Spring 5-22-2019

Online Local Communities

Mrudula Murali
San Jose State University

Online Local Communities

A Project

Presented to

The Faculty of the Department of Computer Science

San Jose State University

In Partial Fulfillment

of the Requirements for

CS298

by

Mrudula Murali

Spring 2019

The Designated Project Committee Approves the Project Titled

Online Local Communities

by

Mrudula Murali

APPROVED FOR THE DEPARTMENTS OF COMPUTER SCIENCE

SAN JOSE STATE UNIVERSITY

Spring 2019

Dr. Katerina Potika	Department of Computer Science
Dr. Christopher Pollett	Department of Computer Science
Dr. Sami Khuri	Department of Computer Science

ABSTRACT

Online Local Communities

by Mrudula Murali

A community in a network is a group of nodes that are densely and closely connected to each other, get sparsely connected to the nodes outside the community. Finding communities in a large network helps solve many real-world problems. But detecting such communities in a complex network by focusing on the whole network is not feasible. Instead, we focus on finding communities around one or more seed node(s) of interest. Therefore, in this project, we find local communities. Moreover, we consider the online setting where the whole graph is unknown in the beginning and we get a stream of edges, i.e., pair of nodes, or a stream of higher order structures, i.e., triangles of nodes.

We created a new dataset that consists of web pages and their links by using the Internet Archive. We extended an existing online local graph community detection algorithm, called COEUS, for higher order structures such as triangles of nodes. We provide experimental results and comparison of the existing method and our proposed method using two public datasets, the Amazon and the DBLP as well as for our new Webpages dataset. In the experimental results, we see that the proposed method performs better than the existing method for one out of three test cases for the public dataset but not for our Webpages dataset. This is because the Webpages dataset has a large number of nodes with degree 1 which poses a problem for modified COEUS because it takes triangles as an input stream.

Keywords - Community detection, Local graph clustering, Online community

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my advisor Dr. Katerina Potika, who was a constant support throughout my project and steered me in the right direction.

I would also like to thank Dr. Christopher Pollet and Dr. Sami Khuri for their valuable feedback, encouragement and support.

I am grateful to my parents, brother and friends for their love and continuous support throughout my years of study.

Contents

Chapter

1 Introduction 1

1.1 Problem Statement 2

1.2 Applications of community detection 2

2 Preliminaries 4

2.1 Conductance 4

2.2 Community Participation 5

2.3 Count Min Sketch 5

3 Related Work 7

3.1 Online community detection 7

3.2 Overlapping community detection 8

3.3 Higher order community detection 8

3.4 Local community detection using Page rank 8

4 Datasets 10

4.1 Internet Archive Dataset 10

4.2 Creating webpage link dataset 12

5 Methodology 14

5.1 Community Detection via Seed-set Expansion on Graph Streams (COEUS) 14

5.1.1 COEUS Algorithm 14

5.2 Modified COEUS 20

6 Experimental Results 25

6.1 Dataset 25

6.1.1 Amazon 25

6.1.2 DBLP 26

6.1.3 Webpages 26

6.2 COEUS 28

6.3 Modified COEUS 29

6.4 Comparing results of COEUS and modified COEUS 31

7 Conclusion and Future Work 33

LIST OF REFERENCES 35

List of Tables

1 Graphs of our dataset 25

2 COEUS on Amazon dataset 29

3 COEUS on DBLP dataset 29

4 COEUS on Webpage dataset 29

5 Graphs of our dataset with triangles 30

6 Modified COEUS on Amazon dataset 30

7 Modified COEUS on DBLP dataset 30

8 Modified COEUS on Webpage dataset 31

9 Comparison of accuracy for COEUS and modified COEUS on Amazon dataset 33

10 Comparison of accuracy for COEUS and modified COEUS on DBLP dataset 33

11 Comparison of accuracy for COEUS and modified COEUS on Webpage dataset 34

List of Figures

1 COUNT-MIN Sketch update process. 6

2 A stream comprising the edges of an undirected graph and a set of communities initialized with a few seed nodes. 15

3 A subset of the Webpage network. 27

4 Degree distribution of the Webpage network. 27

CHAPTER 1

Introduction

Graphs are formed by a set of vertices (nodes) and edges. They model the real-world systems, like social networks, where nodes are people and edges are friendships or interests. One way to analyze complex networks is by finding communities (clusters). The community detection problem is defined as the one where we seek to partition the nodes into groups, sometimes disjoint [1, 2] sometimes overlapping [3, 4]. In this project, we extended an existing online local graph community detection algorithm, called COEUS, by considering motifs, also called higher order structures such as triangles of nodes. Additionally, we created a new Webpage dataset by using the Internet Archive. This dataset along with two public datasets, the Amazon and the DBLP were used to test the accuracy of the proposed method and compare it to the existing COEUS method.

There are many clustering algorithms to find communities in a graph, such as spectral clustering method. Also, there are some heuristic search methods that provide a better quality of community structure in a network. Recently, some algorithms first represent the graph by embedding the nodes into a low dimensional Euclidean space. But these distance-based algorithms are not appropriate for the real world networks of small world phenomena, where the edges between the nodes are small. Page rank can be used to find communities in a network since it provides a structural relationship between nodes in a graph. It is more effective than reducing the dimension of a large graph.

Page rank is the distribution of a random walk that stays put with probability α

at each time step and walks to a random neighbor with probability $\alpha - 1$. The unique solution to the linear system can be represented by the page rank vector pr_α(s). The distribution of a random walk is:

$$pr_\alpha(s) = \alpha s + (1 - \alpha) pr_\alpha(s) W. \tag{1}$$

Here, α is called the teleport probability which is a constant in [0, 1] and 's' is a starting vector (seed-set). W is the transition probability matrix. If the starting vectors are not uniform then it is personalized page rank.

1.1 Problem Statement

With the increasing popularity of online social networking services such as Facebook and Twitter, detecting communities becomes more relevant in the study of networks. In this era of big data, processing massive networks by considering it as a static graph poses a problem. Therefore, it is good to consider a data stream model, in which the edges of a graph is considered as streams [5].

