Experimental Study of a Binary Block Sorting Compression Scheme*

Daniel A. Nagy† and Tamás Linder†‡
†Department of Mathematics and Statistics
‡Department of Electrical and Computer Engineering
Queen's University, Kingston, Ontario, Canada.
Email: `{nagydani,linder}@mast.queensu.ca`

In this work, we provide an experimental evaluation of a block-sorting compression scheme that operates on bit level. We demonstrate that even such a simple technique, which ignores byte boundaries and uses a very simple modeling scheme for the output of the block-sorting transform, outperforms some of the best industry standard compressors for sources that are not byte-aligned, while providing reasonable compression ratios for byte-aligned sources. Although the scheme can be substantially improved using more sophisticated modeling and coding techniques, the preliminary experimental results point out the potential advantages of our approach.

The compression algorithm comprises block-sorting, modeling, and coding steps. The block-sorting algorithm yields the Burrows-Wheeler transform [1] on the binary representation (in which the end-of-sequence character ranks between 0 and 1 in the alphabetic order) through the creation and subsequent traversal of the (binary) suffix tree, resulting in linear time and memory costs [2], albeit with a considerable constant coefficient. The model is a displaced power-distribution of the runlengths, which is encoded using a Huffman-based scheme. The parameters of the power-distribution are stored in order to be able to recreate the codebook on the receiving end.

The experimental evaluation is performed on the Calgary Corpus for byte-aligned input and the first-order Huffman-coded version thereof for the non-byte-aligned case. The results show that our simple compressor performs somewhat worse than the sophisticated `bzip2` compressor on the ascii coded inputs, but substantially outperforms it on the Huffman-coded versions. In particular, our compressor compressed the original and the Huffman coded files to essentially the same length, while `bzip2`'s byte-based approach breaks down in the case of variable symbol length.

References

[1] M. Burrows and D. J. Wheeler, 'A Block-sorting Lossless Data Compression Algorithm," *SRC Research Report 124*, DSRC, Palo Alto, CA, May 1994.

[2] E. Ukkonen, 'On-line construction of suffix trees," *Algorithmica*, 14(3), 1995.

*This work was supported in part by NSERC of Canada and PREA of Ontario.