Set-based Approach for Lossless Graph Summarization using Locality Sensitive Hashing

Kifayat Ullah Khan
Supervisor: Young-Koo Lee
Expected Graduation Date: Fall 2015
Deptartment of Computer Engineering
Kyung Hee University
Republic of Korea
Email: {kualizai, yklee}@khu.ac.kr

***Abstract*—Graph summarization is a valuable approach for in-memory processing of a big graph. A summary graph is compact, yet it maintains the overall characteristics of the underlying graph, thus suitable for querying and visualization. To summarize a big graph, the idea is to compress the similar nodes in dense regions of the graph. The existing approaches find these similar nodes either by nodes ordering or pair-wise similarity computations. The former approaches are scalable but cannot simultaneously consider the attributes and neighborhood similarity among the nodes. In contrast, the pair-wise summarization methods can consider both the similarity aspects but are impractical for a big graph. In this paper, we propose a set-based summarization method that aggregates the sets of similar nodes in each iteration, thus provides scalability. To find each set, we approximate the candidate similar nodes without nodes ordering and explicit similarity computations by using Locality Sensitive Hashing, LSH. In conjunction with an information theoretic approach, we present the scalable solutions for lossless summarization of both attributed and non-attributed graphs.**

I. Introduction

With a continuous and overwhelming interest from people with diverse backgrounds, the graphs like social networks, world wide web, academic, citation networks, and chemical compounds have become ubiquitous. Furthermore, with each blink of a human's eyes, the size of these graphs is continuously increasing[1]. As a result, these graphs have shaped into knowledge warehouses since their every change is triggered by the real life necessities. However, retrieving the right information at right time is hard, since the massive size does not allow their in-memory processing. In this situation, graph summarization is a valuable solution for in-memory processing of a big graph. It compresses a large sized graph into a compact summary, and maintains the properties of its underlying graph. Such compact version is useful to identify communities and influential nodes, information propagation and visualization.

The dense homogenous regions in the graphs play the key role for their summarization. There exists a high overlap of common neighborhood among the nodes in such regions, which is directly exploited towards compression. The existing work is classified into two main streams to find such dense regions, i.e. compression-based and aggregation-based. The goal of compression-based methods is to find an intrinsic ordering of the nodes in a graph, so that the compression algorithm can exploit the naturally occurring properties like locality of reference and neighborhood similarity. There exists a plethora of research works on this aspect [1] [2] [3] [4] [5], however the ordering produced by combining the Layered Label Propagation and WebGraph framework in [6] provides better compression than all the existing techniques [7]. On the other hand, the aggregation-based methods perform pair-wise summarization to collapse a pair of nodes into a super node and their corresponding edges into a super edge [8] [9] [10] [11]. These methods create a highly compact summary graph which is directly useable for querying and visualization.

We understand that the strengths and weaknesses of both the compression and aggregation-based methods, also depend on the type of the graph. In case of a non-attributed graph, the former approaches are highly scalable but does not support direct querying and visualization since the compressed graph is stored in a compact data structure. Furthermore, the low degree nodes are also cause of in-compressibility, specially in social networks [1]. However, the pair-wise super node creation technique is insensitive to the degree of the nodes and compresses both the nodes and edges into a summary graph. Unfortunately, the pair-wise aggregation method is impractical to summarize a big graph. On the other hand, the nodes ordering cannot consider the attributes similarity among the nodes when applied to a graph where each node is attached with multiple attributes. In this situation, the pair-wise method provides a superior performance since each pair-wise merge can investigate both the neighborhood and attribute similarity. Considering these facts, it is desirable to have a unified graph summarization solution which is not only scalable but also applicable to variety of the graphs, with some minor modifications.

