Lossless Graph Summarization using Dense Subgraphs Discovery

Kifayat Ullah Khan
Dept of Computer Engineering,
Kyung Hee University.
Seocheon-dong, Giheung-gu,
Yongin-si,
Gyeonggi-do, 446-701,
Republic of Korea
kualizai@khu.ac.kr

Waqas Nawaz
Department of Computer Engineering,
Kyung Hee University.
Seocheon-dong, Giheung-gu,
Yongin-si,
Gyeonggi-do, 446-701,
Republic of Korea
wicky786@khu.ac.kr

Young-Koo Lee
Department of Computer Engineering,
Kyung Hee University.
Seocheon-dong, Giheung-gu,
Yongin-si,
Gyeonggi-do, 446-701,
Republic of Korea
yklee@khu.ac.kr

ABSTRACT

Dense subgraph discovery, in a large graph, is useful to solve the community search problem. Motivated from this, we propose a graph summarization method where we search and aggregate dense subgraphs into super nodes. Since the dense subgraphs have high overlap of common neighbors, thus merging such subgraphs can produce a highly compact summary graph. Whereas the member nodes of each dense subgraph have many edges in common, they also have some edges not belonging to a given subgraph; which in turn reduce the compression ratio. To solve this problem, we propose the concept of *AutoPruning* that effectively filters the dense subgraphs having higher ratio of common to that of non-common neighbors. To summarize the dense subgraphs, we use the Minimum Description Length (MDL) principle to obtain a highly compact summary with least edge corrections for lossless compression. We propose two alternatives to trade-off the computation time and compression ratio while creating a super node from each dense subgraph. Through experiments on two publicly available real world graphs, we compare the proposed approach with the well known Minimum Degree Measure (MDM), for dense subgraph discovery, and observe very encouraging results.

Categories and Subject Descriptors

G.2 [**DISCRETE MATHEMATICS**]: Graph Theory
Graph Algorithms, *Hypergraphs*

General Terms

Algorithms, Measurement, Performance

IMCOM '15, January 08 - 10, 2015, Bali, Indonesia.

http://dx.doi.org/10.1145/2701126.2701157.

Keywords

Graph Summarization; Dense Subgraphs; Minimum Degree Measure; MDL

1. INTRODUCTION

With an increase in online interaction among the people, social networks and web graphs are rigorously being used. Such overwhelming interactions hold various real life phenomenons like communities formation, influential persons and viral marketing. However, understanding such concepts is hard since these graphs cannot be fit in the memory due to their massive size. In this situation, summary graph is a valuable solution since it compresses the graph by preserving the underlying semantics.

The main theme of graph summarization is to search and compress the natural community structures, dense subgraphs, in a large graph. It is useful to target the communities since their constituent nodes have high overlap of common neighbors thus provide good compression. For instance, Figure 1 shows graphs containing communities of nodes in the shape of clique and bipartite graph. In a clique, the overlap of common neighborhood is high since each node is neighbor of every other node. Similarly in a bipartite graph, the set of nodes share many common neighbors. Thus the nodes sharing common neighbors are merged into a super node and their edges are compressed into a super edge. To find the community for a query node, the search can be either global or local. A global search is computationally expensive since it traverses the entire graph [14]. On the other hand, the local search is effective since only the nodes in certain vicinity of the query node, are explored [6]. In both the approaches, a community is a dense subgraph that is found by maximizing the minimum degree of its member nodes.

MDM is well known to find the dense induced subgraph in a large graph [13, 1, 4]. Given a graph $G(V, E)$, MDM removes the nodes having minimum degree and filters out a dense induced sub-structure. An induced subgraph contains nodes $S \subseteq V$ where for each edge $(v_i, v_j) \in E$, $v_i \in S$ and $v_j \in S$ [4]. Since each super node in the summary graph also constitutes of a dense subgraph, Figure 1, thus we can utilize MDM for this purpose. However, MDM only aims to identify induced subgraphs. The induced subgraphs are dense and can provide good compression but they do not

incorporate an edge $(v_i, v_k) \in E$ where either $v_i \notin S$ or $v_k \notin S$. A super node from such subgraph is infeasible since it introduces many edge corrections (Edge corrections to be formally explained in Section 3.1). For instance, if we create a super node from the induced subgraph from nodes {1, 2, 3, 4, 5, 6} in Figure 1 (b), the member node 2 introduces large edge corrections since it has many non-common neighbors.

