


An effective graph summarization and compression technique for a large-scaled graph

Hojin Seo¹ · Kisung Park¹ · Yongkoo Han¹ ·
Hyunwook Kim¹ · Muhammad Umair¹ ·
Kifayat Ullah Khan¹ · Young-Koo Lee¹ 

© Springer Science+Business Media, LLC, part of Springer Nature 2018

Abstract Graphs are widely used in various applications, and their size is becoming larger over the passage of time. It is necessary to reduce their size to minimize main memory needs and to save the storage space on disk. For these purposes, graph summarization and compression approaches have been studied in various existing studies to reduce the size of a large graph. Graph summarization aggregates nodes having similar structural properties to represent a graph with reduced main memory requirements. Whereas graph compression applies various encoding techniques so that the resultant graph needs lesser storage space on disk. Considering usefulness of both the paradigms, we propose to obtain best of the **both worlds by combining summarization and compression approaches**. Hence, we present a greedy-based algorithm that greatly reduces the size of a large graph by applying both the compression and

✉ Young-Koo Lee
yklee@khu.ac.kr

Hojin Seo
hojinseo1989@gmail.com

Kisung Park
kspark224@gmail.com

Yongkoo Han
ykhhan@khu.ac.kr

Hyunwook Kim
hyunwook@khu.ac.kr

Muhammad Umair
umairchauhan9@outlook.com

Kifayat Ullah Khan
kualizai@khu.ac.kr

¹ Department of Computer Science and Engineering, Kyung Hee University, Yongin, South Korea

summarization. We also propose a novel **cost model** for calculating the compression ratio considering both the compression and summarization strategies. The algorithm uses the proposed cost model to determine whether to perform one or both of them in every iteration. Through comprehensive experiments on real-world datasets, we show that our proposed algorithm achieves a better compression ratio than only applying summarization approaches by up to 16%.

Keywords Graph · Summarization · Compression · Minimum Description Length

1 Introduction

Graphs have been widely used to represent relationships between entities in various applications such as social networks, semantic webs, and XML. In almost every application domain, the use of graph-based data modeling is increasing rapidly and the size of resultant graphs is becoming larger too, where some of them having nodes reaching to billions. For example, the number of users of Facebook has now reached to over 2 billion in 2017.¹ The need to store and analyze large graphs arises for various types of applications. For example, numerous analysis techniques have been proposed for social networks like link prediction, community discovery, graph structure learning, visualization, and so on. However, these processes suffer from scale of the underlying data as either the given graphs cannot reside in main memory or require massive I/O operations on disk.

Recently, graph summarization and compression techniques have been studied to reduce the size of large graphs independently. The *graph summarization* techniques [1–13] decrease the structural complexity of a graph by representing its dense regions, having high inner-connectivity among the members nodes, via collapsing them into supernodes. The *graph compression* techniques [11, 14–18], on the other hand, reduce the description length by compressing sparse adjacent edges of a node using different encoding methods. Therefore, both of the paradigms are quite effective to compress both the dense and sparse regions of a large graph. Therefore, we understand that applying both the techniques together on a large graph can produce a highly compact summary graph. This is so since, graph summarization approaches cannot produce better results for sparse regions, whereas graph compression approaches are mostly used to reduce size of a graph for reduced disk space. For this purpose, the batch processing is a simple procedure to apply both the techniques, where first summarizes dense subgraphs and then compresses sparse subgraphs or vice versa. However, such batch processing is not effective: (i) it is time-consuming as we have to traverse the graph multiple times, (ii) we may summarize the edges which could have provided better compactness by encoding or vice versa.

It is observed that graph summarization and compression approaches have different cost models. These models determine how much compression is achieved as a result of merging or encoding the given subgraph. The graph summarization removes inner edges and outer edges, where inner edges connect the nodes members of a supernode

¹ Source: <https://techcrunch.com/2017/06/27/facebook-2-billion-users/> Last accessed on 09/22/2017.

