Compact Encoding of the Web Graph Exploiting Various Power Laws

Statistical Reason Behind Link Database

Yasuhito Asano[1], Tsuyoshi Ito[2], Hiroshi Imai[2],
Masashi Toyoda[3], and Masaru Kitsuregawa[3]

[1] Department of System Information Sciences, Tohoku University,
05 Aoba, Aramaki, Aoba-ku, Sendai, 980-8579 Japan
[2] Department of Computer Science, The University of Tokyo,
7-3-1 Hongo, Bunkyo-ku, Tokyo, 113-0033 Japan
[3] Institute of Industrial Science, The University of Tokyo,
4-6-1 Komaba, Meguro-ku, Tokyo, 153-8505 Japan

Abstract. Compact encodings of the web graph are required in order to keep the graph on main memory and to perform operations on the graph efficiently. Link2, the second version of the Link Database by Randall et al., which is part of the Connectivity Server, represented the adjacency list of each vertex by the variable-length nybble codes of delta values. In this paper, the fact is shown that certain variables related to the web graph have power distributions, and the reason is explained why using variable-length nybble codes in Link2 led to a compact representation of the graph from the statistical viewpoint on the basis of the relationship between power distributions and generalization of the variable-length nybble code. Besides, another encoding of the web graph based on these fact and relationship is proposed, and it is compared with Link2 and the encoding proposed by Guillaume et al. in 2002. Though our encoding is slower than Link2, it is 10% more compact than Link2. And our encoding is 20% more compact than the encoding proposed by Guillaume et al. and is comparable to it in terms of extraction time.

1 Introduction

The world wide web has evolved at a surprisingly high speed both in its size and in the variety of its content. The analyses of the structure of the web have become more and more important for the information retrieval from the web.

While the text and markups in each page contain vast amount of information, the structure of the hyperlinks among pages is often used extensively, both by itself [3, 4] and in combination with the content of pages [5]. One of the reasons the link structure is useful for the information retrieval is that the meaning of links is usually independent of language: a link from a page u to a page v means the author of u thinks v has relevant or valuable information to u.

The abstraction of how pages are linked together is the web graph. The web graph is a directed graph whose vertices represent web pages and whose edges

hyperlinks among them. Among the studies on the structure of the web graph are [1, 3–5]. The strong component decomposition is one method to describe the structure of the graph, and it requires the depth-first search of the graph. Other methods also require the operations on the graph.

When designing an algorithm to treat a large data like the web graph, one must be careful not only with its time complexity but also its space consumption. Once it requires more than the amount of the main memory available on the system, it usually requires much more time than when all the necessary data is on the main memory, sometimes to the extent that the algorithm itself becomes useless.

It is a good idea to encode the web graph in a compact format to keep it on the main memory. Though general compression algorithms such as bzip2 or gzip gives high compression ratio, they have one significant drawback: the extraction of small fragments of the data is slow. Usually many small fragments of the graph are needed for the operation on the graph such as the depth-first search. It is better to use the encoding methods which support the extraction of fragments as needed.

The web graph has several properties which distinguish itself from other graphs. One of them is "power law" about the distribution of the out-degrees and the in-degrees of vertices. By using these properties appropriately, a compact encoding method is obtained.

Let us review two previous studies on efficient encodings of the web graph: Link2 described in the citation [6] and the encoding proposed by Guillaume et al. [2]. They had a common point that pages were numbered sequencially in the lexicographical order of their URLs, and that the graph was represented by the list of the adjacency lists of all the vertices. They were different in the representation of the adjacency list of a vertex.

Now their difference will be described.

The Link Database [6] is part of the Connectivity Server which provides access to a large web graph, and Link2 is the second version of the Link Database. In Link2, each adjacency list was sorted in ascending order and represented by the list of the delta values of its elements. The delta value of the first element of the list is its difference from the source of the link, and the delta value of each of the other elements is the difference between its previous element and itself. They observed that the delta values tended to be close to zero, resulting from a kind of locality of the web graph that the destinations of the links originating at the same page are often near from each other. According to this observation, they used variable-length nybble codes to represent the delta values in their encoding of the web graph. By combining this representation with other techniques, Link2 encoded a graph in 11.03 bits per edge on average. The latest version Link3 used some other techniques to compress the whole graph to achieve less than 6 bits per edge. However, the compression techniques used in Link3 took longer time to decompress.