In this paper, we present a graph summarization method based on local dense subgraph discovery approach. In this method, our objective is to find the dense subgraphs by performing a local search. This is a paradigm shift in aggregation-based graph summarization since we directly target the dense subgraphs in the graphs without any node, edge ordering [15, 5, 2, 3, 7] or graph partitioning [8]. In the proposed approach, we iteratively retrieve the dense subgraph for each query node and merge it into a super node. We show that dense subgraphs found using the well known MDM, are not suitable for compression, so we propose the concept of Auto Pruning. Auto Pruning is a parameter-less approach to find the dense subgraphs of suitable size by filtering the minimum and large degree nodes. Our summarization approach has strong information theoretic basis of the MDL principle. Using MDL, we create a highly compact summary with edge corrections for lossless compression. We perform experiments on two publicly available real world graphs. We compare the performance of the proposed approach with the MDM and obtain valuable results.

The rest of the paper is organized as follows. We provide the related work in Section 2. Preliminaries and our problem statement is explained in Section 3. Section 4 presents the details of the proposed solution. We present the experiments in Section 5. Finally, we conclude the paper in Section 6.

2. RELATED WORK

In this section we provide the related work in the aspects of graph summarization and dense subgraph search methods.

2.1 Graph Summarization

Graph summarization has received a considerable attention in the past few years due to its wide effectiveness. There exist different techniques for graph summarization. For instance, [5, 2, 3, 7] exploit the node ordering to find the nodes having similar neighborhood. [2] uses natural lexicographic order in the nodes whereas [5, 3, 7] sort the graph by computing the minhash signatures using each node's out-links. On the other hand, [11, 15, 16] merge the pair of nodes having similar neighbors into a super node and their corresponding edges into a super edge. We observe that compression by edge ordering globally identify the communities of nodes; thus they have to traverse the entire graph. Similarly, super nodes-based summarization methods perform iterative pair-wise compression. These approaches are local in nature however, merging the pairs do not make use of the communities in the form of dense subgraphs, rather may divide a single community into multiple super nodes.

We differ from these studies since we directly identify the dense subgraphs in their natural existence. We do not perform any kind of node ordering since it does not exist in social, co-authorship and citation networks [5]. Similarly using an approximate technique for nodes ordering, may incur accuracy loss. Furthermore, we identify the dense subgraphs for compression using query nodes.

2.2 Dense subgraphs for Community Search

The dense subgraphs in a large graph, illustrate the community structures. Seidman [13] presented the pioneering work to find the dense subgraph in a large graph using MDM. Since finding a dense subgraph is an optimization problem, [1, 4, 10] used a greedy algorithm for dense subgraph discovery in undirected and directed graphs. Their greedy approach use the MDM to prune the nodes having least degree to increase the density of the subgraph. Recently [14, 6] use the same measure to find the communities of nodes for the given query nodes. Their greedy algorithm is the modification of the algorithm in [1, 4].

Our objective is to utilize the concept of local community search and evaluate how we can use it for graph summarization. In this regard, we pick the MDM to analyze its effectiveness to identify the dense subgraphs for compression.

3. PRELIMINARIES

In this section, we explain MDL, MDM for graph summarization and dense subgraph discovery respectively as the preliminary work. We then formally present our problem statement.

3.1 MDL for Graph Summarization

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 [12]. In case of graphs, the minimized theory is the summary graph and the list of edge corrections is the associated data to reconstruct the original graph, if required. Our objective to use MDL for graph summarization is to create a highly compact summary graph with least edge corrections. In this paper, we use the MDL rules from [11] for edge corrections with a minor change for super edges creation.

MDL representation for an undirected graph $G(V, E)$, with no self-loops and multi-edges, is formally represented as $G_{MDL} = (S_G, C_r)$ where S_G is the summary graph, $S_G(V_s, E_s)$, and C_r is the list of edge corrections. Each node $v \in V_s$ is a super node, corresponding to set A_v where $\forall v \subseteq A_v$ is $v \in V$ in G. 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. Figure 1 shows graphs G and their MDL representations. In Figure 1 (a), the nodes 1, 2, 3 and 4 are merged into super node S and their common edges are compressed into a self super edge. However, the mistake (no edge between nodes 4 and 5) caused by this super node is kept as a positive edge correction.

