Graph Compression by BFS

Mert Bektas [1]

Abstract

The paper introduces a compression method for Web Graphs that provides quick information recalls while saving from space about 10% over existing methods.

1. Introduction

Due to poor data retrieval speeds, some compression algorithms have recently been developed for large graphs to reduce the number of bits per link. Unlike other methods, the algorithm presented in the paper does not need to consult URLs, which makes the approach applicable to the graphs with more general nature. In addition, the method can do quick testing if two input pages share a hyperlink.

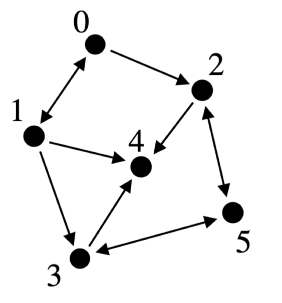
2. Preliminaries

The Web Graph is a directed graph *G = (V, E)*, where *V* is the group of URL indices or identifiers and *E* is the links between them. From the perspective of compression, sorting the URLs lexicographically and then assigning each page its appropriate rank in the ordering is a convenient way to assign indices. However, the method shown in the paper is based on ordering nodes using the Breadth First Search (BFS).

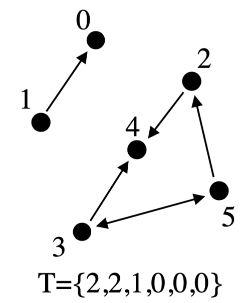
3. Encoding by BFS

Instead of using underlying URLs, the team's method is based on the topological structure of the Web Graph. There are two compression phases. In the first phase, they perform BFS to *G*, and the index of each node correspondent with the way they expand. For the last phase, they compress all the remaining links.

When expanding a node during Phase 1, they assign consecutive integer indices to its node while storing its value. All the links in the breadth-first tree are encoded in a sequence once the traversal is complete. They named this a traversal list. After their tests, they saw that this traversal removes almost $|V| - 1$ links from the graph. Figure 1 shows an example of Phase 1, where Figure 1(a) is before and Figure 1(b) is after the BFS, and below Figure 1(b), there is the traversal list.



(a) Indices assigned to the nodes.



(b) Remaining links after BFS and the *traversal list T*.

Figure 1. Illustrating Phase 1.

After this, they compress a consecutive chunk of nodes according to a value called compression level, which is chosen deliberately. In addition, items from the traversal list that are relevant to the chunk's nodes are prefixed to each compressed chunk.

During Phase 2, an encoding happens for the adjacency list of each node in a chunk in increasing order. Every encoding operation comprises the distance between the list's adjacent elements and a type indicator, which is required for decoding. Moreover, an encoding will be referred to by its type.

This encoding makes it possible that two nodes are likely to be assigned close index values if they are connected. Since neighboring nodes in the graph commonly share many neighbors, similar consecutive lines will be present in the adjacency lists. This situation results in four sorts of "redundancies," the exploitation of which is explained in the following manner.

1. Encoding a run of identical lines by assigning a multiplier to the first line in the sequence;

2. The degrees of ensuing nodes are gap-encoded because there are intervals of constant node degrees;

Mert Bektas [1]Institute of Computer Science, University of Tartu, Tartu, Estonia. Correspondence to: Mert Bektas <mert.bektas@ut.ee>.

3. The sequence is run-length encoded when a series of at least $\mathcal{L}_{min}$ identical elements exist for some appropriately fixed $\mathcal{L}_{min}$;

4. In the end, run-length encoding is applied to a box of identical rows that exceeds a predetermined threshold size $\mathcal{A}_{min}$.

4. Experiments

In Figure 2, their compressed graphs' sizes are shown. The datasets used by the team were collected by Boldi and Vigna (1), and they were gathered using UbiCrawler (2). From Figure 2, it can be seen that the current method performed much better than BV (1), BC (3), and Asano et al. (4), with a compression level $\ell = 10^4$. BV's highest compression scores ($\mathcal{R} = \infty$) are similar to those obtained at level 8. In addition, their general usage results ($\mathcal{R} = 3$) are comparable to the team's method's level 4.

