Online local communities with motifs

Mrudula Murali
Department of Computer Science,
San Jose State University,
San Jose, USA
Email: mrudula.murali@sjsu.edu

Katerina Potika
Department of Computer Science,
San Jose State University,
San Jose, USA
Email: katerina.potika@sjsu.edu

Chris Pollett
Department of Computer Science,
San Jose State University,
San Jose, USA
Email: chris@pollett.org

***Abstract*—A community in a network is a set of nodes that are densely and closely connected within the set, yet sparsely connected to nodes outside of it. Detecting communities in large networks helps solve many real-world problems. However, detecting such communities in a complex network by focusing on the whole network is costly. Instead, one can focus on finding overlapping communities starting from one or more seed nodes of interest. Moreover, on the online setting the network is given as a stream of higher order structures, i.e., triangles of nodes to be clustered into communities.**

In this paper, we propose an on online local graph community detection algorithm that uses motifs, such as triangles of nodes. We provide experimental results and compare it to another algorithm named COEUS. We use two public datasets, one of Amazon data and the other of DBLP data. Furthermore, we create and experiment on a new dataset that consists of web pages and their links by using the Internet Archive. This latter dataset provides insights to better understand how working with motifs is different than working with edges.

Keywords - Community detection, Local communities, Online community, Motifs, Triangles, Higher order structures, seed sets, graph streams

I. Introduction

Graphs model real-world systems, like social networks, where nodes are users 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 the overlapping setting a node can belong to multiple communities. We focus our attention in this work on online local graph community detection that produces overlapping communities, by considering streams of motifs (also called higher order structures), such as triangles of nodes. With the use of two public datasets, the Amazon and the DBLP dataset, and a new Webpage dataset, we test the accuracy of the proposed method and compare it to the COEUS method for finding online local communities from a stream of edges.

Detecting such communities is very important in many fields in order to understand and extract information from such complex systems. The problem is very hard and has been studied extensively for the past few years.

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 realistic to consider a data stream model, in which the edges of a graph is considered as streams [5]. Moreover, processing the whole graph is often inefficient and we can focus on subgraphs and local communities of special nodes, called seeds.

Detecting communities can help solve many real-world problems. Some of the applications of community detection are:

- Social networks - To preform recommendations to 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.
- Marketing networks - Learning preferences of a user to display related, useful ads.
- Citation network - Identify the citation patterns of the authors and uncover the relationship among disciplines.

In this paper, we follow a recent approach on finding communities starting from specific seed nodes, focusing on local communities [6]. Additionally, we consider the online setting, where the graph evolves. In this setting we have to decide in a greedy manner, which communities to keep and which to discard.

Our contribution consists of two parts: we create a Web pages dataset, by considering WARC files from the Internet Archive, and we design a local stream graph community detection algorithm that considers motifs, such as triangles.

The paper is organized as follows. In Section II, we define some preliminaries. Section III provides an overview of the related work. Section IV describes the created new web dataset. In Section V we present our designed method. Section VI contains the experimental results and the various datasets. Finally, Section VII concludes.

II. Preliminaries

We now introduce the definitions and data structures that we will use for the rest of this paper.

A. Conductance

We begin by introducing the notion of conductance. Conductance is a popular objective function that is used for local community detection by many algorithms.

Let $G = (V, E)$ be an undirected graph and let $S \subset V$ be a set of graph nodes.

Conductance is defined over a cut $(S, \bar{S})$ (a **cut** is a partition of the nodes into two sets S and $\bar{S} = V \setminus S$) as:

$$\phi = \frac{adj(S, V \setminus S)}{min(adj(S, S), adj(V \setminus S, V \setminus S))} \quad (1)$$

where:

$$adj(S_i, S_j) = \mid \{(u, v) \in E : u \in S_i, v \in S_j\} \mid .$$

For community detection, we are interested in the conductance of a community, and the cut we use is over the nodes in a community C and the remaining nodes $V \setminus C$. A lower conductance indicates that a greater proportion of edges are within either the community or outside of the community than between the community and its outside. The minimum conductance over all cuts is called the **graph conductance** ϕ_G.

