Efficient Compression of Web Graphs

Yasuhito Asano[1], Yuya Miyawaki[2], and Takao Nishizeki[2]

[1] Graduate School Informatics, Kyoto University, Yoshidahonmachi, Sakyo-ku, Kyoto, 606-8051, Japan
asano@i.kyoto-u.ac.jp

[2] Graduate School of Information Sciences, Tohoku University, Aza-Aoba 6-6-05, Aramaki, Aoba-ku, Sendai, 980-8579, Japan
miyawaki@nishizeki.ecei.tohoku.ac.jp, nishi@ecei.tohoku.ac.jp

Abstract. Several methods have been proposed for compressing the linkage data of a Web graph. Among them, the method proposed by Boldi and Vigna is known as the most efficient one. In the paper, we propose a new method to compress a Web graph. Our method is more efficient than theirs with respect to the size of the compressed data. For example, our method needs only 1.99 bits per link to compress a Web graph containing 3,216,152 links connecting 325,557 pages, while the method of Boldi and Vigna needs 2.84 bits per link to compress the same Web graph.

1 Introduction

A *Web graph* is a directed graph, whose vertex set consists of Web pages, and whose edge set consists of hyperlinks connecting these pages. A Web graph plays a central role in data mining on the Web. For example, search engines, including Google and Yahoo!, score Web pages by analyzing a Web graph containing billions of links [3],[6],[14]. Some other algorithms cluster Web pages by finding dense subgraphs in a Web graph [2],[12],[16],[23]. Such a Web graph has too many pages and links to be stored in the main memory of a computer. Thus, compressing a Web graph is indispensable for many applications, including search engines and clustering algorithms.

Several methods have been proposed for compressing a Web graph [1],[4],[5], [7],[9],[13],[19],[20],[22]. The previously known most efficient method, proposed by Boldi and Vigna [7], compresses the link data of a Web graph as little as about three bits per link. The so-called "localities of a Web graph" are utilized by most of the existing methods including that of Boldi and Vigna. The localities stem mainly from the fact that there are much more "intra-host links" than "inter-host links;" an *intra-host link* is a link between two pages in the same host computer, while an *inter-host link* is a link between two pages in distinct hosts. However, the fact has not been fully utilized by previous methods.

In the paper, we first propose a new method to compress a Web graph. Our method fully utilizes the localities of a Web graph together with the fact that there are much more intra-host links than inter-host links. The main idea of our method is twofold: one is to deal with intra-host links separately for each host; the other is to use six types of "blocks" to cover all 1's in an adjacency matrix representing intra-host links for a host; each block consists of consecutive 1's in the matrix. (See Figure 1 in Section 4.2) Each type of blocks corresponds to some locality of intra-host links. The matrix can be represented by a sequence of blocks, each of which is represented by the "type," "beginning element" and "dimension" of a block. Thus the data of intra-host links can be efficiently compressed. We regard inter-host links as intra-host links of special type, and compress intra-host links and inter-host links all together. We then compare our method with that of Boldi and Vigna with respect to the size of compressed data and the retrieval speed. The size of data compressed by our method is smaller than 79% of that by their method. For example, our method compresses a Web graph containing 3,216,152 links and 325,557 pages as little as 1.99 bits per link, while their method compresses the same Web graph as little as 2.84 bits per link. Our method retrieves all the links of an original Web graph faster than their method, although our method could be slower than their method when retrieving only the links emanating from a specified page.

2 Localities of a Web Graph

Most of the existing methods to compress a Web graph pagenate all pages with integers in lexicographic order of their URIs, and hence two pages have close page numbers if their URIs share a long common prefix. We call the page number of a page the *index* of the page, and often call a page with index i simply *page* i. The following three facts, called the *localities of a Web graph*, hold true.

Locality (A): Two pages connected by a link often have close indices, that is, the difference of their indices are small.

Locality (B): Pages linked from the same page often have close indices.

Locality (C): Pages with close indices often have "similar" links. More precisely, if page i has a link to page j, then page h such that $|i - h|$ is small often has a link to the same page j.

These three kinds of localities stem from the following two facts on the Web: (1) all the pages in the same host have close indices; and (2) there are much more intra-host links than inter-host links in the Web.