The rules to create the super edges and edge corrections are based on the super nodes. 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 highly compact summary with least edge corrections, so we create either of super edges with positive edge corrections or only negative edge corrections, which have minimum memory requirements.

The cost of the summary graph depends upon its storage cost and edge 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 $cost(G_{MDL}) = |E_s|+|C_r|$.

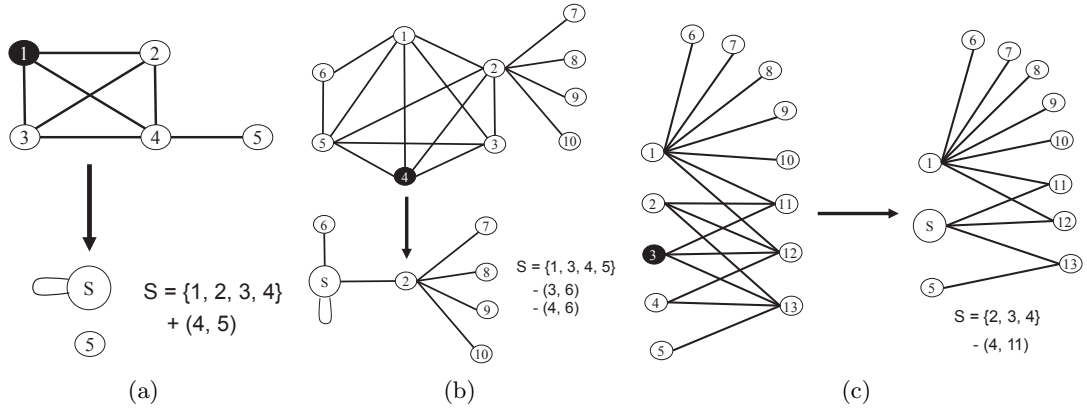


Figure 1: Super Node Creation from (a) clique with single non-common neighbor (b) clique with many non-common neighbors (c) bipartite graph

3.2 Minimum Degree Measure

MDM has recently gained much attention in dense subgraph discovery methods. Given a graph [14] or subgraph [6], it aims to maximize the minimum degree of the nodes to find the dense induced subgraph. Using intersection count of neighbors to measure the density of a subgraph, the nodes having minimum degree increase the sparseness since they minimize the intersection count.

A careful analysis of MDM makes it applicable to find dense subgraphs in bipartite graphs too. Since it focuses to maximize the minimum degree, so nodes decreasing the intersection count in bipartite graphs, are good candidates to be removed. For instance in Figure 1 (c), removing the node 5 from a bipartite graph from nodes {1, 2, 3, 4, 5}, the intersection count increases to two. Thus we observe that MDM can improve the density of different types of subgraphs, other than dense induced subgraphs.

3.3 Notion and Problem Statement

We require the dense subgraphs to summarize G. Since creating a super node by directly merging the entire subgraph, is not optimal so we first formally define the subgraphs which after merge yield least edge corrections. We then present our problem statement.

Definition 1. (*Dense Subgraph*) Given a graph G and a query node $q \in V$, we call a subgraph containing q as its dense subgraph (DSG), where percentage of common neighbors from member nodes of the subgraph is higher than that of their non-common neighbors.

With such definition any subgraph whether complete clique, partial clique or bipartite graph, is a DSG for a query node and can result in a super node with least edge corrections. For example the member nodes of each super node in Figure 1, are DSGs of their respective query nodes (colored as black). In each such subgraph, all the constituent nodes have higher percentage of common neighbors to that of their non-common neighbors.

Definition 2. (*Problem Statement*) Given a graph G, our objective is to create its least cost MDL summary graph by searching a DSG for each query node and merging it into a super node and its edges into super edges.

4. PROPOSED FRAMEWORK FOR GRAPH SUMMARIZATION

In this section, we explain the proposed graph summarization framework. We first present how we perform local search to find DSGs and then provide the algorithm to create MDL-based summary.