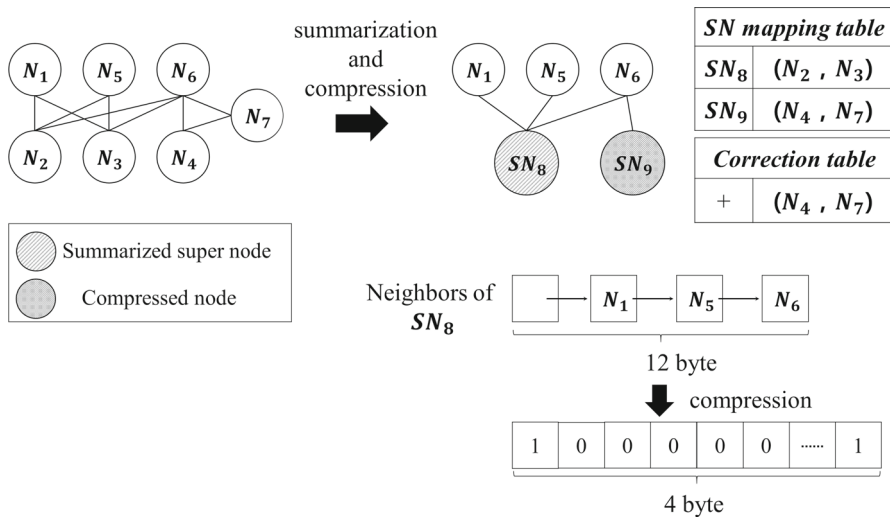


Fig. 1 An example of a summarized and compressed graph

and outer edges connect nodes between supernodes to their common neighbors. The number of nodes in supernodes, inner and outer edges determines the cost of graph summarization. On the other hand, the graph compression removes the description length to express edges of supernodes by encoding techniques like bitmap compression, reference encoding, delta encoding, or by using Word Aligned Hybrid (WAH). The number of edges of supernodes determines the cost of graph compression. Moreover, the IDs of the neighbors of supernodes significantly effect the compression ratio since many graph compression techniques have to perform nodes reordering to achieve better compression. Therefore, there is a need for a novel cost model that simultaneously considers the cost of summarization and compression to obtain a highly compact summary graph. Furthermore, the corresponding computational model (the algorithm) must also be capable of handling aforementioned objectives. We illustrate the proposed approach by using Fig. 1 for the sake of clarity.

We now summarize contributions of our paper as below:

1. We introduce a novel cost model for calculating the compression ratio considering both compression-based and summarization-based approaches to create a compact summary of a large graph.
2. We develop a greedy algorithm that repeatedly picks the best pair of nodes by using our cost model and merges them to a supernode. The algorithm uses the cost model to determine whether to perform one or both of compression and summarization.
3. Through comprehensive experiments on real-world graph datasets, we show that our proposed algorithm achieves better compression ratio than the state-of-the-art graph summarization algorithm by up to 16%.

Rest of the paper organized as follows. Section 2 describes related work, and Sect. 3 describes the problem definition for graph summarization and compression. Section 4 describes our cost model and the proposed greedy algorithm. In Sect. 5, we show

the experiments on real-world graph datasets. In Sect. 6, we conclude our work and describe the future work.

2 Related work

We present the related studies in this section and divide them based on compression and summarization categories.

2.1 Graph Compression

Graph compression [14–24] and methods have been studied extensively devise them methods where a large sized graph can be accommodated in a short storage space on disk. These approaches compress sparse adjacent edges without changing the structure of the graph. Most of the graphs in real-world applications normally are represented using sparse matrices, as they consist of a large number of nodes having only few neighbors. Therefore, most of the studies need to perform reordering of the node ids to obtain better compression ratio. As a result of nodes reordering, they represent neighbor nodes of each node as a binary bit vector and then compresses the neighborhood by using Word Aligned Hybrid (WAH) compression technique or other like reference encoding, delta encoding, or similar. WAH [18] is a bitmap indexing method that provides a high compression ratio for a sparse binary bit vector [21]. Various encoding techniques have been utilized in one of the pioneering study for graph compression [14], where the goal is to compress the neighborhood of a list of nodes by setting a certain node as a reference row (reference encoding) or to find consecutive gaps in them based on simple addition or subtraction (delta encoding). We observe that aforementioned type of strategies greatly compresses a large graph; however, the result is not a graph so cannot directly be utilized for various analysis services like visualization. The compressed graph needs to be decoded to perform querying and analysis. On the other hand, we aim to create such a summary graph which not only requires lesser storage space but also be directly mined and used for analysis.

2.2 Graph summarization

The aim of graph summarization approach is to reduce the structural complexity and description length of the large-scaled graph(s). A common aim of these studies is to create a meaningful summary that approximately preserves the structural properties (like degree distribution, centrality score, and so on) of the underling graph. These approaches create a summary graph by compressing the dense subgraphs into supernodes and superedges. There studies can be classified into summarization of static [3, 6, 9, 25], dynamic [7, 13], weighted [26, 27], or attributed graphs [5, 12, 28, 29]. A detailed survey can also be found in [11, 30] for comprehensive understanding. The grounds for all aforementioned research directions have been set by the pioneering work in [9] to summarize a static simple graph and in [12] to create summary of a static attributed graph. Navlakha et al. [9] presents greedy algorithm that provides the

highly compact summary graph based on the MDL principle. This algorithm, although, inefficient for a large graph, however, still provides the most compact summary of a large graph. Khan et al. [5] proposed to efficiently solve this problem by proposing a set-based summarization approach that uses LSH (locality similarity hashing). Inspired by the efficient set-based approach by Khan et al. [5], our approach is also a set-based strategy. We further extend the existing research by making use of a novel graph compression method that exploits both graph summarization and compression techniques.