	BV [3]				This Paper		
	$R = 3$		$R = \infty$				
	BFS		BFS		$l = 10^4$	$l = 8$	$l = 4$
cnr-2000	3.92	3.56	3.23	2.84	1.87	2.64	3.33
in-2004	3.05	2.82	2.35	2.17	1.43	2.19	2.85
eu-2005	4.83	5.17	4.05	4.38	2.71	3.48	4.20
indochina-2004	2.03	2.06	1.46	1.47	1.00	1.57	2.09
uk-2002	3.28	3.00	2.46	2.22	1.83	2.62	3.33
arabic-2005	2.58	2.81	1.87	1.99	1.65	2.30	2.85

Figure 2. Compressed sizes in bits per link.

The following figure illustrates that the BV general usage version ($\mathcal{R} = 3$) operates at a speed equivalent to the team's method, while the BV high compression mode is slower than theirs. However, their method is quicker when the team settles on $\ell = 4$.

	BV [3]		This Paper	
	$R = 3$	$R = \infty$	$l = 8$	$l = 4$
cnr-2000	1.66 μs	1.22 ms	1.40 μs	0.95 μs
in-2004	1.89 μs	0.65 ms	1.55 μs	1.13 μs
eu-2005	2.58 μs	2.35 ms	3.16 μs	2.09 μs
indochina-2004	2.31 μs	0.93 ms	2.42 μs	1.72 μs
uk-2002	2.35 μs	0.20 ms	2.16 μs	1.52 μs
arabic-2005	2.80 μs	1.15 ms	3.09 μs	2.11 μs

Figure 3. Average times to retrieve the adjacency list of a node.

Lastly, Figure 4 displays the requirement of the actual main memory (top) by BV, the team's method, and the space-time tradeoff (bottom). When the team's method's compression level is 8, the space requirement of it is equal to 80% of BV at $\mathcal{R} = 3$.

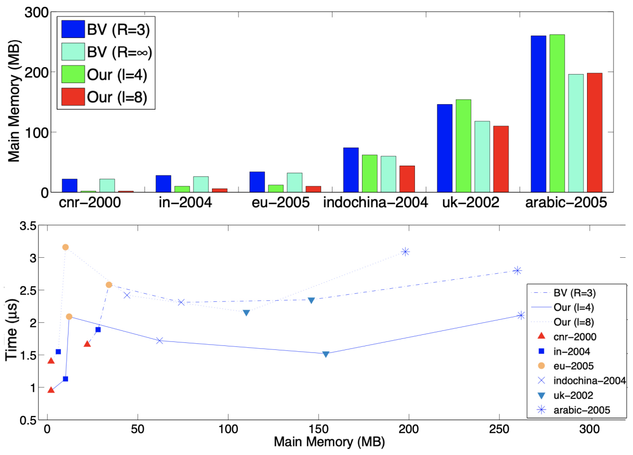


Figure 4. Main memory usage (top) by BV and the present method and the space-time tradeoff (bottom).

5. Conclusion

The paper introduced a novel approach to compress the Web Graphs and graphs with similar structures. Unlike former works, the team used a traversal to order nodes. The size of the compressed files is even smaller than the current state of the art (4). In addition, they presented a quick query to determine whether two nodes are connected without having to construct a whole adjacency list.

References

[1] Boldi, P.; Vigna, S. The web graph framework I: Compression techniques. Thirteenth International World Wide Web Conference, New York, NY, USA, 2004; ACM Press: Manhattan, USA, 2004, pp. 595-601.

[2] Boldi, P.; Codenotti, B.; Santini, M.; Vigna, S. UbiCrawler: a scalable fully distributed web crawler. Software: Pract. Exp. 2004, 34, 711-726.

[3] Buehrer, G.; Chellapilla, K.A scalable pattern mining approach to web graph compression with communities. International conference on Web search and web data mining, Palo Alto, CA, USA, February 11-12, 2008; ACM: New York, NY, USA, 2008; pp. 95-106.

[4] Asano, Y.; Miyawaki, Y.; Nishizeki, T. Efficient compression of web graphs. 14th annual international conference on Computing and Combinatorics, Dalian, China, June 27 - 29, 2008; Springer-Verlag: Berlin, Heidelberg, Germany, 2008, pp. 1-11.