In this paper, we present a novel set-based graph summarization technique to iteratively aggregate the Sets of Similar Nodes, $SSNs$. A set-based method is effective since aggregating multiple nodes, facilitates efficient traversal of the entire graph for its fast summarization. Since, the members nodes of a given set can have intra-connections apart from having common neighbors, so the proposed approach is suitable to identify any dense subgraph structure like clique, $\gamma-$clique and bipartite subgraph. However, locating such sets is the major challenge without nodes ordering and explicit similarity computations. For this purpose, we find each SSN by approximating the

[0]Corresponding Author: Young-Koo Lee

[1]Total number of minutes spent on Facebook each month: 640 Million. http://www.statisticbrain.com/facebook-statistics/. Last accessed on 07/21/2014

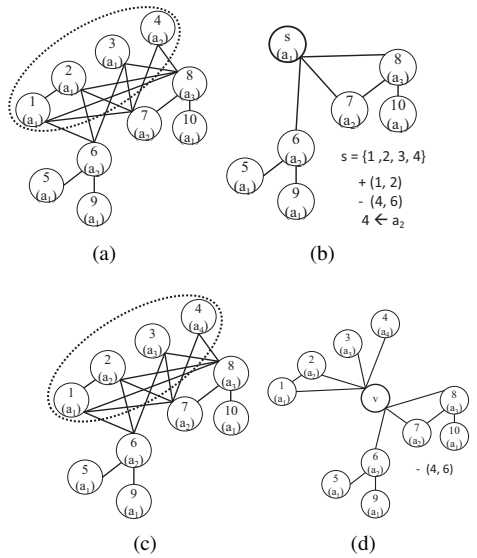

Fig. 1. Set-based approach to compress Graph G (V, E, A) (a) Set of nodes with maximum same attributes (b) Compression using super node (c) Set of nodes having unique attributes for each node (d) Compression using virtual node

Candidate Sets of Similar Nodes, $CSSNs$, using LSH. Since LSH is an approximate technique so each CSSN may contain nodes with varying similarities among each other. For this purpose, we propose a pruning technique based on a heuristic that focuses the similarities of degrees of nodes, to filter out a SSN from each CSSN. In case of an attributed graph, LSH cannot simultaneously consider both the neighborhood and attributes similarity among the nodes. To solve this problem, we modify its hashing scheme to generate SSNs, containing nodes having high neighborhood as well as attribute similarities. For summarization, we use the information theoretic Minimum Description Length, MDL, principle that produces a highly compact summary graph with corrections. In case of non-attributed graph, MDL facilitates merging a SSN into a super node with least edge corrections. Similarly for a multi-attributed graph, MDL helps to compress a SSN into a super node or aggregate its edges by adding a new virtual node, based on highest compression with least attribute and edge corrections[2].

II. MDL Rules and Problem Statement

A. MDL Rules for a Summary Graph

MDL is an information theoretic principle used to minimize the sum of the size of the theory and the associated data to express the knowledge. In case of graphs, the minimized theory is the summary graph and the list of corrections is the associated data to reconstruct the original graph, if required. Our objective to use MDL for graph summarization is to create a compact summary graph with least corrections. Below, we explain the MDL rules for an attributed graph which are applicable for a non-attributed graph, while ignoring the details related to attributes and edge aggregation using a virtual node. Following MDL explanation is adopted from [8] but we modify it for attributes, virtual node and summary graph with least edge corrections.

[2]We do not provide the comparative analysis due to lack of space

An undirected graph $G(V, E, A)$ consists of nodes V and edges E where each $v \in V$ is attached with the list of attributes A. The MDL representation of G is formally represented as $G_{MDL} = (S_G, C_r)$ where S_G is the summary graph, $S_G(V_s, E_s, A_s)$, and C_r is the list of corrections. Each node $v \in V_s$ is either a super or virtual node. The super node corresponds to set A_v where $\forall v \subseteq A_v$ is $v \in V$ in G. A virtual node vn is a new node added to S_G to compress the super edges. Similarly each edge $(A_u, A_v) \in E_s$ is a super edge and is the set of all edges between members of A_u and A_v in G, except the super edge $(A_u, A_{vn}) \in E_s$. In Figure 1 (b) and (d), we show the MDL representation of the graphs in (a) and (c) after compression using super and virtual node, along with the corrections.