3 Problem formulation

We define various key concepts of our paper in this section and our problem statement. Our input graph is an unweighted graph $G = (V_G, E_G)$. Similarly, a summarized graph $G_S = (V_S, E_S)$ consists of a set of supernodes V_S and a set of superedges E_S . The *supernode* $v \in V_S$ is an aggregated structure, corresponding to a set A_u of nodes from V_G . The *superedge* $e \in E_S$ represents an edge between all pair of nodes in A_u and A_v . A compressed graph $G_C = (V_C, E_C)$, on the other hand, consists of a set of nodes V_C and a set of edges E_C . Adjacent edges of a node $v \in V_C$ are represented as a set of segments $S_v = \{s_1, s_2, \dots, s_m\}$. A segment is represented as 4-bytes, which is categorized into an edge segment and a non-edge segment. An *edge segment* is used to represent a node connected to an edge, and a *non-edge segment* is used to represent a node that is not connected to any edge. In the edge segment $s_i \in S_v$ for v , a header bit is set to 1, and the remaining 31 bits represent edges, which show existence of neighbor nodes from v_j to v_{j+30} . In the non-edge segment $s_k \in S_v$ for v , the header bit is set to 0, and the remaining 31 bits represent the number of successive segments whose all 31 bits are set to 0.

In this paper, we create a summary graph based on the minimum description length (MDL) principle [31]. MDL aims to obtain the best model of the underlying data. The best model is stated as the model which minimizes the sum of the size of that model M and the size of the data D encoded with the model. In graph summarization, the size of model is that of summarized graph G_S and D represents list of corrections C [9, 31], where the corrections are the added or removed edges during summarization. In this paper, our goal is to find a summarized and compressed graph with the best compression ratio according to the MDL principle. We now define the concept of summarized and compressed graph in Definition 1.

Definition 1 A Summarized and Compressed graph (SCG). Given a graph G , $G_{sc} = (V_{sc}, E_s, E_c)$ consists of a set of supernodes V_{sc} , a set of superedges E_s , and a set of compressed edges E_c . The edges of a supernode may be compressed in the shape of super edges or be present in the non-compressed form.

In our settings, the data are G , the model is G_{sc} , and a set of corrections C is the encoded data of the model according to MDL principle. We define the cost of G_{sc} and C to be the total size of graph and the corrections, denoted as $cost(G_{sc}, C) = |V_{sc}| + |E_s| + |E_c| + |C|$. We enlist all the aforementioned notations used throughout the paper in the Table 1.

Table 1 Notations for graph summarization and compression

Notation	Description
G	An unweighted graph
G_S, G_C	A summarized graph and a compressed graph
G_{sc}	A summarized and compressed graph
V_G, E_G	Set of nodes and a set of edges of G
\bar{V}	Neighbor set of a supernode
C	Set of corrections
C_v	Cost of a supernode v
$Cost(v, u)$	Reduction cost by merging supernodes v and u
$Cost_S(u)$	Summarization cost of u
$Cost_C(u)$	Compression cost of u
$Cost_{sc}(u)$	Summarization and compression cost of supernode u
$S(u, v)$	Benefit of merging pairs (u, v)
$Segment(v)$	The segments by compressing edges of a supernode v
S_v	A set of segments for a supernode v

According to the MDL principle, the graph summarization and compression problem is to find the minimum cost representation (G_{sc}^*, C) of a graph G . A detailed explanation about the cost model and the algorithm for the minimum cost representation is described in Sect. 4. Figure 1 represents an example of result of graph summarization and compression, where the G_{sc} consists of summarized and compressed nodes. Any V_{S_i} and the V_{C_j} represent summarized supernodes and summarized nodes having compressed neighbor edges, respectively.

4 A greedy-based approach for graph summarization and compression

This section presents our proposed cost model and the proposed algorithm to create a summarized compressed graph, SCG .

In our proposed cost mode, for given node u and its sorted neighbors set $\bar{V} = v_1, v_2, \dots, v_l$, the summarization and compression costs for u can be calculated using Eqs. 1 and 2, respectively. It is to be noted that we do not have an explicit phase/step to sort the neighbors of each node, rather we store the neighbors in the sorted manner at the time of creating the graph while reading the data file from disk. Equation 1 [9] tells the summarization cost which is the sum of the number of edges and corrections on creating of a supernode. Whereas the compression cost, Eq. 2, of u is the sum of the number of edge segments and non-edge segments for \bar{V} .

$$Cost_S(u) = \sum_{v \in \bar{V}} \min\{|\Pi_{u,v}| - |A_{u,v}| + 1, |A_{u,v}|\} \quad (1)$$

where $\Pi_{u,v}$ is the set of all the pairs (a, b) in $a \in A_u$ and $b \in A_v$ and $A_{u,v}$ is the set of edges present in the original graph G .

$$Cost_c(u) = \left| \bigcup_{v \in \bar{V}} \left\lfloor \frac{id(v)}{31} \right\rfloor \right| + \sum_{i=0}^{l-1} \{NS(v_i, v_{i+1})\} \quad (2)$$

where $id(v)$ is an ID of a vertex v and if $\left\lfloor \frac{id(v_{i+1})}{31} \right\rfloor - \left\lfloor \frac{id(v_i)}{31} \right\rfloor > 1$, $NS(v_i, v_{i+1})$ returns 1 otherwise it returns 0.