The URIs of pages in the same host share a long common prefix, and hence these pages have close indices. An intra-host link and an inter-host link are formally defined as follows.

Definition 1. *A link between two pages is called an intra-host link if the pages belong to the same host; otherwise, the link is called an inter-host link.*

Intra-host links occupy more than 89% of all the links for the three data sets used in Section 5.

3 Previously Known Methods

In this section, we explain the ideas of two previous methods to compress a Web graph, one by Boldi and Vigna [7], and the other by Claude and Navarro [9].

We first present several definitions. If a link emanates from page p and enters page q, then the pages p and q are called the *source* and *destination* of the link, respectively.

Definition 2. *The destination list L_p of page p contains all indices of pages linked from page p, sorted in increasing order. The adjacency list of a Web graph is the set of all destination lists of pages in the Web graph.*

Methods of compressing a Web graph encode the adjacency list to a binary file, called the *compressed data*.

Boldi and Vigna's method [7] utilizes Localities (B) and (C), and has two positive integer parameters W and α. The parameter W is called a *refer range*. Their method represents the destination list L_p of page p by referring the destination list of one of the W pages preceding page p. They choose one of the W pages, say page q, such that L_q is most similar to L_p, that is, $|L_q \cap L_p|$ is maximum among all q, $p - W \leq q \leq p - 1$. They say that page p *refers* page q and page q is *referred* by page p. The *copy list* is a binary string of $|L_q|$ bits; its i-th bit, $1 \leq i \leq |L_q|$, is set to 1 if the i-th element of L_q is contained in L_p; otherwise, it is set to 0. Their method efficiently represents the subset $L_q \cap L_p$ of L_p by the copy list, while the remaining subset $L_p \setminus L_q$ of L_p is represented by a "differential list," called the *remaining list*; the *differential list* of k integers $i_1 < i_2 < \cdots < i_k$ is defined as a list of k integers $i_1, i_2 - i_1, i_3 - i_2, \cdots, i_k - i_{k-1}$. Their method uses several other techniques to efficiently compress the copy lists and the remaining lists. In particular, they allow that page q refers another page r, page r refers another page s, and so on. A set of pages $p_1, p_2, \cdots, p_k$ for some integer k is called a *copy chain* if page p_i refers page p_{i-1} for each i, $k \geq i \geq 2$, and p_1 does not refer any page. The integer k is called the *length of this copy chain*. The maximum length of a copy chain is bounded above by the parameter α. If α becomes larger, then the size of compressed data tends to become smaller but the retrieval speed becomes slower.

Claude and Navarro's method [9] uses a uniform technique called Re-Pair [17] to compress a Web graph comparably as small as that of Boldi and Vigna, while the retrieval time of links emanating from a specified page is several times faster than that of Boldi and Vigna.

4 Our Method

4.1 Classification of Links

We first partition the set of all links of a Web graph into several subsets, each consisting of all the links whose sources belong to the same host.

For example, consider a Web graph whose pages have URIs and indices written in Table 1, and whose adjacency list is represented in Table 2. The "URI"

Table 1. An example of URIs and indices of pages

URI	Original Index
abc.com/index.html	0
abc.com/link.html	1
abc.com/t0.html	2
abc.com/t1.html	3
ace.com/index.html	4
...	...
ace.com/pic300.html	314
add.com/a1.html	315
add.com/index.html	316
add.com/adv.html	317

Table 2. An example of an adjacency list

Source	Destinations
0	1, 2, 315
1	0, 2, 3, 315, 316
2	3
3	1
4	314
...	...
314	4
315	0, 316, 317
316	1, 317
317	316

column of Table 1 represents the URIs of pages, and the "Original Index" column represents the indices of pages. Let H_0 be a host whose name is `abc.com`, and let H_2 be a host whose name is `add.com`. Pages 0, 1, 2 and 3 belong to H_0, and pages 315, 316 and 317 belong to H_2. The "Destinations" column of each row in Table 2 represents the destination list of the page written in the "Source" column of the same row.