4.1 Local Dense Subgraph Search

Given a query node $q \in V$, our objective is to search a DSG for it. A simple solution is to explore the neighborhood of q with the intention that it shares neighbors with its connected nodes. For instance, with query nodes 1 and 4 in Figure 1 (a) and (b) respectively, their directly connected nodes have higher percentage of common neighbors; thus forming a DSG. However, searching a DSG in this manner may not be valid in all cases and the resultant super node may produce many edge corrections. For example in Figure 1 (b), the query node 4 shares three neighbors with node 2, but they also have four non-common neighbors. In worst case, it is possible that a query node may have no or least common neighbor with any of its directly connected nodes. In this case, we may waste much computation time.

For this purpose, we investigate the 2-hops neighborhood of each query node. This exploration guarantees that each query node shares at least one neighbor with other nodes. For instance in Figure 1 (b) all the nodes are 2-hops away from the query node 4. Similarly in Figure 1 (c) with query node 3, we retrieve nodes {1, 2, 4, 5} to create a bipartite subgraph. From each of these subgraphs, we extract the DSGs to create the super nodes.

4.2 Auto Pruning

A subgraph obtained from the 2-hops away neighbors of each query node, includes nodes of varying degree. A varying degree of the member nodes decreases the overlap of common neighbors. For this purpose, we propose the concept of Auto Pruning which aims to extract a DSG from each subgraph.

The objective of Auto Pruning is to remove the nodes from the subgraph whose percentage of non-common neighbors is higher than that of common neighbors. It is a parameter-less approach to remove the nodes which violate the criteria. For each node in the subgraph, it iteratively compares the percentage of common and non-common neighbors with the

Algorithm 1 Graph Summarization using Dense Subgraphs Search

Require: Graph G
Ensure: Summary S_G, Edge Corrections C_r.
1: $S_G(V_s, E_s) \leftarrow G(V, E)$
2: **while** (*unvisited nodes in* S_G) **do**
3: Choose a query node q
4: $Subgraph_q \leftarrow$ Retrieve 2-hops away neighbors of q
5: Merge($Subgraph_q$)
6: **end while**
7: **for each** $(u, v) \in V_s$ **do**
8: **if** $|A_{uv}| \geq (\Pi_{uv} + 1)/2$ **then**
9: add (u, v) to E_s
10: add $-(a, b)$ to C_r for all $(a, b) \in \Pi_{uv} - A_{uv}$
11: **else**
12: add $+(a, b)$ to C_r for all $(a, b) \in A_{uv}$
13: **end if**
14: **end for**

query node. After each iteration, a node is automatically pruned if it does not satisfy the criteria.

4.3 Creating Summary Graph

The algorithm 1 describes our procedure to compute the MDL-based summary. In line 1, we refer the input graph as the summary graph and declare each node as unvisited node. The lines 2-6 traverse the entire graph and randomly select an unvisited node as a query node. A node become visited if either selected as query node or is merged into a super node. At line 4, we retrieve the 2-hops away neighbors of q to create a subgraph. Note that the subgraph contains q. At line 5, we pass this subgraph to super node creation procedure, explained in subsequent subsections, Section 4.3.1 and Section 4.3.2. When all the nodes have been visited and super node are created, we create the super edges and edge corrections, Lines 7-14, using the rules explained in Section 3.1.

4.3.1 Super Node Creation using Subgraph-wise Merge

In this approach, we filter a DSG for each query node q and merge it directly into a super node. Since all the nodes have high percentage of common neighbors, so less edge corrections are already anticipated.

The algorithm 2 describes the DSG-wise merge. At line 1, we sort the nodes based on their degree. After sorting, q can be either the first, last or somewhere in the middle based on its degree. If q has highest degree then we pick a node having degree equal or less then that of q, Line 4, otherwise we accordingly pick the node, Line 13 or Line 17. When a node is picked from the subgraph, we union its neighbors with those of q and compare the percentage of common and non-common neighbors. Further picking of v stops if it has higher percentage of non-common neighbors. In this way, the rest of the nodes are automatically pruned. At this stage, we obtain a filtered subset of nodes i.e. a DSG from the subgraph. We then directly merge it into a super node at Line 20. Further we remove the member nodes of this DSG from summary graph and add the super node in the summary.