By obtaining the results from Eqs. 1 and 2, the summarization and compression cost for u can be calculated using Eq. 3 which selects the minimum of the two values.

$$Cost_{sc}(u) = \min\{Cost_c(u), Cost_s(u)\} \quad (3)$$

The computed cost $Cost_{sc}(u)$ for u is then utilized in our proposed greedy algorithm. The key idea of our proposed algorithm is to select the node pairs having maximum common neighbors and then merge those pairs to create supernodes which reduce the $Cost_{sc}(u)$ upto maximum. The benefit (\hat{B}) of merging a pair (u, v) is calculated using Eq. 4, where \hat{v} is a supernode created by merging u and v .

$$S(u, v) = \frac{Cost_{sc}(u) + Cost_{sc}(v) - Cost_{sc}(\hat{v})}{Cost_{sc}(u) + Cost_{sc}(v)} \quad (4)$$

Example 1 We show the step-by-step execution of the proposed greedy-based algorithm in Fig. 2, where the id of any supernode starts from 90. We observe that the pair (v_{27}, v_2) provides highest compression (by using Eq. 4) as they share all of their neighbors with each other, so they are merged into a super node first. The next iteration aggregates the pair (v_{60}, v_{51}) into v_{91} and in this way, the process continues till the estimated compression cost does not satisfy the summarization threshold.

We understand that if we only consider the $Cost_{sc}$ as a criterion for selecting node pairs, the non-dense areas would be also compressed since these areas can have higher

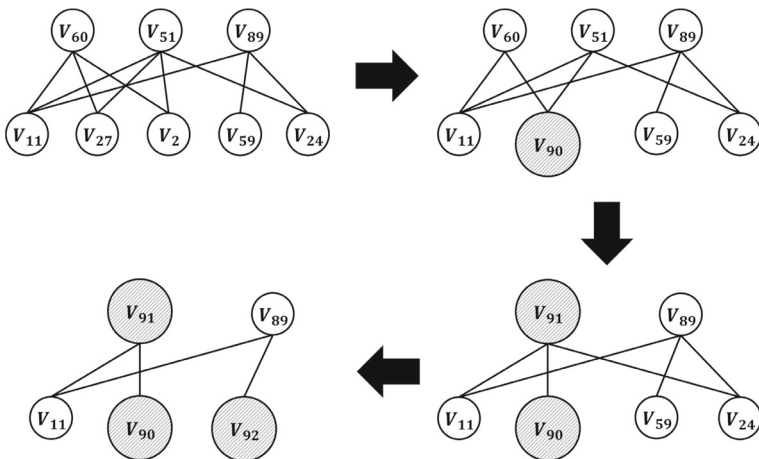


Fig. 2 The step-by-step explanation of the proposed technique

compression cost. As the number of nodes being aggregated into supernodes increases, the graph summarization shows that the summarization cost tends to decrease too as the density of the graph decreases. On the other hand, as the number of nodes being aggregated into supernodes increases, the graph compression shows that the compression cost tends to increase too because the number of adjacent edges of the supernode decreases. If we represent the supernode in such a way that has a higher cost among the summary and compression costs, the sparse regions are summarized too. When a sparse region is summarized into super nodes, the number of corrections is greatly increased. Therefore, the resultant description length increases. Hence, there is a directly proportional relationship between the number of corrections and the description length of the summary graph. To obtain a summary graph having reduced description length, our greedy algorithm first measures the summarization cost and excludes pairs that do not satisfy the threshold.

4.1 The proposed greedy algorithm

Algorithm 1 shows our proposed greedy-based graph summarization and compression approach. It receives an original graph $G = (V_G, E_G)$ and a summarization threshold θ as inputs and returns an S_{CG} . We then create a max-heap structure H in the memory to efficiently select the node pairs with the highest compression cost. Specifically, the pairs of the nodes from the 2-hops neighborhood for each node are inserted into the heap H (lines 2–6). We then select each pair (u, v) that satisfies θ and merge them into a supernode s (lines 9–13). In this way, The supernode s is added to the graph, and u and v are removed (line 13–14) both from the graph as well as from H . With the creation of a new supernode, say w , we recompute its summarization cost against the nodes in its 2-hops neighborhood and add those into H which satisfy θ . This process is repeated H becomes empty (lines 16–23). Finally, in the summarized node set V_{SC} , the edges of the nodes with a compression ratio higher than the summarization ratio are compressed (lines 24–33). In this way, we obtain a summarized and compressed graph G_{SC} .

4.2 Applying compression on the summarized graph

Recall from Algorithm 1 that we apply compression after the summarizing the nodes from the graph. During the compression, the neighborhood of a node u is represented in the form of edge segments. We observe that the number of edge segments for any node greatly depends on the ids of its neighbors. This in turn has a significant effect on $Cost_c(u)$, i.e., higher the diversity of node ids in the neighborhood, larger the number of edge segments required for their representation. Therefore, it becomes necessary to reorder the node ids to reduce the number of edge segments. In our setting a segment consists of 4 bytes i.e., 32 bits. A segment is an edge segment if it contains the neighbors of a node like 00100000 00000000 00100001 00010000. On the other hand, a non-edge segment although reserves 4 bytes but does not contain any node id in it like 00000000 00000000 00000000 00000000.

