Sparsifying Networks while Preserving Properties

Gaurav Viramgami 19110106
Indian Institute of Technology Gandhinagar, India
viramgami.g@iitgn.ac.in

Hrushti Naik 19110088
Indian Institute of Technology Gandhinagar, India
hrushti.n@iitgn.ac.in

Manas Mulpuri 19110093
Indian Institute of Technology Gandhinagar, India
mulpuri.m@iitgn.ac.in

Abstract

Network sparsification is useful for fast clustering, community detection, node classification, etc. on the massive datasets. We have focused on reducing the number of edges while retaining the structures and denseness of the clusters intact. This reduces the problem to finding the edges whose removal does not affect the denseness of the clusters, i.e. finding how likely an edge is to be part of a cluster. We have implemented three already existing sprasification algorithms including Random Sparsification, Global Sparsification, and Local Sparsification. We also experimented by making some changes to these existing algorithms which gave good results over multiple runs. We also propose a new algorithm based on Independent Set of the graph to find the edges whose removal retains the clusters. We have evaluated these methods on five different networks using the modularity score.

2012 ACM Subject Classification Computing methodologies

Keywords and phrases Network Sparsification, Jaccard Similarity, Global Sparsification, Local Sparsification, Independent Set

1 Introduction

Graph sparsification refers to the reduction of the size of networks while preserving structural and statistical properties of interest. Reduction of the size of a network can refer to either node sampling (nodes and edges from the original network are discarded) or edge sampling (preserves all nodes and reduces the number of edges only). In our project, we perform edge sampling - reducing the edge set only, while keeping all nodes of the original graph.

The structural properties of a graph refer to properties like shortest paths, cuts, spectral measures, or network modularity. These properties need to be retained. In addition to the network, we also consider as input a set of communities. The goal is to sparsify the network so as to preserve the network structure with respect to the given communities.

Sparsification is done for various reasons. Some of its applications include:

1. Information Visualisation: Real-life networks are gigantic, with over thousands of nodes and even more edges. Visualising the complete graph is overwhelming, and often, more information can be shown when the graph is reduced. The human eye better detects network structures when a fraction of the edges are properly selected.
2. Improved Processing: By disregarding a large fraction of edges that are unimportant for the task, running times of graph and network analysis algorithms can be reduced. This way, it can improve the processing of networks. It can also be thought of as lossy compression of a graph.
3. From a network science perspective, sparsification can provide valuable information about the importance of relationships and the participating nodes. Given that a sparsification

method tends to preserve a certain property, the method can be used to differentiate between essential and non-essential edges.

2 Problem Formulation

There are multiple sampling and sparsifying algorithms known in theory, but most of them have not been evaluated for community detection strategies. The goal of this project is to take a class of graph sampling techniques, implement them and then evaluate the performance on selected datasets for community detection. We then evaluate the performance of each technique. We also propose some additional changes that can be done on the existing techniques as well as a new sparsification technique which can also be useful for network clustering.

3 Dataset

The datasets required for this project are large-scale networks, with information about ground truth communities. We have used 5 different network datasets as described below:

3.1 email-Eu-core network

The first dataset used in our project is the SNAP email-Eu-core dataset [3]. The network in this dataset is generated using email data from a large European research institution. There is an edge (u, v) in the network if person u sent person v at least one email. The e-mails only represent communication between institution members (the core), and the dataset does not contain incoming messages from or outgoing messages to the rest of the world. The dataset also contains "ground-truth" community memberships of the nodes. Each individual belongs to exactly one of 42 departments at the research institute. We have used this dataset as an undirected network. The statistics of the dataset are given in Table 1.

3.2 General Relativity and Quantum Cosmology collaboration network

The second dataset is the GR-QC collaboration network dataset [3]. It covers scientific collaborations between authors papers submitted to General Relativity and Quantum Cosmology category. Each author is represented as a node. There is an undirected edge between author i and author j, if author i has co-authored a paper with author j. If the paper is co-authored by k authors, then it generates a complete graph between the k nodes of these authors. The statistics of the dataset are given in Table 2.

