EAI Endorsed Transactions
on Scalable Information Systems
Research Article EAI.EU

g-Sum: A Graph Summarization Approach for a Single Large Social Network

Saif ur Rehman[1,*], Asif Nawaz[1], Tariq Ali[1] and Naveed Amin[1]

[1]University Institute of Information Technology, PMAS Arid Agriculture University, Rawalpindi, Pakistan

Abstract

With the advances in computing resources, the processing of huge amount of data becomes possible. However, the human ability to identify patterns from such data has not scaled accordingly. Consequently, efficient computational approaches for condensing and simplifying data are becoming essential for uncovering actionable insights, especially the big social networks. Many large datasets analyzed today are represented with the help of graphs. In many real-world applications, summarizing large graphs is beneficial to achieve a number of advantages, including 1) significant speedup for graph algorithms, 2) graph storage space reduction, 3) faster network transmission, 4) improved data privacy, 5) more effective graph visualization, etc. All these benefits can be obtained from the summarized graph, if it is summarized accurately. The quality of the summary graph is mostly measured using the Reconstruction Error (RE). In the existing literature, RE is a very challenging task. Most of the approaches investigated so far ending up with high value of RE. Hence, the summarized graph does not express the original graph completely due to the high value of the RE. Therefore, in this work we have examined how can we summarize a big graph structure into a compact summary by reducing its RE and ensuring its usefulness. For this task, we have proposed a novel graph summarization approach, called g-Sum, to calculate the graph structure summaries while minimizing RE and compare to the existing work in the domain. The functionality of the g-Sum is decomposed into three different, but interlinked phases; (1)- graph dataset pre-processing; (2)- graph partitioning; and (3)- graph summarization. We have performed an extensive series of experiments on different real-world social graph dataset, including Ego-Facebook, Enron Email and Web-Stanford graph datasets. The computed results show the effectiveness of the g-Sum and also compared with the state-of-the-art graph summarization approaches, S2L and GraSS.

Keywords: Social Network, Social Graph, Graph Data, Graph Mining, Graph Summarization, Graph Partition, Reconstruction Error, Big Graph.

Received on 12 November 2020, accepted on 04 March 2021, published on 23 March 2021.

doi: 10.4108/eai.23-3-2021.169073

1. Introduction

The present day world has witnessed a dramatic change in the access to the Internet and its usage. In the era of the connected world, our social and digital lives are confronted with the networks (or simple graphs) on a daily basis [1]. Graph-based representation of real world problem has been proved to be very beneficial in finding solutions to the problems [2-3]. Graphs are generated from almost every field of today's life; internet browsing means traversing a big network of web pages that is interlinked via clickable (or sometimes hyper) links [4]; online social networks such as Facebook are based on massive networks, in which different people are connected through so-called friendship links (a graph of friends) [5-6]; using mobile accessing one webpage create a few dozen wired or wireless connections between devices in a matter of microseconds [7]. Therefore, networks can be found everywhere around us, and these influenced the system in which we disseminate, search, socialize, navigate and absorb the information.

As technology advances, ability to collect and archive huge amount of generating data also improved. Daily activities like social media interaction, web browsing, product / service, purchasing, itineraries, and wellness sensors generate large amounts of data. The analysis of such data can immediately impact our lives. This abundance of generating data and its velocity call for data summarization which is one of the main data mining tasks.

Graph Summarization (GS) is the process of reducing the size of the original graph. Such that different operations (e.g. Querying, analysis of a summary, etc.) can be performed more efficiently. GS produces an approximate and precise version of original graph where communities are easily identified in a network. In addition, GS uncovers important / interesting information from the original graph as well. For example, Figure 1 (b) shows a summary graph of the original graph in Figure 1 (a). Obviously, it is much easier to

*Corresponding author. Email: saifi.ur.rehman@gmail.com

understand and analyse the graph in (b). For large graphs, the advantage is much more apparent.

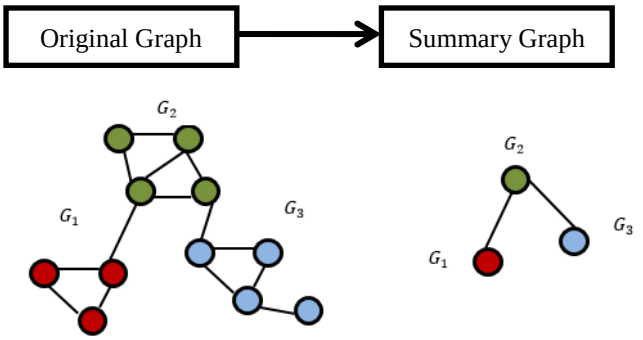


Figure 1. An example of graph summarization (a) original graph; (b) graph summarization

The graph of real-world applications is very large and it is difficult to understand the information stored in a large graph by mere visual inspection. Therefore, effective methods of GS are required to help users to mine and understand the essential information. GS also speeds up the analysis by creating a lossy concise representation of the graph that is loaded into main memory. The graph summarization techniques should allow users to select the attributes and relationships of their interest, and then make use of the selected attributes with corresponding relationship to produce small and informative summaries. Moreover, the users should be able to easily control the resolution of the resultant summaries also [7]. The GS has different benefits [8]:

(i) It helps to reduce the input/output operations and allow the summary graph to easily load into memory.
(ii) The summary of original graph can be understood, analyzed, and queried more easily with existing tools and algorithms.
(iii) The summary graph helps to visualize datasets that are originally huge to load into memory.
(iv) Summarization also helps to remove noise and discover patterns in the data [9][19-21].

