International Journal of Engineering Research and General Science Volume 3, Issue 2, March-April, 2015
ISSN 2091-2730

GMINESYS: Efficient approach to reduce storage area using LZW compression

Ulka A. Panchal, Prof. Vina M. Lomte

Student M.E. (Computer Engineering) Second Year, ulkaapare@gmail.com, 9881911755

Abstract—Now days there are several graph visualization tools exist to represent the graph. However, they are not able to handle large graphs, and/or they do not allow interaction. We are interested on large graphs, with hundreds of thousands of nodes. Such graphs bring two challenges: the first one is that any straightforward interactive manipulation will be prohibitively slow. The second one is sensory overload: even if we could plot and replot the graph quickly, the user would be overwhelmed with the vast volume of information because the screen would be too cluttered as nodes and edges overlap each other. Our GMine system addresses both these issues, by using summarization and multi-resolution. GMine offers multi-resolution graph exploration by partitioning a given graph into a hierarchy of communities-within-communities and storing it into a novel R-tree-like structure which we name G-Tree. GMine offers summarization by implementing an innovative subgraph extraction algorithm and then visualizing its output. Storage and processing of large data of the large graph is an significant challenge so by using LZW compression technique we can reduce the storage area required to store the huge data of the sparse graph without occurring the data loss.

Keywords— Large Graph, Gmine System, Graph Tree, Graph Representation, Graph Partitioning, LZW compression and LZW decompression.

INTRODUCTION:

An important support for graph exploration is interactive visualization, which can help to quickly identify the main components of a graph, its outliers, the most important edges and communities of related nodes. Large graphs can be found in numerous real-life settings: web graphs, computer communication, recommendation, bipartite graphs of web-logs of who visits what page; blogs and similar [1]. At this magnitude, efficient graph visualization becomes prohibitive because of the excessive processing power requirements that prevent interaction. Besides that, hundred-thousand-node drawings result in unintelligible cluttered images that do not aid the user's cognition. To face these challenges we present a system that explores two new ideas to address scalability in large graph visualization [5]. Thus here we put two ideas one is to represent large graph using Gmine system and then second to reduce the storage area required to store the huge data. The rest of this paper is structured as follows. Section 2 introduces the DBLP dataset that will be used along this work. Section 3 describes our graph representation in Gmine system idea and section 4 illustrates LZW compression algorithm to reduce the storage area. Section 5 concludes the work.

2. DBLP DATASET:

Throughout this text we employ the DBLP dataset to illustrate the functionalities of our system. This dataset originates from the Digital Bibliography & Library Project (or DBLP). DBLP is a publicly available database of publication data that embraces authors (also co-authors) from the Computer Science community [7], Titles of the book and their published works. The version of DBLP dataset that we use defines a graph with n = 315,688 nodes and e = 1,659,853 edges, where each node represents an author or publication or title and each edge denotes a relationship between the authors or titles or publications.

3. GRAPH REPRESENTATIONS IN GMINE SYSTEM:

3.1 Construction of the GraphTree:

For this work, initially we need to recursively and hierarchically partition a given graph. For the partitioning task we adopted METIS k-way partitioning [1]. The choice for a specific graph partitioning is independent of the Graph-Tree methodology [15]. Hence, given a graph, we perform a sequence of recursive partitioning to achieve a hierarchy of communities within-communities. At each recursion, each partition is submitted to a new partitioning cycle that will create another set of partitions [20]. This process repeats until we get the desired granularity for the partitions (communities).

3.2 Visualization and Interaction:

We propose an innovative interactive presentation for large graphs. For this purpose, our system promotes the navigation across the levels of the tree that represents the partitioning of a large graph which is show in the figure 1. As the user interacts with the visualization, the system keeps track of the connectivity among communities of nodes at different levels of the partitioned graph [17]. When the user changes the focus position on the tree structure, the system works on demand to calculate and present contextual information [10]. Also system provides different graph representation layout view to view the large graph in an different angle which are Spring layout, KK layout, FR layout, ISOM layout and circle layout. The system provides zoom facility with increase and decrease form to zoom the graph to see a portion or whole large graph in one view clearly and detail. Node information is also get displayed when click on the node in the Node Information text box.