We define by $deg(u)$ the degree of node u in graph G, i.e., the number of adjacent nodes to u and we define by $deg_C(u)$ the community degree of node u in the community C as the number of adjacent nodes of u that are in the same community C.

B. Community Participation

The community participation $cp(u)$ of a node u in a community C, defined in [5], measures a node's u participation level in C. It is defined as:

$$cp(u) = \frac{\mid \{(u, v) \in E : v \in C\} \mid}{\mid \{(u, v) \in E\} \mid} = \frac{deg_C(u)}{deg_V(u)} \quad (2)$$

The community participation of a node measures the fraction of a node's adjacent nodes in the graph that are part of the same community. If the community participation of a node v is higher than a node u, then node v is more closely connected to its community than node u.

C. Count-Min Sketch

A Count-Min sketch is a well-known sub-linear space data structure for the representation of a high-dimensional vector $\vec{a}$. Count-Min sketches enable the answering of queries, such as the value of a component a_i, efficiently, and with strong guarantees of accuracy. We use Count-Min sketches to summarize data streams because Count-Min sketches can handle updates at high rates. A Count-Min (CM) sketch with parameters (ε, δ) is represented as a two-dimensional array of width w and depth d: $count[1, 1] \ldots count[d, w]$ together with a collection of d hash functions. Here $w = \lceil \frac{e}{\varepsilon} \rceil$ and $d = \lceil \ln \frac{1}{\delta} \rceil$. Each entry in the array is initially zero. The hash functions

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

are selected randomly from a pairwise-independent family [7].

An update to the CM sketch is a pair (i_t, c_t) which represents item a_{i_t} is updated by a quantity of c_t. To carry out this update we set $\forall j$, where $1 \leq j \leq d$

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

The value of a_i can be estimated according to the formula:

$$\hat{a_i} = \min_j count[j, h_j(i)]$$

It can be shown $\hat{a_i} \leq a_i + \epsilon N$, where $N = sum_i a_i$, with probability $1 - \delta$. The array of $w \times d$ counts used by Count-Min sketches takes $w \cdot d$ words to store. The defining parameters of the d pairwise hash functions can be stored with two words [7].

III. Related Work

In this section, we discuss previous related work for streams of graphs and local communities detection through seed sets. Additionally, we will review some works on Page Rank based algorithms used for community detection. Instead of Page Rank we use a modified version in our methods to measure the importance of a node in the community.

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 node's 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 version the quality of the new edge is considered. They introduce and use a new node centrality, called community participation cp, instead of page rank. As a last step, they determine in dynamically 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.

A local graph partitioning algorithm in presented in [8] 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.

Regarding other successful methods for community detection in this setting, where overlapping communities are also sought, BigClam [9] is closely related to our approach. However, this method uses matrix factorization in order to discover overlapping and non overlapping communities in large scale networks

Another approach by Ahn, Bagrow and Lehmann [4] that discovers overlapping communities by partitioning edges instead of nodes. Both these approaches work on the global structure of the network.

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 [6], they first generalize the conductance to one that is a motif conductance

and then extend the approximate Personalized PageRank with motifs, which they name the MAPPR algorithm, starting from a seed node and finding a local community such that the minimal motif conductance is achieved.

PageRank vectors can be computed to calculate the importance of node u on other nodes. Jeh and Widom [10] presented an algorithm for computing these PageRank vectors. Making use of this PageRank vector technique, Andersen et al. [11] proposed a local graph partitioning algorithm. It can be used to find communities in an undirected graph for given seed nodes. The sweep technique is used again 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, they select a set with low conductance. Later, Andersen et al. [12] extend the local graph partitioning algorithm to accommodate strongly connected directed graphs.

IV. Internet Archive Dataset

We obtained WARC files from the Internet Archive [1]. The 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.

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 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 within a WARC file. It consists of a sequence of one line index records for a WARC file. Each index record has meta information which summarizes a single web document in a WARC file. The first line in the CDX file is a legend to used to interpret the CDX records, this is followed by the records themselves. 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. For example, the letter 'U' indicates that the column is for a canonized URL, 'D' indicates that the columns provide an offset byte of a record in the WARC file, and 'n' indicates that the columns are used for lengths of WARC record. Then the CDX record lines consist of space-separated fields in this format. For a 1 GB WARC (see Table I), a compressed CDX file is about 20-30 MB, which is more manageable to analyze.