A Social Network (SN) is a collection of entities (which may be individuals or organizations). SN is a platform that allows different users to share their information, data, resources, and views [1]. Email services, the transmission of the disease, and the criminal activities can be modelled by SN. Social Network Analysis (SNA) is the technique, used to map and measure the associations and flows between the individuals, groups, departments, workstations or other information processing object. SNA is a process of quantitative and qualitative *analysis* of a *social network*. SNA offers both a graphical and statistical analysis of human interactions. The data of social network is stored in the form of graphs which contains millions of billions of nodes and edges. The graphs of SN are very large, and it is very difficult to understand such graphs. So, there is a need of summarizing such graphs into a simple and concise one that can easily be understood. A key challenge with graph summarization approaches is the reconstruction error, which is very high (a high value of RE shows that the summary graph does not represent accurately the original graph). Therefore, to address this critical issue, we pose the following key question: How to convert an original single graph into a compact and summarized graph, without compromising its originality (characteristics) and information insight?

Therefore, in this paper, a new graph summarization algorithm, called the g-Sum, is proposed that works in three phases; (1) - graph pre-processing; (2) - graph partitioning using hierarchical agglomerative clustering approach; (3) - graph summarization. The nodes and edges within the partition are converted into super node and the edges between partitions are represented as super edge in the summary graph. The efficiency of the proposed work is assessed by reconstruction error which is the difference of adjacency matrix A on the original graph and the expected adjacency matrix of a summary graph A_s. Another parameter used to assess summaries is the accuracy in answer to queries about original graph computed from summaries. Furthermore, the proposed graph summarization approach is evaluated using different standard social networks, Ego-Facebook network, Enron email and Web-Stanford network datasets. The experiments showed that the proposed approach achieves good results by minimizing the reconstruction error and obtained a summary graph that contains the same properties in the original graph. Finally, we evaluated the proposed graph summarization algorithm with the existing approaches, GraSS [13] and S2L [14].

1.1. Research Contributions:

The contribution of this study may thus be summarized as follows:

(i) Proposal of a new graph summarization approach, called the g-Sum.
(ii) Investigation on the g-Sum algorithm to highlight its core components, highlighting how to get a summary of the big original graph.
(iii) Experimental evaluation of the g-Sum based on different performance metrics
(iv) Performance comparison of the g-Sum with the existing graph summarization approaches
(v) Analysis and evaluation of the discovered summary graph as obtained from the real-world social network datasets.

The remaining paper is organized as follows. Section 2 highlights the preliminaries on graph theory and graph summarization. The related work on graph summarization approaches has been discussed in Section 3. We present our novel graph summarization algorithm in Section 4, where the fundamental principle of the approach is presented. In Section 5, we report the experimental results obtained from the simulations of the proposed approach on various graph networks. Finally, in Section 6, we draw conclusions with final thoughts for future work.

2. Graph summarization preliminaries

In this section, we present the preliminaries required to facilitate discussion in the rest of the paper. We provide some of the basics of graph theory, graph summarization and its enablers.

Definition 2.1 (Graph): A *graph* is the collection of nodes and edges connecting with each other. A Graph is a pair of $G(V, E)$, where V is the *set of nodes* and E is the *set of edges*, formed by pairs of vertices. A sample graph is shown in Figure 2.

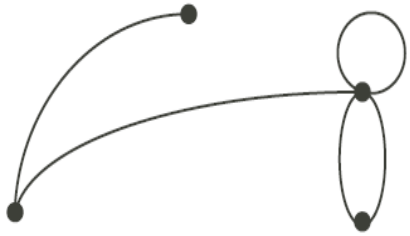


Figure 2. A graph with nodes and vertices

Definition 2.2 (Undirected Graph) : An undirected graph is a pair $G = (V, E)$, where V is a set of vertices (or nodes), and $E \subseteq \binom{V}{2}$ is a set of edges. In an undirected graph, each edge is an unordered pair $e = \{u, v\}$ (or equivalently$e = \{v, u\}$). By this definition, an undirected graph cannot have self-loops since $\{v, u\} = \{v\} \notin \binom{V}{2}$. In Figure 3, there are 4 vertices $\{v_1, v_2, \ldots\ldots, v_4\}$ and edges $\{e_1, e_2, \ldots\ldots, e_4\}$.

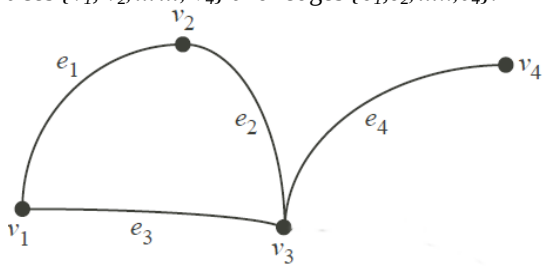


Figure 3. A simple graph with nodes and edges

Definition 2.3 (Directed Graph) : A directed graph or (digraph) is a pair $G = (V, E)$ where V is a set of vertices (or nodes), and $A \subseteq V \times V$ is a set of directed edges (or arcs). In a directed, each arc is an ordered pair $e = (u, v)$. A directed graph can have self-loops (u, v)[13]. Figure 4 shows a directed graph structure.

