Contemporary Engineering Sciences, Vol. 7, 2014, no. 16, 791 – 796
HIKARI Ltd, www.m-hikari.com
http://dx.doi.org/10.12988/ces.2014.4692

An Efficient Algorithm for MDL Based Graph Summarization for Dense Graphs

Kifayat Ullah Khan, Waqas Nawaz and Young-Koo Lee[1]

Department of Computer Engineering
Kyung Hee University, Republic of Korea

Abstract

In-memory visualization and analysis of graphs is hard if they cannot be fit in the memory. For this purpose, creating a summary graph to understand their insights is very useful. In this paper, we present an efficient algorithm for MDL based graph summarization to compress big graphs. We have evaluated the performance of the proposed algorithm on a real graph and observe better execution time than the state of the art, at similar accuracy.

Keywords: Dense Graphs, Graph Summarization, MDL, Randomized algorithm

1 Introduction

These days the graphs are dense and big; thus they cannot fit in the main memory for better analysis and visualization. Graph Summarization is a useful technique to compress these graphs into a memory sized summary graph [1, 4]. In this technique, the nodes sharing common neighbors are merged into a super node and

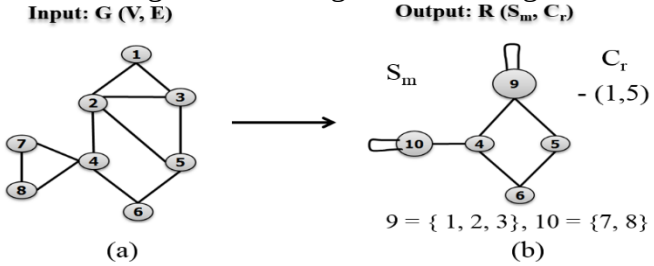

Fig 1. MDL based Graph Summary with Edge Corrections

[1] Corresponding author

and their corresponding edges are merged into a single super edge. This greatly reduces the size of the graph. The summary graph is small in size yet it maintains the charactertics of the underlying graph. This enables the analysts to quickly understand the various real life phenomenon with the little loss of the accuracy due to compression.

Navlakha et.al [1] proposed a lossless graph summarization technique to compress the graphs. They used the Minimum Description Length (MDL) principle [2] to create the summary graph. MDL is a principle from information theory that states that the best theory to represent the data is the one whose sum of the size of the theory and the size of the associated data is minimal. In case of the graph summarization, the theory is the summary graph and the associated data is the list of edge corrections. The edge corrections are maintained to recover the original graph, if required.

The authors in [1] proposed the Randomized algorithm to compute the MDL based summary. In Randomized, the nodes giving best cost reduction (section II) are merged to create the super node and their common edges are merged into super edges. Randomized employs a bottom-up strategy to compute the summary graph. We observe that this strategy is not optimal when the graphs are dense and big.

In this paper, we present the MDL based graph summarization for big graphs. We propose an efficient algorithm, Randomized++, to compute the summary graph. We have evaluated the proposed algorithm on three real graphs. The experimental results show that the proposed algorithm outperforms the existing one on dense graphs and produces the summary of similar accuracy.

The rest of the paper is organized as follows: we present the background of MDL based summary and existing algorithm in section II. Section III presents the proposed algorithm. In section IV, we provide the experimental results. Finally, we conclude the paper and provide the future work in section V.

2 Background

In this section, we provide the concept of MDL based summary and detail the existing Randomized algorithm [1] along with its limitations. These limitations are the motivation for our work in this paper.

2.1 MDL based Summary

The MDL based summary graph for a graph $G = (V, E)$ can be formally represented as $M = (S_r, E_r)$ where S_r is the summary graph and E_r are the list of edge corrections. S_r consists of (V_s, E_s) where V_s is the set of super nodes and E_s are the super edges between the super nodes. E_r are the positive and negative edge corrections. The positive edge corrections indicate the actual edges missing in the summary due to the super edges. On the other hand, the negative edge corrections are the edges which actually does not exist in G but are created due to the super edges. For instance, for the graph in Fig 1. (a), the Fig 1. (b) is the summary graph. In Fig 1. (b), the set of nodes $\{1, 2, 3\}$ and $\{7, 8\}$ are merged into the super

nodes 9 and 10 respectively. A self-edge for the super node 9 shows nodes 1 and 5 as connected which is a mistake. This mistake is avoided by adding a negative edge correction $-(1,5)$.