TABLE I: Description of the WARC column type codes

Column type	Meaning
A	canonized url
B	news group
C	rulespace category
D	compressed dat file offset
F	canonized frame
G	multi-columm language description
H	canonized host
I	canonized image
J	canonized jump point
K	FBIS what's changed
L	canonized link
M*	meta tags (AIF)
N*	massaged url
P	canonized path
Q	language string
R	canonized redirect
S*	compressed record size
U	uniqness
V*	compressed arc file offset
X	canonized url in other href tages
Y	canonized url in other src tags
Z	canonized url found in script
a*	original url
b*	date
c	old style checksum
d	uncompressed dat file offset
e	IP
f	frame
g*	file name
h	original host
i	image
j	original jump point
k*	new style checksum
l	link
m*	mime type of original document
n	arc document length
o	port
p	original path
r*	redirect
s*	response code
t	title
v	uncompressed arc file offset
x	url in other href tages
y	url in other src tags
z	url found in script
#	comment

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

* indicates the column codes used in our WARC CDX file.

Decompressing the whole WARC file takes a lot of time and is not an efficient approach. So we 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 record for respective web document. In the creation of our dataset, we were interested only in HTML pages. So the *file type* helps us to filter out 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. The *origina_url* field helps us make a connection between the 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.

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
```

Each of the URLs (webpage) and the linked urls correspond to a node in the graph and the link between *original_url* and the *linked_url* represent an edge in the graph.

V. Methodology

A community is generally thought to be a set of graph nodes that are closely connected and have fewer outside 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. Several methods attempt to detect low-conductance communities in an effort to develop a set of nodes with a limited number of non-community node ties. However, in terms of time and space, tracking the behavior of all possible communities as we process the edges of the stream is inefficient. Rather, we use community participation $cp(u)$ of a node u in a community, which measures the level of participation of a node in a community. The intuition here is that the low conductance for the community is equivalent to adding nodes that have high values of cp to a community C.

The motivation behind graph stream algorithms is that many real-world networks are now reaching sizes that are just too big. Graph algorithms are therefore unable to store and process the entire graph [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.

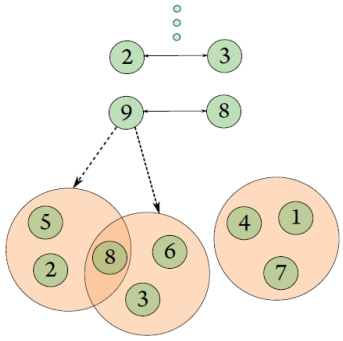

Fig. 1: A stream comprising the edges of an undirected graph and a set of communities initialized with a few seed nodes.

Due to a large amount of data, Count-Min sketches are used to store the frequency data.

A. Example of the stream of edges

The COEUS method proposed by Liakos et al. [5] considers a stream of edges as the input. Consider using Figure 1 as an example to demonstrate how this algorithm works with three communities to be detected. It 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. When edge $\{9, 8\}$ arrives, the degree of both nodes 8 and 9 will first be increased by 1. After that, for each community, we examine whether nodes 9 or 8 are community members. This is true for two communities with node 8. Therefore, for both communities we increase the community degree of node 9 by 1. Furthermore, to both communities to which node 8 belongs we add node 9.

B. Motif-COEUS

In our proposed method we will consider the stream of triangles, i.e., motifs, of nodes as the input.

We propose the Motif-COEUS method that takes two inputs:

1) A set of community seeds $K' = \{K_1, K_2, \ldots, K_k\}$, each of which is $K_i = \{k_1, k_2, \ldots, k_l\} \in V$ and
2) a stream of triangles $S = (t_1, t_2, \ldots, t_m)$, 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), where (e_1, e_2), (e_2, e_3), and $(e_1, e_3) \in E$.

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 $C = \{C_1, C_2, ..., C_k\}$, 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.