We then partition the set of all links whose sources belong to each host into two subsets: the set of all intra-host links and the set of all inter-host links. Table 3 depicts intra-host links and inter-host links for H_2. The destinations of intra-host links are written in the "Intra" column, and those of inter-host links are in the "Inter" column.

Table 3. Intra-host links and inter-host links for H_2

Source	Intra	Inter
315	316, 317	0
316	317	1
317	316	-

Table 4. Intra-destination lists for H_2

Source	Destinations
0	1, 2
1	2
2	1

Finally, for each host, we assign an integer, called a "local index," to each page in the host, and represent every intra-host link by a pair of local indices. The *original index* of a page is the index in lexicographic order of its URI, as described in Section 2. Tables 1–3 above use original indices. The *local index* of a page with original index p is defined to be the difference between p and the smallest original index of pages in the host containing page p. For example, the smallest original index in H_2 is 315, and hence the local index of the page with original index 317 is 2. Thus, a local index is much smaller than an original index. Table 4 represents the local indices of pages belonging to H_2. The "Source" column of each row represents the local index of a source of intra-host links, and the "Destinations" column represents the local indices of the destinations.

Comparing Tables 3 and 4, one can immediately realize the advantage of local indices in compressing intra-destination lists. From now on we call a page with local index i simply page i. The *intra-destination list* of page i is a list of all local indices of the destinations of intra-host links whose sources are page i.

4.2 Compression of Intra-host Links

Our method compresses the data of intra-host links in a Web graph, separately for each host, by extending a technique used for the compression of bi-level images [15].

Only for the sake of explanation, we employ an adjacency matrix A of intra-host links for a host. It should be noted that we use an adjacency list, in place of an adjacency matrix, for implementing our method. Let n be the number of pages in a host. Then the adjacency matrix A of the host is an $n \times n$ matrix such that, for $0 \le i,\ j \le n-1$, an (i,j)-element $A_{i,j} = 1$ if page i has a link to page j, and otherwise, $A_{i,j} = 0$. We say that an element $A_{i,j}$ is a *1-element* if $A_{i,j} = 1$; otherwise, it is called a *0-element*.

Our method finds the following six types of blocks in A, each consisting of 1-elements consecutive in A in some sense.

Definition 3. *(1) A* **singleton block** *consists of an "isolated" 1-element. (2) A* **horizontal block** *consists of two or more horizontally consecutive 1-elements. (3) A* **vertical block** *consists of two or more vertically consecutive 1-elements. (4) An* **L-shaped block** *is a union of a horizontal block and a vertical block sharing the upper leftmost 1-element. (5)A* **rectangular block** *is a submatrix of A such that all the elements in the submatrix are 1's and the submatrix has more than one consecutive rows and more than one consecutive columns. (6)*

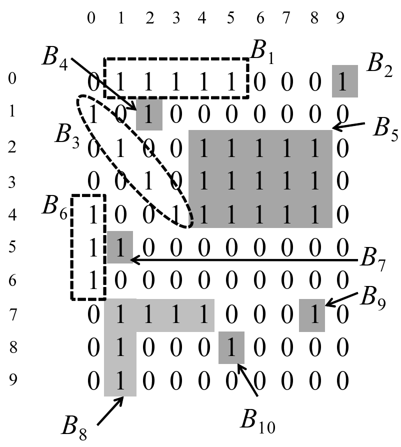

Fig. 1. Adjacency matrix A and blocks $B_1, B_2, \cdots, B_{10}$ in A

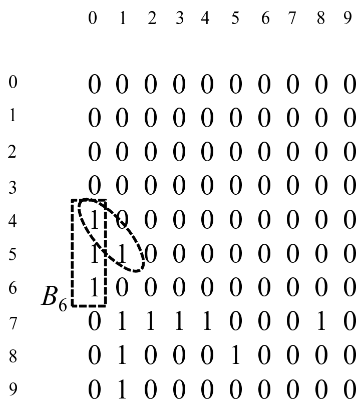

Fig. 2. Matrix A after B_5 is found

A **diagonal block** *consists of two or more 1-elements downward diagonally consecutive from upper left to lower right.*

In Figure 1, each block is either shaded or surrounded by dotted lines. For example, a horizontal block B_1 consists of five 1-elements $A_{0,1}, A_{0,2}, \cdots, A_{0,5}$. Block B_3 is diagonal, B_5 is rectangluar, B_6 is vertical, and B_8 is L-shaped. Blocks B_2, B_4, B_7, B_9, and B_{10} are singletons.

