Decoding of Canonical Huffman Codes with Look-Up Tables

Yakov Nekritch
Department of Computer Science
University of Bonn.
e-mail: yasha@cs.uni-bonn.de

This paper presents an efficient algorithm for decoding canonical Huffman codes with look-up tables.

Canonical codes are a subclass of Huffman codes, that have a numerical sequence property, i.e. codewords with the same length are binary representations of consecutive integers. In case of decoding with look-up tables we read a number of bits at each step of decoding process. We look-up the value of read bit sequence in a table and, if this bit sequence contains a codeword, we output the corresponding symbol. Otherwise we proceed with the decoding, using the next look-up table or using some other method.

We describe an efficient method for decoding of canonical Huffman codes with look-up tables. We read a sequence of bits $b$ and output all symbols, corresponding to codewords, contained in $b$. Thus, more than one codeword can be decoded at each algorithm step, which leads to efficient decoding of short codewords. Besides that, if only the beginning of a codeword is contained in the string $b$, we use this codeword prefix to determine or limit the length of this codeword. An additional look-up table or some other method can be then used for decoding of this codeword. We also propose an algorithm for estimating the source file size from the specification of a canonical Huffman code and suggest dynamic choice of the look-up table size, based on the estimated file size.

We have tested our algorithm on files from Calgary compression corpus (see T.C. Bell, J.G. Cleary, I.H. Witten, “Text Compression”, Prentice Hall, 1990). As a speed criterion we have used the average number of decoded symbols per read bit sequence. For our method number of decoded symbols per bit string ranges from 1.4 to 6.3. Reduction in the number of bit manipulations leads to faster decoding. Concrete results are, however, implementation- and machine-dependent. Practical experiments, performed on SUN Sparc 5 workstation show 10 to 25% reduction in decoding time for large files (over 200 Kbytes), compared with the state-of-the-art look-up method by Moffat and Turpin (see Moffat A. and Turpin A., ” On the Implementation of minimum redundancy prefix codes”, IEEE Trans. Communications, vol. 45, 1200 - 1207).

In our experiments the source alphabet consisted of single characters, occurring in a text. We are planning to perform experiments with large text databases, in which the source alphabet consists of all words occurring in a text.