1.2 Applications of community detection

Detecting communities help solve many real-world problems. Some of its applications are:

- Social networks - Understanding communities in social networks helps to preform recommendations to the users, understand the interests of users so that we can provide specific feeds to them.
- Fraudulent websites - Many false websites tend to link to each other. Finding communities of such websites is extremely useful because the whole network of fraudulent websites can be exposed by finding one.

- Biological networks - Learning preferences of a user from one network can be used to display related, useful ads for similar user/users in another network.
- Citation network - Finding communities in the citation network helps to identify the citation patterns of the authors and uncover the relationship between the disciplines.

CHAPTER 2

Preliminaries

In this chapter, we introduce some preliminaries that we will use in the rest of this project. The sections 2.1 and 2.2 remind the quality function used in community detection. In Section 2.3 we remind the well-known structure of sublinear space data for high-dimensional vector representation.

2.1 Conductance

Conductance is a popular objective function that is used for local community detection in many algorithms.

Definition 2.1.1. (Conductance) Let $G = (V, E)$ be an undirected graph and let $S \subset V$ be a set of graph nodes. The following equation defines the conductance of a cut $(S, \bar{S})$ as:

$$\phi_S = \phi_{\bar{S}} = \frac{\mid E_{S,\bar{S}} \mid\mid E \mid}{\mid E_S \mid\mid E_{\bar{S}} \mid} \tag{2}$$

The minimum conductance over all cuts is defined by the graph conductance ϕ_G.

A widely used quality function in the field of community detection is the conductance of a community. More specifically, conductance $\phi(C)$ of a community C is formally defined as:

$$\phi = \frac{adj(C, V \mid C)}{min(adj(C, V), adj(V \mid C, V))} \tag{3}$$

where:

$$adj(C_i, C_j) = \mid \{(u, v) \in E : u \in C_i, v \in C_j\} \mid$$

.

2.2 Community Participation

Community participation $cp(u)$ of a node u in a community, that measures a node's u participation level in a community. In particular the community participation of node u in community C is defined as:

$$cp(u) = \frac{\mid \{(u,v) \in E : v \in C\} \mid}{\mid \{(u,v) \in E\} \mid} \tag{4}$$

Community participation of a node in a community is the fraction of its adjacent nodes in the graph that are part of the community [5].

If community participation of one node is higher than the other, then the node with higher community participation is considered to be more closely connected to the community than the node which exhibits lower community participation

2.3 Count Min Sketch

The COUNT-MIN sketch for the representation of high-dimensional vectors is a well-known sub-linear space data structure. COUNT-MIN sketches enable the answering of fundamental queries efficiently and with strong guarantees of accuracy. They are especially useful to summarize data streams because they can handle updates at high rates.

Data Structure. A Count-Min (CM) sketch with parameters (ε, δ) is represented by a two-dimensional array of width w and depth d: $count[1,1]...count[d,w]$ counts. Set $w = \lceil \frac{e}{\varepsilon} \rceil$ and $d = \lceil \ln \frac{1}{\delta} \rceil$, given parameters (ε, δ). Initially, each entry in the array is zero. Furthermore, d hash functions

$$h_1...h_d : \{1...n\} \rightarrow \{1...w\}$$

are selected randomly from a pairwise-independent family [6].

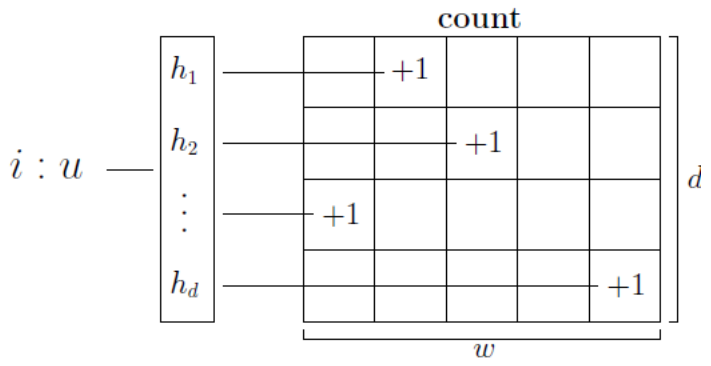

Figure 1: COUNT-MIN Sketch update process.

Update Procedure. When an update (i_t, c_t) arrives, which means item a_{i_t} is updated by a quantity of c_t, then c_t is added to one count in each row; h_j is used to determine the counter. Formally, set $\forall 1 \leq j \leq d$

$$count[j, h_j(i_t)] \leftarrow count[j, h_j(i_t)] + c_t$$

The array of wd counts is the space used by Count-Min sketches, that takes wd words, and d hash functions, each of which can be stored with two words when using the pairwise functions [6].

CHAPTER 3

Related Work

In this chapter, we give a brief review of the previous related works. Since our work uses streams of graphs and local communities detection through seed sets we present the related work that covers both as well the ones that cover each of these. Additionally, we will review some works on Page Rank based algorithms used for community detection.

3.1 Online community detection

In the work of [5], the authors propose a local community detection algorithm that receives the graph as an edge stream. They call their algorithm COEUS. By processing a stream of edges, without restriction on the arrival order, and maintaining limited information about the respective graph, such as the nodes' degrees, the community participation of nodes and the nodes in each community they manage to stay in sub-linear space to the number of edges. Additionally, they have two versions of their algorithm. In the first one, they greedily merge the endpoints of the new arriving edge to the same communities and check after some steps that their communities are not very big. In the second the quality of the new edge is considered. They introduce and use a new node centrality, called community participation, instead of page rank. As the last step, they determine in real-time the size of each community by removing some nodes. In their experiments, they measure time and space. Summarizing, their approach is one that can deal with large-scale community detection. In the description of our methods, more details will be given in order to compare and contrast.

3.2 Overlapping community detection

A local graph partitioning algorithm in presented in [7] that finds cuts with an approximate computation and use of the PageRank vectors. Each of the PageRank vectors they compute uses a seed node and then can use that vector to determine a cut that partitions the local graph into two communities. This cut is found through a sweep method over the vector and the computation of the conductance of the resulting sets.

3.3 Higher order community detection

A network motif is a higher-order structure and such structures are important aspects of the graph. A motif can be an edge or a triangle of nodes. In the work of [8], first the generalize the conductance to one that is a motif conductance and then extends the approximate Personalized PageRank with motifs, MAPPR algorithm, starting from a seed node and finding a local community such that the minimal motif conductance is achieved.