3.3 High Energy Physics - Theory collaboration network

The third dataset is the HEP-TH collaboration network dataset [3]. It covers scientific collaborations between authors papers submitted to High Energy Physics - Theory category. In this dataset also authors are represented as nodes, and edge between two authors denote they have co-authored in some paper. For k co-authors of a paper, a complete graph will be made between all co-authors. The statistics of the dataset are given in Table 3.

3.4 Social circles: Facebook

The fourth dataset contains the social circles (friends lists) from Facebook [3]. The data was collected from survey participants from the Facebook app. Nodes represent the users

(profiles) and undirected edges represent followers. The statistics of the dataset are given in Table 4.

Dataset Statistics	
Nodes	1005
Edges	25571
Nodes in largest WCC	986 (0.981)
Edges in largest WCC	25552 (0.999)
Nodes in largest SCC	803 (0.799)
Edges in largest SCC	24729 (0.967)
Average clustering coefficient	0.3994
Number of triangles	105461
Fraction of closed triangles	0.1085
Diameter (longest shortest path)	7
90-percentile effective diameter	2.9

Table 1 email-Eu-core Network

Dataset Statistics	
Nodes	5242
Edges	14496
Nodes in largest WCC	4158 (0.793)
Edges in largest WCC	13428 (0.926)
Nodes in largest SCC	4158 (0.793)
Edges in largest SCC	13428 (0.926)
Average clustering coefficient	0.5296
Number of triangles	48260
Fraction of closed triangles	0.3619
Diameter (longest shortest path)	17
90-percentile effective diameter	7.6

Table 2 GR-QC Collaboration Network

Dataset Statistics	
Nodes	9877
Edges	25998
Nodes in largest WCC	8638 (0.875)
Edges in largest WCC	24827 (0.955)
Nodes in largest SCC	8638 (0.875)
Edges in largest SCC	24827 (0.955)
Average clustering coefficient	0.4714
Number of triangles	28339
Fraction of closed triangles	0.1168
Diameter (longest shortest path)	17
90-percentile effective diameter	7.4

Table 3 HEP-TH Collaboration Network

Dataset Statistics	
Nodes	4039
Edges	88234
Nodes in largest WCC	4039 (1.000)
Edges in largest WCC	88234 (1.000)
Nodes in largest SCC	4039 (1.000)
Edges in largest SCC	88234 (1.000)
Average clustering coefficient	0.6055
Number of triangles	1612010
Fraction of closed triangles	0.2647
Diameter (longest shortest path)	8
90-percentile effective diameter	4.7

Table 4 Social circles: Facebook

3.5 LastFM Asia Social Network

The fifth dataset is the LastFM Asia Social Network dataset [3]. LastFM users from Asian countries are represented as nodes and edges represent the mutual follower relationships between them. The possible tasks that can be performed on this dataset are multinomial

node classification, link prediction, community detection and network visualization. The statistics of the dataset are given in Table 5.

Dataset Statistics	
Nodes	7624
Edges	27806
Density	0.001
Transitivity	0.179

Table 5 LastFM Asia Social Network

4 Dataset Preprocessing

All of the datasets mentioned in the above section, have been used as undirected and unweighted networks. We have also removed self edges from all of the networks if they are present. Then we have created an adjacency list from the edges and then implemented the sparsification algorithms.

5 Sparsification Algorithms

A network sparsification algorithm should ideally have the following properties [5]:

1. The task for which we are doing sparsification i.e. clustering, community detection, node classification, etc. should be much faster to solve on sparsed graph as compared to the original graph.
2. The accuracy of the task (clustering, community detection, etc.) on sparsed graph should be close to its accuracy on the original graph.
3. The sparsification algorithm itself should be fast on massive networks.