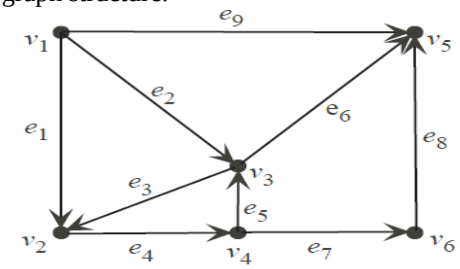
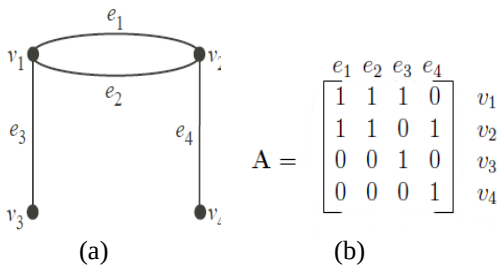


Figure 4. A directed graph with 6 nodes and 9 edges

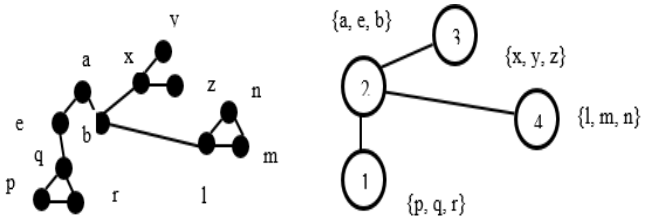
Definition 2.4 (Adjacency Matrix) : Adjacency Matrix is an array of size $V \times V$ where V is the number of vertices in a graph. Let the array be $adj[\ ][\ ]$, a slot $adj[i][j] = 1$ indicates that there is an edge from vertex i to vertex j. Adjacency matrix for undirected graph is always symmetric. Adjacency Matrix is also used to represent weighted graphs [30]. Figure 5 (b) shows the adjacency matrix of a given graph Figure 5 (a).



(a) (b)

Figure 5. Adjacency Matrix (a) of a graph in (b)

Definition 2.5 (Summary Graph) : We can formally define summary graphs as follows: Given a graph $G = (V,E)$, and a partition of V, $\Phi = \{G1, G2, \ldots, Gk\}$ $(U_{i=1}^{K}\, Gi = V$ and $\forall i \neq j\ Gi \cap Gj = \emptyset)$, the *summary graph* based on Φ is $G_s = (V_s, E_s)$, where $V_S = \Phi$, and $E_s = \{(Gi, Gj) | \exists u \in Gi, v \in Gj, (u, v) \in E\}$.



(a) Original Graph G (b) Summary Graph G_S

Figure 6. Summary graph $G_S$of Original Graph G

In Figure 6 (b) the summary graph G_S of an original graph G is shown. In a summary graph, each node, called a *subgraph*, corresponds to one subgraph in the partition of the original node set, and each edge, called a subgraph relationship, represents the connections between two corresponding sets of nodes. A graph relationship between two subgraphs exists if and only if there exists at least one edge connecting some nodes in the two graphs [22].

Definition 2.6 (Reconstruction Error): Reconstruction error shows the accuracy of summary graph. The reconstruction error is computed by finding the difference of adjacency matrix A of original graph $G(V, E)$ and adjacency matrix A_s of summary graph $S(V_s, E_s)$. The procedure to compute the reconstruction error is given later on in Section 4.

Definition 2.7 (Query Error): When we query a summary graph, then it returns an answer. Query error represents how approximate the answer is to the query or how the answer is different when the same query is sent to the original graph structure. Later on, we have presented the measures used to compute the query error.

3. Related Work

In this section, we have reviewed the different graph summarization approaches and algorithms proposed in the literature. We have highlighted their methodologies, the strengths and weaknesses of each graph summarization technique.

The problem of graph summarization with content linked to the node is proposed in [18]. Based on the topological similarities and attribute similarities, authors have suggested an algorithm for graph summarization, named AGSummary. They used Minimal Description Length (MDL) principle to model the problem of GS into the code cost function. Furthermore, they adopted a neighbourhood greedy algorithm to find an optimal summary of the input graphs. To evaluate the performance of the AGSummary, it is compared with others existing GS algorithms. They used the well-known graph datasets including political books, political blogs and DBLP for experimentation. Different results showed that AGSummary outperformed the existing approaches.

A new method for the summarization of SN is presented in [10]. Their approach used the graph pruning method which removes less important nodes and edges of SN graphs and the summary graph is produced after graph pruning that retains the original graph property. Their technique is based on the *k*-core graph and also reduced the complexity of large scale SN graph. The graph pruning method helps to prune a large number of nodes while leaving the core node intact with the remaining nodes in the graph structure. A scale-free distribution method is used in graph pruning which reduces the graph size in a very fast fashion such as edge deletion and node deletion iteratively to reach a certain size, contracting adjacent nodes and graph induction, etc.

In another study a new framework proposed for graph summarization of given graph G which consist of two parts, graph summary S and set of correction C such that G(S, C). S is the summary graph in which node corresponds to a set of nodes and each edge represents a relation between pairs of edges [11]. The second part of their proposed work is a set of correction C which is edge correction that is helpful to reconstruct original graph G. As claimed in [11], most of the existing work on graph summarization supports lossless compressing. The authors proposed works are highly compact and also allow both lossy and lossless graph compression with the bonded error. In addition to this, they used the MDL principle to produce a coarse level summary of the given input graph G. They also proposed two parameters-less algorithms, first is the greedy algorithms and the second one is the randomized algorithm, which compressed the graph with a size small and high accuracy. Greedy algorithm produced a highly compressed graph by repeatedly picking the best pair of nodes in the entire graph. On the other hand, the randomized algorithm picks randomly selected nodes from the graph to produce a compressed graph and this algorithm is faster than the previous one.

Another work for summarizing a large graph is suggested in [12], focusing on the compressing edge-weighted graphs. Their approach suggested the merging of nodes with similar relationships to other objects (structurally equivalent nodes) in such a way that the compression is maximized and approximation error is minimized. When merging nodes to obtain a compressed graph, the algorithm either supports weights on the edge or the strengths of links to a certain number of hops. In particular, in the simplest version of the solution, each super-edge allocated the average weight from all the edges it represents. In summary, the best path between every two nodes are "approximately equally good" in original graphs and compressed graphs, but the path may be different. The definition of path “goodness” is application and data-dependent.