Motif-COEUS (Algorithm 2) does the initialization of the seed-set communities (Line $1 - 7$ of Algorithm 2) as the COEUS Algorithm does. In contrast to COEUS our Motif-COEUS is now ready to process the stream of trianges instead of edges. Because of the size of the network, we do not keep the entire triangle stream. Instead, we keep track of the

Algorithm 2 Motif-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: $deg_V[u] += 2$;
10: $deg_V[v] += 2$;
11: $deg_V[w] += 2$;
12: **for each** $C \in C'$ **do**
13: **if** $u \in C$ **then**
14: $deg_C[v] += 2$;
15: $deg_C[w] += 2$;
16: **end if**
17: **if** $v \in C$ **then**
18: $deg_C[u] += 2$;
19: $deg_C[w] += 2$;
20: **end if**
21: **if** $w \in C$ **then**
22: $deg_C[u] += 2$;
23: $deg_C[v] += 2$;
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: $procElements += 1$;
35: **if** $procElements \mod W == 0$ **then**
36: $C \leftarrow prune(C, s, deg_V, deg_C)$
37: **end if**
38: **end for**
39: **end while**

following aspects as we process the stream of edges as in COEUS:

1) Each node's degree in a graph
2) Community degree
3) Nodes that form the community

We first increase the degree of each node in the incoming triangle of edges of the stream (Lines $8-11$). We then examine whether each of the nodes in the triangle is a member of a community already. If this is the case, we will increase the other nodes' community degree. Furthermore, if the other nodes are not community members, the nodes will be added to that community (Lines $12-32$).

As the diameters exhibited by real-world networks are small and often decrease as the network grows, the size of communities often grows considerably through the above process. However, we want to focus on nodes for each community that is closely connected to each other. Therefore, after processing a number of W triangles, called the window, we prune (Lines $33-37$) all the communities in order to keep in each community the most highly involved nodes as done in COEUS. The prune process, calculates for each node in a community the $cp(c)$. A min-heap holds the nodes with the highest values for community involvement. Additionally, a maximum size s of each community is used as a threshold. We use the same as in the COEUS, i.e., $s = 100$ and $W = 10k$.

Instead of increasing the node's community degree by 2 (SIMPLE approach) for all adjacent nodes that are community members we can adapt Eq. 2 which is equal to the fraction of the node adjacent nodes which are also members of the community concerned in order to get the degree. 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. Increment the community degree of a node using the community participation value of its adjacent node instead of 2 enables to maintain the focus in the community. In particular, this variation (EDGE_QUALITY approach) favors nodes that are adjacent to well-established members of the community, as such nodes receive a significant increment to their community degree. In contrast, nodes that exhibit low values of community participation provide insignificant increments to the participation levels of their adjacent nodes. Thus, the potential that nodes exhibiting low values of community participation will shift the focus of the community is limited.

Thus, by replacing Lines $12-24$ of Algorithm 2 by the steps in Procedure 3 we deal with triangles and we get the $Edge\_Quality$ approach for motifs.

Furthermore, we follow the approach of finding a smaller size community by removing any irrelevant nodes. This approach is the DROP_TAIL of [5] (see Procedure 4). It is based on the fact that irrelevant nodes have weak ties to the actual community and therefore their respective values of community participation are insignificant compared to the values of other nodes included in the community. The distribution of values of community participation cp varies depending on the graph, as well as the community concerned. Therefore, it is not an option to set a constant threshold values and must discard nodes that exhibit lower values of community participation to remove such tails. Instead, it is adjusted to each particular community and isolate the nodes that belong to the tail through clustering. In order to do that, it calculates the average distance between two consecutive nodes by their associated community participation values after ranking them. Then, the