4.3.2 Super Node Creation using Incremental Merge

In Section 4.3.1, we directly merge the DSG into a super node. Since a DSG has high overlap of common neighbors, so a super node results with low edge corrections. However, the auto pruning consumes much time since we compare the neighbors of each node one by one with the neighbors accumulated in previous iteration(s). This delays the creation of super nodes till auto pruning happens.

Algorithm 2 Dense Subgraph-wise Merge

Require: Subgraph $Subgraph_q$, Query node q
Ensure: Super Node
1: Sort the nodes based on degree count
2: $UnionNbrs \leftarrow Nbrs_q$
3: **if** (q has lowest degree) **then**
4: **while** $(degree(v) \in Subgraph_q \geq degree(q)$ **do**
5: $UnionNbrs \leftarrow Nbrs_v$
6: **if** $(\%CommonNbrs > \%NonCommonNbrs)$ **then**
7: $DSG \leftarrow v$
8: **else**
9: $Break$ //$AutoPruning$
10: **end if**
11: **end while**
12: **else if** q has highest degree **then**
13: **while** $(degree(v) \in Subgraph_q \leq degree(q)$ **do**
14: $Filter\ DSG\ like\ lines\ 5-10$
15: **end while**
16: **else**
17: $Move\ towards\ nodes\ having\ higher$ $and\ lower\ degree\ than\ q$
18: $Filter\ DSG\ like\ lines\ 5-10$
19: **end if**
20: $SuperNode \leftarrow DSG$
21: $V_s \leftarrow V_s - DSG \cup SuperNode$

To speed up the super node creation time, we use an alternate procedure for auto pruning. Here we sort the nodes based on their cost reduction, Definition 3. Using CR, we compute the amount of compression to achieve for each member node in the subgraph against q. This clearly distinguish the nodes which cannot be merged with q. A node giving highest CR against q is merged first with it to create a super node. The algorithm 3 describes the super node creation in incremental manner.

Definition 3. (*Cost Reduction*) Given a pair of the nodes $u, v \in V$, the cost reduction (CR) from their merge is a ratio defined as the cost after merging u and v (into a new supernode w) divided by the combined cost of u and v before their merge. The equation (1) shows how to compute the CR. Here c_u, c_v and c_w are the degrees of the nodes u, v and w respectively [11].

$$CR(u, v) = \frac{c_u + c_v - c_w}{c_u + c_v} \tag{1}$$

5. EXPERIMENTS

In this section, we evaluate the proposed method on publicly available real world graphs from Stanford datasets collection[1], summarized in Table 1. We evaluate the proposed

[1]http://snap.stanford.edu/data/index.html. Last accessed on 07/12/2014

Algorithm 3 Incremental Merge

Require: Subgraph $Subgraph_q$, Query node q
Ensure: Super Node
1: Sort the nodes in $Subgraph_q$ against q using equation 1
2: **for each** $v \in Subgraph_q$ **do**
3: **if** $(CR(u, v) > 0)$ **then**
4: $supernode \leftarrow u \cup v$ $//u = q$ if $1st$ $iteration$ $otherwise$ $u = SuperNode$
5: $V_s \leftarrow V_s - \{u, v\} \cup SuperNode$
6: **else**
7: $Break\ Prune$
8: **end if**
9: **end for**

method on three evaluation criteria: execution time, cost of summary and compression. For execution time evaluation, we compare the execution time of Auto Pruning using DSG-wise merge and incremental merge. The execution time for pruning using MDM is very high so we do not consider it for comparison. To compare the cost of summary and compression, we evaluate the pruning methods with MDM [2].

We use the Intel Core i7-3960X with 3.30GHz processor and 36 GB main memory, having 64 bit Windows 7 enterprise edition for experiments. We create the subgraph for each comparison by sequentially traversing the given number of edges from the entire graph, stored in edge list format.

5.1 Execution Time Evaluation

Figure 2 displays the execution time comparison of super node creation by direct DSG-wise merge and incremental merge. We observe that DSG-wise merge consumes greater time than its counterpart. The reason is that given a subgraph, DSG-wise merge iteratively compares the percentage of common as well as non-common neighbors for each node. These comparisons are performed for each query node along with their all 2-hops away neighbors. In a large sized graph, the connectivity between the nodes is high. As a result of this, the 2-hops neighbors count for each node is usually high, which increases the computation time to extract DSGs. On the other hand in incremental merge, we keep on merging the nodes giving positive CR against the query node or super node created in previous iteration. Thus there is no direct filtration of DSGs. Rather all the nodes whose CR is positive is termed a DSG for a given query node. This makes the incremental merge to show a better execution time than the DSG-wise merge.