We represent a block by the "beginning element," the "type," and the "dimension" of the block. We denote by $type(B)$ the type of a block B. For example, $type(B_1) = Horizontal$. The upper leftmost element of a block B is called the *beginning element* of B, and is denoted by $b(B)$. Let $br(B)$ be the row number of $b(B)$, and let $bc(B)$ be the column number, then $b(B) = A_{br(B),bc(B)}$. Let $er(B)$ be the row number of the lowest element in B, and let $ec(B)$ be the column number of the rightmost element in B. We call $A_{er(B),ec(B)}$ the *ending element* of a block B unless B is L-shaped. An L-shaped block has two ending elements, the rightmost one $A_{br(B),ec(B)}$ and the lowermost one $A_{er(B),bc(B)}$. The *dimension* $d(B)$ of an L-shaped or rectangular block B is definded to be an ordered pair $(er(B) - br(B) + 1, ec(B) - bc(B) + 1)$. The *dimension* $d(B)$ of block B of the other types is defined to be the number of elements in B. For the example depicted in Figure 1, the beginning element of B_1 is $A_{0,1}$. The dimension of B_1 is 5, while the dimensions of B_5 and B_8 are $(3, 5)$ and $(3, 4)$, respectively. We can represent a block B in A by a quadraplet $(br(B), bc(B), type(B), d(B))$, called the *signature* $sig(B)$ of B. For example, $sig(B_5) = (2, 4, Rectangular, (3, 5))$. A singleton block can be represented without the dimension, because the dimension of every singleton block is 1. For example, $sig(B_2) = (0, 9, Singleton)$. The *size* of a block B is the number of 1-elements in B, and is denoted by $size(B)$.

The five types of blocks, other than a singleton block, corresponds to localities of intra-host links; some are variants of the three localities mentioned in Section 2, and the remainder are newly found in the paper. We call pages with consecutive local indices simply *consecutive pages*. (i) A horizontal block corresponds to a variant of Locality (B); a page often has intra-host links to consecutive pages. (ii) A vertical block corresponds to a variant of Locality (C); in a host, consecutive pages often have an intra-host link to the same page. (Some of the previously known methods explicitly used (i) and (ii) [1],[4],[7],[13],[19],[20],[22].) (iii) A rectangular block corresponds to a newly found variant of Localities (B) and (C); in a host, several consecutive pages often have intra-host links to common consecutive pages. (iv) An L-shaped block also corresponds to a newly found variant of Localities (B) and (C); a page, say page p, often has intra-host links to several consecutive pages $q, q+1, \cdots, q+h$, and several consecutive pages $p+1, p+2, \cdots, p+k$ often have intra-host links to page q. For example, a site has such a locality if the site consists of ten pages `page0.html`, `page1.html`, `page2.html`, $\cdots$, `page9.html`, the top page `page0.html` has intra-host links to the remaining nine pages, and they have a link to `page0.html` for returning to the top page. There are a number of sites similar to this example. (v) A diagonal block corresponds to another newly found locality in intra-host links; if page p has an intra-host link to page q, then some consecutive pages $p + i$, $1 \leq i \leq k$, often have intra-host links to page $q+i$. For example, a site has such a locality if a site has ten pages `page0.html`, `page1.html`, `page2.html`, $\cdots$, `page9.html` and each page except the last page has an intra-host link to the next page. There are a number of sites similar to this example. We do not adopt an upward diagonal block from lower left to upper right, because there are very few such blocks.

We now explain how to find a set of blocks which cover all 1-elements in an adjacency matrix A. We first find the leftmost 1-element $A_{i,j}$ in the uppermost row containing 1-elements, then find a block B containing $A_{i,j}$ as its beginning element, then output the signature $sig(B)$, and finally replace all the 1-elements in B by 0's. We repeat the operation above from top to bottom and from left to right until there is no 1-element in A.