3.4 Local community detection using Page rank

PageRank vectors can be computed to calculate the importance of node u on other nodes. Jeh and Widom [9] presented an algorithm for computing these PageRank vectors. Making use of this PageRank vector technique, Andersen et al. [10] proposed a local graph partitioning algorithm. It can be used to find communities in an undirected graph for a given seed node(s). Sweep technique is used in this algorithm to sweep over the PageRank vectors which selects a set that minimizes or maximizes some scoring function. Conductance is one such scoring function. To find a good high-quality cluster, select a set with low conductance. Later, Andersen et al.

extended the local graph partitioning algorithm to accommodate strongly connected directed graphs in [11]. In this project, we will be using the page rank to determine which nodes to consider in the local cluster of the seed.

CHAPTER 4

Datasets

We will be using three datasets in our project for experiment purpose. Two of them are publicly available datasets, Amazon and DBLP [12] that are undirected and contain the ground-truth communities. The third dataset is one which we create from the web crawls. Creating webpage dataset is the first part of the project.

4.1 Internet Archive Dataset

We download the WARC files obtained from the Internet Archive [1]. WARC, or Web ARChive, is a successor to the previous ARC format used by the 1996 Internet Archive to store web crawls. The WARC format is standardized by the International Internet Preservation Consortium (IIPC), a consortium of national libraries, research laboratories, and technology organizations, with version 1.1 being the latest version from 2017.

The Internet Archive makes many of its web crawls available to the public. A typical web crawl is stored as a WARC file sequence where each WARC file, in turn, consists of a sequence of WARC records. Usually, a WARC file is used to store a gigabyte of data. Each record in it is often compressed using `gzip`, and these compressed records are concatenated, allowing the entire file to be decompressed using `gzip -d`, but also allowing individual records to be read and uncompressed without the need to decompress the entire file if an offset and a compressed length are known. A record starts with a line declaring the WARC format in use followed by a sequence of header-name value lines specifying record properties such as the type

[1] https://web.archive.org/web/*/warc

and date of the record. This is followed by a line of Content-Length and the actual content of the record is followed in turn. This content is a web page most often.

A smaller file called a CDX file is used to facilitate random access to a WARC file. It consists of a sequence of one line recording in a WARC file, each with meta information about one record which summarizes a single web document. The first line in the file is a legend to interpret the data, and the following lines contain the data to reference the corresponding host pages. The file's first character is the field delimiter used in the rest of the file. It is followed by the literal "CDX" and a space-separated list of letters used as column type codes. Below are some of the column type codes present in our WARC file.

- $k-$ checksum
- $g-$ warc file name
- $S-$ record size (compressed)
- $V-$ file offset (compressed)
- $N-$ massaged url
- $s-$ response code
- $m-$ file type
- $a-$ original url

Then the CDX record lines consist of space-separated fields in this format. For a 1 GB WARC, a compressed CDX file is about 20-30 MB, which makes it much more manageable.

4.2 Creating webpage link dataset

To create the Webpage dataset we read the CDX file line by line and obtain the necessary fields to extract each web document. The necessary fields are:

- $g-$ warc file name
- $S-$ compressed record size
- $V-$ compressed arc file offset
- $m-$ file type
- $a-$ original url

Decompressing the whole WARC file takes a lot of time and not an efficient approach. So we try to decompress only part of the WARC file where the web document is stored. This can be done by using the *offset* and the *record size* available in each of the CDX line for respective web document. In the creation of our Webpage dataset, we are interested with only HTML pages. So the *file type* helps us to filter only the HTML file type and ignore the rest. After we decompress part of the WARC file for a web document, we scan it to check for the web page links (URL). *original url* field helps us make a connection between the webpage URL and the links the webpage contains. Let's call each link in the webpage of a URL as *linked_url*. This process is detailed in Algorithm 1.

The *original url* and all the *linked_url* is stored in 'node' table. Each of the URL in the 'node' table is represented as a node in a graph and the link between *original url* and the *linked_url* is stored in 'edge' table and represented by an edge in the graph.

Algorithm 1 FetchWebLinks

```
1: Procedure FETCHWEBLINKS(cdx)
2:     for each line in cdx file do
3:         open WARC file
4:         seek(offset)
5:         read(record size)
6:         webpage = Decompress the read section of WARC file
7:         Scan the webpage to get the links cited in the webpage
8:     end for
9: end Procedure
```

The Webpage dataset we created by this process has 1,977,975 nodes and 2,484,651 edges. The ratio of nodes to edges is very less

CHAPTER 5

Methodology

In this chapter, we first explain the existing COEUS algorithm in section 5.1 and then provide a modification to this algorithm(proposed algorithm) in the section 5.2

5.1 Community Detection via Seed-set Expansion on Graph Streams (COEUS)

A community is generally thought to be a set of graph nodes that are closely connected and have very few links to the rest of the nodes of the graph [13]. A widely used [14], [15] community detection function is the conductance of a community as defined in definition 2.4.1. Conductance is the most used quality function when detecting communities. However, tracking the behavior of all possible communities as we process the edges of the stream is inefficient w.r.t time and space. Rather, we use community participation $cp(u)$ of a node u in a community here, which measures the level of participation of a node in a community.

Graph algorithms are unable to store and process the entire graph if the graph is large [16]. Graph stream algorithms, on the other hand, process a stream comprising the graph edges in the order in which these edges arrive over time using limited memory. Due to a large amount of data, COUNT-MIN sketches are used to store the frequency data.

