LETTER *Special Section on Discrete Mathematics and Its Applications*

Compact Encoding of the Web Graph Exploiting Various Power Distributions

Yasuhito ASANO[†a)], **Tsuyoshi ITO**[††], ***Nonmembers***, **Hiroshi IMAI**[††], ***Member***, **Masashi TOYODA**[†††], ***Nonmember***, ***and*** **Masaru KITSUREGAWA**[†††], ***Member***

SUMMARY Compact encodings of the web graph are required in order to keep the graph on the main memory and to perform operations on the graph efficiently. In this paper, we propose a new compact encoding of the web graph. It is 10% more compact than Link2 used in the Connectivity Server of Altavista and 20% more compact than the encoding proposed by Guillaume et al. in 2002 and is comparable to it in terms of extraction time.

key words: *Web graph, encoding, power distributions*

1. Introduction

The World Wide Web has evolved at a surprisingly high speed both in its size and in the variety of its content. Since several information retrieval methods utilizing graph algorithms on the web graph have been developed in recent years such as [3]–[5], compact encodings of the web graph enough for the main memory have been desired in order to perform the graph algorithms efficiently. Note that the web graph is a directed graph whose vertices represent web pages and whose edges hyperlinks among them.

Though general compression algorithms such as bzip2 or gzip gives high compression ratio, the extraction of small fragments of the data is too slow to use it for several kinds of operations, such as the depth-first search. Thus, encodings for this purpose have been developed in recent years. The Link Database [6] is a well-known example, and used in the Connectivity Server of Altavista. Link2 is the second version of the Link Database and has achieved compression with 11.03 bits per edge and practical extraction time. Guillaume et al. [2] have proposed another compact encoding achieving 12.3 bits per edge and practical extraction time.

In this paper, we propose a new compact encoding of the web graph utilizing the power distribution of several kinds of elements representing the web graph. It is 10% more compact than Link2 and 20% more compact than the encoding proposed by Guillaume et al., and comparable to the latter in terms of extraction time.

Manuscript received August 25, 2003.
Manuscript revised November 20, 2003.
Final manuscript received January 5, 2004.

[†]The author is with Graduate School of Information Sciences, Tohoku University, Sendai-shi, 980-8579 Japan.
[††]The authors are with Graduate School of Information Science and Technology, the Univeristy of Tokyo, Tokyo, 113-0033 Japan.
[†††]The authors are with Institute of Industrial Science, the Univeristy of Tokyo, Tokyo, 153-8505 Japan.
a) E-mail: asano@nishizeki.ecei.tohoku.ac.jp

2. Encoding Using Power Distribution

Let $\alpha > 1$. A random variable taking positive integer values is said to have the power distribution of the exponent $-\alpha$ when its probability function $f(n)$ satisfies $f(n) = \frac{1}{cn^\alpha}$ $(n \geq 1)$ where $c = \sum_{n=1}^{\infty} \frac{1}{n^\alpha}$. Integers obedient to a power distribution are encoded efficiently in a generalization of the variable-length nybble code, which we call a *variable-length block code*. In the variable-length block code with k-bit blocks, a positive integer n is first represented in the base 2^k. Each digit in this base 2^k number is represented in k bits. This k-bit sequence is called a *block*. A bit 1 is appended for each block except for the last block, for which a bit 0 is appended. Thus, the variable-length block code with k-bit blocks encodes a positive integer n in $((k+1)/k)\log n$ bit asymptotically.

If X has the power distribution of the exponent $-\alpha$, the following fact is obtained from Kraft's inequality about instantaneous codes: the most efficient instantaneous code in terms of average codeword length is the variable-length block code with $1/(\alpha - 1)$-bit blocks.

3. Proposed Encoding

Our encoding adopts the following ideas used in Link2: pages are indexed in the lexicographical order of URLs, and the graph is represented by the list of the adjacency lists of all the vertices, where each adjacency list is sorted in ascending order. Let the adjacency list for a vertex v be $(a_1, a_2, \cdots, a_d)$. Our encoding and Link2 encode the following list $(v - a_1, a_2 - a_1, a_3 - a_2, \cdots, a_d - a_{d-1})$, instead of the original adjacency list, in order to keep the absolute value of each element small by utilizing the locality of hyperlinks in the web. The first element of this list $v - a_1$ is called *initial distance*, and the other elements are called *increments*. Link2 encodes both the initial distance and the increments by using the variable-length nybble code. On the other hand, our encoding encodes them distinctively by using the variable-length block code with respective block sizes, since the initial distance and the increments have different distributions as described below. Moreover, in our encoding a sublist of consecutive 1s in this list are compressed using the *run-length encoding* (i.e. instead of the sublist, only the length of the sublist is encoded), since the length will be long when a directory index page has links

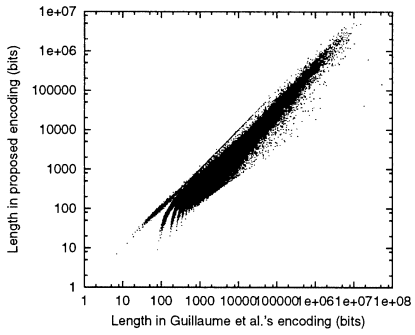

Fig. 1 The comparison of the compactness of Guillaume et al.'s encoding with Huffman codes and that of the proposed encoding. Both axes are in logarithmic scale.