In all of the implemented algorithms, we mainly focus on removing the edges which are not likely to be part of any cluster (i.e. edges joining two clusters) as removing them will not affect the structure and denseness of the cluster much. We try not to remove edges which are highly likely to be inside a cluster, as it will affect the denseness of the clusters. So, we have narrowed down our problem to finding out which edges are not likely to be part of any cluster and will be part of the set of edges joining two different clusters.

So, we need to give an edge score to each edge of the network using some measures to denote the chance of that edge to be inside a cluster or between clusters. Some of the different measures to perform this task are explained in the below.

5.1 Measuring Methods

5.1.1 Edge Betweenness Centrality

The edge betweenness centrality [4] of an edge e is proportionate to the total number of shortest paths between any two vertices that pass through the edge e. Edges with higher edge betweenness centrality are the bottleneck edges of the graph and thus they are highly likely to be part of the set of edges joining two different clusters. Let n be the number of nodes of the graph and m be the number of edges of the graph. Then finding edge betweenness centrality of every edge takes $\mathcal{O}(mn)$ time [4].

5.1.2 Similarity in adjacency lists

The edge e is more likely to be part of a cluster if both of its endpoints have highly overlapping adjacency lists. We can find the similarity between two adjacency lists using Jaccard similarity. In statistics, it is a measure of how similar two sets are. In this case, the sets are the adjacency lists of the given nodes, i.e. neighbours of the given nodes. To assign an edge score to an edge *(u, v)*, we use the Jaccard similarity between the adjacency lists *Adj(u)* and *Adj(v)* as follows:

$$JS(u,v) = \frac{|Adj(u) \cap Adj(v)|}{|Adj(u) \cup Adj(v)|} \tag{1}$$

Let *T(u, v)* denotes the total number of common neighbours of nodes u and v, and *d(u)* and *d(v)* denotes the degree of nodes u and v respectively. Then, the Jaccard similarity of nodes u and v can also be written as below:

$$JS(u,v) = \frac{T(u,v)}{d(u) + d(v) - T(u,v)}$$

Edges with higher Jaccard similarity are more likely to be a part of the cluster. It will take $(d(u) + d(v))$ operations to find the Jaccard similarity of two nodes given their adjacency lists, where *d(u)* and *d(u)* are the degree of node u and v respectively. On each node u, we need to apply Jaccard similarity *d(u)* times as there will be *d(u)* edges to this node. Thus, the total number of operations for finding Jaccard similarity of all the edges will be at max $\sum_i d(i)^2$ and at least nd_{avg}^2, where n is the number of nodes of the graph, and d_{avg} is the average degree of the graph.

5.1.3 Triangle count of edges

A triangle is a complete graph of 3 vertices. The triangle count of an edge e denotes the total number of triangles of which e is part of in the graph. Edges with higher triangle count are more likely to be a part of the denser region, i.e. inside some cluster.

We denote the fraction of edges that we want to retain in the network as *sparse_ratio* and total edges of the network as m. We are using similarity in adjacency lists as the measure.

5.2 Existing Algorithms

We have implemented three already existing sparsification algorithms [5] which are explained below.

5.2.1 Random Sparsification

In Random Sparsification [2], we uniformly and randomly select $m \times sparse\_ratio$ edges and remove the remaining edges from the network. We do not use any edge score measure in this method. The advantage of random sparsification is that a linear time algorithm in the number of edges can be implemented for it, and since the selection is uniform and random each edge selection can be done parallelly.

5.2.2 Global Sparsification

In Global Sparsification [5], we calculate the Jaccard similarity for each edge using Equation 1, and assign it as the edge score for that edge. Then, we sort the edges in descending order of their edge score, and take the top *sparse_ratio* fraction of the edges to retain. We remove the remaining edges from the network.