MDL based summary ensures that the summary should be the best representative of the underlying graph. For this purpose, the nodes which share the maximum number of common neighbors, are merged. The merger of such nodes produce the super nodes with least edge corrections. Before merging any two nodes, we compute their compression or reduction cost. This cost can be formally defined as "*the ratio of the reduction in cost as a result of merging u and v (into a new supernode w), and the combined cost of u and v before the merger*" [1]. Equation (1) shows how to compute the cost *c* of any two nodes. For instance, for the nodes 7 and 8 in Fig 1. (a), the cost is 0.5 since they share their only neighbor.

$$c(u,v) = (c_u + c_v - c_w)/(c_u + c_v) \quad (1)$$

The cost of MDL based summary is the sum of the edge corrections and the super edges.

2.2 Randomized, the existing algorithm

It is a randomized merging algorithm to efficiently compute the summary graph. In randomized, a node is randomly selected from the entire graph and is merged with the best node in its 2-hops neighborhood.

The super node creation policy in Randomized is iterative. Randomized divides the nodes into finished and unfinished nodes. Finished nodes represent a set of the nodes which cannot be further merged with any of the nodes in the graph. Initially all the nodes are considered as unfinished nodes. In each step of iterative merging, we randomly choose a node u from the unfinished nodes and compute the cost reduction against all its 2-hops away neighbors. Any node v giving the highest cost reduction with u in all its 2-hops neighborhood, is merged with u to create the super node w. After creating the super node, the nodes u and v are moved from unfinished to finished category. If u does not give positive cost reduction with any of the node, then it is confirmed that it cannot be considered further so it is moved to the finished category. This process is repeated until all the nodes from unfinished are marked finished. At this moment, we generate the summary graph and the edge corrections.

2.3 Limitations of the Existing Algorithm

We observe that the computational model of the Randomized algorithm does not enable it to scale well to the dense and big graphs. In case of the big graphs, the connectivity between the nodes is high which increases their degrees. An increase in the degree of each node, also increases the count of its 2-hops neighbors. When the Randomized is applied to such graphs, it consumes the significant time in computing the cost reduction against each 2-hops away neighbor. When the cost reduc-

tion is computed against all such neighbors by performing large number of computations, only two nodes are merged to create a super node. This style of super node creation makes the Randomized least scalable to the dense and the big graphs.

3. Randomized++, The Proposed Algorithm

In this section, we present the proposed algorithm, Randomized++ to compute the MDL based summary for the dense and the big graphs. Randomized++ reduce the number of cost reduction computations in the Randomized algorithm and to make it scalable to the dense and the big graphs.

In Randomized++, we classify the nodes of the input graph G into finished and unfinished nodes like Randomized. We then randomly choose a node u from the unfinished category and retrieve all its 2-hops away neighbors. We then compute the cost reduction against each such neighbor and maintain the cost against each in the form of a set. It is worthwhile to consider that we are focusing the dense graphs, so the count of 2-hops neighbors is large. Thus an optimal super node creation strategy is required that makes highly use of the time spent to compute the cost reductions against all the 2-hops neighbors. For this purpose, we sort all the neighbors against u with respect to their cost reductions. This sort brings the nodes, giving the highest cost reduction, to the top of the list. We then merge the nodes in the set iteratively till negative cost reduction occurs. At this stage, we perform no more merger and release the rest of the nodes in the set so that they can be considered in upcoming iterations. This computation model of the Randomized++, makes it suitable for the big graphs since it makes use of each computation performed.

4. Experiments

In this section, we present the evaluation analysis for execution time and summary cost. We have performed experiments on 3 real world network, Table 1.

	Road Network	Citation Network	DBLP
|V|	19,65,206	34,546	3,17,080
|E|	55,33,214	4,21,578	10,49,866
Density (|E|/|V|)	2.81	12.2	3.31

Table 1: Dataset Information

4.1 Execution Time Evaluation

The objective of execution time evaluation is to show how the two algorithms behave on dense and sparse graphs. Recall that in dense graphs, the connectivity between the nodes is high which increases the degree of each node in the graph and vice versa for sparse graphs. Figure 2 shows the execution time comparison. On the road network, having the lowest density, we observe that the existing Randomized performs slightly better than our proposed Randomized++. The reason is that degree of the nodes is small so it does not perform much computations to find the best

node against the query node. Hence the performance is better. On the other hand, in citation network and DBLP where the degree is high, Randomized suffers from its computation model. A number of cost reduction computations of Randomized are wasted because it has to merge the best node with the query node. The Randomized++, makes use of each computation performed and merges the query node with all of its 2-hops away neighbors who give positive cost reduction.