Algorithm 1: GreedyCS(G)**Input** : A graph $G = (V_G, E_G)$, A threshold θ **Output**: A summarized and compressed graph G_{SC}

```

1   $V_{SC} \leftarrow V_G, H \leftarrow \emptyset$ 
2  for each  $v$  in  $V_{SC}$  do
3      get a set of nodes  $V' \subset V_{SC}$  within 2 hops of  $v$ 
4      for each  $u$  in  $V'$  do
5          insert a pair  $(u, v)$  and  $S(u, v)$  into  $H$ 
6      end
7  end
8  while  $H \neq \emptyset$  do
9      get a pair  $(u, v)$  from  $H$ 
10     if  $S_S(u, v) \leq \theta$  then
11         continue
12     end
13     construct a supernode  $s$  by merging  $u$  and  $v$ 
14      $V_{SC} \leftarrow V_{SC} - \{u\} - \{v\} \cup \{s\}$ 
15     get a set of nodes  $V' \subset V_{SC}$  within 2 hops of  $v$  or  $u$ 
16     for each  $w$  in  $V'$  do
17          $p \leftarrow (u, w), p' \leftarrow (w, v)$ 
18          $H \leftarrow H - p - p'$ 
19         if  $S(s, w) > \theta$  then
20             insert a pair  $(s, w)$  and  $S(s, w)$  into  $H$ 
21         end
22     end
23 end
24 for each  $v$  in  $V_{SC}$  do
25     get a set of edges  $E_v$  of  $v$ 
26     if  $Cost_S(v) > Cost_C(v)$  then
27         compress  $E_v$  of  $v$ 
28         Add  $E_v$  to  $E_C$ 
29     end
30 else
31     Add  $E_v$  to  $E_S$ 
32 end
33 end
34 return  $G_{SC}$ 

```

We now present details of the effectiveness of node ids reordering using an example in Fig. 3. We find that V_0 has 7 neighbors in Fig. 3a, so it is represented as 3 edge segments and 1 non-edge segment. For instance, the neighbor ids 0–31 are represented in the 1st segment, 32–63 in the 2nd, 64–95 in 3rd, and 96–127 are stored in the 4th segment. Using this arrangement, the 1st, the 2nd, and the 4th segments are the edge segments since they contain the neighbor ids. On the other hand, there is no neighbor id in the 3rd segment, so it is called as the non-edge segment. Hence, the $Cost_C(V_0)$ is 4 due to 3 edge segments and 1 non-edge segment. In order to reduce the $Cost_C(V_0)$, Fig. 3b shows the neighbor ids of V_0 in the reordered form. Using this action, $Cost_C(V_0)$ becomes 1 since all the neighbor ids are stored in only 1 edge segment. Therefore, it is possible to maximize the compression ratio by re-assigning the node ids.

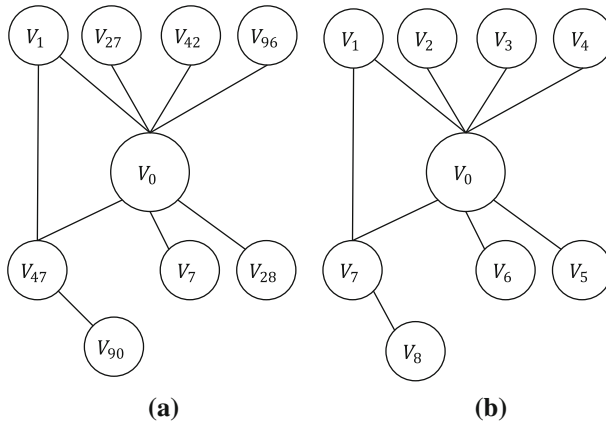


Fig. 3 An example of a graph before (a) and after (b) reordering of node ids

For the required summary graph S_{CG} , the aim of the nodes reordering is to search for the vertex set V^* which provides the best compression performance. We compute the said performance of compression, i.e., the benefit of certain reordering using Eq. 5.

$$B(V) = \sum_{v \in V} Cost_c(v) - \sum_{v \in O(V)} Cost_c(v) \quad (5)$$

where $O(V)$ returns a set of vertices which are reordered having minimum diversity like the ids in Fig. 3b.

It is observed that re-assigning ids for every combination of nodes requires a long execution time. To reduce the search space in this regard, we apply the forward greedy algorithm to efficiently reorder the node ids. In this way, the proposed greedy algorithm re-measures the benefit for all nodes $v \in V$ at each step and selects those with the highest benefit. During this course of action, if the neighbors of a certain node have already been reassigned some ids, then they will not be modified. The process is repeated until there are no more nodes re-assignment of ids can take place.

The proposed node reordering algorithm, using forward greedy selection, guarantees $(1 - 1/e)$ -approximation to the optimal solution if the benefit measurement B is a submodular set function [10]. A benefit function $B(V)$ is said to be submodular on a set C if for $C' \subset V \subseteq C$ and $v \in C$, $B(V)$ is a non-decreasing function and $B(\emptyset) = 0$. This can be shown using Eq. 6.

$$B(C' \cup \{v\}) - B(C') \geq B(V \cup \{v\}) - B(V) \quad (6)$$

5 Experiments

In this section, we present the experimental results and analyze the performance of the proposed algorithm, GreedyCS. Section 5.1 describes the experimental settings, the datasets used, and the existing graph summarization and compression algorithms.