A formal semantics for answering queries on summaries of graph structures was introduced in GraSS [13], which targets accurate query handling. Their summarization method supports important graph summarization queries such as adjacency, degree, and eigenvector centrality can be answered efficiently and in a closed form using these semantics. The summary graph of original graph is obtained by greedily merging the pair of nodes in such a way that reconstruction error is minimized, A is the original adjacency matrix of the graph and $\bar{A}$ is the real-valued approximate adjacency matrix, each entry of which intuitively represents the probability of the corresponding edge existing in the original graph given the summary. GraSS leverages MDL to automatically find the optimal number of super-nodes in the summary graph.

Later studies showed that the GraSS [13] does not produce a quality summary. Therefore, a new method proposed in [14] that convert nodes and edges into super-nodes and super edges with guarantee quality. The main target of [14] is to produce the graph summary by minimizing the reconstruction error. The main task is to produce supergraph which contains properties similar to the original graph so the no information in the summary graph is missed as in the original graph. The experimental results obtained clearly depict the supremacy of their work when compared to GraSS approach. Their approach uses sampling, sketching and approximate partitioning for generating the summary graph. The summary graph produced is helpful in query answering such as degree centrality and Betweenness etc.

The edge grouping method is another graph summarization technique in which the edges in the original graph are converted into super edges. A graph de-densification technique proposed in [15] to summarize a graph. Graph de-densification is an edge grouping method

that compresses multiple edges in the original graph into few super-edges in the summary graph. Their proposed approach follows the edge grouping that compresses neighbourhood in high degree nodes. In their work, they add a new node in a graph which is called a compressor node in which multiple edges are reduced to fewer edges by passing through the compressor node.

4. g-Sum: A Proposed approach for Graph Summarization

In this section, the proposed graph summarization approach, g-Sum, is discussed, highlighting its main phases. Moreover, the mathematical foundations and a sample working example to elaborate how the algorithm works has been given in the subsequent subsections. Different graph notations used throughout the paper are given in Table 1.

Table 1. Different notations used in this paper

Notation	Description
A	Adjacency Matrix of the original graph
A_S	Adjacency Matrix of the summary graph
G	A Graph or Original Graph
G_D	Graph Dataset
G_S	Graph Summarization/Summary Graph
K	Number of Super-node
MDL	Minimal Description Length
RE	Reconstruction Error
SNs	Social Networks
SNA	Social Network Analysis

The functionality of the g-Sum has been decomposed into three different phases, as shown in Figure 7.

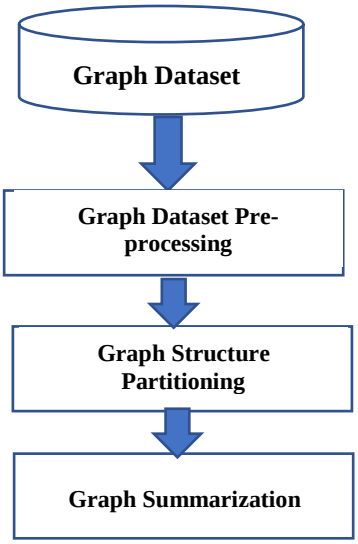


Figure 7. Phases of the proposed graph summarization approach g-Sum

As shown in Figure 7, the proposed g-Sum approach takes graph dataset as an input. In the first phase, graph dataset is pre-processed. In preprocessing, g-Sum removes the ambiguities such as duplicate or less important node from dataset. After the preprocessing, g-Sum partition the graph, where the input graph is converted into the required number of partitions, say *K*-partitions, using hierarchal agglomerative clustering approach; and in the final phase g-Sum produces the summary of the partition graph where each partition is converted super-nodes.

The proposed algorithm for the g-Sum approach is given in Figure 8.

Algorithm: *Proposed Algorithm for Graph Summarization g-Sum(G_D,K, G_S)*

Input: G_D: Graph dataset, *K*: Number of partitions (super-node)
Output: G_s as a summary graph

```
1   Begin
2     Graph pre-processing
3     Generate V × V adjacency matrix of an input graph
      dataset
4     i = count(Number of nodes in the dataset)
5     foreach i ≥ K do
6       bestPair← Max (adjacency matrix value)
7       MergeNode ← (bestPair)
8           foreach neighbour of bestPair do
9               compute max distance
10              MergeNode ← Max (distance)
11          end for
12    end for
13  Convert each partition into super-node in summary
    graph, G_s
14  Nodes and edges within the partition are represented in a
    super-node.
15  The edges between different partitions are represented as
    super edge in summary graph
16  end
```

Figure 8. g-Sum: A Proposed Algorithm for Graph Summarization

Now, the algorithm is explained further with an example, following the algorithm explanations.

Line No. 1 Graph Pre-processing: In g-Sum, first step is to pre-process the given graph dataset, in which graph dataset is clean up. Dataset cleaning is performed by removing the low degree nodes (a node with in-degree =1 or out-degree <=1) from the graph dataset. In addition to this, a graph structure has any associated features including node label, edge label, node in-degree and out-degree, weight on node and weight on the edges between nodes. During the pre-processing phase, the important graph features are selected

for using in the next phase of g-Sum approach. The g-Sum deals with the unweighted graph structures.

