About Graph Index Compression Techniques

Antonio Cruciani
Daniele Pasquini
antonio.cruciani@students.uniroma2.eu
daniele.pasquini@uniroma2.it
Dipartimento di Ingegneria dell'Impresa, University of Rome "Tor Vergata"
Rome, Italy

Giambattista Amati
gba@fub.it
Fondazione Ugo Bordoni
Rome, Italy

Paola Vocca
vocca@unitus.it
Department of Humanities, Communication and Tourism (DISUCOM), University of Tuscia,
Viterbo, Italy

ABSTRACT

We perform a preliminary study on large graph efficient indexing using a gap-based compression techniques and different node labelling functions. As baseline we use the Webgraph + LLP labelling function. To index the graph we use three labelling functions: Pagerank, HITS, and Pagerank with random walks choosing restart nodes with HITS authority scores. To compress the graphs we use Varint GB, with and without d-gaps, derived by rank value of the labelling function. Overall, we compare 8 different methods on different datasets composed by the WebGraph eu-2005, uk-2007-05@100000, cnr-2000, and the social networks, enron, ljournal-2008, provided by the Laboratory for Web Algorithmics (LAW).

CCS CONCEPTS

• **Information systems → Information retrieval**; *Query representation*; • **Theory of computation → Data compression**;

KEYWORDS

Graph compression, Webgraph LLP, Varint GB, PageRank, HITS

1 INTRODUCTION

The analysis of the structure of real graphs requires many data mining tasks (e.g., detecting specific nodes, identifying interest groups, estimating measures of centrality, evaluating distance based measures etc. [12, 13]). Often, graph algorithms used to implement these mining tasks assume that the graph is stored into the main memory. However, this assumption is far from trivial when dealing with large graphs, and this is actually the case when social networks are considered. In these cases, even the successor lists would require hundreds of terabytes of memory.

Store and access large graphs is a central issue in the field of information retrieval [1, 8, 9]; a *good* compression technique allows to efficiently manage large graphs; better understand the social network structure; and derive and exploit possible regularities. In this context, a compressed data structure for a graph, also known as *succinct* representation [10], must provide both very fast amortized access to an edge (link) and require an amount of space that is "close" to the information-theoretic lower bound, as opposed to other *compressed representation* whose only evaluation criterion is the number of bits per link. While this definition is not formal, it excludes methods in which the successors of a node are not directly accessible unless, for instance, a large part of the graph is scanned.

We use the *Varint GB* [5] compression schema. In order to take into account the structure of the graph, we use three labelling functions which label the nodes of the graph in not increasing order of: 1) *PageRank* [3]; 2) HITS (Hyperlink-Induced Topic Search) also known as hubs and authorities [11]; and 3) PageRank having as seed the HITS authority number, that is, PageRank with random walks choosing restart nodes with HITS authority scores. As an additional compression step, we use the d-gap compression of the adjacency lists. We choose as a baseline the labelling function WebGraph+LLP [1, 2]. Hence, we overall test 8 different compression techniques, derived by mixing 4 labelling functions (PageRank HITS, mixed PageRank-HITS, LLP), with or without d-gaps compression of the adjacency lists, and the encoding *Varint GB*. The experimentation is performed using the datasets provided by the Laboratory for Web Algorithmics (LAW) [1, 2]. More precisely, we consider three graphs derived from the web crawling (eu-2005, uk-2007-05@100000, cnr-2000) and three social networks (enron, ljournal-2008).

We derive the following results:

- Without d-gaps representation of the adjacency lists of nodes, the best labelling strategy is the greedy one, based on the use of node degrees ranking, the entropy being the lower bound of the compression, that is $m \cdot E$ bits where E is the entropy of the system and m is the number of the edges.
- With d-gaps different labelling strategy of the nodes are required according to what the graph represents. For example, in order to minimize gaps and thus compression, nodes in a cluster need to be consecutive in a linear ordered sequence. Using PageRank strategy to visit a graph, we first visit nodes in a cluster and then we teleport (with the dumping factor probability) to a new node that will be lead us to surf a portion of the graph (high a hubness score), but in the meantime containing a high authoritative nodes, that is nodes where many randoms walks end into them. This intuition leads us to provide a labelling/ranking node strategy, which is also a visiting/crawling strategy that assigns lower values to high PageRank scores nodes (they must compare many ranks before others in any adjacency list) and when we jump to a new subgraph we restart with new nodes that have high PageRank scores, that is the ones that are also good hub

IIR 2019, September 16–18, 2019, Padova, Italy

nodes. HITS authority nodes are indeed the authoritative nodes that contain also good hub nodes in their random walks. This remark suggests us to use PageRank with restart nodes. These restart nodes are those that have the highest HITS authoritative scores.