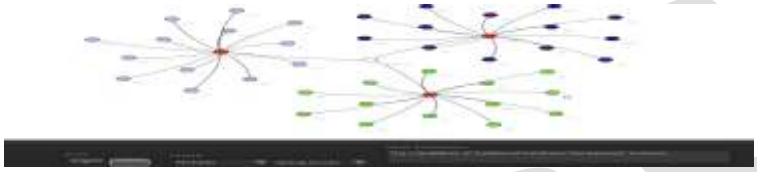

Figure3.1Large graph representation using Gmine system

3.3 Graph Search:

The system provides search facility to search a graph according to Author or Title or Publication by entering author name/ publication name/ title name whatever selected user can search the particular graph and view it**.**

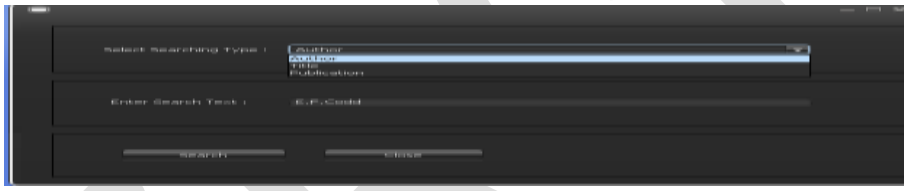

Figure 3.2 Search Graph

4. **LZW COMPRESSION AND DECOMPRESSION TECHNIQUE**:

LZW is a data compression and decompression method that takes advantage of this repetition. LZW compression technique converts [27] a big file into a smaller file using a table-based lookup algorithm invented by Abraham Lempel, Jacob Ziv, and Terry Welch. A particular LZW compression algorithm takes each input sequence of bits of a given length (for example, 12 bits) and creates an entry in a table (sometimes called a "dictionary" or "codebook") for that particular bit pattern, consisting of the pattern itself and a shorter code and for decompression it reverse the process and get the original data back [27].

4.1 LZW Compression Algorithm:

1. Initialize the existing dictionary

2. Take the input character say s.

3. Check s is exist in the dictionary

4. If exist then

{take next input character say c

Check s+c exist in the dictionary

If s+c exist in the dictionary then s= s+c

else add s+c in the dictionary with new code

}
else add s in the dictionary with a new code

5. Repeat this process until end of the file =0

4.2 LZW Decompression Algorithm:

1. Initialize the existing dictionary with code and its appropriate characters.

2. Take the input code

3. Initially input string is empty

{Take the first input code

Write the character represented for the first input code in the dictionary}

4. Take the next input code

5. Write the character for next input code

6. Add the previous code character and first character in the next input code character.

7. Print it in the output string.

8. Assign a new code for {(previous code character)+(next input code character)}

9. Repeat this process until EoF=0

LZW Compression and Decompression Result:

Now take the input string is "ABABBABCABABBA", and apply the LZW compression algorithm which gives output codes as 1 2 4 5 2 3 4 6 1. Instead of sending 14 characters, only 9 codes are here. After decompression of these codes we get apparently, the output string is "ABABBABCABABBA", a truly lossless result! [27].

Compression Result Table:

Result Table1.1.For LZW Compression

S	C	Output	Code	String
			1	A
			2	B
			3	C
A	B	1	4	AB
B	A	2	5	BA
A	B			
AB	B	4	6	ABB
B	A			
BA	B	5	7	BAB
A	C	2	8	BC
C	A	3	9	CA
A	B			
AB	A	4	10	ABA
A	B			
AB	B			
ABB	A	6	11	ABBA
A	EOF	1		

Result Table 1.2 for LZW Decompression

S	K	Entry/Output	Code	String
			1	A
			2	B
			3	C
Nil	1	A		
A	2	B	4	AB
B	4	AB	5	BA
AB	5	BA	6	ABB
BA	2	B	7	BAB
B	3	C	8	BC
C	4	AB	9	CA
AB	6	ABB	10	ABA
ABB	1	A	11	ABBA
A	EOF			

4.3. Result:

The result of the proposed system to reduce the storage area required to store the huge graph data is given as below. This graph shows the memory size comparison for the existing and proposed system which results the memory size of proposed system used is less than the existing system required.