If there are two or more blocks whose beginning elements are $A_{i,j}$, then we choose B as follows. (One can easily find B using an adjacency list in place of an adjacency matrix.)

1. If there is a rectangular block whose beginning element is $A_{i,j}$, then we choose the *largest one*, that is, the block of the largest size, among these rectangular blocks.
2. Otherwise and if there is an L-shaped block whose beginning element is $A_{i,j}$, then we choose the largest one among them.
3. Otherwise, let B_h, B_v and B_d be the largest blocks with beginning element $A_{i,j}$ among the horizontal, vertical and diagonal blocks, respectively, and we choose the largest one among B_h, B_v and B_d. If there are no such blocks B_h, B_v and B_d, then we choose the singleton block B consisting only of the beginning element $A_{i,j}$.

For example, consider the adjacency matrix A in Figure 1, for which our algorithm finds $B_1, B_2, \cdots, B_{10}$ in this order. Figure 2 depicts matrix A just after the first five blocks $B_1, B_2, \cdots, B_5$ in Figure 1 are found and all the 1-elements in these blocks are replaced by 0's. $A_{4,0}$ is now the leftmost 1-element in the uppermost row containing 1-elements, and there are two blocks having $A_{4,0}$ as the beginning element: a vertical block B_6 and a diagonal block consisting of two 1-elements $A_{4,0}$ and $A_{5,1}$. Since the former is larger than the latter, we choose B_6 as the block containing $A_{4,0}$.

After all the blocks B covering all 1-elements in the adjacency matrix A are found, we encode $sig(B) = (br(B), bc(B), type(B), d(B))$ of all the blocks B in A. In order to encode an integer $br(B)$ to a binary string, we choose the best code among the following three kinds of variable-length codes: γ-code [11], δ-code [11] and ζ-code [8]. The best one depends on the distribution of the values of row numbers $br(B)$. Similarly, we choose the best code for encoding $bc(B)$ and $d(B)$.

We encode $type(B)$ to a binary string so that a type which often appears is encoded to a shorter string, but the details are omitted here.

4.3 Compression of Inter-host Links

For each host, we regard all the inter-host links for the host as intra-host links by giving "new local indices" to the destinations of inter-host links. We then compress the inter-host links and the intra-host links all together by the method described in Section 4.2.

For each host H, our method first gives "new local indices" to all the destinations of inter-host links whose sources belong to H. Let n be the number of pages in H. Let m be the number of destinations of inter-host links for H, and let $t(1) < t(2) < \cdots < t(m)$ be the original indices of all the destinations of inter-host links. The *new local index* $N(t(i), H)$ of a destination $t(i)$ is defined to be an integer $n + i - 1$.

Our method then constructs a "new intra-destination list" of each page p in host H. If $t(x_1) < t(x_2) < \cdots < t(x_k)$ are the original indices of destinations of inter-host links for page p, then the *new intra-destination list* of page p is the union of two lists: one is the intra-destination list of page p, and the other is the list of new local indices $N(t(x_1), H), N(t(x_2), H), \cdots, N(t(x_k), H)$.

We thus compress intra-host links and inter-host links all together for each host by the method in Section 4.2. The input to the method is the new intra-destination lists for the host. Let p be the local index of a page in host H, and let $t(x_i)$ be the i-th smallest orignal index of destinations of inter-host links of page p. Then, for most inter-host links, $|N(t(x_i), H) - p|$ is much smaller than $|t(x_i) - p|$. Thus, by using new local indices, we can cover all 1-elements in the adjacency matrix by a relatively small number of blocks, and can efficiently compress inter-host links together with intra-host links. Experimental analyses will be presented in Section 5.

For each host, it is necessary to store, in a table, a pair of the new local index and original index of the destination of each inter-host link. Using the table, one can retrieve orignal indices from new local indices. We efficiently represents the table by a differential list of the original indices.

5 Experiments

For the computational experiments, we use three data sets of Web graphs, named `cnr-2000`, `in-2004`, and `eu-2005`, collected by Boldi and Vigna [7]. These data sets can be downloaded from their site [21].