5.2.3 Local Sparsification

Local Sparsification [5] is a modified method of global sparsification. In global sparsification, it may happen that the network has a dense cluster as well as a sparse cluster. But the algorithm will remove nearly all the edges from sparse cluster as edges in sparse cluster will have low Jaccard similarity. In Local sparsification, we define $e = sparse\_exponent$ which can be calculated from the *sparse_ratio*. Then, for each node v in the network, we take the edges whose one endpoint is v, sort them in descending order of their Jaccard similarity and choose top $d(v)^e$ many edges to retain. We remove the remaining edges from the graph whose one endpoint is v. Here, $d(v)$ is the degree of the vertex v. This algorithms ensures that we select at least one edge incident to each node, and thus also retaining the sparse clusters in the resulted network.

5.3 Additional changes in the existing Algorithms

We experimented with some changes in these existing algorithms. These additional methods are explained below:

5.3.1 Probability based Global Sparsification

This method is nearly same as global sparsification. Only change is during the selection of edges. In this method, instead of choosing top *sparse_ratio* fraction of edges, we sample *sparse_ratio* fraction of edges, where each edge will have a selection probability proportionate to its edge score, i.e. Jaccard similarity.

5.3.2 Probability based Local Sparsification

This method is nearly same as local sparsification. Here too, the only change is during the selection of edges. For each node v, instead of choosing top $d(v)^e$ fraction of edges from the entire graph, we sample $d(v)^e$ fraction of edges, where each edge will have a selection probability proportionate to its edge score, i.e. Jaccard similarity.

6 Proposed Algorithm - Independent Set

In the above mentioned existing algorithms (Random, Global and Local sparsification), we focus on only removing the edges which are not part of any cluster. The main idea behind this algorithm is to also remove some of the edges which are inside clusters along with the edges which are not part of the clusters such that the same clusters can also be identified in the resulted network. Observe that if there exist some dense cluster in the network, and we remove some of the edges from that cluster, then also there is a high chance that we will retain that cluster as it is still relatively denser that the other regions of the network. Intuitively, we can make all the clusters as sparse as the sparsest cluster of the network, and there is a high chance that the same clusters will still be identified in the sparsified network. To implement the same, we will use the Independent Set of the given network or graph.

▶ **Definition 1.** ***Independent Set*** *of an undirected graph G is the set of vertices of G in which no two vertices are adjacent to each other. Maximum sized Independent Set of the graph G is called the* ***Maximum Independent Set****.*

Let I be the Maximum Independent Set of the graph G. So, no two vertices of the set I are adjacent to each other. In some sense there is a high chance that all of the vertices of the Independent Set I are part of different clusters, i.e. it is very less probable that more than k vertices of the set I belongs to the same cluster for some $k > 0$.

▶ Note. We can upper bound k with the max size among Maximum Independent Set of the clusters of the given network.

Now, assuming that each vertex of the set I belongs to different clusters, for each vertex $v \in I$, removing f ($f \geq 0$) edges incident to v will make all of the clusters sparse. But their relative denseness will remain almost same. So, the idea is to take all of the vertices of the Maximum Independent Set of the graph G, find the average degree d_{avg} of these vertices, and then for the vertices whose degree is greater than d_{avg}, remove some of the incident edges to make its degree equal to d_{avg}.

One critical problem in the above approach is that finding the Maximum Independent Set is NP-Complete. So, we will need to use approximation algorithm for the same. We will define Vertex Cover of a graph as it will be required for the approximation algorithm.

▶ **Definition 2.** ***Vertex Cover*** *of an undirected graph G is the set of vertices of G such that for every edge (u, v) of G, either u or v is in the Vertex Cover. Minimum sized Vertex Cover of the graph G is called the* ***Minimum Vertex Cover****.*

▶ **Lemma 3.** *Let G be an undirected graph and V is the set of all vertices of G. Observe that, if set S is the vertex cover of G, then set $V - S$ is the Independent Set of G and vice versa.*

Proof. Suppose the set $V - S$ is not an Independent Set. Then from the definition of the Independent Set, there exist and edge $e = (u, v)$ such that $u \in V - S$ and $v \in V - S$. But it means both of the endpoints of the edge e are in the set $V - S$, and thus none of them are in the set S, which contradicts the assumption that S is a Vertex Cover. ◁

▶ Note. The above lemma also infers that if S is the Minimum Vertex Cover of the graph G, then $V - S$ will be the Maximum Independent Set of G, where V is the set of all vertices.