Line No. 3 to 12 Graph Partition The main task in the g-Sum is the graph partitioning. For graph partitioning, we have used the hierarchical agglomerative algorithm. In Line 3, an adjacency matrix of the input graph is computed. As the g-Sum deals with an undirected and un-weighted graph structure, therefore values in the adjacency matrix contains 0 or 1. The value 0 represents that there is no edge between the two corresponding nodes and 1 presents that there is an edge between two nodes. Line 4 to 12, once the adjacency matrix is generated, the next step is the selection of largest value from adjacency table. Line 4 returns the total number of nodes in the graph. Next, on line 5 a loop is started which is executed upto *i* (count of nodes in the graph). Line 6-7, selection of the best pairs in the matrix, as there are only 0's and 1's in the matrix so the g-Sum uses greedy approach to pick randomly any 1's from the adjacency matrix and merge their corresponding nodes. Line 8-11, after merging, the distance is computed (either 0 or 1) on the basis of the adjacency matrix for whose nodes affected after merging the pair of nodes and assign largest value to the corresponding merge node and affected node. These steps (Line 8-11) are executed for each neighbour in the best pair, selected on line 6-7 in the algorithm. When it finishes, then the whole process is repeated until the required number of partitions (*K*) is obtained, line 4. All this process is further explained with an example in this section.

Line 13-15 Graph Summarization: When the graph partitioning is completed, the next task is the summarization of the partitioned graph. In summary graph, the numbers of partitioned are converted into a super-node (super-node is a user provided value). The nodes in the partition and the number of edges within the partition are represented in super-node of a summary graph and edges between different partitions are represented as super-edge (super-node can contain one or more super-edges) in summary graph.

Example: The proposed algorithm is explained with the help of an example graph, as shown in Figure 9.

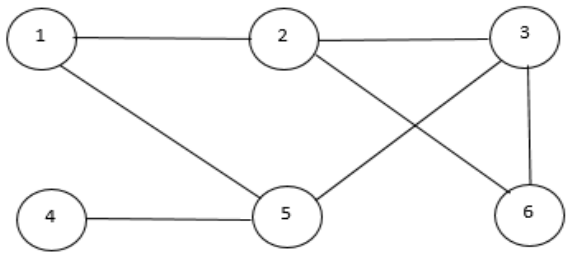


Figure 9. A Simple Graph with 7 Nodes and 7 edges

The adjacency matrix of the graph given in Figure 9 is shown below,

Table 2. Adjacency Matrix for Graph given in Figure 10.

Node	1	2	3	4	5	6
1	0	1	0	0	1	0
2	1	0	1	0	0	1
3	0	1	0	0	1	1
4	0	0	0	0	1	0
5	1	0	1	1	0	0
6	0	1	1	0	0	0

To start merging, we have to select any two nodes from the adjacency matrix given in Table 2. Therefore, we have selected the largest value from the adjacency matrix and merge their corresponding nodes. In this example, 1 is taken as large value. Hence, we merged the node 1 and node 2. Table 3 shows the resulting adjacency matrix after first merge of the graph nodes is executed.

Table 3. Adjacency Table after First Merge

Node	(1,2)	3	4	5	6
(1,2)	0	1	0	1	1
3	1	0	0	1	1
4	0	0	0	1	0
5	1	1	1	0	0
6	1	1	0	0	0

Next, compute the distance after merging the nodes (1, 2) with the remaining nodes and select the maximum distance.

$$D\big((1,2),3\big) = \max\big(D(1,3), D(2,3)\big) = \max(0,1) = 1$$
$$D\big((1,2),4\big) = \max\big(D(1,4), D(2,4)\big) = \max(0,0) = 0$$
$$D\big((1,2),5\big) = \max\big(D(1,5), D(2,5)\big) = \max(1,0) = 1$$
$$D\big((1,2),6\big) = \max\big(D(1,6), D(2,6)\big) = \max(0,1) = 1$$

Table 4 shows the results after second merging of the graph nodes.

Table 4. Adjacency Table after Second Merge

Node	((1,2),3)	4	5	6
((1,2),3)	0	0	1	1
4	0	0	1	0
5	1	1	0	0
6	1	0	0	0

Again repeat the previous step and then merge those nodes whose value is maximum. Merge node 4 and 5 with maximum value 1.

$$D\big((4,5),((1,2),3)\big) = \max\big(D(4,((1,2),3), D(5,((1,2),3))\big)$$
$$= \max(0,1) = 1$$

$$D(6,(4,5)) = \max\big(D(6,4), D(6,5)\big) = \max(0,0) = 0$$

Table 5. Adjacency Table after Third Merge

Node	((1,2),3)	(4,5)	6
((1,2),3)	0	1	1
(4,5)	1	0	0
6	1	0	0

Now stop merging the further nodes as the required numbers of partitions are obtained (in this example 3 partitions). So the three partitions of given graph G is $\mathbf{P_1}$ (1,2,3), $\mathbf{P_2}$ (4,5) and $\mathbf{P_3}$ (6) and now summarize the graph after partitioning. The partitioning of the input graph obtained after applying the hierarchical agglomerative clustering is shown in Figure 10.

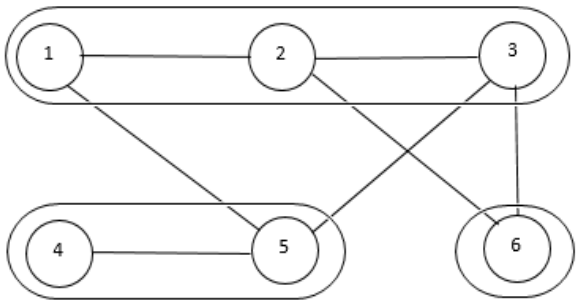


Figure 10. Partition of Input Graph $\boldsymbol{G(V,E)}$

Finally, the summary graph of an input graph after applying summarization algorithm is shown in Figure 11.