Guillaume et al. [2] proposed another compact encoding of the web graph. In their encoding, each element of an adjacency list was stored as a signed integers

representing the length of the link, which is defined as the difference between the indices of the source and the destination pages of the link. The locality utilized here is that many links are short, which is more restricted than that utilized in Link2. They observed the length of a link had a power distribution. However, in their encoding, the length of a link was represented in either a Huffman code, 16-bit integer or 32-bit integer according to its absolute value, and the property of the power distribution was not utilized very much.

Our encoding utilizes the locality which is the same as that utilized by Link2, and the exact relationship between the distribution of the delta values of the adjacency lists and the optimal encoding of them.

The rest of this paper is organized as follows. Section 2 explains the fact that integers which have a power distribution are encoded most efficiently in a generalization of the variable-length nybble code. In section 3, the characteristics of web graph are described which were obtained from observation of an actual web graph, and the reason Link2 is efficient is explained. Section 4 proposes another encoding of the web graph according to our observation in the previous section. Sections 5 gives the details about and the results of the experiments to show the usefulness of the proposed encoding. Section 6 concludes this study.

2 Power Distribution and Variable-length Block Codes

Let $\alpha > 1$. A random variable which takes positive integer values is said to have the power distribution of the exponent $-\alpha$ when its probability function $f(n)$ satisfies

$$f(n) = \frac{1}{cn^{\alpha}} \quad (n \geq 1) \quad \text{where } c = \sum_{n=1}^{\infty} \frac{1}{n^{\alpha}}.$$

Integers which have 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.

For example, an integer 92 is represented as 1011100 in binary, and it is represented as 01$\underline{1}$01$\underline{1}$11$\underline{1}$00$\underline{0}$ in the variable-length block code with 2-bit blocks.

The variable-length block code with 3-bit blocks is called the *variable-length nybble code*. A "nybble" means a 4-bit-long sequence. The name of the variable-length nybble code comes from the fact that each block including the appended bit is four bits long.

A positive integer n is represented in asymptotically $\big((k+1)/k\big)\log n$ bits long in the variable-length block code with k-bit blocks.

The following fact is obtained from Kraft's inequality about instantaneous codes: when a random variable X has a probability function $f(X)$, the instantaneous code which gives the minimum average codeword length when used to represent X is the code in which the codeword for n is $-\log_2 f(n)$ bit long.

Therefore, if X has the power distribution of the exponent $-\alpha$, the most efficient instantaneous code in terms of average codeword length is the variable-length block code with $1/(\alpha - 1)$-bit blocks.

Note that Huffman encoding is the shortest instantaneous code only when we do not count the space needed to store the code table.

3 Observation of Actual Web Graph and Reason Behind Link2

We explained how delta values were used to encode the adjacency list of a vertex in Link2. Note that there are two different kinds of delta values. The delta value of the first element in a list is the difference between the source and the destination of the link, and the delta values of the rest are the differences between their previous elements and themselves. We call the delta value of the first element *initial distance*, and the delta values of the rest *increments*. Because initial distance and increment are different things, they may well have different distributions, hence different optimal representations.

To observe how the initial distances and increments in our encoding are distributed when used to represent an actual web graph, we analyzed Toyoda and Kitsuregawa's web graph [7] of pages in .jp domain collected in 2002, after removing pages whose URLs do not look like HTML files. This graph consists of 60,336,969 pages of 297,949 servers.

We extracted the subgraph representing each server from this graph, and analyzed the distributions of the absolute value of initial distance and of the value of increment independently. These subgraphs have 221,085,322 edges in total.

Figure 1 shows that the absolute value of initial distance has the power distribution with the exponent of about $-7/6$, and the increment has the power distribution with the exponent of about $-4/3$.

These facts mean initial distances and increments will be represented efficiently in the variable-length block codes with 6-bit and 3-bit blocks, respectively.