5.2 Summary Cost and Compression Evaluation

Figures 3 and 4 respectively display the comparison of summary cost and compression ratio by direct DSG-wise, incremental and merge using MDM. Compression ratio is the ratio of the summary cost and the original cost. A lower percentage of reduction cost indicates a better compression. Recall that the summary cost is the sum of super edges and edge corrections.

In both the datasets, we observe that the performance of the proposed approach is better than that using MDM. MDM does not show better performance since it aims to remove only those nodes who have minimum degree in the subgraph. At the same time, it cannot differentiate between the nodes who although have good contribution in increasing the density of the subgraph but have many non-common neighbors too. Thus it results in many edge corrections. On the other hand, in proposed approach merging the DSGs already ensure least edge corrections, thus result in higher performance.

[2]We improve the proposed method and provide its comparison with some other existing works in the journal version of this paper [9].

Table 1: Description of Datasets

	DBLP	Web Stanford
Nodes (N)	3,17,080	2,81,903
Edges (E)	10,49,866	23,12,497
Density (E/N)	3.31	8.20

It is interesting to observe that DSG-wise merge shows better performance than the incremental merge. The difference between the performance increases with the increase in the size of the graph, from DBLP to Web Stanford. In DBLP, the connectivity between the nodes is low thus the size of DGSs using both the approaches is similar; thus the result is almost similar. On the other hand, the density of the Web Stanford graph is high so the nodes have much varying degree than in DBLP. The auto pruning for DSG-wise merge effectively filter all the nodes having higher percentage of non-common neighbors. However the incremental merge keep on merging nodes till positive CR is achieved. Since a pair of nodes even sharing single common neighbor but having many non-common neighbors, produces positive CR, thus such merge yield many edge corrections. It is possible to stop the further merging in incremental merge by using a threshold for CR. However the right selection of a threshold for any task is a hard problem.

6. CONCLUSION AND FUTURE WORK

In this paper, we propose a graph summarization approach where we locally find the dense subgraphs using query nodes. A local search is effective since it searches the dense subgraphs by avoiding the computationally expensive traversal of the entire graph. Our summarization technique is valuable since we use an information theoretic approach to create a highly compact and lossless summary graph. We propose an effective dense subgraph discovery concept to create super nodes from the nodes having higher ratio of common to that of non-common neighbors.

In future, we plan to make our proposed approach applicable to graphs having billions of nodes and edges.