5.1.1 COEUS Algorithm

In this section, the existing COEUS algorithm is explained in order to understand the proposed algorithm which is the modification of the existing COEUS. The

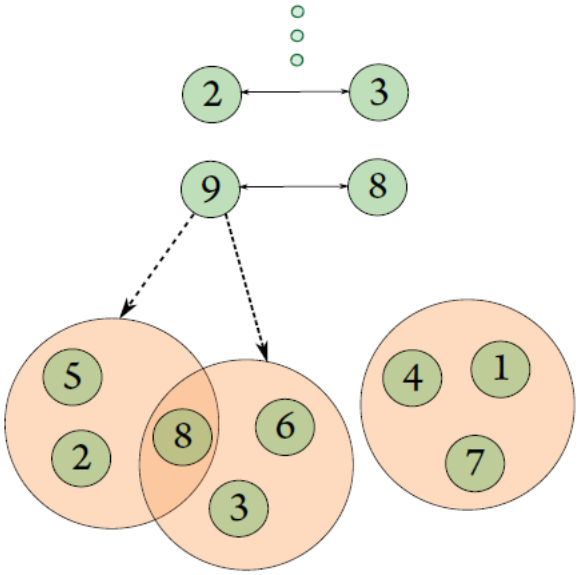

Figure 2: A stream comprising the edges of an undirected graph and a set of communities initialized with a few seed nodes.

pseudocode of COEUS is given in Algorithm 2.

COEUS algorithm takes two inputs: (i) A set of community seeds $K' = \{K_1, K_2, ..., K_s\}$, each of which is $K_i = \{k_1, k_2, .K_l\} \in V$ and (ii) a stream $S = (e_1, e_2, ..., e_m)$, where $e_i \in E$, and E is the set of edges of the undirected graph $G = \{V, E\}$ defined by S. Each edge in the graph stream is processed one at a time and added to the initial seed-set communities K' to extend it. At the end of this algorithm, we get a set of communities $C = \{C_1, C_2, ..., C_s\}$, with community C_i corresponding to K_i's seed set, as the output. This output is available at any point in time during the processing.

Line 1-7 of Algorithm 2 (COEUS algorithm) does the initialization of the seed-set communities. This is a simple procedure whereby we create an additional set to hold the nodes of the respective communities for each of the community seed sets. The seed-sets and community sets allow us to check whether a node is a seed or a

Algorithm 2 COEUS (S,K') SIMPLE

Input: A set of community seed-sets K', and a graph stream S.
Output: A set of communities C'.

```
1: for each K ∈ K' do
2:     C ← {};
3:     for each k ∈ K do
4:         C[k] = 1;
5:     end for
6:     C'.put(C);
7: end for
8: while ∃(u, v) ∈ S do
9:     degree_V[u]+ = 1;
10:    degree_V[v]+ = 1;
11:    for each C ∈ C' do
12:        if u ∈ C then
13:            degree_C[v]+ = 1;
14:        end if
15:        if v ∈ C then
16:            degree_C[u]+ = 1;
17:        end if
18:        if u ∈ C then
19:            C.put(v);
20:        end if
21:        if v ∈ C then
22:            C.put(u);
23:        end if
24:        processedElements+ = 1;
25:        if processedElementsmodW == 0 then
26:            C ← prune(C, s, degree_V, degree_C)
27:        end if
28:    end for
29: end while
```

member of a community efficiently at any time. Consider using Figure 2 as an example that we want three communities to be detected. COEUS begins with three seed sets describing these communities, namely $\{2, 5, 8\}$,$\{3, 6, 8\}$, and $\{1, 4, 7\}$. COEUS creates three community sets consisting of these nodes in this setting.

COEUS is ready to process the stream after initializing the communities (Lines

8-29). The size of the network is large, so we avoid storing the whole network. Hence we keep track of the following aspects as we process the stream of edges:

1. Each node's degree in a graph
2. community degree
3. nodes that form the community

We first increase the degree of each of the adjacent nodes in the graph by 1 for each incoming edge of the stream (Lines 9-10). We then examine whether each of the adjacent nodes is a member of the community for each community we wish to extend. If this is the case, we will increase the other node's community degree. Furthermore, if the other node is not a community member, the node will be added to the community (Lines 12-23). Returning to the example of Figure 2, when edge $\{9, 8\}$ arrives, the degree of both nodes 8 and 9 will first be increased by 1 by COEUS. After that, for each community, COEUS will examine whether nodes 9 or 8 are community members. This is true for two communities with node 8. Therefore, for both communities, COEUS will increase the community degree of node 9 by 1. Furthermore, to both communities to which node 8 belongs, COEUS will add node 9.

We want to focus on nodes for each community that is closely connected to each other. So we consider a window of size W for this purpose. The communities can freely grow in size during a window as new edges arrive. When the window closes, however, COEUS prunes all communities and keeps only each community's most highly involved nodes (Lines 25-27).

With Algorithm 3 and pruneComm function, which uses Eq 4, this process is detailed. We iterate over the nodes in a community and find the community par-

Algorithm 3 pruneComm

```
1: Procedure PRUNECOMM(C, s, degree_V, degree_C)
2:     minheap ← []
3:     for each c ∈ C do
4:         cp(c) = degree_C[c] / degree_V[c]
5:         if minheap.size() < s then
6:             minheap.push((c, cp(c)));
7:         else if cp(c) > minheap[0] then
8:             minheap.pop();
9:             minheap.push(c, cp(c));
10:        end if
11:    end for
12:    return set(minheap);
13: end Procedure
```

ticipation for that node in a community (Line 4). Then we push the community participation value of the node into a min-heap. We repeat this procedure until it reaches a set maximum size, 's' of the min-heap (Lines 5-6). Once the min-heap reaches its maximum size, the community participation value of a node is compared with the first value in min-heap because this would be the lowest community participation value (Line 7). If the community participation of a node is higher than the first value of min-heap, the first value of min-heap is removed from the min-heap and the community participation of the node is pushed into the min-heap (Lines 8-9). At the end of this procedure, we will have 's' nodes which exhibit the highest community participation value. The function outputs a set containing these 's' nodes (Line 12).

COEUS can be terminated at any point in time because the node members of a community are always readily available. In Algorithm 2 we consider a finite edge stream. So once all the edges in the stream are processed, Algorithm 2 terminates.

In Algorithm 2, we increase the community degree of a node by 1 if an adjacent member of that node is in the community. We do not take in to account the level

of involvement of a node in the community. But considering the level of involvement of a node to decide if a node belongs in a community is important since this helps us to differentiate between important nodes and irrelevant nodes. So COEUS offers a variant of Algorithm 2 by considering edge quality when increasing the degree of a community. This variation is based on the concept used in PageRank, that employs the network's link structure to improve over the in-degree measure [10].

