International Journal of Engineering Research and General Science Volume 3, Issue 5, September-October, 2015
ISSN 2091-2730

A Review: Sparse Web Graph Compression using Eulerian Data Structure and its Optimization

Ankita Singh

UPTU University, ankjai121292@gmail.com

Abstract— The purpose of this paper is to review the sparse web graph compression technique using Eulerian data structure and optimization of the designing of compressed scheme based on careful application. Web graphs form the foundation of social networks but their storage is a big challenge due to their large size. Compression techniques are proposed in the recent time in which queries can be carried out without decompressing the graph. This paper reviews graph compression using Eulerian data structure and multiposition linearization which is capable of solving in and out query and hence exclude the need of storing transpose of Web graph for in-neighbor query determination. Some optimization approaches are also proposed which depends on compression ratio and k- factor.

Keywords— Graph, compression, optimization, sparse, query, application, Eulerian data

INTRODUCTION

The World Wide Web has become the focus of research in the recent times. Social networks, network consisting of individuals or organization as nodes, are an important part of World Wide Web. In social networks, nodes are connected by one or more type of interdependency such as friendship and kinship. Compression of networks, using Eulerian data structure and multiposition linearization, such that neighbor queries can be carried out on the same graph without decompressing the graph or storing its transpose. Various factors such as optimal value of k and compression ratio are to be analyzed.

NOTIONS

1. Network-: A network is modeled as directed graph G = (V, E) where V is a set of vertices and E ⊆ V × V is a set of edges. We also refer to V by V(G) and E by E(G). For an edge e= (u,v), we refer to u as the source of e and v as the destination of e.(u,v) ≠e(v,u).
2. Undirected Graph-: For an undirected graph G, we can obtain the directed version G of G by placing two directed edges (u,v) and (v,u) in Gd for each undirected edge {u, v) in G.
3. Transpose-: For a graph G, the transpose of G, is denoted by GT, is a graph such that V(GT) = V(G) and (u,v) belongs to E(GT) if and only if (v, u) belongs to E(G).
4. Reciprocal-: In a graph G, an edge (u,v) belongs to E is called reciprocal if (v,u) belongs to E as well. In such a case, u and v are immediately connected in both directions. Let Fre(G) be the fraction of reciprocal edges in E(G), i.e.,
5. In- Neighbor Query-: For a vertex u belongs to V(G) v2 belongs to V(G) is an in-neighbor of u if (v2, u) belongs to E(G).
6. Out- Neighbor Query-: For a vertex u belongs to V(G) v2 belongs to V(G) is an out neighbor of u if (u, v1) belongs to E(G).

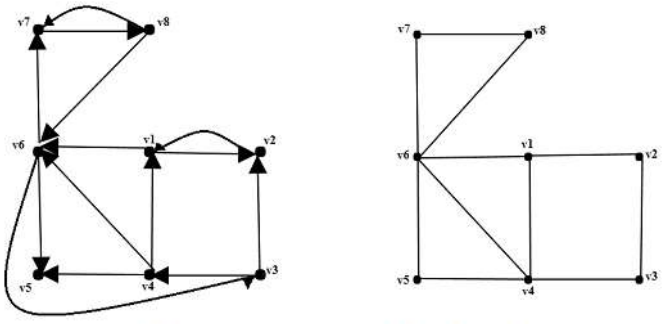

Figure 1: A directed graph G and its underlying undirected graph G'

FORMULATION OF PROBLEM AND ITS SOLUTION

A directed network model is given and its MP1 linearization is required. For a sequence S, the S-distance between u and v, denoted by S-dist(u,v), is the minimum norm-1 distance among all pairs of appearances of u and v. An Mpk linearization of a Graph G is a sequence S of vertices of the graph with possible replication, such that S covers G and for all (u, v) belongs to E(G), S- dist(u,v)<= k. The length of an Mpk linearization is equa to length of S. Eulerian Data Structure stores MPk linearization L of G using an array of same length as L. Two piece of information are kept. First, local information of two bits specifying if edges (V(i-1), v(i) and (V(i), V(i-1)) belongs to E(G) respectively. Second, a pointer to the next appearance of v(i). If this is the last appearance of v(i), then the pointer points to the first appearance of the vertex.