Table 2 The statistics of datasets

No	Dataset	# of nodes	# of edges
1	DBLP	317,080	1,049,866
2	web-Google	875,713	5,105,039
3	web-NotreDame	325,729	1,497,134
4	web-BerkStan	685,230	7,600,595

Finally, we show the experimental results including the evaluation of the GreedyCS in terms of compression ratio and execution time in Sect. 5.2.

5.1 Experimental setup

A comprehensive performance evaluation is conducted using real-world datasets. These datasets are obtained from the SNAP dataset collection [32]. The detailed information of the datasets used is displayed in Table 2.

We use the compression ratio measure to evaluate quality of summarization and compression. Given a original graph G and a compressed or summarized graph G' , the compression ratio of G' is calculated by Eq. 7.

$$\text{Compression Ratio}(G, G') = \frac{\text{size}(G) - \text{size}(G')}{\text{size}(G)} \quad (7)$$

where $\text{size}(G)$ tell the description length which is the sum of the number of edges and corrections of G , the correction is stored in order to recover the original graph if needed. If G is compressed, the description length of G is the sum of the number of segments and the corrections.

To evaluate the effectiveness of our proposed algorithm, we compare it against following three algorithms in terms of the compression ratio and the execution time.

1. **Graph summarization** This algorithm is based on the graph summarization algorithm of [9]. The cost model $S_s(u, v)$ is designed to determine the summarization effect. The threshold θ is used to prune the redundant pairs with lower compression ratio.
2. **Graph compression** This algorithm is based on WAH compression [18]. In this algorithm, the cost model $\text{Cost}_c(u)$ is used for enumerating the number of segments without compression.
3. **Batch algorithm** This algorithm employs a batch process, which first summarizes the original graph using θ and then compresses the summarized graph. In this algorithm, the two cost models $S_s(u, v)$ and $\text{Cost}_c(u)$ are used for summarization and compression, respectively.

All the algorithms are implemented in C++, and the experiments are conducted on a machine with Windows OS 10 Intel(R) Core(TM) i7-6950X CPU@3.0 GHz, and 32 GB of RAM.

5.2 Experimental results

Figure 4 shows the compression ratio at varying threshold values (0.1, 0.2, and 0.3) on the web-Google dataset, where the horizontal axis represents the compression ratio. Here, we compare the three algorithms except the compression algorithm since it requires any threshold value. The vertical axis, on the other hand, represents the threshold value used in the summarization step. We find that the proposed algorithm obtains the highest compression ratio compared to the existing algorithms at the threshold values. Especially, the compression rate of the proposed algorithm is about 11% higher than that of the batch algorithm and about 12% higher than that of the summarization algorithm at the threshold 0.1.

We display the compression ratio on various datasets at threshold $\theta = 0.1$ in Fig. 5. We observe that our proposed algorithm shows better compression ratio than all the existing algorithms in every dataset. In particular, the proposed algorithm improves the performance in the web-Google dataset by about 8.4% over the batch algorithm. It also improves the performance by about 16% in the web-NotreDame dataset. We find that the proposed algorithm shows better performance in compression-efficient datasets (web-NotreDame, BerkStan). It happens because the lower the density of a graph, the higher the compression efficiency.

We show the execution time by setting the threshold $\theta = 0.3$ for over different datasets in Fig. 6. We find that the proposed approach shows similar execution time to all other datasets except web-Google. This result implies that the simultaneous consideration of summarization and compression does not significantly affect the search

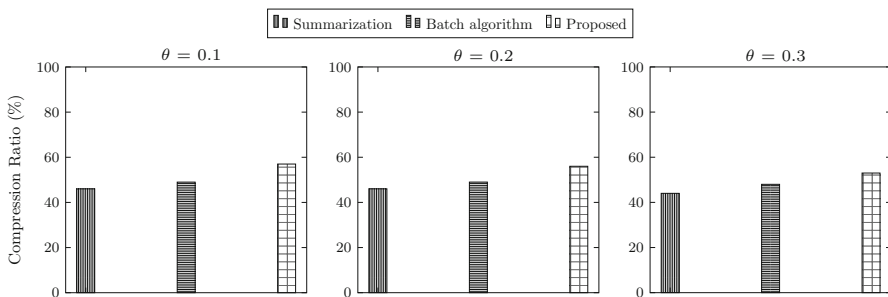


Fig. 4 Comparison of the compression ratio with various thresholds (dataset = web-Google)

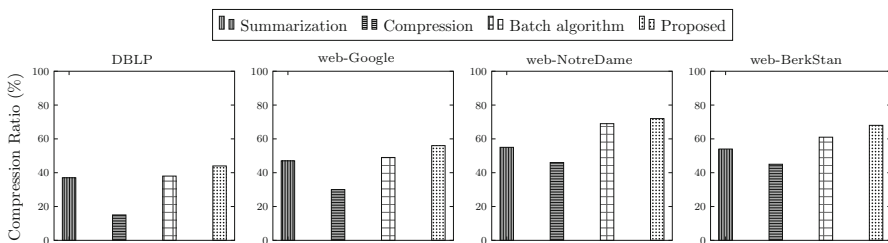


Fig. 5 Comparison of the compression ratio with various datasets ($\theta = 0.1$)