- Experiments show that the best strategy for web collections (uk-2007-05@100000 and cnr-2000) is the webGraph+LLP strategy. For other kind of graphs the other labelling functions produce better results, even if we still need to assess which is the best strategy which allows to obtain an optimal compression. In the specific case of social networks, for example, we need to combine hub and authority scores to linearly order the nodes of graphs and thus compress the graph indexes by the d-gap. It seems that using PageRank scores with restart nodes with HITS authoritative scores is a very intuitive and promising strategy to model minimal d-gap distributions in the adjacency lists of the nodes.

The paper is organized as follows: In Section 2 we formally define all the tools used; Section 3, the development platform is presented, while in Section 4 we describe the results of the experimentations. Finally, in Section 5, we conclude and present further investigation directions.

2 BASIC DEFINITIONS AND FRAMEWORK DESCRIPTION

In this section we describe both the compression schema and the labelling functions used to exploit the graph structure and, hence, to obtain a more efficient compression. For lack of space we do not describe the framework WebGraph+LLP that we use as a benchmark for our analysis. A detailed description can be found in [1, 2].

2.1 Compression schema

Varint GB. The VarInt GB or Group VarInt is a variant of the VarInt compression scheme proposed by Jeff Dean at the 2009 WSDM conference[5]. The idea of this technique is to encode the integers in groups of four at a time. Each group is preceded by a prefix of a byte which is composed of four pairs of bits that specify the length of each integer following the prefix. For example, the following prefix byte:

$$00, 00, 01, 10$$

indicates that the following four integers are of length, respectively: one byte, one byte, two bytes and three bytes. So each group of four integers occupies a space ranging from 5 to 17 bytes. A simple search table based on the prefix bytes can therefore be used to decode the encoding. The advantage of this coding schema is that the values can be decoded with fewer branching errors and less bit-to-bit operations.

Another important coding is the **Elias γ-code**, that we may use in the experimentation, but it is not important to the final aim of this preliminary work because it is the d-gap generation that matters in the comparison of labelling strategies. For sake of completeness we remind that the Elias γ-code is an efficient encoding algorithm widely used in practice [7]. The idea of this technique is as follows: Given an integer $x \geq 1$ it is "broken" into two components $n = 1 + \lfloor \log_2 x \rfloor$ which encode, in unary, the length of x in binary and $r = x - 2^{\lfloor \log_2 x \rfloor}$ the carry, which is coded in binary. The unary component, n, specifies the number of bits needed to encode x, and the binary component, r, is encoded using $\lfloor \log_2 x \rfloor$ bits.

2.2 Labelling Functions

The compression schema above described are designed to compress any type of information without taking into account the specificity of the data set under examination. In the case of graphs represented by adjacency lists it is possible to perform some optimizations.

Generally, input graphs are represented by lists of arcs or edges, that is, each node is represented by an integer and each row of the input file represents an arc (clearly an arc is represented by a pair of integers (u, v)). So first we need to convert the list of edges into an adjacency list and then apply the encoding algorithms on the adjacency list. Since we have to deal with large graphs it is very likely that there are nodes with indexes represented by very large integers.

We now tackle the problem on how to encode the adjacency list of the nodes of a graph in order to have the optimal compression scheme.

We first try to label nodes in not increasing order of frequencies (node with higher frequency are encoded with a lower label). Unfortunately, the frequency it is not sufficient to capture the structure of real large graphs, for which the degree distribution is not enough to minimize d-gap cost labelling of complex graphs. Instead, we use three different labeling functions:1) *PageRank* [3]; 2) HITS (Hyperlink-Induced Topic Search) also known as hubs and authorities [11]; and 3) PageRank having as seed the HITS number.