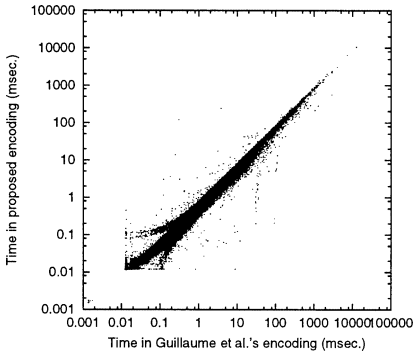

Fig. 2 The comparison of the time taken by the depth-first search of the graphs encoded in Guillaume et al.'s method with Huffman codes and that in the proposed method. Both axes are in logarithmic scale.

to all the files in that directory and such a situation occurs frequently in the actual web.

To observe how the initial distance, increments, and the run-lengths in our encoding are distributed, we have analyzed Toyoda and Kitsuregawa's web graph [7] of pages in .jp domain collected in 2002. This graph consists of 60,336,969 pages in 297,949 servers and 221,085,322 edges in total. For every adjacency list, the initial distance (i.e. $v - a_1$ in the list used in our encoding) has the power distribution with the exponent of about -1.17, and the increments (e.g. $a_2 - a_1$ or $a_3 - a_2$) have the exponent about -1.33, and the run-lengths have the exponent about -2.67. Thus, the initial distance is represented in the variable-length block codes with 6-bit ($1/(1.17 - 1)$, rounded off) blocks by using the fact described in the previous section. Similarly, the increments and the run-lengths are represented in the variable-length block codes with 3-bit and 1-bit blocks, respectively. Refer [1], the preliminary version of this paper, for the detail.

In addition to the encoding of the adjacency list, we use a balanced binary tree T with each leaf representing one adjacency list to locate the adjacency list of a given vertex in $O(\log n)$ time where n is the number of the vertices of the graph. Each subtree T' of this binary tree is represented by the concatenation of the encodings of the left child tree of T' and of the right child tree of T', preceded by the length of the first part represented in the variable-length block code with 2-bit blocks.

4. Experiments, Results and Discussions

Figure 1 shows the comparison of compression ratio of our encoding with Guillaume et al.'s encoding. Our encoding produced 9.7 bits per edge on the average while Guillaume et al.'s produced 27.0 bits per edge in this result.

In the citation [2], their encoding with Huffman codes produced 12.3 bits per edge. Compared to this figure, our method gives 20% less number of bits per edge than Guillaume et al.'s. According to the citation [6], Link2 produced 11.03 bits per edge on the average when used to encode their dataset with 61 million vertices and 1 billion edges. Compared to this, our encoding produces 10% shorter encoding.

From these observation, our encoding successfully utilizes a wide range of locality and the distributions of various variables (including initial distances and increments) in the web graph.

Figure 2 shows that Guillaume et al.'s and our method are comparable in extraction time. Guillaume et al.'s method took 5.1 μsec. per edge and ours took 3.5 μsec. per edge.

Benchmark program was written in C++, compiled with GNU C++ 3.0.1 and executed on Solaris 2.6 with UltraSPARC-II 360MHz CPU and 1GB memory.

5. Conclusion

This paper has proposed a new efficient encoding of the web graph using the distinct power distributions of the initial distances and the increments. Note that Link3 [6], a newer version of Link2, achieves about 5.6 bits per edge on the average, though its extraction time is over four times longer than the extraction time of Link2. Our method has several parameters and it hopefully has application to other kinds of graphs than the web graphs.

References

[1] Y. Asano, T. Ito, H. Imai, M. Toyoda, and M. Kitsuregawa, "Compact encoding of the Web graph exploiting various power laws—Statistical reason behind link database," Proc. 4th International Conf. on Web-Age Information Management, LNCS 2762, pp.37–46, 2003.

[2] J.L. Guillaume, M. Latapy, and L. Viennot, "Efficient and simple encodings for the Web graph," Proc. 3rd International Conf. on Web-Age Information Management, LNCS 2419, pp.328–337, 2002.

[3] J.M. Kleinberg, "Authoritative sources in a hyperlinked environment," J. ACM, vol.46, no.5, pp.604–632, 1999.

[4] S.R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins, "Trawling the Web for emerging cybercommunities," Comput. Netw., vol.31, no.11–16, pp.1481–1493, 1999.

[5] L. Page, S. Brin, R. Motwani, and T. Winograd, "The PageRank citation ranking: Bring order to the Web," Technical Report, Stanford University, 1998.

[6] K. Randall, R. Stata, R. Wickremesinghe, and J.L. Wiener, "The link database: Fast access to graphs of the Web," Research Report 175, Compaq Systems Research Center, 2001.

[7] M. Toyoda and M. Kitsuregawa, "Observing evolution of Web communities," Poster Proc. 11th International World Wide Web Conference, 2002.