The rules to create the super edges and corrections vary for super nodes and virtual nodes. For edge correction using super node, we define Π as the set of all possible edges between $(A_u, A_v) \in V_s$ and A_{uv} as the actual edges between members of A_u and A_v. A super edge is created between A_u and A_v if $A_{uv} \geq (|\Pi|+1)/2$ otherwise positive edge corrections are created between their members. The negative edge corrections are created for the edges $\Pi - A_{uv}$. Since we are interested in a compact summary with least corrections, so we create either super edges with positive edge corrections or only negative edge corrections, which have minimum memory requirements. For each super node, we set its attributes by picking the values which have maximum repetition in the given SSN. The remaining attribute values are marked as attribute corrections. Using virtual nodes, only edge corrections are possible. In this case, we create negative edge corrections for a pair of nodes, appearing as connected through virtual node and positive edge correction in vice versa case.

The cost of the summary graph depends upon its storage cost and corrections. The storage cost of the mapping of super nodes is small compared to those of E_s and C_r, so we ignore it. Thus the cost of summary is computed as $cost(G_{MDL}) = |E_s|+|C_r|$.

B. Problem Statement

Given an undirected graph G, our goal is to efficiently find the SSNs to create a compact summary graph with least corrections.

In a non-attributed graph, there exist high neighborhood similarities among the member nodes of each SSN. The summary graph of such a graph is attached with edge corrections only. On the other hand, there are attribute as well as edge corrections for summary of an attributed graph. Similarly, the member nodes of each SSN are similar from both the neighborhood as well as attributes similarity.

III. Summarizing a non-attributed graph

In this section, we illustrate how we apply LSH on graph $G(V, E)$ and briefly explain the proposed pruning technique. It should be noted that the we identify the SSNs using a query node-based approach, where we iteratively select a query node q to identify its SSN. This approach provides each node at least one chance of merger. We skip the explanation for bucket-based SSN retrieval approach and related details due to lack of space.

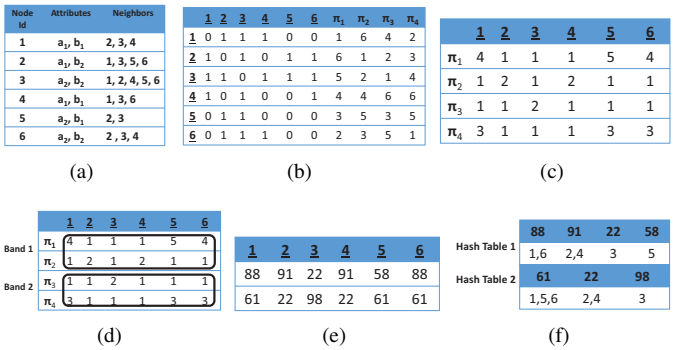

Fig. 2. LSH applied on graph (a) G (V, E, A) (b) Binary representation of neighbors list and hash functions (c) Minhash matrix (d) Division of matrix into bands (e) Combined hash codes for each portion of columns (f) Hash tables containing buckets of candidate similar nodes

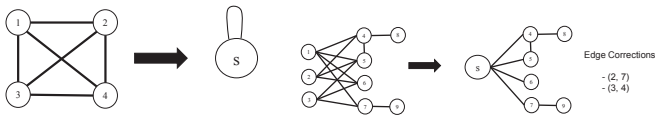

Fig. 3. Relationship between the neighborhood similarity and degree of nodes. (a) Nodes having same degrees and same similarities (b) Variation in degree of nodes, producing edge corrections

A. *Identifying SSNs*

We create the super nodes using SSNs. To find the SSNs, we apply LSH on G using steps in Figure 2, and retrieve CSSNs against each q. Since each SSN is a subset of its corresponding CSSN, so we propose a heuristic based on similarities among degrees of nodes and use it to propose a pruning technique.