PageRank Index *PageRank index* was introduced in [3] and it recursively quantifies a "value" or the PageRank of a node based on: (i) the number of links it receives, (ii) the link propensity of the linkers (that is, the number of outgoing links of each in-going node), and (iii) the centrality of the linkers, that is their PageRank. According to Google[1]: *PageRank index works by counting the number and quality of links to a page to determine a rough estimate of how important the website is. The underlying assumption is that more important websites are likely to receive more links from other websites.* It is mathematically defined as:

$$C_{PR}(v) = \alpha \sum_{v \in N^-(u)} a_{vu} \frac{C_{PR}(v)}{|N^+(v)|} + \frac{1-\alpha}{n}, \tag{1}$$

where $N^+(u)$ $(N^-(u))$ is the set of vertices $v \in V$ linked to u with outgoing (in-coming) edges, that is such that $(u, v) \in E$ ($(v, u) \in E$, respectively) and $n = |V|$. Additionally, α is the *damping factor* which models the probability that, in a random walk on the network the walk will stop. It is generally assumed that the damping factor will be set around 0.85 [3]. Hence, according to the PageRank index, a node is important if it is linked from other node with high PageRank index and, at the same time, it links parsimonious nodes or if it is highly linked.

HITS Hypertext Induced Topics Search (HITS) was developed by Jon Kleinberg [11] and it identifies good *authorities* and *hubs* for a topic by assigning two numbers to a page: an authority and a

[1]Facts about Google and Competition. https://web.archive.org/web/20111104131332/https://www.google.com/competition/howgooglesearchworks.html
Archived from the original on 4 November 2011. Retrieved 12 July 2014.

hub weight. An authority is a page that many hubs link to; a hub is a page that links to many authorities. The weights are defined recursively. A higher authority weight occurs if the page is pointed to by pages with high hub weights. A higher hub weight occurs if the page points to many pages with high authority weights.

2.3 d-Gap

To obtain a better compression, we can use a technique used in Information Retrieval, that is to encode the differences of the ranks of the elements of the adjacency list instead of the ranks themselves. Since the lists of each node are sorted in ascending order, the differences (called *d-gaps*) must be positive integers greater than zero. We show an example of this technique: Let $\{i_1, i_2, i_3, i_4, \ldots\}$ be the adjacency list of a node x, then the *d-Gap encoding* is $\{i_1, i_2 - i_1, i_3 - i_2, i_4 - 1_3, \ldots\}$. This technique allows us to obtain smaller dimensions than the original list and therefore also smaller encodings than those applied to the original adjacency list. It is clear that this technique is *Lossless*, since there is no lost of information because the original integers, given the integers encoded with d-gap, can be easily reconstructed.

3 IMPLEMENTATION

3.1 Development environment

The development environment is based on Apache Spark. As for the data structure used, we used the `Bytes` structure for the VarInt GB encoding. The compression of the instances and the gaps construction have been made in a distributed environment with Spark Python on a cluster of 8 CX Server machines *2550 Fujitsu* (CX 400 M1) with 32 GB of memory each and CPU Xeon e5 1620 v4.
All the data are stored on distrubuted files (HDFS). ‘ To optimize the encoding time of the posting lists we used bitwise operations.

3.2 Apache Spark Implementation

We used the Apache Spark framework to convert the original graph data structure from the edge list to the posting list; to relabel and then to compress the graph. First we perform a **`map`** job for reading the edge list and to obtain the RDD of the edge list where each element is a tuple ⟨source, target⟩; then, for what concern the relabeling task, given the RDDs where vertices are ordered in not increasing order of HITS, Pagerank or HITS+Pagerank, we:

1) relabel each node in increasing order, starting from 0, of HITS, Pagerank or HITS+Pagerank scores;
2) excute a **`map`** job on the RDD edge list assigning to each vertex the new label obtained in 1).