Because most delta values are actually increments, it is a good approximation to represent delta values altogether in the variable-length block codes with 3-bit blocks, a.k.a. the variable-length nybble codes. This is why the encoding used by Link2 is compact.

4 Proposed Encoding

As we saw in the previous section, the two kinds of delta values, namely initial distances and increments, have the power distributions of different exponents. By utilizing this fact and the distributions of other variables related to the web graph, a new encoding of the web graph is obtained.

A high-level description of our encoding is the same as that of Link2: pages are represented by sequence numbers in the lexicographical order of URLs, and

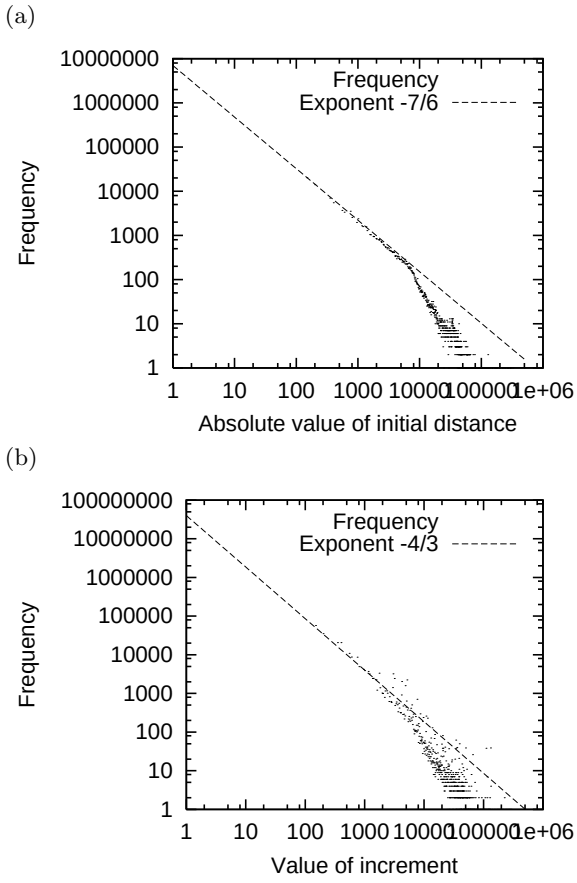

Fig. 1. The distributions of (a) initial distances and (b) increments. Both axes are in logarithmic scale.

the graph is represented by the list of the adjacency lists of all the vertices, where each adjacency list is sorted in ascending order and each element of an adjacency list is represented by its delta value. The difference lies in the way each delta value is represented.

According to the observation in the previous section, we propose the following encoding of the web graph.

Encoding adjacency list of one vertex. Suppose a vertex v has out-degree d. Then the adjacency list of v has one initial distance and the list of $(d-1)$ increments. Consecutive 1s in the list of increments are compressed using the run-length encoding.

We treat 1s in the list of increments specially because the increments of 1 appear frequently because they appear when a directory index page has links to all the files in that directory.

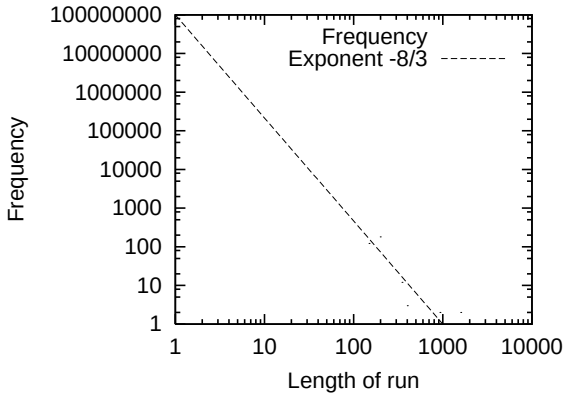

Fig. 2. The distributions of the length of the runs consisting of the increments of 1. Both axes are in logarithmic scale.

From the actual data, the lengths of these runs have the power distribution with the exponent of about $-8/3$ as shown in Figure 2. Because we now stick to instantaneous codes for simple and fast decoding, the best we can do is to represent them in the variable-length block code with 1-bit blocks.