Finding Minimum Vertex Cover is also NP-Complete. So, we will first find the Vertex Cover S using approximation algorithm and from the above Lemma 3, the set $V - S$ will give the Independent Set. We have used logn-approximation algorithm for the Vertex Cover using highest degree algorithm, which will give us an Independent Set of size, $|I| \geq \log n|OPT| - n(\log n - 1)$, where, OPT is the Maximum Independent Set and n is the total number of vertices in G. We can also use 2-approximation algorithm for Vertex Cover using Maximal Matching, which will give an Independent Set of size, $|I| \geq 2|OPT| - n$. Independent Set can also be found using Minimum Degree algorithm, which gives $\frac{1}{\Delta+1}$-approximation, i.e. $|I| \geq \frac{|OPT|}{\Delta+1}$, where Δ is the maximum degree of any vertex in G.

6.1 Independent Set Sparsification (Random Selection)

Suppose we want the degree of any vertex of the Independet Set to be less than or equal to d. Let $d(v)$ denote the degree of the vertex v. Now, in this method, we first find the Independent Set I as described in the algorithm, and then for each vertex $v \in I$, if $d(v) > d$,

we select $d(v) - d$ edges uniformly and randomly from all of the incident edges to v, and remove them from the network. Here also, we can parallelize the process of selecting edges for each vertex $v \in I$.

6.2 Independent Set Sparsification (Top Edges Selection)

In this method we use the similarity measure to decide which edges to remove. After finding the Independent Set I, for each vertex $v \in I$, if $d(v) > d$, then we find the Jaccard similarity for all of the edges incident to v, sort them in ascending order of Jaccard similarity and remove the top $d(v) - d$ edges from it.

6.3 Independent Set Sparsification (Probability-based Selection)

This method is nearly same as the above method. The only change is during the selection of edges. In this method, instead of choosing the top $d(v) - d$ edges, we sample d edges from $d(v)$ edges, where each edge will have a selection probability proportionate to its edge score, i.e. Jaccard similarity, and we remove the edges which are not sampled.

7 Implementation

For the implementation of above algorithms, we have used `NumPy` and `Pandas` libraries for dataset preprocessing, and `NetworkX` and `Pyvis` libraries for network visualization. Link to the GitHub repository containing all the code files used in the project:
https://github.com/GauravViramgami/CS328-Network-Sparsification.

8 Evaluation and Results

Modularity: Modularity measures the relative density of edges inside communities with respect to edges outside communities. It ranges between -0.5 (non modular clustering) to 1.0 (fully modular clustering).
Louvain Method: We have used Louvain method [1] for the community detection on the original network as well as the sparsed networks generated by different algorithms. It focuses on optimizing the value of modularity using heuristic algorithms. First small communities are detected by optimizing the modularity on all nodes locally, then each of the small community is grouped into one node and the first step is repeated on these nodes.

We have used `CDlib` library which has the Louvain method implemented, and it also gives the modularity score based on the detected communities. The results obtained are as follows:

8.1 email-Eu-core network

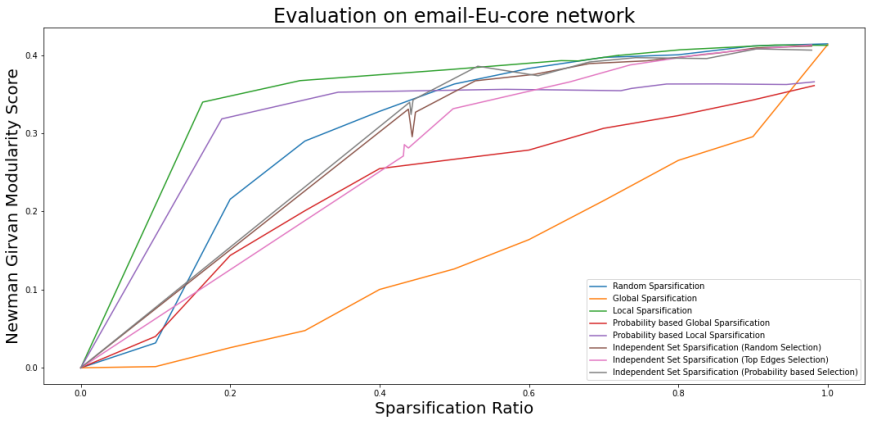


Figure 1 Sparsification Ratio vs Modularity Score

(a) Random Sparsification (f = 1.0)

(b) Random Sparsification (f = 0.7)

(c) Random Sparsification (f = 0.5)

(d) Random Sparsification (f = 0.3)

(e) Global Sparsification (f = 1.0)

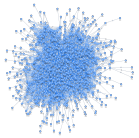
(f) Global Sparsification (f = 0.7)

(g) Global Sparsification (f = 0.5)

(h) Global Sparsification (f = 0.3)

(i) Local Sparsification (f = 1.0)

(j) Local Sparsification (f = 0.72)



(k) Local Sparsification (f = 0.66)

(l) Local Sparsification (f = 0.5)

(m) Independent Set Sparsification (f = 0.97)

(n) Independent Set Sparsification (f = 0.75)

(o) Independent Set Sparsification (f = 0.60)

(p) Independent Set Sparsification (f = 0.44)