1) Similar Degree Heuristic: Given a CSSN, the similar degree heuristic aims to identify its subset of nodes by choosing the nodes having similar degree to that of q. By having the similar degrees, the similarity among the member nodes of each SSN increases, resulting in super nodes with least edge corrections. Note that the nodes having same degree to that of q, are considered by default. Normally, the nodes having varying degree reduce the neighborhood overlap. Thus we find that there is a relationship between the similarity of nodes and their degrees. Consider a $CSSN = \{1, 2, 3, 4\}$ in Figure 3 (a). From this CSSN, the SSN contains all the nodes which on aggregation produces a super node with no edge corrections. We observe that in this SSN, the neighborhood of any given node n_i is same to every other node (1) and also to that of q (2).

$$Nbrs\,(n_i) = Nbrs\,(n_1) \cap ... \cap Nbrs\,(n_c) \tag{1}$$

where $\forall n_i \in SSN$ and $n_i \neq q$

$$Nbrs\,(n_i) = Nbrs\,(q) \tag{2}$$

Investigating the degree of each node in the given SSN, we find that degree of any $n_i \in SSN$ is also equal to every other node in it (3) and also to that of q (4).

$$Degree\,(n_i) = Degree\,(n_1) = ... = Degree\,(n_c) \tag{3}$$

$$Degree\,(n_i) = Degree\,(q) \tag{4}$$

Lets evaluate a case where we obtain some edge corrections, when the similarities between the nodes in a SSN are slightly varying. Consider the bipartite graph in Figure 3 (b) which on aggregation, produces a super node with two edge corrections. This illustrates that there are no edge corrections when the degrees are same, Figure 3 (a), but they occur when a change happens in the degrees, Figure 3 (b). So it is evident that the edge corrections from a super node, depend upon the deviation in the degree of each n_i from that of q (5).

$$|Degree\,(q) - Degree\,(n_i)\,| \;\leq\; \delta \times Degree\,(q) \tag{5}$$

We add any $n_i \in CSSN$ to SSN if its degree difference with q satisfies $\delta \times Degree\,(q)$ (5). Otherwise, there are more edge corrections from a super node compared to the desired compression ratio. We bound δ by $0 \leq \delta \leq 0.5$, where the lower limit targets each n_i having complete neighborhood overlap with q. Similarly, the upper limit of δ enforces to have more common to that of non-common neighbors between n_i and q. Thus we observe that the nodes having similar degrees in CSSN, have higher probability to be more similar. By defining the range of δ, pruning the least similar nodes in a CSSN is facilitated. Using (5), we also obtain the lower and upper bounds to extract a SSN from a given CSSN (6).

$$\delta_{min} \times Degree\,(q) \leq Degree_{ni} \leq \delta_{max} \times Degree\,(q) \tag{6}$$

When each n_i have similar degree to that of q then there is a high probability to have large neighborhood overlap between n_i and q. However, simply relying on similar degrees, produce inconsistent results in some of the cases. Fortunately, we resolve this inconsistency and identify the required nodes using proposed pruning method.

2) Auto Pruning: The objective of auto pruning is to filter the nodes from a CSSN, which minimize the overlap of common neighbors. The similar degree heuristic directly helps prune the nodes having large deviation from the degree of q.

To prune the least similar nodes, we first sort the CSSN using degree of nodes. This sort brings the nodes having similar degree to q closer to it in the sorted order. We then start the search of SSN by iteratively comparing the overlap ratio from the nearby nodes in the sorted order. We add a node into SSN if its neighbors satisfy the bounds in (6) . On narrowing the bounds, we tend to extract only the highly similar nodes which result in lower compression ratio. Having such bounds, the likely similar nodes at least get the chance for overlap computation. An end to the traversal happens either when we find the nodes violating the bounds or entire list is exhausted if each member of CSSN has degree within the bounds.