Table 5. The size of the compressed data

Data set	cnr-2000	in-2004	eu-2005
Pages	325,557	1,382,908	862,664
Links	3,216,152	16,917,053	19,235,140
Hosts	722	4,409	417
BV, $\alpha = 3$(bit/link)	3.56	2.82	5.17
BV, $\alpha = \infty$(bit/link)	2.84	2.17	4.38
Ours (bit/link)	1.99	1.71	2.78
Ratio(%)	70.1	78.8	63.5

Table 5 depicts the size of each data set compressed by our method and that of Boldi and Vigna [7]. The numbers of pages and links in each data set are written in rows “Pages” and “Links,” respectively. Each cell in the row “Ours” represents the size of the compressed data per link, obtained by our method. Similarly, the rows “BV, $\alpha = \infty$” and “BV, $\alpha = 3$” represent those by Boldi and Vigna’s method with $\alpha = \infty$ and $\alpha = 3$, respectively, where α is the maximum length of a copy chain as described in Section 3. The “Ratio” row represents the ratio of the size of the data compressed by our method to that by the “Boldi and Vigna, $\alpha = \infty$.” On average, the size of the data compressed by our method is smaller than 70.8% of theirs.

We use the method of Boldi and Vigna implemented by themselves, which is available on their site [21]. Our method is implemented with Java, and the experiments run on a PC with Core2 Duo E6600 (2.40GHz) and 2GB main memory.

Table 6. Retrieval time for the whole compressed data

Data set	cnr-2000	in-2004	eu-2005
BV, $\alpha = 3$	1.48s	5.87s	7.34s
BV, $\alpha = \infty$	1.25×10^3s	1.54×10^3s	5.38×10^3s
Ours	0.68s	3.73s	1.94s

Table 7. Retrieval time for a specified page

Data set	cnr-2000	in-2004	eu-2005
BV, $\alpha = 3$	3.51×10^{-2}ms	6.07×10^{-2}ms	4.07×10^{-2}ms
BV, $\alpha = \infty$	4.35ms	1.22ms	6.73ms
Ours	2.34ms	2.38ms	28.72ms

Table 6 depicts the time required to retrieve the whole compressed data. Table 7 depicts the time to retrieve the destination list of a specified page, which is the average time for randomly selected 10,000 pages for each data set. The retrieval time of our method is written in the column “Ours.” Similarly, that of Boldi and Vigna’s method with $\alpha = 3$ is written in the column “BV, $\alpha = 3$,” which is much faster than their method with $\alpha = \infty$, written in the column “BV, $\alpha = \infty$.”

Both our method and theirs take time $O(M)$ to retrieve the whole compressed data, where M is the number of links in a Web graph. However, our method is experimentally several times faster than theirs with $\alpha = 3$, and is about 1000 times faster than theirs with $\alpha = \infty$.

For a request to retrieve a specified page, our method must retrieve all the links whose sources belong to the same host as the specified page. On the other hand, their method must retrieve the destination lists of all the pages in a copy chain, whose length is at most α. Our method retrieves a specified page experimentally much slower than their method with $\alpha = 3$. For `cnr-2000` data set, our method is faster than their method with $\alpha = \infty$, although our method is slower than theirs for `in-2004` and `eu-2005` data sets. Our method takes much time particularly for `eu-2005` data set, because the data set have several hosts containing a huge number of links and pages.

6 Concluding Remarks

We have proposed a new efficient method of compressing a Web graph. We have introduced six types of blocks to cover all 1-elements in an adjacency matrix to fully utilize localities of intra-host links. We have also proposed a technique for compressing inter-host links and intra-host links all together by giving new local indices to the destinations of inter-host links. The size of data compressed by our method is about 70.8%, on average, of that by Boldi and Vigna's method which has been known as the most efficient method of compressing a Web graph [7]. The retrieval of our method for the whole compressed data is faster than their method, although that for a specified page could be slower than their method. Thus, one of the possible future works is to improve the retrieval speed for a specified page.