We use Eq. 4 in our variation instead of increasing the node's community degree by 1 for all adjacent nodes that are community members. Eq. 4 is equal to the fraction of the node adjacent nodes which are also members of the community concerned. This fraction is essentially an estimate of the likelihood that a one-step random walk starting from the node will result in a node being a community member in question. Therefore, The involvement of each node in the community increases. If this value is high, then there is also a high likelihood of an adjacent node being a community member. Incrementing the community degree of a node using the community participation value of its adjacent node instead of 1 helps COEUS ensure that the nodes that exhibit strong ties in a community will have a significant contribution to the community than the nodes which exhibit weak ties.

Algorithm 4 details the approach described above and can replace Algorithm 2 lines 11 - 23. The difference in functionality is in Algorithm 4 lines 4 and 7, which use an Community participation estimate to increase a node's participation level in the community for the adjacent node in question.

One problem with COEUS is that it also includes the nodes which have lower community participation value. But these nodes are not relevant to the community as they do not exhibit strong ties with the community. COEUS resolves this issue by removing the nodes with the lowest community participation value.

Algorithm 4 COEUS addToCommByEdgeQuality

```
1: Procedure addToCommByEdgeQuality
2:     for each C ∈ C′ do
3:         if u ∈ C then
4:             degree_C[v] += degree_C[u] / degree_V[u];
5:         end if
6:         if v ∈ C then
7:             degree_C[u] += degree_C[v] / degree_V[v];
8:         end if
9:         if u ∈ C then
10:            C.put(v);
11:        end if
12:        if v ∈ C then
13:            C.put(u);
14:        end if
15:    end for
16: end Procedure
```

Algorithm 5 details drop-tail, which first ranks the nodes in the community in the decreasing order of their community participation value and then finds the mean distance between two adjacent nodes using their community participation value. The average distance is calculated by considering all the nodes in a community (Lines 5-10). Then the drop-tail iterates over all the nodes in each community in the decreasing order of their community participation value and finds the distance between that node and the previous node. If this distance is less than the average distance then that node is removed from the community. If drop-tail find the distance between two nodes greater than the average distance it terminates the algorithm. Because all the remaining nodes have a gap between them greater than the set threshold (Line 13-23).

5.2 Modified COEUS

In the previous section, we saw how [5] implements the COEUS to find communities of a graph stream. In this section, we will propose a modification to the existing

Algorithm 5 dropTail

```
1: Procedure DROPTAIL
2:     Ĉ ← reverseSort(C);
3:     totalDifference ← 0;
4:     previous ← 0;
5:     for each C ∈ Ĉ do
6:         if previous > 0 then
7:             totalDifference ← cp(c) − previous;
8:         end if
9:         previous ← cp(c);
10:    end for
11:    averageDifference ← totalDifference / (Ĉ.size()−1);
12:    previous ← 0;
13:    for each C ∈ Ĉ do
14:        if previous > 0 then
15:            difference ← cp(c) − previous;
16:        end if
17:        previous ← cp(c);
18:        if difference < averageDifference then
19:            Ĉ.remove(c);
20:        else
21:            break;
22:        end if
23:    end for
24: end Procedure
```

COEUS algorithm.

The COEUS algorithm proposed by Liakos et al. [5] considers a stream of edges as input. In our proposed algorithm we will consider the stream of triangles of nodes as an input.

The Modified COEUS algorithm takes two inputs: (i) A set of community seeds $K' = \{K_1, K_2, ..., K_s\}$, each of which is $K_i = \{k_1, k_2, .K_l\} \in V$ and (ii) a triangle stream $S = (t_1, t_2, ..., t_m)$, where $t_i \in T$, and T is the set of triangles of the undirected graph $G = \{V, E\}$ defined by S and T is of the form (e_1, e_2, e_3). Each triangle in the graph stream is processed one at a time and added to the initial seed-set

communities K' to extend it. At the end of this algorithm, we get a set of communities $C = \{C_1, C_2, ..., C_s\}$, with community C_i corresponding to K_i's seed set, as the output. This output is available at any point in time during the processing.

Line 1-7 of Algorithm 6 (Modified COEUS algorithm) does the initialization of the seed-set communities and is the same as we have seen in Algorithm 2. Modified COEUS is now ready to process the stream after initializing the communities (Lines 8-42). Because of the size of the network, even in modified COEUS, we don't keep the entire triangle stream. Instead, we keep track of the following aspects as we process the stream of edges:

1. Each node's degree in a graph
2. community degree
3. nodes that form the community

We first increase the degree of each of the adjacent nodes in the graph by 1 for each incoming triangle of edges of the stream (Lines 9-11). We then examine whether each of the adjacent nodes is a member of the community for each community we wish to extend. If this is the case, we will increase the other nodes' community degree. Furthermore, if the other nodes is not a community member, the nodes will be added to the community (Lines 12-36). The pruning on communities once it reaches the window size, is the same as the existing COEUS.

We also modify the Algorithm 4 of the existing COEUS to process nodes of three (triangles) instead of edges. The modified edge quality is given in Algorithm 7.

Algorithm 6 Modified COEUS (S,K')

Input: A set of community seed-sets K', and a triangle graph stream S.
Output: A set of communities C'.

1: **for each** $K \in K'$ **do**
2: $C \leftarrow \{\}$;
3: **for each** $k \in K$ **do**
4: $C[k] = 1$;
5: **end for**
6: $C'.put(C)$;
7: **end for**
8: **while** $\exists(u, v, w) \in S$ **do**
9: $degree_V[u] += 1$;
10: $degree_V[v] += 1$;
11: $degree_V[w] += 1$;
12: **for each** $C \in C'$ **do**
13: **if** $u \in C$ **then**
14: $degree_C[v] += 1$;
15: $degree_C[w] += 1$;
16: **end if**
17: **if** $v \in C$ **then**
18: $degree_C[u] += 1$;
19: $degree_C[w] += 1$;
20: **end if**
21: **if** $w \in C$ **then**
22: $degree_C[u] += 1$;
23: $degree_C[v] += 1$;
24: **end if**
25: **if** $u \in C$ **then**
26: $C.put(v); C.put(w)$;
27: **end if**
28: **if** $v \in C$ **then**
29: $C.put(u); C.put(w)$;
30: **end if**
31: **if** $w \in C$ **then**
32: $C.put(u); C.put(v)$;
33: **end if**
34: $processedElements += 1$;
35: **if** $processedElements mod W == 0$ **then**
36: $C \leftarrow prune(C, s, degree_V, degree_C)$
37: **end if**
38: **end for**
39: **end while**