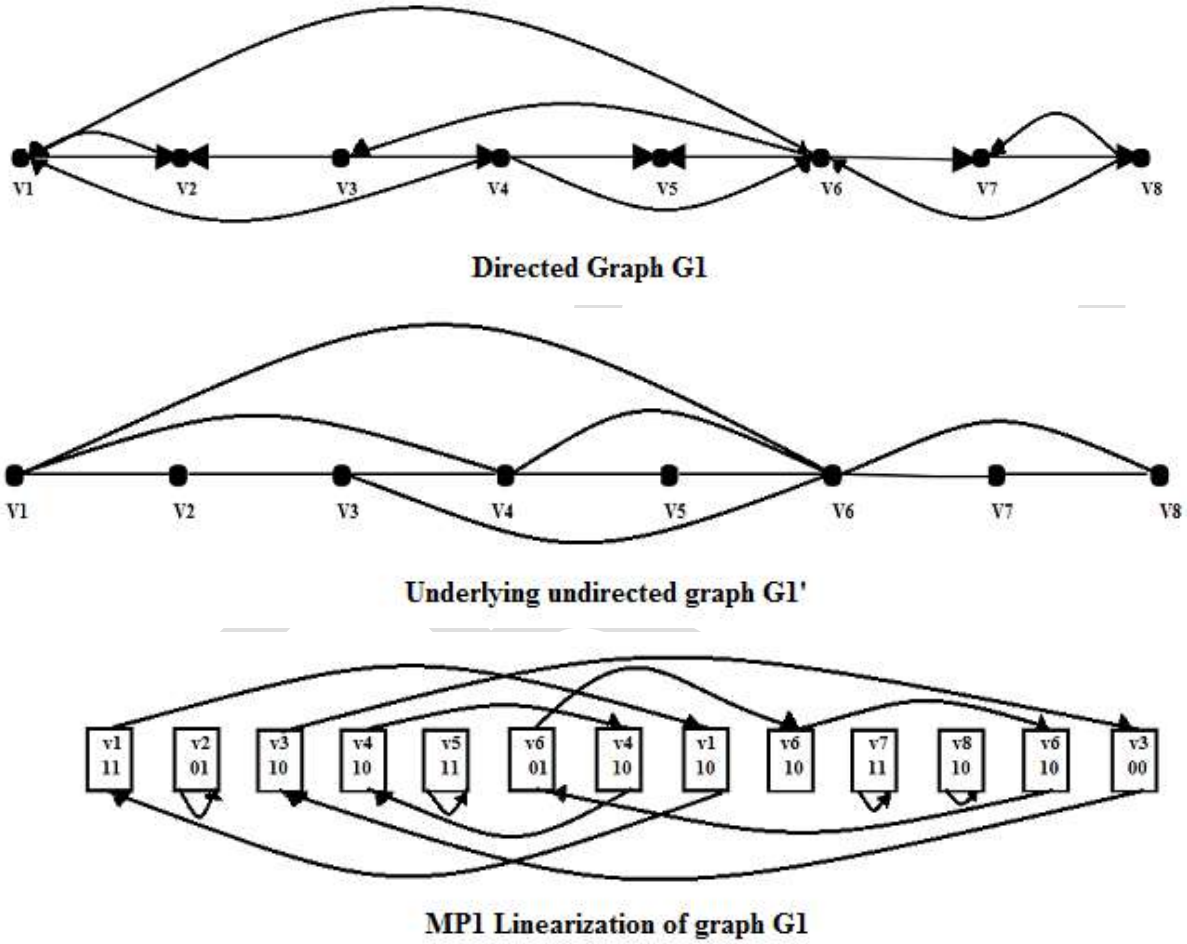

Figure 2: MP1 linearization of the graph G presented in Figure 1

The Eulerian data structure of the above shown graph G1 using an MP1 linearization is illustrated. Here we show the pointers by arcs. Since the length of the linearization is 18, we need log2(13) = 4 bits to encode each pointer. Therefore, for each position we need 4+2(local information) bits. In total we need 13×(4+2)=78 bits, which make a compression rate of 78/13 ≈ 6 bits per edge. As the value of k is increased from 1 to 2 there is a significant decrease in required bits per edge is seen. But after a certain value inefficiency in compression rate is noticed.

MP1 linearization algorithm-:Both graph G and its transpose is required to be stored in order to answer neighborhood queries such as in and out neighbor queries but Eulerian data structure stores MP1 linearization L of G.

- Start from an odd degree node, if there is no such node, start from an arbitrary node.
- Choose an edge whose deletion does not disconnect the graph, unless there is no such choice left.
- Move across the edge and remove it.
- Keep removing edges until getting to a node that does not have any remaining edge to choose.
- If the graph is not empty go to step 1.

This algorithm partitions the edges to exactly Nodd/2 edge-disjoint paths, where Nodd is the number of vertices with odd degree (assuming Nodd>0). It can be implemented in O(|E|).

OPTIMIZATION

A greedy approach for Mpk linearization is used, since it gives the optimized version.

MPk linearization greedy algorithm [10]:

- Start with a random vertex.
- At each step, add to the list the vertex having the largest number of edges with the last k nodes in the list.
- Remove these edges from the graph
- Iterate until no edge is left.
- If none of the last k vertices in the list have a neighbor, and graph is not empty go to step 1.

This requires 2k bits to encode the local information for each position. Having a fixed k all the time is not a good idea since the rear part of the linearization may have very few new edges to encode.