IV. Summarizing an Attributed Graph

In this section, we present the proposed approach to unify the topological and attributes information in LSH, since the LSH steps in Figure 2 does not consider the attributes similarity.

For each node in an attributed graph $G(V, E, A)$, the adjacency list $Nbrs$ and attributes set $Attribs$ are semantically two different entities. Whereas, the adjacency list of each node holds its structural information, the set of attributes describe its identity. Apart from their semantic differences, usually the neighborhood of each node has much larger size compared to its attributes set, $|Nbrs| >> |Attribs|$. Since in real world graphs like social networks, users often do not provide all the attribute information. On the other hand, the graphs having power law properties have large number of low degree nodes, leading to $|Nbrs| < |Attribs|$. Both of these information can be considered as the two dimensions of each node and the problem is how to unify or reduce them such that their output preserve the intrinsic properties from both the aspects. In our scenario, we refer this problem as the "curse of dimensionality" due to the semantic differences between the two dimensions, rather than upon their count.

Since the existing dimension reduction methods, both linear and non-linear, operate on n-dimensional data where n can be large but have semantic homogeneity like spatial information. So we observe that these techniques are not applicable for $G(V, E, A)$. However, influenced by the projection techniques we propose to project the two dimensions of each node into a single unified granularity level. To be precise, we union the attributes set of each node with its adjacency list and perform LSH on the unified list. By such unification, the subsequent minhash signatures of each node contains the representation from both the aspects. This is so since any value in the combined list can become the minimum value against a given hash function π_k. Thus, the resulting CSSNs have high probability to be similar from both the aspects. We now formally define the proposed unified information in Definition 1.

Definition 1. $[Neighbors\ Attributes\ List]$ *Given a node $v_i \in V$ along with its list of neighbors N_i and attributes A_i, the Neighbors Attributes List (NAL) is a unified list that concatenates N_i and A_i.*

Figure 4 (a) shows a graph $G(V, E, A)$ where each $v_i \in V$ is attached with two attributes. In Figure 4 (b), we map the attributes into numeric values to unite them with the adjacency lists. The Figure 4 (c) illustrates the NALs for nodes 1 and 6.

With the unification of attribute and neighborhood information, we can compute the minhash values using hash functions. Since now we consider both the information for minhash computation, it is desirable that their representation must be preserved in hash functions. We now formally define the hash function in Definition 2.

Definition 2. $[Unified\ Hash\ Functions]$ *The random permutations $\{\pi_1, \pi_2, ..., \pi_m\}$ is a set of hash functions where π_i is a permutation of $\{1, 2, ..., n, n+1, ..., k\}$. The set $\{1, 2, ..., n\}$ are the total nodes in G and $\{n+1, ..., k\}$ is the union of all cardinal values from each attribute.*

We illustrate the sample hash function in Figure 4 (d). Here each function is the random permutation from total nodes in the G and possible attribute values from all the attributes, after mapping.

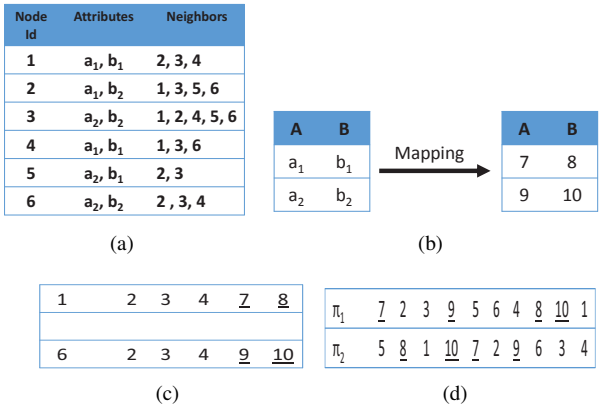

Fig. 4. Illustrating the Unified LSH concept (a) Graph G (V,E,A) (b) Attributes Mapping (c) Sample Nodes from G with their NALs (d) Hash functions