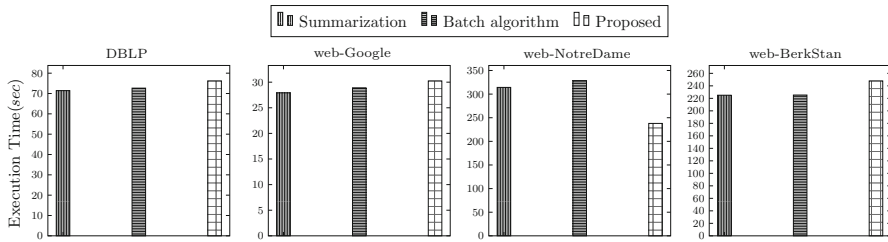


Fig. 6 Comparison of the execution time with various datasets ($\theta = 0.3$)

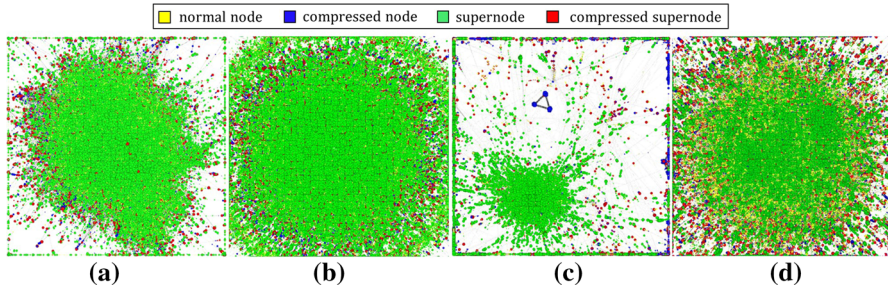


Fig. 7 Visualization of a SCG. **a** DBLP, **b** web-Google, **c** web-NotreDame and **d** web-BerkStan

space. In the web-NotreDame dataset, the execution time of the proposed technique is improved by about 27% than that of the graph summarization. In this dataset, the proposed algorithm creates the supernodes which provide higher compression ratio since the graph is much denser, so the number of iterations in the algorithm is reduced and so the execution time decreases too.

Finally, Fig. 7 shows visualization of the obtained summarized and compressed graph obtained by using the proposed GreedCS algorithm. In Fig. 7, the yellow color refers to the non-compressed or non-summarized nodes, whereas the blue color represents nodes whose edges are compressed. Similarly, the green color represents the summarized nodes without compression. Finally, the red color shows the nodes which are both the summarized and the compressed. By differentiating compressed nodes using different coloring scheme, we aim to show the the proposed approach is effective as large number of nodes are being compressed. In particular, we aim to emphasize the red colored nodes which tells the special effect of our proposed greedy algorithm that it is able to find the nodes which are summarized as well as compressed. Through the visualization, we observe that DBLP and web-Google datasets contain large number of densely populated subgraphs; hence, they are often summarized rather than compressed and we do not find large number of red nodes in them. Web-NotreDame dataset, on the other hand, is quite sparse compared to DBLP and web-Google, as can be seen via the visualization. A sparse graph does not contain many dense region, so using our proposed method we do not obtain large number of green nodes (except one large portion in the bottom-left region). However, a sparse graph can be significantly compressed using WAH compression. Hence, majority of the sparse regions in this graph (Web-NotreDame) are compressed using red colored nodes. Finally, the web-

BerkStan dataset is more denser than the Web-NotreDame however, lesser than DBLP and web-Google datasets. Therefore, we find a large number of red colored nodes in this dataset compared to DBLP and web-Google.

6 Conclusion

In this paper, we have proposed an algorithm that creates summary of a large graph by performing greed-based summarization and compression simultaneously. We have also presented a new cost model that considers graph compression and summarization at the same time. Our proposed approach of summarization is a new direction that achieves the benefit of both types of summary creation philosophies, i.e., graph summarization and graph compression. In this way, we apply both types of graph compression methods using the proposed cost model to create a highly compressed summary graph. The experiments on real-world datasets show the superiority of the proposed algorithm, and hence prove our claim. In future, we aim to extend the proposed algorithm to be operated on a distributed platform like Spark in order gain better efficiency for compression of very large graphs.

Acknowledgements This work was supported by the National Research Foundation of Korea (NRF) Grant funded by the Korea government (MEST) (No. 2015R1A2A2A01008209).