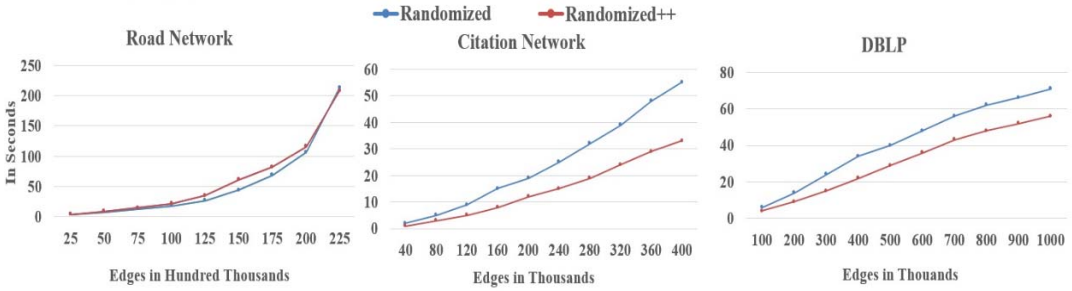

Figure 2. Execution Time Comparison

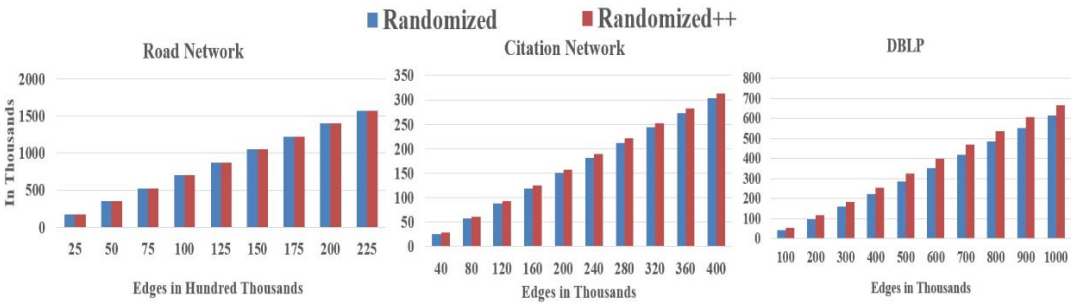

Figure 3. Cost Comparison

4.2 Cost Comparison

In Figure 3, we compare the cost of the two algorithms on the three datasets we used. Recall that cost is composed of the positive, negative edge corrections and the super edges in summary graph. We observe that both the algorithms show equal cost in the road network. On the other hand, in citation network and DBLP, the Randomized++ generates the summary of slight higher cost. The reason is that in our case, the size of the super node is large because we keep on merging all the 2-hops away neighbors of the query node till negative cost reduction. But the Randomized, merges only the best node with the query node that produces minimal edge corrections.

5. Conclusion

In this paper, we have presented the MDL based graph summarization. MDL is a concept from information theory that helps to create a highly compressed representation of the big graphs. We have proposed an efficient algorithm to compute the MDL based summary. The proposed algorithm overcomes the weakness of the existing algorithm on dense graphs, hence show better execution time.

Acknowledgements.
This research was supported by the MSIP (Ministry of Science, ICT & Future Planning), Korea, under the ITRC (Information Technology Research Center) support program (NIPA-2014(H0301-14-1020)) supervised by the NIPA (National IT Indus-try Promotion Agency).

References

[1] Navlakha, Saket, Rajeev Rastogi, and Nisheeth Ackley, Graph summarization with bounded error, *SIGMOD*, (2008), 419-432.
[2] J. Rissanen, Modelling by shortest data description, *Automatica*, (1978), 465–471.
[3] http://snap.stanford.edu/data/index.html
[4] Tian, Yuanyuan Richard A. Hankins, and Jignesh M. Patel, Efficient aggregation for graph summarization, *SIGMOD*, (2008), 567-580.

Received: May 1, 2014