A. Balance between Neighborhood and Attribute Information

Having NAL for each node and unified hash functions, we can find the CSSNs. However, this procedure may still not reveal the nodes similar from both the aspects. For instance, consider the NAL_i of a node v_i. Using (7), we find that each element in NAL_i has the equal probability to be minimum against a given hash function, π_k. On the other hand, since the sizes of adjacency lists and attributes sets are not same. So $\forall v_i \in V$, whose $|Nbrs| >> |Attribs|$, their attributes have very low probability to be minimum in any of the hash function, compared to that of their neighbors, (8). This results in creation of minhash signature columns preserving the neighborhood information only, thus minimizing the influence of attribute information. In worst case, a pair of nodes $(u, v) \in V$ having a complete overlap of neighbors but no attributes similarity or vice versa, will produce exactly same minhash signatures. For example, consider the node 1 in Figure 4 (a). Using (7), the probability of each element in its NAL_1 to be considered as minimum is 0.2. However, the probability of any neighbor in NAL_1 to become minimum is larger than that of any of its attribute value, 0.6 and 0.4 respectively. Now in Figure 4 (c) consider the nodes 1, 6 as worst case scenario, where both share all of their neighbors but have no attribute in common. Conclusively, their Jaccard similarity does not depict the real situation and may produce more attribute corrections on aggregation into a super node.

$$Pr\left(min\left(\pi\left(NAL_i\right)\right) = \forall x \in NAL_i\right) = \frac{1}{|NAL_i|} \quad (7)$$

$$Pr\left(min\left(\pi\left(NAL_i\right)\right) = \forall x \in Nbrs_i\right) \gg Pr\left(min\left(\pi\left(NAL_i\right)\right) = \forall y \in Attribs_i\right) \quad (8)$$

Ideally the minhash signatures of each node must have the representation from its neighbors as well attributes. For this purpose, we propose to enforce representation from both the

aspects in the minhash signatures so that the bias towards either aspect is minimized.

In our approach, we use a parameter p to preserve the neighborhood and attributes information in the minhash signatures for each node. Specifying p guarantees that certain proportion of minhash values are computed from each component, restricting the large percentage of minhashes from either component. Using p, the equations (9) and (10) state the fraction of minhash values from each of the two components in NAL_i for each v_i. We understand that the right selection of p plays a key role to assign certain weight to each of the two components in NAL_i. For instance, if $p = 15$ and total hash functions are 30 then using (9) and (10), the minhash signature column for the node 1 in Figure 4 (c), contains 50% minhash values from each of the component, (11). Thus, we find with the inclusion of p, the probability to have minhash values from attributes increases from 0.4 to 0.5.

$$Minhash_{v_i}^{Nbrs} = \frac{m-p}{m} \tag{9}$$

$$Minhash_{v_i}^{Attribs} = \frac{p}{m} \tag{10}$$

$$Pr\left(min\left(\pi\left(NAL_i\right)\right) = \forall x \in Nbrs_i\right) = Pr\left(min\left(\pi\left(NAL_i\right)\right) = \forall y \in Attribs_i\right) \tag{11}$$

We observe that the enforced inclusion of neighborhood and attribute information in the minhash signatures, may produce erroneous results when we generate CSSNs in hash tables. Figure 2 (d) displays the step when we split the minhash matrix into b bands of r rows each, to create CSSNs on the basis of hash codes. If a given band gets a portion of minhash column where all the minhash values are either from neighbors or attributes then the corresponding buckets will contain the nodes similar from that very aspect only. There is a higher probability for this situation in the case when we use higher count of hash functions, since then the length of resulting minhash column for each node is also high. This situation is possible in both the random and sequential selection of minhash values for each band. To overcome this limitation, we propose a concept called $Bi - Minhashing$ which produces minhash signatures preserving the representation from both the aspects from NAL_i of each v_i.