To be adaptive, a relaxed version of the linearization notion is used. For start, a relatively large value of k is taken. Average local density is considered for the graph node positions. If it drops down below a predefined density threshold DT, k is to be reduced by multiplying it to another predefined factor known as reducing factor RF. As the edges are removed, graph become sparse, now the value of k can be decreased so that local information in Euler data structure decreases.

Peter-hamster dataset is used for testing our algorithm for different k values. That graph was having around 22,900 edges with 2400 vertices. and Fre(G) = 0.55 For different k values we obtain different compression ratios.

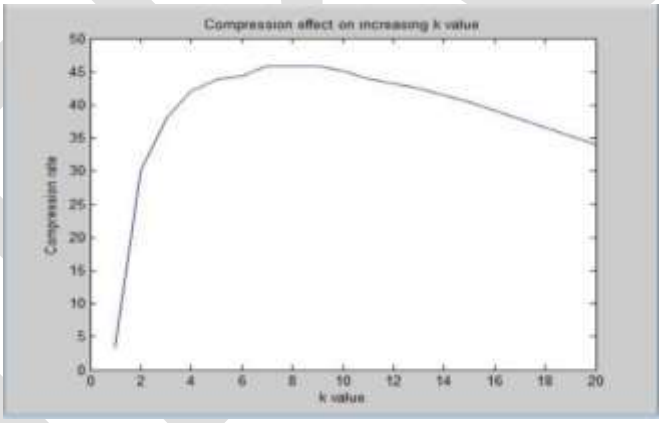

Figure 3: Relation between Compression rate and k value for Peter-hamster dataset

Here we can see as we increase the value of k compression ratio increases but after a certain value it start decreasing. Here maximum compression in on k=8 around 46.2% without using heuristic.

Name	Description	V(G)	E(G)	Fre(G)
Peter-hamster	Social network datasets of peter-hamster	2422	22962	0.55

Figure 4: Details of Peter-hamster dataset

For a large sparse graph where each vertex has the same out degree, and their destinations are picked randomly, increasing k would not influence the length of the linearization significantly. However, for a large random dense graph G where the existence of an edge from every node to another is independently determined by a probability of 50%, increasing k up to |V (G)|, the number of vertices, is actually beneficial.

While considering different datasets k = 5 to 10 proved to be a good value.

APPLICATION

Neighborhood queries are most essential operations on any network. Various other operations can be built using neighborhood queries. One such operation is outlier detection. Outlier detection is the observation of the deviation from a set measurement such that a suspicion is aroused that it does not fit into a certain criteria. A technique for outlier detection is, by using k-nearest neighbor graph. Moreover, an effective way for a k-nearest neighbor graph to be built for a large sparse network is, by using the Eulerian data structure and multiposition linearization. Another operation is community finding, where nodes belonging to similar profiles are clustered. This technique is useful for various other operations as well such as pattern mining.

CONCLUSION

The web graph compression using Eulerian data structure is an efficient way to compress the graph network which can be queried without decompressing the graph. There are various factors affecting the efficiency of this technique, such as value of k, compression ratio and structure of graph. It is observed that $5 \leq k \leq 10$, provides a good range for efficient compression.

REFERENCES:

[1] Hossein Maserrat and Jian Pei, In ACM-KDD Cup: 2010, "Neighbor Query Friendly Compression of Social Networks".
[2] Jérôme Kunegis, "KONECT - the Koblenz Network Collection". Online konect.uni-koblenz.de, 2013.
[3] P. Boldi, M. Santini, and S. Vigna. Permuting web graphs, Berlin, Heidelberg, 2009, In Proceedings of the 6th International Workshop on Algorithms and Models for the Web-Graph (WAW'09).
[4] M. Adler and M. Mitzenmacher, In Data compression conference: 2001, "Towards compressing web graphs".
[5] K. H. Randall et al., In DCC :2002, "The link database: Fast access to graphs of the web".
[6] P. Boldi and S. Vigna, WWW :2004, "The web-graph framework I: compression techniques".
[7]] P. Boldi and S. Vigna, In Data Compression Conference :2004, "The web- graph framework II: Codes for the world-wide web".
[8] G. Buehrer and K. Chellapilla. In WSDM :2008, "A scalable pattern mining approach to web graph compression with communities".
[9] Fang Zhou, In Data compression conference :2009, "Graph Compression".
[10] F. Chierichetti et al. , ACM-KDD Cup, 2009, "On compressing social networks",
[11] Torsten Suel and Jun Yuan, In proc. IEEE Data Compresson Conference 2001, "Compressing the Graph structure of the Web".
[12] Susana Ladra González, In Tesis Doctoral, "Algorithms and compressed data structures for information retrieval".
[13] Sriram Raghavan and Hector Garcia-Molin, In Proc. Of the IEEE Intl. Conference on Data Engineering, 2003. "Representing Web graphs".
[14] John M. Kleinberg, In journal of the ACM, 46(5):604-632, September 1999, " Authoritative sources in hyperlinked environment".