As a next step, we produce the adjacency list data structure from the renamed RDD edge list. We execute a **`map`** job for converting the target vertex of each tuple in a list of one element, ⟨source, [target]⟩, and then perform a **`reduceByKey`** function using the RDD with the relabeled tuples, concatenating the target value lists, so obtaining the adjacency list. Now, as for the *d-Gap* compression, we use a **`map`** job to apply to each list of the adjacency list the **d-`Gap`** function. As a last step, we execute again the **`map`** function to the new RDD (the adjacency list data structure) in order to execute the *Varint GB* encoding function.

3.3 Datasets

As a test set we used six graphs from the WebGraph datasets [1, 2]. Table 1 reports the graphs dimensions.

- **eu-2005**: A small crawl of the .eu domain, mainly useful for debugging and testing purposes. This graph exhibits a very low locality, probably because the crawl was quite shallow (and the chosen domain is quite artificial anyway).
- **uk-2007-05@100000**: This graph has been artificially generated from uk-2007-05. It is a ball (in the graph metric sense) of 100000 nodes centered at a random node. It simulates the result of a small breadth-first crawl around the node.
- **enron**: This dataset was made public by the Federal Energy Regulatory Commission during its investigations: it is a partially anonymised corpus of e-mail messages exchanged by some Enron employees (mostly part of the senior management). We turned this dataset into a directed graph, whose nodes represent people and with an arc from x to y whenever y was the recipient of (at least) a message sent by x.
- **ljournal-2008**: LiveJournal is a virtual-community social site started in 1999: nodes are users and there is an arc from x to y if x registered y among his friends. It is not necessary to ask y permission, so the graph is directed). This graph is the snapshot used in [4].
- **cnr-2000**: A very small crawl of the Italian CNR domain, mainly useful for debugging and testing purposes.

graph	nodes	arcs
eu-2005	862664	19235140
uk-2007-05@100000	100000	3050615
enron	69244	276143
ljournal-2008	5363260	79023142
cnr-2000	325557	3216152

Table 1: Datasets dimensions

4 ANALYSIS

Our main focus is to determine on the best graphs labelling technique for compression. The behavior of a labelling technique, among other features, mostly depends on the type or nature of the graph that we want to compress. Indeed, our conjecture is that the LLP strategy is not optimal when the graph is generated by a Social Network while it is optimal when the graph a portion/instance of the Web Graph. This is due to the topological differences of this class of graphs and to the crawling techniques used to discover them. Considering the Web Graphs, a typical crawling policy is to use a BFS to visit in a lexicographical order urls pointing to other pages. By doing this, we cluster implicitly nodes by their hosting domains, with a higher probability that many links are internal to the domain, and thus it can capture the scale free property of the network. Our experiments show (see Table 2 that LLP is optimal for this class of graphs because for each cluster it labels the adjacent vertices with close integer values allowing to get small size gaps for each posting list of each node of each cluster. However, from our experiments, for Social Network graphs there is a better labelling technique that

	Varint GB Compression								
Graph	LLP	Gap-LLP	HITS-Labeling	Gap-HITS Labeling	PageRank Labeling	Gap-PageRank Labelling	HITS-PageRank Labeling	GapHITS PageRank Labeling	Original size of the Graph
enron	22.94	18.14	20.77	17.64	20.48	**16.89**	20.42	17.15	95.60
uk-2007-05@100000	20.97	**11.28**	20.97	11.8	19.41	12.06	20.45	12.06	93.88
cnr-2000	26.86	**14.43**	25.12	14.92	28.87	15.42	24.38	14.92	106.46
Live Journal 2008	3.40	3.40	3.40	**3.24**	**3.24**	**3.24**	**3.24**	**3.24**	7.64
eu-2005	25.66	24.83	26.61	**13.72**	25.79	14.56	25.79	**13.72**	106.64

Table 2: Results of the experimentation in terns of space occupancy, values are in bits per link, in bold, the best results

is based on the use of the PageRank with initial seeds (restarting nodes) given by the highest authorities given by HITS algorithm. The interpretation is that, in a Social Network, the centroid of a cluster, i.e. celebrities, politicians, etc., can be seen as vertices with the highest authority scores. These nodes are the end of many random walks and are good candidates to be PageRank restart nodes. The intuition is if we can obtain a better compression schema if we label vertices having higher PageRanks with smaller integers since they more frequently appear in other vertices posting lists.