B. Bi-Minhashing

In bi-minhashing, we compute the minhash signatures of each v_i where both of its dimensions (neighbor or attribute) has the equal probability to be chosen as minimum against any hash function, π_k. Using (9) and (10), we first compute the minhash signatures M_{NAL} from neighbors and attributes. Next, we again apply minhasing on M_{NAL} to compute its minhash signatures using the same hash functions to create $M_{NAL}^{/}$. This second level hashing completely randomizes the minhash representation of both the aspects from NAL_i of each v_i. Thus it rightly approximates the true Jaccard similarity between any pairs of nodes (v_i, v_j).

We find the significance of bi-minhashing is two-folded. First, it creates a balance between neighborhood and attribute information, i.e., when $|Nbrs| >> |Attribs|$ or vice versa. Since setting p to 50%, the initial hashing creates the minhash signatures where each of the two component has balanced representation. Secondly, the equal minhash representation is itself the solution of selecting the appropriate value of p.

V. Conclusion and Future Work

In this paper, we propose a set-based approach for graph summarization. A set-based approach is scalable since it aggregates multiple nodes in each iteration. We understand that the proposed approach adds a new dimension in aggregation-based summarization since we directly target the inherent occurrence of phenomenon like $friends\ of\ friends\ have\ many\ common\ friends$ in social networks without nodes ordering or explicit similarity computations. Currently, our approach is applicable to both attributed and non-attributed graphs, but we plan to build a unified solution to summarize further types of graphs like heterogeneous and weighted graphs.

VI. Acknowledgment

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government(MEST) (No. 2014R1A2A1A05043734).

References

[1] F. Chierichetti, R. Kumar, S. Lattanzi, M. Mitzenmacher, A. Panconesi, and P. Raghavan, "On compressing social networks," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2009, pp. 219–228.

[2] P. Boldi and S. Vigna, "The webgraph framework i: compression techniques," in *Proceedings of the 13th international conference on World Wide Web*. ACM, 2004, pp. 595–602.

[3] U. Kang and C. Faloutsos, "Beyond'caveman communities': Hubs and spokes for graph compression and mining," in *Data Mining (ICDM), 2011 IEEE 11th International Conference on*. IEEE, 2011, pp. 300–309.

[4] G. Buehrer and K. Chellapilla, "A scalable pattern mining approach to web graph compression with communities," in *Proceedings of the 2008 International Conference on Web Search and Data Mining*. ACM, 2008, pp. 95–106.

[5] I. Safro and B. Temkin, "Multiscale approach for the network compression-friendly ordering," *Journal of Discrete Algorithms*, vol. 9, no. 2, pp. 190–202, 2011.

[6] P. Boldi, M. Rosa, M. Santini, and S. Vigna, "Layered label propagation: A multiresolution coordinate-free ordering for compressing social networks," in *Proceedings of the 20th international conference on World Wide Web*. ACM, 2011, pp. 587–596.

[7] P. Liakos, K. Papakonstantinopoulou, and M. Sioutis, "Pushing the envelope in graph compression," in *Proceedings of the 23rd ACM International Conference on Information and Knowledge Management*. ACM, 2014, pp. 1549–1558.

[8] S. Navlakha, R. Rastogi, and N. Shrivastava, "Graph summarization with bounded error," in *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. ACM, 2008, pp. 419–432.

[9] H. Toivonen, F. Zhou, A. Hartikainen, and A. Hinkka, "Compression of weighted graphs," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2011, pp. 965–973.

[10] K. LeFevre and E. Terzi, "Grass: Graph structure summarization." in *SDM*. SIAM, 2010, pp. 454–465.

[11] Y. Tian, R. A. Hankins, and J. M. Patel, "Efficient aggregation for graph summarization," in *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. ACM, 2008, pp. 567–580.