References

1. Koutra D, Kang U, Vreeken J, Faloutsos C (2014) VoG: summarizing and understanding large graphs. In: Proceedings of the 2014 SIAM International Conference on Data Mining, pp 91–99
2. Toivonen H, Zhou F, Hartikainen A, Hinkka A (2011) Compression of weighted graphs. In: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, pp 965–973
3. Khan KU, Nawaz W, Lee YK (2015) Set-based approximate approach for lossless graph summarization. *Computing* 97(12):1185–1207
4. Koutra D, Kang U, Vreeken J, Faloutsos C (2015) Summarizing and understanding large graphs. *Stat Anal Data Min ASA Data Sci J* 8(3):183–202. <https://doi.org/10.1002/sam.11267>
5. Khan KU (2015) Set-based approach for lossless graph summarization using locality sensitive hashing. In: 31st IEEE International Conference on Data Engineering Workshops (ICDEW), 2015. IEEE, pp 255–259
6. LeFevre K, Terzi E (2010) Grass: graph structure summarization. In: Proceedings of the SIAM International Conference on Data Mining, SDM 2010, Columbus, pp 454–465
7. Shi L, Tong H, Tang J, Lin C (2014) Flow-based influence graph visual summarization. In: 2014 IEEE International Conference on Data Mining (ICDM), pp 983–988. <https://doi.org/10.1109/ICDM.2014.128>
8. Shi L, Tong H, Tang J, Lin C (2015) Vegas: visual influence graph summarization on citation networks. *IEEE Trans Knowl Data Eng* 27(12):3417–3431
9. Navlakha S, Rastogi R, Shrivastava N (2008) Graph summarization with bounded error. In: Proceedings of the 2008 ACM SIGMOD International Conference on Management of data. ACM, pp 419–432
10. Nemhauser GL, Wolsey LA, Fisher ML (1978) An analysis of approximations for maximizing sub-modular set functions—I. *Math Program* 14(1):265–294
11. Liakos P, Papakonstantinou K, Sioutis M (2014) Pushing the envelope in graph compression. In: Proceedings of the 23rd ACM International Conference on Information and Knowledge Management. ACM
12. Tian Y, Hankins RA, Patel JM (2008) Efficient aggregation for graph summarization. In: Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, pp 567–580

13. Tang N, Chen Q, Mitra P (2016) Graph stream summarization: from big bang to big crunch. In: Proceedings of the 2016 International Conference on Management of Data. ACM
14. Boldi P, Vigna S (2004) The webgraph framework I: compression techniques. In: Proceedings of the 13th International Conference on World Wide Web. ACM, pp 595–602
15. Chierichetti F, Kumar R, Lattanzi S, Mitzenmacher M, Panconesi A, Raghavan P (2009) On compressing social networks. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp 219–228
16. Maserrat H, Pei J (2010) Neighbor query friendly compression of social networks. In: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining
17. Hernandez C, Navarro G (2014) Compressed representations for web and social graphs. *Knowl Inf Syst* 40(2):279
18. Wu K, Shoshani A, Otoo E (2004) U.S. Patent No. 6,831,575. U.S. Patent and Trademark Office, Washington, DC
19. Apostolico A, Drovandi G (2009) Graph compression by BFS. *Algorithms* 2(3):1031–1044
20. Faloutsos C, Megalooikonomou V (2007) On data mining, compression and Kolmogorov complexity. *Data Min Knowl Discov* 15:3–20
21. Seo H, Kim H, Park K, Han Y, Lee YK (2015) Summarization technique on a compressed graph for massive graph analysis. *Korean Soc Big Data Serv* 2(1):25–35
22. Otoo EJ, Shosahni A, Nordberg H (2001) Notes on design and implementation of compressed bit vectors. Lawrence Berkeley National Laboratory, Berkeley
23. Lim Y, Kang H, Faloutsos C (2014) Slashburn: graph compression and mining beyond caveman communities. *IEEE Trans Knowl Data Eng* 26(12):3077–3089
24. van Schaik SJ, de Moor O (2011) A memory efficient reachability data structure through bit vector compression. In: Proceedings of the 2011 ACM SIGMOD International Conference on Management of data. ACM
25. Riondato M, Garcia-Soriano D, Bonchi F (2014) Graph summarization with quality guarantees. In: 2014 IEEE International Conference on Data Mining (ICDM). IEEE, pp 947–952
26. Liu W, Kan A, Chan J, Bailey J, Leckie C, Pei J, Kotagiri R (2012) On compressing weighted time-evolving graphs. In: Proceedings of the 21st ACM International Conference on Information and Knowledge Management. ACM, pp 2319–2322
27. Khan KU et al (2017) Faster compression methods for a weighted graph using locality sensitive hashing. *Inf Sci* 421:237–253
28. Zhang N, Tian Y, Patel JM (2010) Discovery-driven graph summarization. In: 2010 IEEE 26th International Conference on Data Engineering (ICDE). IEEE, pp 880–891
29. Khan KU, Nawaz W, Lee YK (2017) Set-based unified approach for summarization of a multi-attributed graph. *World Wide Web* 20(3):543–570
30. Liu Y, Dighe A, Safavi T, Koutra D (2016) A graph summarization: a survey. <http://arxiv.org/abs/1612.04883>
31. Rissanen J (1978) Modeling by shortest data description. *Automatica* 14(5):465–471
32. SNAP Stanford Large Network Dataset Collection. <http://snap.stanford.edu/data/index.html>