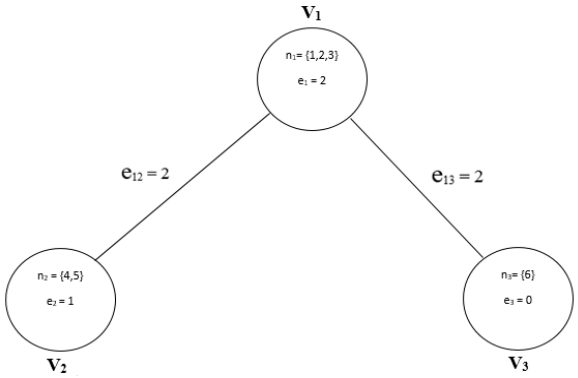


Figure 11. Summary Graph $S(V_s, E_s)$

Construction of Summary Graph Adjacency Matrix: To compute the adjacency matrix of the summary graph, we have devised the different rules. These rules specify the values of the cells in the adjacency matrix.

Rule 1: If there are distinct nodes (u, v) in the same super-node V_i then adjacency values $A_s(u, v)$of summary graph is calculated equation 1 [13],

$$A_s(u,v) = \frac{2E_i}{|V_i|(|V_i|-1)} \qquad (1)$$

Rule 2: If there are distinct nodes (u, v) in different super-nodes V_i and V_j then adjacency values $A_s(u, v)$ of summary graph is calculated by equation 2 [13],

$$A_s(u,v) = \frac{E_{ij}}{|V_i| \times |V_j|} \qquad (2)$$

Rule 3: If there are two same node $(u = v)$ in summary graph then the adjacency values $A_s(u, v)$ is calculated by equation 3 [13],

$$A_s(u,v) = 0 \qquad (3)$$

We have followed these three rules to construct the adjacency matrix of the summarized graph. Table 6 shows the adjacency matrix of the summarized graph given in Figure 11. Further, the quality of the summarized graph structure and the answer to the user query accuracy has been discussed in details in the forthcoming section, Results and Analysis.

Table 6. Adjacency Matrix of Summary Graph

Node	1	2	3	4	5	6
1	0	2/3	2/3	1/3	1/3	2/3
2	2/3	0	2/3	1/3	1/3	2/3
3	2/3	2/3	0	1/3	1/3	2/3
4	1/3	1/3	1/3	0	1	0
5	1/3	1/3	1/3	1	0	0
6	2/3	1/3	2/3	0	0	0

5. Results and analysis

In this section we discuss the experimental evaluation results with the following goals: (1) to characterize the structure of the graph summaries constructed by g-Sum algorithms; (2) to evaluate the quality of the graph summaries in terms of the reconstruction errors and the cut-norm error and of their usefulness in answering queries; (3) to compare the performances of the g-Sum algorithms with those of GraSS [31 and S2L [14]. We first describe the evaluation metrics followed by the dataset description. Next, we presented a set of experiments to evaluate the performance of the proposed algorithm by comparing with already existing algorithms GraSS [13] and S2L [14].

5.1. Evaluation Metrics

The efficiency of the g-Sum algorithm is assessed by finding the Reconstruction Error (RE) of the summary graph. RE shows the accuracy of summary graph, which is computed by finding the difference of adjacency matrix $\boldsymbol{A}$ of original graph $\boldsymbol{G}(\boldsymbol{V}, \boldsymbol{E})$ and adjacency matrix $\boldsymbol{A_s}$ of summary graph $\boldsymbol{S}(\boldsymbol{V_s}, \boldsymbol{E_s})$. The equation to compute the reconstruction error is given in equation 4 [13].

$$\mathrm{RE}(\mathrm{G}|\mathrm{S}) = \left(\sum_{i=1}^{|V|}\sum_{j=1}^{|V|}|\mathrm{A_s}(i,j) - \mathrm{A}(i,j)|\right) \quad (4)$$

The RE score represents how summary graph is similar to the original graph and how the summary graph preserves the properties of an original graph. A small value of RE represents that summary graph represents the same characteristics as that of original graph. Moreover, RE score is zero if the summary graph contains the same properties as original graph.

Another parameter used to assess summaries is the accuracy in answer to queries about original graph computed from summaries only. In the proposed work, we used two types of queries, includes degree and centrality. We compute error of degree and centrality query of the original graph by using summary graph. The degree value for queries computed using equation 5 [13-14].

$$\mathrm{d_s}(v) = \sum_{j=1}^{|V|}\mathrm{A_s}(v,j) \quad (5)$$

Similarly, the value of centrality measure is computed using equation 6 [16-17].

$$\mathrm{p_s}(v) = \frac{\mathrm{d_s}(v)}{2|E|} \quad (6)$$

5.2. Experimental Setup

We performed a series of experiments to evaluate the performance of g-Sum. The proposed technique is implemented in Java, using Eclipse Juno as an IDE. All experiments were conducted on HP ProBook 4530 Core i5-2430M CPU @ 2.40 GHz, 4Gb RAM) running Windows 7 Home Premium 64-bit version.

5.3. Datasets Description

We expended significant time, effort, and resources to obtain high quality hand-labelled data. We were able to obtain ego-networks and ground-truth from two major social networking sites: Facebook, Enron-email data. The datasets used for experimentation are ego-Facebook network [16], web-Stanford network, and email-Enron network [17].

Ego-Facebook Network: In the case of Facebook, the average ego-network has around 190 nodes [16] [23] [27], while the largest network we encountered has 4,964 nodes.

Email-Enron: It is a communication network that covers the email communication within a dataset. Nodes are email addresses, and edges denote interactions between emails. Text descriptions of this dataset are full email message text. The size of the vocabulary is 29523, and the average length of the text is 149. We filter 36,692 nodes and 1,83,831 edges from the original dataset [17] [28-29].

Web-Stanford Network: Nodes represent pages from Stanford University and edges represent hyperlinks between the pages. There are 2,81,903 nodes and 19,92,636 edges in the dataset [24-26].