Algorithm 7 Modified COEUS addToCommByEdgeQuality

1: **Procedure** addToCommByEdgeQualityModified
2: **for each** $C \in C'$ **do**
3: **if** $u \in C$ **then**
4: $degree_C[v] += \frac{degree_C[u]}{degree_V[u]}$;
5: $degree_C[w] += \frac{degree_C[u]}{degree_V[u]}$;
6: **end if**
7: **if** $v \in C$ **then**
8: $degree_C[u] += \frac{degree_C[v]}{degree_V[v]}$;
9: $degree_C[w] += \frac{degree_C[v]}{degree_V[v]}$;
10: **end if**
11: **if** $w \in C$ **then**
12: $degree_C[u] += \frac{degree_C[w]}{degree_V[w]}$;
13: $degree_C[v] += \frac{degree_C[w]}{degree_V[w]}$;
14: **end if**
15: **if** $u \in C$ **then**
16: $C.put(v)$;
17: $C.put(w)$;
18: **end if**
19: **if** $v \in C$ **then**
20: $C.put(u)$;
21: $C.put(w)$;
22: **end if**
23: **if** $w \in C$ **then**
24: $C.put(u)$;
25: $C.put(v)$;
26: **end if**
27: **end for**
28: **end Procedure**

CHAPTER 6

Experimental Results

In this chapter, we present an evaluation of our algorithms. In Section 6.1, we describe the datasets we use in our experiments. Then in Section 6.2, we evaluate the COEUS on three networks. Next, we evaluate the modified COEUS on the same 3 networks in Section 6.3. Finally in Section 6.4, we compare the results of COEUS and modified COEUS.

6.1 Dataset

As we mentioned already, our experiments include three datasets. Two of the them are publicly available [12], one is the Amazon co-purchasing network and the other is the DBLP co-authorship network. Both these datasets are undirected and contain the ground-truth communities. The last one is the one we created in the first part of the project, namely consisting of webpages. The details of the provided datasets are listed in Table 1.

Table 1: Graphs of our dataset

Dataset	**Type**	**Nodes**	**Edges**
Amazon	Co-purchasing	334,863	925,872
DBLP	Co-authorship	317,080	1,049,866
Webpages (own)	Link citation	1,977,975	2,484,651

6.1.1 Amazon

The Amazon co-purchasing network dataset is obtained from the SNAP library [12]. The SNAP library collected this data by crawling the Amazon website.

The data is based on Amazon website's feature 'Customers Who Bought This Item Also Bought'. If a product i is often co-purchased with product j, an undirected edge from i to j exists in the graph. Each category of products provided by Amazon defines each community of ground-truth. They considered each connected component to be a separate ground-truth community in a product category. Then they removed communities with less than 3 nodes of ground-truth.

6.1.2 DBLP

The DBLP co-authorship network dataset is also obtained from the SNAP library [12]. The bibliography of the DBLP computer science provides a comprehensive list of computer science research papers. The SNAP library built a network of co-authorships where two authors are connected if at least one paper is published together. Publication venue like journal or conferencewas used to define an individual ground-truth community; a community is formed by authors who have published in a particular journal or conference.

They considered each connected component in a group to be a separate community of ground-truth. They removed communities with less than 3 nodes of ground-truth.

6.1.3 Webpages

The webpages dataset is obtained from the web crawl as described in section 4.2. The graphical representation of the subset of the network obtained from section 4.2 is shown in Figure 3.

The degree distribution of the whole network is shown in Figure 4. In Figure 4, we observe that the number of nodes with degree $1(10^0)$ is large which might pose

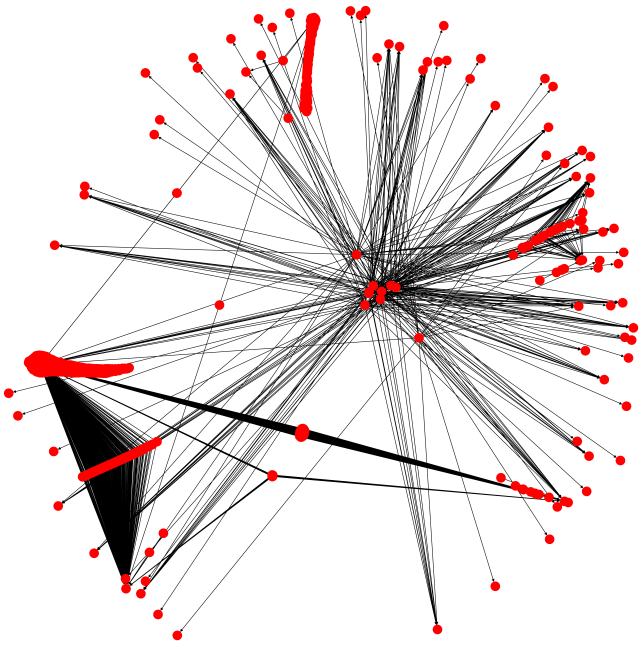

Figure 3: A subset of the Webpage network.

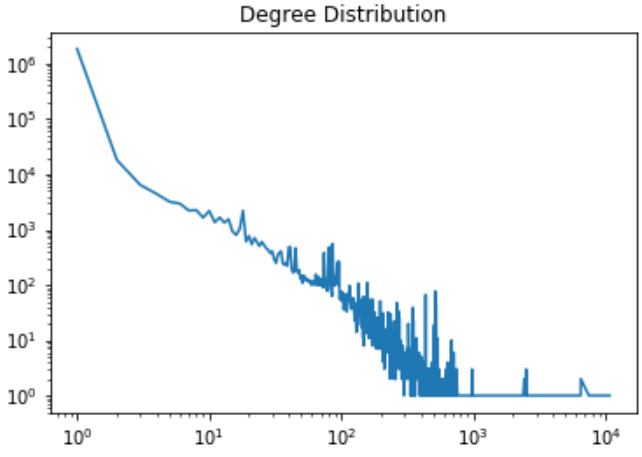

Figure 4: Degree distribution of the Webpage network.

a problem when the Webpage dataset is used to compare the existing COEUS and modified COEUS.