References

1. Asano, Y., Ito, T., Imai, H., Toyoda, M., Kitsuregawa, M.: Compact Encoding of the Web Graph Exploiting Various Power Laws: Statistical Reason Behind Link Database. In: Dong, G., Tang, C.-j., Wang, W. (eds.) WAIM 2003. LNCS, vol. 2762, pp. 37–46. Springer, Heidelberg (2003)
2. Asano, Y., Nishizeki, T., Toyoda, M., Kitsuregawa, M.: Mining Communities on the Web Using a Max-Flow and a Site-Oriented Framework. IEICE Trans. Inf. Syst. E89-D (10), 2606–2615 (2006)
3. Asano, Y., Tezuka, Y., Nishizeki, T.: Improvements of HITS Algorithms for Spam Links. In: Dong, G., Lin, X., Wang, W., Yang, Y., Yu, J.X. (eds.) APWeb/WAIM 2007. LNCS, vol. 4505, pp. 479–490. Springer, Heidelberg (2007)
4. Bharat, K., Broder, A., Henzinger, M., Kumar, P., Venkatasubramanian, S.: The Connectivity Server: Fast Access to Linkage Information on the Web. In: Proc. of the 7th WWW, pp. 469–477 (1998)
5. Blandford, D.K., Blelloch, G.E., Kash, I.A.: Compact Representation of Separable Graphs. In: Proc. of the 14th SODA, pp. 679–688 (2003)
6. Brin, S., Page, L.: The Anatomy of a Large-Scale Hypertextual Web Search Engine. In: Proc. of the 7th WWW, pp. 14–18 (1998)
7. Boldi, P., Vigna, S.: The Web Graph Framework I: Compression Techniques. In: Proc. of the 13th WWW, pp. 595–601 (2004)
8. Boldi, P., Vigna, S.: Codes for the World Wide Web. Internet Mathematics 2(4), 405–427 (2005)
9. Claude, F., Navarro, G.: A Fast and Compact Web Graph Representation. In: Ziviani, N., Baeza-Yates, R. (eds.) SPIRE 2007. LNCS, vol. 4726, pp. 118–129. Springer, Heidelberg (2007)
10. Cormen, T.H., Leiserson, C.E., Rivest, R., Stein, C.: Introduction to Algorithms. 2nd edn. MIT Press, Cambridge (2001)
11. Elias, P.: Universal Codeword Sets and Representaions of the Integers. IEEE Transactions on Information Theory 21, 194–203 (1975)
12. Flake, G.W., Lawrence, S., Giles, C.L.: Efficient Identification of Web Communities. In: Proc. of the 6th KDD, pp. 150–160 (2000)
13. Guillaume, J.L., Latapy, M., Viennot, L.: Efficient and Simple Encodings for the Web Graph. In: Meng, X., Su, J., Wang, Y. (eds.) WAIM 2002. LNCS, vol. 2419, pp. 328–337. Springer, Heidelberg (2002)

14. Kleinberg, J.: Authoritative Sources in a Hyperlinked Environment. In: Proc. of the 9th SODA, pp. 668–677 (1998)
15. Kou, W.: Digital Image Compression: Algorithms and Standards. Springer, Heidelberg (1995)
16. Kumar, R., Raghavan, P., Rajagopalan, S., Tomkins, A.: Trawling the Web for Emerging Cyber-Communities. Computer Networks 31(11-16), 1481–1493 (1999)
17. Larsson, N.J., Moffat, A.: Off-Line Dictionary-Based Compression. Proc. IEEE 88(11), 1722–1732 (2000)
18. Levenstein, V.E.: On the Redundancy and Delay of Separable Codes for the Natural numbers. Problems of Cybernetics 20, 173–179 (1968)
19. Randall, K., Stata, R., Wickremesinghe, R., Wiener, J.L.: The Link Database: Fast Access to Graphs of the Web. Research Report 175, Compaq Systems Research Center, Palo Alto, CA (2001)
20. Suel, T., Yuan, J.: Compressing the Graph Structure of the Web. In: Proc. of the Data Compression Conference, pp. 213–222 (2001)
21. WebGraph Homepage, `http://webgraph.dsi.unimi.it/`
22. Wickremesinghe, R., Stata, R., Wiener, J.: Link Compression in the Connectivity Server. Technical Report, Compaq Systems Research Center, Palo Alto, CA (2000)
23. Zhang, Y., Yu, J.X., Hou, J.: Web Communities: Analysis and Construction. Springer, Berlin (2006)