5.4. Experimental Results

Summary Characterization: We studied the structure of the summaries created by our algorithms in terms of the distribution of the sizes of the super nodes, the distributions of the internal and cross densities, the (reconstruction or cut-norm) error of the generated summaries, and the running time of our algorithms. Nevertheless, there are also large super nodes containing hundreds or thousands of vertices, which helps explain the relatively large standard deviation. As k grows, the standard deviation shrinks faster than the average size (n=k), suggesting that super node sizes become more uniform.

Comparison with GraSS and S2L: We compared the runtime and the summary quality of the g-Sum approach with two well-known graph summarization approaches, Grass [13] and S2L [14]. The RE results obtained from the proposed g-Sum against S2L and GraSS are given in Table 2. Tabular results shows the graph datasets used, number of super node($\boldsymbol{K}$) and the S2L, GraSS, g-Sum approaches. We used the different combination of the supernode on each graph datasets.

In Table 7, the results displayed shows that the proposed graph summarization approach, g-Sum, outperformed on all the three graph datasets by minimizing the RE as compared to the well-known graph summarization approaches. We compared our results with GraSS and S2L approaches.

For example, using Ego-Facebook graph structure with $\boldsymbol{K}$ = 1500, g-Sum returned the summarized graph much similar to the original graph. Moreover, g-Sum reduced the RE by 2.72 and 2.92 as compared to the S2L and GraSS approaches in comparison. Similarly, on the Enron graph dataset, the results were also very promising. In all the cases a number of super node the performance of the g-Sum was outclass in comparison to S2L and GraSS approaches. While using $\boldsymbol{K}$ = 28000 supernode, the summary graph returned by the g-Sum was almost similar to the input graph structure of the Enron dataset. Finally, Table 7 also shows the results computed from the Web-Stanford graph dataset, which are also showing the good summarization accuracy of the g-Sum approach.

Table 7. Reconstruction Error Comparison on ego-Facebook dataset of g-Sum with S2L and GraSS Approaches

Graph	*K*	S2L	GraSS	g-Sum	RE-Reduced by g-Sum	
					S2L	GraSS
Ego-Facebook	1500	5.78	6.02	3.06	2.72	2.96
	2000	3.74	3.86	2.91	0.83	0.95
	2500	5.33	5.76	2.80	2.53	2.53
	3000	2.27	2.54	1.23	1.04	1.31
	3500	0.57	0.96	0.33	0.24	0.63
Enron	8000	1.03	1.17	0.94	0.09	0.23
	16000	0.95	1.02	0.59	0.36	0.43
	20000	0.30	0.37	0.22	0.08	0.15
	24000	0.18	0.23	0.15	0.03	0.08
	28000	0.10	0.16	0.07	0.03	0.09
Web-Stanford Network	20000	1.27	1.42	1.03	0.24	0.39
	25000	1.13	1.39	1.02	0.11	0.11
	30000	0.95	1.04	0.89	0.06	0.06
	35000	0.84	0.97	0.79	0.05	0.05
	40000	0.78	0.92	0.71	0.07	0.07

Table 8. Computation of the Query Error Using the proposed g-Sum in Ego-Facebook Network Graph Dataset (*K*-represents Number of Super-Node)

Graph	*K*	Average Degree		Average eigenvector Centrality Error	
		g-Sum	S2L	g-Sum	S2L
Ego-Facebook	2500	1.09	2.76	0.311	0.345
	3000	0.52	0.54	1.1	0.476
	3500	0.17	0.26	0.507	0.752
Enron	16000	0.40	0.56	0.55	0.87
	20000	0.25	0.27	0.34	0.49
	24000	0.17	0.32	0.24	0.78
Web-Stanford	25000	0.20	0.42	0.27	0.31
	30000	0.18	0.40	0.26	0.28
	35000	0.14	0.34	0.26	0.27

The results displayed in Table 8 shows a comparative study of the proposed g-Sum to the S2L approach on two graph datasets. Results are showing that on different number of super-node, g-Sum performed very well on the two graph datasets by returning most approximate answers to the different used provided queries. For example, in case of Ego-Facebook dataset (Super-Node = 3500), g-Sum average degree value was computed as 0.17 and average eigenvector centrality error was 0.507, while the S2L computed value for these two parameters was 0.26 and 0.752 respectively. This case clearly depicts that the user-queries were answered more accurately by g-Sum in comparison to S2L. Similar performance was observed on different number of Super-Node on Enron email graph dataset as well. Therefore, the query answers are very close to the true values (answers to the queries), signifying the fact that the graph summaries obtained

from the proposed g-Sum do preserve the essential structure of the original big graph.

6. Conclusion and Future Work

In this paper, an efficient graph summarization algorithm, called g-Sum, is proposed for summarizing the large graphs. The working of the proposed algorithm decomposed in three phases. The first phase is the pre-processing of graph dataset, the second phase is the partitioning of the graph and the last phase of the algorithm is the summarization of the partitioned graph. The quality of the summary graph is assessed by reconstruction error.

The proposed algorithm showed promising results by minimizing reconstruction error, as the proposed g-Sum outperform the two state-of-the-art graph summarization approaches on three different graphs represented datasets. In addition to this performance comparison, we have also shown the computation of the query error on the graph datasets, which also confirms the supremacy of the proposed g-Sum approach. In the future, we will employ this approach to the directed and labelled graph with weights. Furthermore, this approach can be easily extended to dynamic and time series graph structures well.