6.2 COEUS

We use the three datasets described above to run experiments on the existing COEUS approach. The COEUS approach forms communities by tacking edges as streams. We initialized the following parameters so that we obtain 99% confidence and $\epsilon < 0.00001$:

1. $d = 7$
2. $w = 200,000$

We consider three random nodes of each ground truth community as the input seed set. Since we have the ground truth communities for the publicly available datasets we use F1 score to measure the accuracy of the algorithm. There are two techniques in the coeus algorithm. Let's call it SIMPLE technique When we increment the community degree of a node by 1 and EDGE_QUALITY technique when the community degree of a node is incremented by the community degree of the adjacent node. Result of each of this technique is compared against the ground-truth community, GROUND_TRUTH, and the F1 score is calculated. A novel clustering algorithm (Algorithm 5) is applied to the resulting COEUS community to separate the nodes that exhibit weak community ties and are removed. Let's call this approach as DROP_TAIL. We compare the efficiency of Algorithm 5 by comparing the F1 scores of DROP_TAIL and EDGE_QUALITY. The results of this approach for Amazon dataset is listed in Table 2 and the results of this approach for DBLP dataset is listed in Table 3. The results of this approach for the webpage dataset created in this project is listed in Table 4.

Table 2: COEUS on Amazon dataset

Test Case	Test Parameter 1	Test Parameter 2	F1-score
1	SIMPLE	GROUND_TRUTH	0.76
2	EDGE_QUALITY	GROUND_TRUTH	0.85
3	EDGE_QUALITY	DROP_TAIL	0.80

Table 3: COEUS on DBLP dataset

Test Case	Test Parameter 1	Test Parameter 2	F1-score
1	SIMPLE	GROUND_TRUTH	0.38
2	EDGE_QUALITY	GROUND_TRUTH	0.43
3	EDGE_QUALITY	DROP_TAIL	0.25

Table 4: COEUS on Webpage dataset

Test Case	Test Parameter 1	Test Parameter 2	F1-score
1	SIMPLE	GROUND_TRUTH	0.90
2	EDGE_QUALITY	GROUND_TRUTH	0.90
3	EDGE_QUALITY	DROP_TAIL	0.74

6.3 Modified COEUS

In this project we have modified the existing COEUS as explained in Section 5.2 and did some experiments on it. In the modified COEUS we consider a stream of triangles instead of edges. So first we need to find the triangles in each dataset. We added the edges of the network in to MySQL database where two nodes of the edges were considered as two columns and named it node1 and node2. Say the database name is 'edge_data', then the following SQL return the triangles for a given network.

```
SELECT e1.node1 as U, e2.node1 as V, e3.node2 as W
  FROM edge_data e1, edge_data e2, edge_data e3
    WHERE e1.node2 = e2.node1 AND e2.node2 = e3.node2 AND e3.node1 = e1.node1;
```

The number of triangles for the provided datasets are listed in Table 5.

Table 5: Graphs of our dataset with triangles

Dataset	**Type**	**Nodes**	**Edges**	**Triangles**
Amazon	Co-purchasing	334,863	925,872	667,129
DBLP	Co-authorship	317,080	1,049,866	2,224,385
Webpages (own)	Link citation	1,977,975	2,484,651	1,269,112

We maintained the same setting as before and initialized the following parameters so that we obtain 99% confidence and $\epsilon < 0.00001$:

1. $d = 7$

2. $w = 200,000$

The results of the modified COEUS for Amazon dataset is listed in Table 6. The results of the modified COEUS for DBLP dataset is listed in Table 7. The results of the modified COEUS for the webpage dataset created in this project is listed in Table 8.

Table 6: Modified COEUS on Amazon dataset

Test Case	**Test Parameter 1**	**Test Parameter 2**	**F1-score**
1	SIMPLE	GROUND_TRUTH	0.83
2	EDGE_QUALITY	GROUND_TRUTH	0.83
3	EDGE_QUALITY	DROP_TAIL	0.80

Table 7: Modified COEUS on DBLP dataset

Test Case	**Test Parameter 1**	**Test Parameter 2**	**F1-score**
1	SIMPLE	GROUND_TRUTH	0.40
2	EDGE_QUALITY	GROUND_TRUTH	0.40
3	EDGE_QUALITY	DROP_TAIL	0.33

Table 8: Modified COEUS on Webpage dataset

Test Case	Test Parameter 1	Test Parameter 2	F1-score
1	SIMPLE	GROUND_TRUTH	0.47
2	EDGE_QUALITY	GROUND_TRUTH	0.47
3	EDGE_QUALITY	DROP_TAIL	0.47

6.4 Comparing results of COEUS and modified COEUS

Comparing Table 2 and Table 6 we see that for the test case 1 (SIMPLE and GROUND_TRUTH) the modified COEUS gives much better result than the existing COEUS for the Amazon dataset. Test case 3 (EDGE_QUALITY and DROP_TAIL) gives almost the same result for both COEUS implementations. But the test case 2 (EDGE_QUALITY and GROUND_TRUTH) does not provide better result for the Modified COEUS compared to existing COEUS.

Similarly, comparing Table 3 and Table 7 we see that for the test case 1 (SIMPLE and GROUND_TRUTH) the modified COEUS gives better result than the existing COEUS for the DBLP dataset. Test case 3 (EDGE_QUALITY and DROP_TAIL) also gives better result for modified COEUS than the existing COEUS. But the test case 2 (EDGE_QUALITY and GROUND_TRUTH) does not provide better result for the Modified COEUS compared to existing COEUS.

Comparing Table 4 and Table 8 we see that modified COEUS fails to perform better than the existing COEUS for any of the test cases. The ratio of nodes to edges in the Webpage dataset is high and this might be one of the reasons for the failure of the modified COEUS. The modified COEUS considers triangles as the input stream and the Webpage dataset has less number of triangles due to the large number of nodes with degree 1. We also do not have a ground-truth community for the Webpage dataset which is used to pick the initial seed-set. Also, the accuracy of

existing COEUS and modified COEUS is calculated by comparing the output of these methods against the ground-truth communities.