5 CONCLUSIONS AND FUTURE WORKS

We have showed that for web collections like uk-2007-05@100000 and cnr-2000 the LLP strategy is optimal but for other types of graphs we still need to assess better strategies to obtain an optimal compression of the graphs. For social networks, for example, we need to combine hub and authority scores to linearly order the nodes of graphs and thus compress the graph indexes by the d-gap. It seems that using PageRank scores with restart nodes with HITS authoritative scores is a very intuitive and promising strategy to model minimal d-gap distributions in the adjacency lists of the nodes. There are three main future directions: 1) to consider real large graphs different from social networks and web graphs, such us, for example, biological networks, and of different dimension; 2) to implement and test other compression schemes and labelling function, as for example, the **Elias γ-code** for the graph compression and the labelling function presented in [6] based on recursive graph bisection; 3) besides to the space occupancy, it would be important to evaluate encoding and decoding time, as well as query or the adjacency list time.

ACKNOWLEDGMENTS

This work was conducted in the Laboratory of Big Data of ISCOM-MISE (Institute of communication of the Italian Ministry for Economic Development).

REFERENCES

[1] Paolo Boldi, Marco Rosa, Massimo Santini, and Sebastiano Vigna. 2011. Layered Label Propagation: A Multiresolution Coordinate-free Ordering for Compressing Social Networks. In *Proceedings of the 20th International Conference on World Wide Web (WWW '11)*. ACM, New York, NY, USA, 587–596. https://doi.org/10.1145/1963405.1963488

[2] Paolo Boldi and Sebastiano Vigna. 2004. The WebGraph Framework I: Compression Techniques. In *Proc. of the Thirteenth International World Wide Web Conference (WWW 2004)*. ACM Press, Manhattan, USA, 595–601.

[3] Sergey Brin and Lawrence Page. 1998. The Anatomy of a Large-scale Hypertextual Web Search Engine. *Comput. Netw. ISDN Syst.* 30, 1-7 (April 1998), 107–117. https://doi.org/10.1016/S0169-7552(98)00110-X

[4] Flavio Chierichetti, Ravi Kumar, Silvio Lattanzi, Michael Mitzenmacher, Alessandro Panconesi, and Prabhakar Raghavan. 2009. On Compressing Social Networks. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '09)*. ACM, New York, NY, USA, 219–228. https://doi.org/10.1145/1557019.1557049

[5] Jeffrey Dean. 2009. Challenges in Building Large-scale Information Retrieval Systems: Invited Talk. In *Proceedings of the Second ACM International Conference on Web Search and Data Mining (WSDM '09)*. ACM, New York, NY, USA, 1–1. https://doi.org/10.1145/1498759.1498761

[6] Laxman Dhulipala, Igor Kabiljo, Brian Karrer, Giuseppe Ottaviano, Sergey Pupyrev, and Alon Shalita. 2016. Compressing Graphs and Indexes with Recursive Graph Bisection. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016.* 1535–1544. https://doi.org/10.1145/2939672.2939862

[7] Peter Elias. 1975. Universal Codeword Sets and Representations of the Integers. *IEEE Trans. Inf. Theor.* 21, 2 (Sept. 1975), 194–203. https://doi.org/10.1109/TIT.1975.1055349

[8] Paolo Ferragina, Francesco Piccinno, and Rossano Venturini. 2015. Compressed Indexes for String Searching in Labeled Graphs. In *Proceedings of the 24th International Conference on World Wide Web (WWW '15)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland, 322–332. https://doi.org/10.1145/2736277.2741140

[9] Anna Gilbert and Kirill Levchenko. 2004. Compressing network graphs. In *In LinkKDD*.

[10] Guy Joseph Jacobson. 1988. Succinct static data structures. (1988).

[11] Jon M. Kleinberg. 1998. Authoritative Sources in a Hyperlinked Environment. *J. ACM* 46 (1998), 668–677.

[12] David Knoke and Song Yang. 2008. *Social Network Analysis*. Vol. 154. https://doi.org/10.1007/978-1-4614-6170-8_362

[13] Stanley Wasserman and Katherine Faust. 1994. *Social Network Analysis: Methods and Applications*. Cambridge University Press. https://doi.org/10.1017/CBO9780511815478