As a result, this adjacency list is represented by its initial distance, followed by mixture of actual increments and run-lengths of increments of 1, followed by the end-of-list mark. The initial distance, the increments and the run-length are represented in the variable-length block codes with 6-bit, 3-bit and 1-bit blocks, respectively.

Encoding a whole graph. The number of vertices is represented in the variable-length block code with 3-bit blocks, and the adjacency lists of the vertices are represented using the encoding described above.

To make random access possible, a balanced binary tree T with each leaf representing one adjacency list is used. 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. The whole graph is represented by the representation of T. This way, the adjacency list of a given vertex can be located in $O(\log n)$ time where n is the number of the vertices of the graph.

Note that our encoding have four parameters: the lengths of blocks to represent initial distances, increments, run-lengths of the increments of 1, and the lengths of the representations of left-children of inner nodes of the binary tree T. If another, slightly different graph has to be encoded, one can tune these parameters according to the actual graph.

5 Experiments, Results and Discussions

Experiments were performed to see how compact the proposed encoding method is and how efficient its decoding is.

The graphs used in them are Toyoda and Kitsuregawa's web graph divided into each server, as described in section 3.

5.1 Compression Ratio

The proposed encoding algorithm was implemented and the web graphs were encoded. Guillaume et al.'s encoding with Huffman codes was also implemented and the results were compared.

Figure 3 shows the result of the comparison. Our encoding produced 9.7 bits per edge on average while Guillaume et al.'s produced 27.0 bits per edge. These numbers have to be treated with care. In our method, the length of blocks of variable-length block codes of integers were adjusted to produce the best result for our dataset, while Guillaume et al.'s was used as it was with Huffman code table adjusted to be optimal.

It may be fair to compare the number of bits our method produced per edge for our dataset with the number of bits their method produced per edge for their dataset. 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 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, the proposed encoding successfully utilized a wide range of locality the web graph has and the distributions of various variables related to the web graph including initial distances and increments, and these

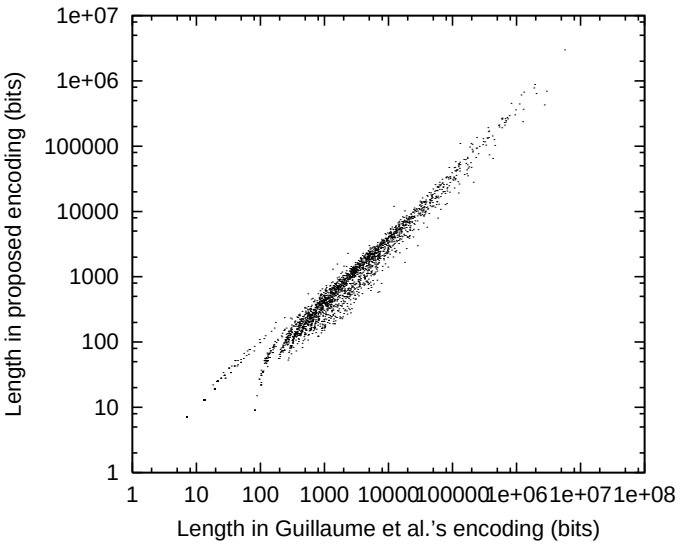

Fig. 3. The comparison of the compactness of Guillaume et al.'s encoding with Huffman codes and that of the proposed encoding. Both axes are in logarithmic scale.

facts resulted in a better compression ratio than Guillaume et al.'s method and Link2.

5.2 Extraction Time

A corresponding decoding algorithm for the proposed encoding method was implemented and the time taken by the depth-first search on the encoded web graphs was measured. The comparison with the case using Guillaume et al.'s encoding with Huffman codes was also performed.

For each graph, depth-first search starting from each vertex was performed to visit every vertex at least once and follow every edge exactly once.

Benchmark program was written in C++, compiled with GNU C++ Compiler 3.0.1 and executed on Solaris 2.6 on Sun Ultra 60 with UltraSPARC-II 360Hz CPU and 1GB memory.

Figure 4 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.

This means that the use of variable-length block codes did not impact time needed to decode adjacency lists of the vertices of the graph compared to Guillaume et al.'s method. It seems this is because most of the decoding time is taken to find the adjacency list of an appropriate vertex in the balanced binary tree.

