratively by reducing in each step source size by $r-1$ elements, the only difference is that the r value may change from step to step (reasonable values of r are prime numbers). Search for optimal code is a trial and error process, nevertheless, usually they are several suboptimal mixed Huffman codes that are better than the binary ones, hence, exhaustive search for optimal solution is not necessary. It is worth to note that not described in this paper generated using simplified search rules dynamic mixed Huffman codes [2] are usually better than their binary counterparts, too. The mixed Huffman code is coded and decoded using modified Huffman tree, in which some nodes have r instead of 2 offsprings. The r-nary elements are grouped to form r-nary numbers being close, but slightly smaller than some powers of number two, e.g. three 5-nary digits define numbers up to 124, which can be coded using seven bits. In fact, bit number can be lowered by variable-length coding, in the example above one 5-nary element is coded using 2.3333 bits, the value can be diminished to 2.3253 bits, while $\log 5 = 2.3219$. This technique implies that before sending r-nary elements should be collected from more than one codeword, leading to delays on the decoder side, which may be important in on-line applications. It is shown in the paper that delays can be kept short, while improved coding efficiency retained. For example, if each codeword contains at least one r-nary element, the delay introduced by this element is shorter than the size of buffer for its storing, e.g. in the example above for $r = 5$ the delay is shorter than three codewords. As noted before there are usually many mixed Huffman codes better than the binary one, hence, the chance that such minimum delay and efficient code exists is high. Of course, in many applications coding delay is not a problem. Summarizing, coders and decoders for practical mixed Huffman codes are simple and fast, while theoretical considerations and experiments with true data show that indeed, their use usually leads to better data compression, if compared to that for the binary Huffman code.

References

[1] R. Stasinski and G. Ulacha, "Huffman codes revisited," in *Proc. 24th Symp. on Information Theory in the Benelux*, 2003, pp. 63–70.

[2] G. Ulacha and R. Stasinski, "Dynamic mixed Huffman codes," in *Proc. ICSES'04*, 2004, pp. 537–540.

2022 Data Compression Conference (DCC) | 978-1-6654-7893-9/22/$31.00 ©2022 IEEE | DOI: 10.1109/DCC52660.2022.00094

2375-0359/22/$31.00 ©2022 IEEE
DOI 10.1109/DCC52660.2022.00094