Adapting the Knuth-Morris-Pratt Algorithm for Pattern Matching in Huffman Encoded Texts

Ajay Daptardar and Dana Shapira
{amax/shapird}@cs.brandeis.edu
Computer Science Department, Brandeis University, Waltham, MA

We perform *compressed pattern matching* in Huffman encoded texts. A modified Knuth-Morris-Pratt (KMP) algorithm is used in order to overcome the problem of *false matches*, i.e., an occurrence of the encoded pattern in the encoded text that does not correspond to an occurrence of the pattern itself in the original text. We propose a bitwise KMP algorithm that can move one extra bit in the case of a mismatch, since the alphabet is binary. To avoid processing any encoded text bit more than once, a preprocessed table is used to determine how far to back up when a mismatch is detected, and is defined so that the encoded pattern is always aligned with the start of a codeword in the encoded text. We combine our KMP algorithm with two Huffman decoding algorithms which handle more than a single bit per machine operation; Skeleton trees defined by Klein [1], and numerical comparisons between special canonical values and portions of a sliding window presented in Moffat and Turpin [3]. We call the combined algorithms *sk-kmp* and *win-kmp* respectively.

The following table compares our algorithms with *cgrep* of Moura et al. [2] and *agrep* which searches the uncompressed text. Columns three and four compare the compression performance (size of the compressed text as a percentage of the uncompressed text) of the Huffman code (*huff*) with *cgrep*. The next columns compare the processing time of pattern matching of these algorithms. The "decompress and search" methods, which decode using skeleton trees or Moffat and Turpin's sliding window and search in parallel using *agrep*, are called *sk-d* and *win-d* respectively. The search times are average values for patterns ranging from infrequent to frequent ones.

Files	Size (bytes)	Compression		Search Times (sec)				
		cgrep	*huff*	cgrep	sk-kmp	win-kmp	sk-d	win-d
world192.txt	2,473,400	50.88	32.20	0.07	0.13	0.08	0.21	0.13
bible.txt	4,047,392	49.70	26.18	0.05	0.22	0.13	0.36	0.22
books.txt	12,582,090	52.10	30.30	0.21	0.69	0.39	1.21	0.74
95-03-erp.txt	23,976,547	34.49	25.14	0.18	1.10	0.65	1.80	1.11

As can be seen, the KMP variants are faster than the methods corresponding to "decompress and search" but slower than *cgrep*. However, when compression performance is important or when one does not want to re-compress Huffman encoded files in order to use *cgrep*, the proposed algorithms are the better choice.

References

[1] Klein S.T., Skeleton Trees for efficient decoding of Huffman encoded texts, *Information Retrieval*, **3**, 7-23, 2000.

[2] Moura E.S., Navarro G., Ziviani N. and Baeza-Yates R., Fast and flexible word searching on compressed Text, *ACM TOIS*, **18(2)**, 113-139, 2000.

[3] Turpin A., Moffat A., Fast file search using text compression, *20th Proc. Australian Computer Science Conference*, 1-8, 1997.