It is difficult to compare our encoding to Link2 in terms of the extraction time because we have not executed the algorithm of Link2 on our environment. However, at our best guess, Link2 is faster than our encoding because Link2 is

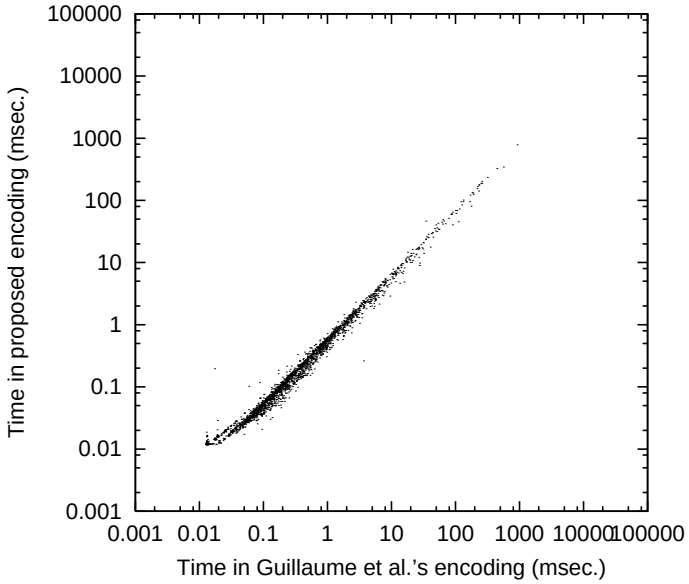

Fig. 4. The comparison of the time taken by the depth-first search of the graphs encoded in Guillaume et al.'s method with Huffman codes and that in the proposed method. Both axes are in logarithmic scale.

fixed for the variable-length nybble code. The variable-length nybble code can be decoded faster than general variable-length block codes because in the variable-length nybble code, it is sufficient to simply split one byte into two nybbles to extract blocks, while in general variable-length block codes, operations on bit by bit are needed.

6 Conclusion

It was clarified that the compact encoding by Link2 came from the fact that the delta values of the adjacency lists had the power distribution with the exponent of about $-4/3$ by observing the actual data. Furthermore, it was found that the initial distances and the increments had the power distributions of the different exponents. By using the distributions of initial distances, increments and other variables, we obtained another compact encoding of the web graph. Especially, the connection between power distributions and variable encodings has turned to be useful to encode the web graph efficiently.

The qualitative comparison of our method with Link2 and Link3 in terms of compression ratio and extraction time is yet to be performed.

Widely-known "power law" about the distribution of the out-degrees and the in-degrees of vertices is not used in our method. Using it and the distribution of other variables may lead to more efficient encoding method.

Our method has several parameters and it has hopefully application to other kinds of graphs than the web graphs. The optimal parameters of the variable encodings used in the method may be different for different kinds of graphs. Currently, to adopt our method to other kinds of graphs, it is required to tune the parameters by observing the distributions of initial distances and increments of the actual graph. However, our method have few parameters and the tuning will be easier than encodings with more parameters. The general discussion for usefulness and optimal parameters of our method to other kinds of graphs needs more experiments using different graphs.

References

1. A. Z. Broder, S. R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins and J. Wiener. Graph structure in the web. In *Proceedings of the 9th International World Wide Web Conference*, pp. 309–320, 2000.
2. J.-L. Guillaume, M. Latapy and L. Viennot. Efficient and Simple Encodings for the Web Graph. In *Proc. the 3rd International Conference on Web-Age Information Management, LNCS 2419*, pp. 328–337, 2002.
3. J. M. Kleinberg. Authoritative Sources in a Hyperlinked Environment. *Journal of ACM*, **46**(5):604–632, 1999.
4. S. R. Kumar, P. Raghavan, S. Rajagopalan and A. Tomkins. Trawling the Web for Emerging Cybercommunities. *Computer Networks*, **31**(11–16):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, Palo Alto, CA, 2001.
7. M. Toyoda and M. Kitsuregawa. Observing evolution of Web communities. In *Poster Proceedings of the 11th International World Wide Web Conference (WWW2002)*, 2002.