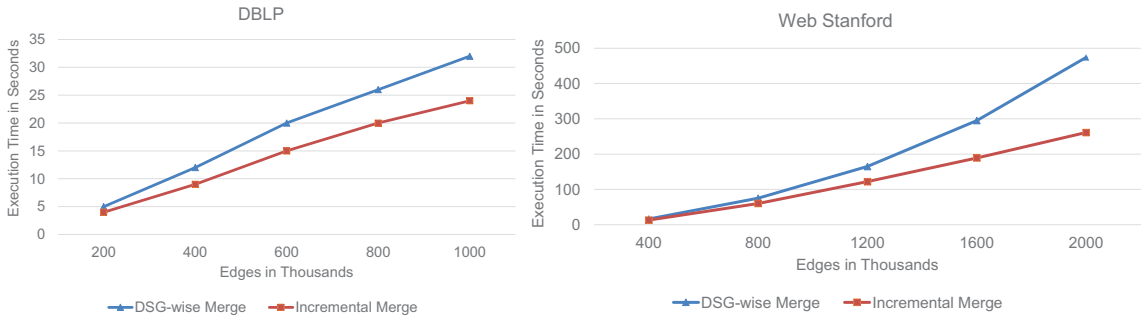
7. ACKNOWLEDGMENT

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

8. REFERENCES

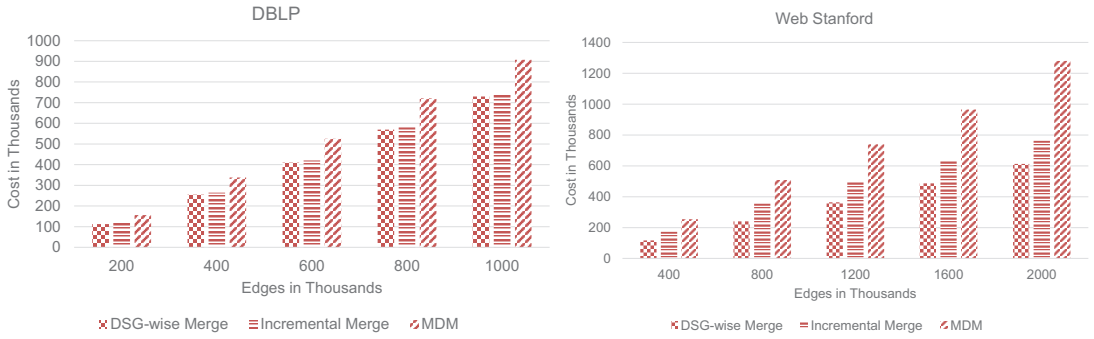
[1] Y. Asahiro, K. Iwama, H. Tamaki, and T. Tokuyama. Greedily finding a dense subgraph. *Journal of Algorithms*, 34(2):203–221, 2000.

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



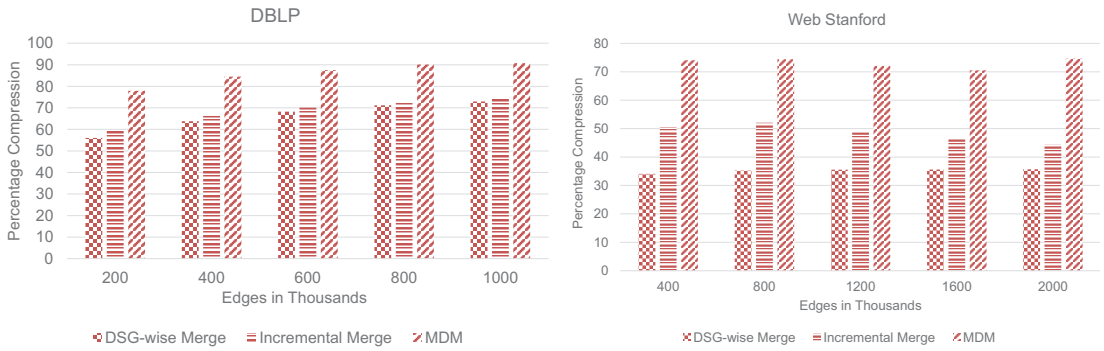
(a) (b)

Figure 2: Execution Time Comparison. Lower the better.



(a) (b)

Figure 3: Summary Cost Comparison. Lower the cost, better the effectiveness.



(a) (b)

Figure 4: Graph Compression Comparison. Lower the reduction cost, higher the compression.

[3] 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*, pages 95–106. ACM, 2008.

[4] M. Charikar. Greedy approximation algorithms for finding dense components in a graph. In *Approximation Algorithms for Combinatorial Optimization*, pages 84–95. Springer, 2000.

[5] 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*, pages 219–228. ACM, 2009.

[6] W. Cui, Y. Xiao, H. Wang, J. Hong, and W. Wang. Local search of communities in large graphs. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*. ACM, 2014.

[7] C. Hernández and G. Navarro. Compressed representations for web and social graphs. *Knowledge and Information Systems*, pages 1–35, 2013.

[8] 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*, pages 300–309. IEEE, 2011.

[9] K. U. Khan, N. Waqas, and Y.-K. Lee. Set-based approximate approach for lossless graph summarization. *The Computing Journal- Under Review*, 2014.

[10] S. Khuller and B. Saha. On finding dense subgraphs. In *Automata, Languages and Programming*, pages 597–608. Springer, 2009.

[11] 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*, pages 419–432. ACM, 2008.

[12] J. Rissanen. Modeling by shortest data description. *Automatica*, 14(5):465–471, 1978.

[13] S. B. Seidman. Network structure and minimum degree. *Social networks*, 5(3):269–287, 1983.

[14] M. Sozio and A. Gionis. The community-search problem and how to plan a successful cocktail party. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 939–948. ACM, 2010.

[15] 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*, pages 567–580. ACM, 2008.

[16] 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*, pages 965–973. ACM, 2011.