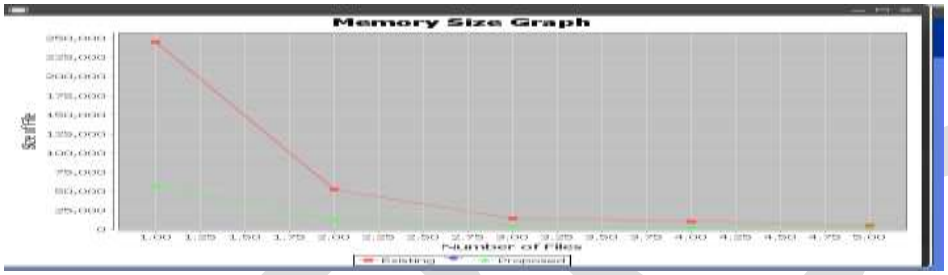

Figure 4.1 Comparison Graph for memory Size

Figure 4.2 memory size Result of proposed system.

This is the output result of the application which shows the data analysis for the selected dblp data file in memory size.

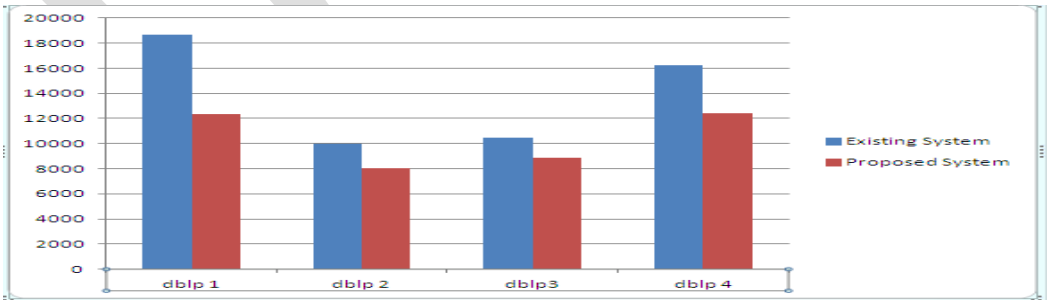

Figure 4.3 Memory size comparison for different dblp files

This is the column graph for the comparison in the proposed system and existing system for memory size required for the dblp file for graph representation.

5. CONCLUSION AND FUTURE SCOPE:

We have demonstrated a system that supports the visualization of large graphs in an interactive environment with different layouts and storage area can be reduced by applying LZW compression and decompression techniques. Also any graph can be searched and viewed clearly and neatly. In future we can apply the both algorithms on different pictures images and calculates the final results for very large and complex graphs.

REFERENCES:

[1] "Large Graph Analysis in the GMine System" Jose F. Rodrigues Jr., Member, IEEE, Hanghang Tong, Member, IEEE, Jia-Yu Pan, Agma J.M. Traina, Senior Member, IEEE, Caetano Traina Jr., Senior Member, IEEE, and Christos Faloutsos, Member, IEEE.
[2] Abello, F. van Ham, and N. Krishnan, "Ask-Graphview: A Large Scale Graph Visualization System," IEEE Trans. Visualization and Computer Graphics, vol. 12, no. 5, pp. 669-676, Sept./Oct. 2006.
[3] S.B. Akers, "Binary Decision Diagrams," IEEE Trans. Computers, vol. C-27, no. 6, pp. 509-516, June 1978
[4] D. Archambault, T. Munzner, and D. Auber, "Grouseflocks: Steerable Exploration of Graph Hierarchy Space," IEEE Trans. Visualization and Computer Graphics, vol. 14, no. 4, pp. 900-913, July/Aug. 2008
[5] D. Auber, Y. Chiricota, F. Jourdan, and G. Melanc¸on, "Multiscale Visualization of Small World Networks," Proc. IEEE Ninth Conf. Information Visualization (InfoVis), pp. 75-81, 2003.
[6] V. Batagelj, W. Didimo, G. Liotta, P. Palladino, and M. Patrignani, "Visual Analysis of Large Graphs Using (x,y)-Clustering and Hybrid Visualizations," Proc. IEEE Pacific Visualization Symp. (PacificVis), pp. 209-216, 2010.
[7] A.L. Buchsbaum and J.R. Westbrook, "Maintaining Hierarchical Graph Views," Proc. ACM-SIAM Symp. Discrete Algorithms, pp. 566-575, 2000.
[8] E. Dahlhaus, J. Gustedt, and R.M. McConnell, "Efficient and Practical Algorithms for Sequential Modular Decomposition,"J. Algorithms, vol. 41, pp. 360-387, 2001
[9] B.B. Dalvi, M. Kshirsagar, and S. Sudarshan, "Keyword Search on External Memory Data Graphs," Proc. VLDB Endowment, vol. 1, pp. 1189-1204, 2008.
[10] P. Vaz de Melo, L. Akoglu, C. Faloutsos, and A. Loureiro, "Surprising Patterns for the Call Duration Distribution of Mobile Phone Users," Proc. European Conf. Machine Learning and Knowledge Discovery in Databases (ECML-PKDD), pp. 354-369, 2010[11] P. Eades and Q. Feng, "Multilevel Visualization of Clustered Graphs," Proc. Symp. Graph Drawing, pp. 101-112, 1997.
[12] P. Eades and M.L. Huang, "Navigating Clustered Graphs Using Force-Directed Methods," Graph Algorithms and Applications, vol. 4, pp. 157-181, 2000. [13] C. Faloutsos, K.S. McCurley, and A. Tomkins, "Fast Discovery of Connection Subgraphs," Proc. ACM 10th Int'l Conf. Knowledge Discovery and Data Mining (SIGKDD), pp. 118-127, 2004.
[14] Finocchi, "Hierarchical Decompositions for Visualizing Large Graphs," PhD thesis, Univ. of Rome, 2002.
[15] S. Fortunato, "Community Detection in Graphs," Physics Reports, vol. 486, pp. 75-174, 2010
[16] E.R. Gansner, Y. Koren, and S.C. North, "Topological Fisheye Views for Visualizing Large Graphs," IEEE Trans. Visualization and Computer Graphics, vol. 11, no. 4, pp. 457-468, July/Aug. 2005.
[17] R. Gentilini, C. Piazza, and A. Policriti, "Computing Strongly Connected Components in a Linear Number of Symbolic Steps," Proc. Symp
[18] M. Gomez-Rodriguez, J. Leskovec, and A. Krause, "Inferring Networks of Diffusion and Influence," Proc. 16th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining, pp. 1019-1028, 2010.
[19] H. David, "Statecharts: A Visual Formalism for Complex Systems," Science Computer Programming, vol. 8, pp. 231-274, 1987.
[20] F. van Ham and J.J. van Wijk, "Interactive Visualization of Small World Graphs," Proc. IEEE Symp. Information Visualization (InfoVis), pp. 199-206, 2004.
[21] D. Harel and Y. Koren, "Graph Drawing by High-Dimensional Embedding," Proc. Revised Papers from 10th Int'l Symp. Graph Drawing, pp. 207-219, 2002.
[22] M.L. Huang and Q.V. Nguyen, "A Space Efficient Clustered Visualization of Large Graphs," Proc. Fourth Int'l Conf. Image and Graphics, pp. 920-927, 2007.
[23] D.J. Watts, Small Worlds: The Dynamics of Networks between Order and Randomness. Princeton Univ. Press, 2003.
[24] G. Karypis and V. Kumar, "Multilevel Graph Partitioning Schemes," Proc. IEEE/ACM Conf. Parallel Processing, pp. 113-122, 1995.
[25] J.F. Rodrigues Jr., A.J.M. Traina, C. Faloutsos, and C. Traina Jr., "Supergraph Visualization," Proc. IEEE Eighth Int'l Symp. Multi-media (ISM), pp. 227-234, 2006.
[26] L. Page, S. Brin, R. Motwani, and T. Winograd, "The Pagerank Citation Ranking: Bringing Order to the Web," technical report, Stanford, 1998. [27] "Simrandeep kaur, V.Sulochana Verma "Design and Implementation of LZW Data Compression Algorithm" International Journal of Information Sciences and Techniques (IJIST) Vol.2, No.4, July 2012