Figure 2 email-Eu-core network Visualization

8.2 General Relativity and Quantum Cosmology collaboration network

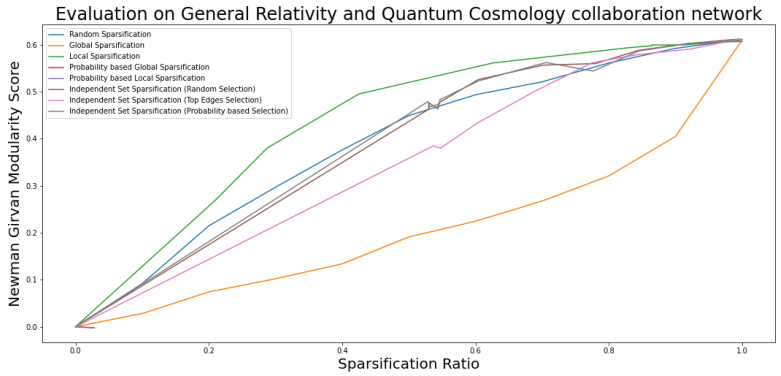


Figure 3 Sparsification Ratio vs Modularity Score

8.3 High Energy Physics - Theory collaboration network

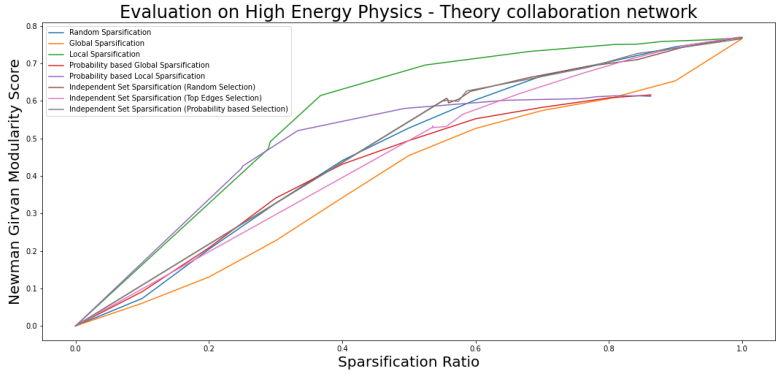


Figure 4 Sparsification Ratio vs Modularity Score

8.4 Social circles: Facebook

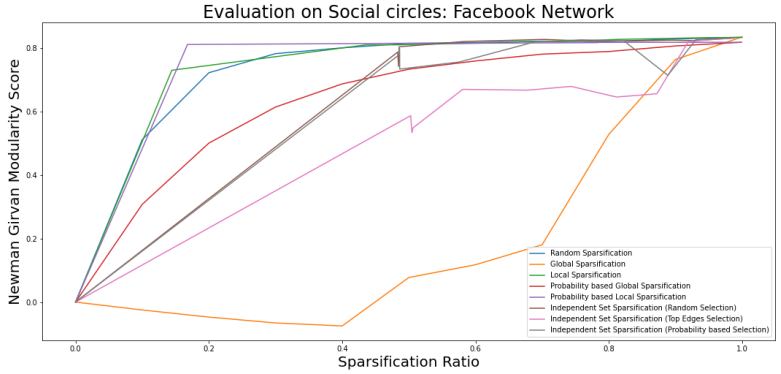


Figure 5 Sparsification Ratio vs Modularity Score

8.5 LastFM Asia Social Network

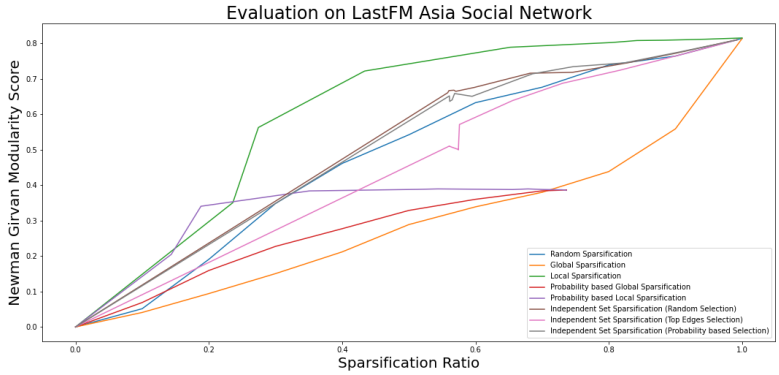


Figure 6 Sparsification Ratio vs Modularity Score

9 Conclusion

The Local Sparsification method gives a better modularity score as compared to others when the sparsification ratio is low. However, Global Sparsification gives the worst modularity score for low sparsification ratio. When we increase the sparsification ratio, Local Sparsification, Random Sparsification and Independent Set Sparsification give nearly the same results. Using the Independent Set method, edges cannot be removed after a certain limit, as the size of the independent set is fixed for a given network. So, in the plots there are no data points for Independent Set when sparsification ratio is below 0.4. To overcome this problem, we can also try combining Independent Set saparsification with Local sparsification in future. Overall, it gave good results when sparsification ratio is higher, as shown in the plots. Also, observe that in most of the cases, probability based sparsification yields better results than top edges in Global and Independent Set sparsification.

10 Contribution

The three of us (Gaurav, Hrushti and Manas) have worked on all aspects of this project together. We divided the readings (references) amongst ourselves, then discussed the main parts of those readings and decided which methods to implement. Implementation and evaluation were also done together by three of us and later, we brainstormed the idea of sampling and independent set, and experimented with it.

References

1 Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, 2008(10):P10008, 2008.

2 Michael Hamann, Gerd Lindner, Henning Meyerhenke, Christian L. Staudt, and Dorothea Wagner. Structure-preserving sparsification methods for social networks. *CoRR*, abs/1601.00286, 2016. URL: `http://arxiv.org/abs/1601.00286`, `arXiv:1601.00286`.

3 Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`, June 2014.

4 Mark EJ Newman and Michelle Girvan. Finding and evaluating community structure in networks. *Physical review E*, 69(2):026113, 2004.

5 Venu Satuluri, Srinivasan Parthasarathy, and Yiye Ruan. Local graph sparsification for scalable clustering. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data*, SIGMOD '11, page 721–732, New York, NY, USA, 2011. Association for Computing Machinery. `doi:10.1145/1989323.1989399`.