References

1. Li, J., Liu, C., Yu, J. X., Chen, Y., Sellis, T., & Culpepper, J. S. (2016). Personalized influential topic search via social network summarization. IEEE transactions on knowledge and data engineering, 28(7), 1820-1834.
2. Rehman, S. U., & Asghar, S. (2020). Online social network trend discovery using frequent subgraph mining. *Social Network Analysis and Mining*, *10*(1), 1-13.
3. Rehman, S. U., Asghar, S., & Fong, S. J. (2018). Optimized and Frequent Subgraphs: How Are They Related? IEEE Access, 6, 37237-37249.
4. West, J. D., Wesley-Smith, I., & Bergstrom, C. T. (2016). A recommendation system based on hierarchical clustering of an article-level citation network. IEEE Transactions on Big Data, 2(2), 113-123.
5. Dlodlo, N. (2015). Salient indicators of mobile instant messaging addiction with selected socio-demographic attributes among tertiary students in South Africa. South African Journal of Psychology, 45(2), 207-222.
6. Ozonat, M. K., & Bartolini, C. (2016). U.S. Patent No. 9,264,505. Washington, DC: U.S. Patent and Trademark Office.
7. Tian, Y., Hankins, R. A., & Patel, J. M. (2008, June). Efficient aggregation for graph summarization. In Proceedings of the 2008 ACM SIGMOD international conference on Management of data (pp. 567-580).
8. Liu, Y., Safavi, T., Dighe, A., & Koutra, D. (2018). Graph summarization methods and applications: A survey. ACM Computing Surveys (CSUR), 51(3), 1-34.
9. Rehman, S. U., Khan, A. U., & Fong, S. (2012, August). Graph mining: A survey of graph mining techniques. In Seventh International Conference on Digital Information Management (ICDIM 2012) (pp. 88-92). IEEE.
10. Alamsyah, A., Priyana, Y., & Rahardjo, B. (2017). Fast summarization of large-scale social network using graph pruning based on k-core property. Journal of Theoretical & Applied Information Technology, 95(16).
11. Navlakha, S., Rastogi, R., & Shrivastava, N. (2008, June). Graph summarization with bounded error. In Proceedings of the 2008 ACM SIGMOD international conference on Management of data (pp. 419-432).
12. Toivonen, H., Zhou, F., Hartikainen, A., & Hinkka, A. (2011, August). Compression of weighted graphs. In Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 965-973).
13. LeFevre, K., & Terzi, E. (2010, April). GraSS: Graph structure summarization. In Proceedings of the 2010 SIAM International Conference on Data Mining (pp. 454-465). Society for Industrial and Applied Mathematics.
14. Riondato, M., García-Soriano, D., & Bonchi, F. (2017). Graph summarization with quality guarantees. Data mining and knowledge discovery, 31(2), 314-349.
15. Maccioni, A., & Abadi, D. J. (2016, August). Scalable pattern matching over compressed graphs via dedensification. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 1755-1764).
16. Leskovec, J., & Mcauley, J. J. (2012). Learning to discover social circles in ego networks. In Advances in neural information processing systems (pp. 539-547).
17. Liu, Y., Wang, X., & Kurths, J. (2019). Framework of evolutionary algorithm for investigation of influential nodes in complex networks. IEEE Transactions on Evolutionary Computation, 23(6), 1049-1063.
18. Wu, Y., Zhong, Z., Xiong, W., & Jing, N. (2014, April). Graph summarization for attributed graphs. In 2014 International Conference on Information Science, Electronics and Electrical Engineering (Vol. 1, pp. 503-507). IEEE.
19. Rehman, S. U., Asghar, S., Zhuang, Y., & Fong, S. (2014). Performance evaluation of frequent subgraph discovery techniques. Mathematical Problems in Engineering, 2014.
20. Rehman, S. U., & Asghar, S. (2019). A-RAFF: A Ranked Frequent Pattern-growth Subgraph Pattern Discovery Approach. Journal of Internet Technology, 20(1), 257-267.
21. Rehman, S. U., Asghar, S., & Fong, S. (2018, February). An Efficient Ranking Scheme for Frequent Subgraph Patterns. In Proceedings of the 2018 10th International Conference on Machine Learning and Computing (pp. 257-262).
22. Zhang, N., Tian, Y., & Patel, J. M. (2010, March). Discovery-driven graph summarization. In 2010 IEEE 26th International Conference on Data Engineering (ICDE 2010) (pp. 880-891). IEEE.

23. Sanatkar, M. R. (2020). The dynamics of polarized beliefs in networks governed by viral diffusion and media influence. Social Network Analysis and Mining, 10(1), 1-21.
24. Beg, M. A., Ahmad, M., Zaman, A., & Khan, I. (2018, June). Scalable approximation algorithm for graph summarization. In Pacific-Asia Conference on Knowledge Discovery and Data Mining (pp. 502-514). Springer, Cham.
25. Wang, C., Deng, Y., Li, X., Chen, J., & Gao, C. (2019). Dynamic Community Detection Based on a Label-Based Swarm Intelligence. IEEE Access, 7, 161641-161653.
26. Zhao, Y., She, Y., Chen, W., Lu, Y., Xia, J., Chen, W., ... & Zhou, F. (2018). EOD edge sampling for visualizing dynamic network via massive sequence view. IEEE Access, 6, 53006-53018.
27. Vu, M. M., & Hoang, H. X. (2019, November). Location-Based Competitive Influence Maximization in Social Networks. In International Conference on Computational Data and Social Networks (pp. 133-140). Springer, Cham.
28. Vardasbi, A., Faili, H., & Asadpour, M. (2019). Solving submodular text processing problems using influence graphs. Social Network Analysis and Mining, 9(1), 21.
29. Singh, R. R., Iyengar, S. R. S., Chaudhary, S., & Agarwal, M. (2018). An efficient heuristic for betweenness estimation and ordering. Social Network Analysis and Mining, 8(1), 66.
30. Rehman, S. U., & Khan, M. N. A (2010). An incremental density-based clustering technique for large datasets, in Computational Intelligence in Security for Information Systems 2010, 3-11