CHAPTER 7

Conclusion and Future Work

In this project, we proposed modified COEUS which is a modification to the existing novel graph stream community detection algorithm, COEUS that expands seed-sets of nodes into communities [5]. The modified COEUS is based on motifs, also called higher order structures, such as edges and triangles.

Table 9: Comparison of accuracy for COEUS and modified COEUS on Amazon dataset

Test Case	**COEUS**	**modified COEUS**
SIMPLE & GROUND_TRUTH	0.76	**0.83**
EDGE_QUALITY & GROUND_TRUTH	**0.85**	0.83
EDGE_QUALITY & DROP_TAIL	**0.80**	**0.80**

Table 10: Comparison of accuracy for COEUS and modified COEUS on DBLP dataset

Test Case	**COEUS**	**modified COEUS**
SIMPLE & GROUND_TRUTH	0.38	**0.40**
EDGE_QUALITY & GROUND_TRUTH	**0.43**	0.40
EDGE_QUALITY & DROP_TAIL	0.25	**0.33**

In Table 9 and Table 10 we see that modified COEUS performs better than COEUS for SIMPLE technique and for EDGE_QUALITY with DROP_TAIL technique, modified COEUS performs equivalent to or better than COEUS.

In the future work, we can try to improve the accuracy of EDGE_QUALITY technique.

In Table 11 we see that modified COEUS fails to perform better than the existing COEUS for any of the test cases. The ratio of nodes to edges in the Webpage dataset

Table 11: Comparison of accuracy for COEUS and modified COEUS on Webpage dataset

Test Case	**COEUS**	**modified COEUS**
SIMPLE & GROUND_TRUTH	**0.90**	0.47
EDGE_QUALITY & GROUND_TRUTH	**0.90**	0.47
EDGE_QUALITY & DROP_TAIL	**0.76**	0.37

is high and this might be one of the reasons for the failure of the modified COEUS. The modified COEUS considers triangles as the input stream and the Webpage dataset has less number of triangles due to the large number of nodes with degree 1. We also do not have a ground-truth community for the Webpage dataset which is used to pick the initial seed-set. Also, the accuracy of existing COEUS and modified COEUS is calculated by comparing the output of these methods against the ground-truth communities.

In the future work, we can try to improve the dataset creation so as to reduce the number of nodes with degree 1.

LIST OF REFERENCES

[1] M. E. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Physical Review E*, vol. 69, no. 2, p. 026113, 2004.

[2] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of statistical mechanics: theory and experiment*, vol. 2008, no. 10, p. P10008, 2008.

[3] R. Andersen and K. J. Lang, "Communities from seed sets," in *Proceedings of the 15th international conference on World Wide Web, WWW 2006, Edinburgh, Scotland, UK, May 23-26, 2006*, 2006, pp. 223–232. [Online]. Available: http://doi.acm.org/10.1145/1135777.1135814

[4] Y.-Y. Ahn, J. P. Bagrow, and S. Lehmann, "Link communities reveal multiscale complexity in networks," *Nature*, vol. 466, no. 9, pp. 761– 764, 2010.

[5] P. Liakos, A. Ntoulas, and A. Delis, "COEUS: community detection via seed-set expansion on graph streams," in *2017 IEEE International Conference on Big Data, BigData 2017, Boston, MA, USA, December 11-14, 2017*, 2017, pp. 676–685. [Online]. Available: https://doi.org/10.1109/BigData.2017.8257983

[6] G. Cormode and S. Muthukrishnan, "An improved data stream summary: the count-min sketch and its applications," *J. Algorithms*, vol. 55, no. 1, pp. 58–75, 2005. [Online]. Available: https://doi.org/10.1016/j.jalgor.2003.12.001

[7] R. Andersen, F. R. K. Chung, and K. J. Lang, "Using pagerank to locally partition a graph," *Internet Mathematics*, vol. 4, no. 1, pp. 35–64, 2007. [Online]. Available: https://doi.org/10.1080/15427951.2007.10129139

[8] H. Yin, A. R. Benson, J. Leskovec, and D. F. Gleich, "Local higher-order graph clustering," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, August 13 - 17, 2017*, 2017, pp. 555–564. [Online]. Available: http://doi.acm.org/10.1145/3097983.3098069

[9] G. Jeh and J. Widom, "Scaling personalized web search," in *Proceedings of the Twelfth International World Wide Web Conference, WWW 2003, Budapest, Hungary, May 20-24, 2003*, 2003, pp. 271–279. [Online]. Available: https://doi.org/10.1145/775152.775191

[10] R. Andersen, F. R. K. Chung, and K. J. Lang, "Local graph partitioning using pagerank vectors," in *47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006), 21-24 October 2006, Berkeley, California, USA, Proceedings*, 2006, pp. 475–486. [Online]. Available: https://doi.org/10.1109/FOCS.2006.44

[11] R. Andersen, F. R. K. Chung, and K. J. Lang, "Local partitioning for directed graphs using pagerank," *Internet Mathematics*, vol. 5, no. 1, pp. 3–22, 2008. [Online]. Available: https://doi.org/10.1080/15427951.2008.10129297

[12] J. Leskovec and A. Krevl, "SNAP Datasets: Stanford large network dataset collection," http://snap.stanford.edu/data, June 2014.

[13] M. E. J. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Phys. Rev. E*, vol. 69, no. 2, p. 026113, Feb. 2004. [Online]. Available: http://link.aps.org/doi/10.1103/PhysRevE.69.026113

[14] J. J. Whang, D. F. Gleich, and I. S. Dhillon, "Overlapping community detection using seed set expansion," in *22nd ACM International Conference on Information and Knowledge Management, CIKM'13, San Francisco, CA, USA, October 27 - November 1, 2013*, 2013, pp. 2099–2108. [Online]. Available: https://doi.org/10.1145/2505515.2505535

[15] T. S. Evans and R. Lambiotte, "Line graphs, link partitions, and overlapping communities," *Phys. Rev. E*, vol. 80, p. 016105, Jul 2009. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevE.80.016105

[16] A. McGregor, "Graph stream algorithms: a survey," *SIGMOD Record*, vol. 43, no. 1, pp. 9–20, 2014. [Online]. Available: https://doi.org/10.1145/2627692.2627694