Algorithm 3 *Edge_Quality* Motifs

Procedure ADDTOCOMMBYEDGEQUALITYMOTIF
▷ (u, v, w) a triangle
  for each $C \in C'$ **do**
    if $u \in C$ **then**
      $degree_C[v] += \frac{degree_C[u]}{degree_V[u]}$;
      $degree_C[w] += \frac{degree_C[u]}{degree_V[u]}$;
    end if
    if $v \in C$ **then**
      $degree_C[u] += \frac{degree_C[v]}{degree_V[v]}$;
      $degree_C[w] += \frac{degree_C[v]}{degree_V[v]}$;
    end if
    if $w \in C$ **then**
      $degree_C[u] += \frac{degree_C[w]}{degree_V[w]}$;
      $degree_C[v] += \frac{degree_C[w]}{degree_V[w]}$;
    end if
  end for
end Procedure

Algorithm 4 DROP_TAIL [5]

Procedure DROPTAIL
  $\hat{C} \leftarrow reverseSort(C)$;
  $totalDifference \leftarrow 0$;
  $previous \leftarrow 0$;
  for each $C \in \hat{C}$ **do**
    if $previous > 0$ **then**
      $totalDifference \leftarrow cp(c) - previous$;
    end if
    $previous \leftarrow cp(c)$;
  end for
  $averageDifference \leftarrow \frac{totalDifference}{\hat{C}.size()-1}$;
  $previous \leftarrow 0$;
  for each $C \in \hat{C}$ **do**
    if $previous > 0$ **then**
      $difference \leftarrow cp(c) - previous$;
    end if
    $previous \leftarrow cp(c)$;
    if $difference < averageDifference$ **then**
      $\hat{C}.remove(c)$;
    else
      break;
    end if
  end for
end Procedure

DROP_TAIL examines iteratively, starting from the last node, the value distance of two nodes in this ranking. When this distance is found to be greater than the average node distance, DROP_TAIL stops because it has found a significant gap between the two consecutive node values.

VI. EXPERIMENTAL RESULTS

We now describe some experiments we conducted to evaluate our algorithms. We start by detailing the three network datasets used and how we ran our algorithms on these datasets. Finally, we present and compare the results of our algorithms to the ones from COEUS.

A. Datasets

Our experiments include three datasets. Two of the them are publicly available [17]: the Amazon co-purchasing network and the DBLP co-authorship network. Both of these datasets are undirected and contain ground-truth communities. Our last dataset is the one we created from a crawl of webpages (Section IV). The details of the provided datasets are listed in Table II.

TABLE II: Graphs of our dataset

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

1) Amazon: The Amazon co-purchasing network dataset is obtained from the SNAP library [17]. 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.

2) DBLP: The DBLP co-authorship network dataset is also obtained from the SNAP library [17]. 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 conference was 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.

3) Webpages: The webpages dataset is obtained from the web crawl as described in Section IV. The graphical representation of a subset of the network is shown in Figure 2, and one can see a lot of one degree nodes.

The degree distribution of the whole network is shown in Figure 3. We observe that the number of nodes with degree

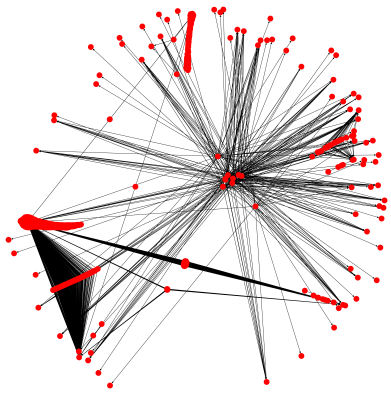

Fig. 2: A subset of the Webpage network.

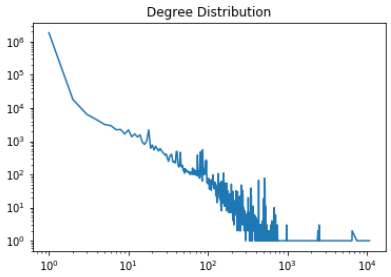

Fig. 3: Degree distribution of the Webpage network.

$1(10^0)$ is large which we will see pose a problem in detecting communities when we use streams of motifs.

B. COEUS

We used the three datasets described above to run experiments on the existing COEUS approach. We initialized the following parameters so that we obtain 99% confidence and $\epsilon < 0.00001$: $d = 7$ and $w = 200,000$.

We considered three random nodes of each ground truth community as the input seed set. Since we have the ground truth communities for the publicly available data sets we use the $F1$ score to measure the accuracy of the algorithm. There are two approaches in the COEUS algorithm as in our. Let's call it SIMPLE approach when we increment the community degree of a node by 1 and EDGE_QUALITY approach when the community degree of a node is incremented by the community degree of the adjacent node. The results of each of this technique is compared against the ground-truth community, GROUND_TRUTH, and the $F1$ score is calculated. The DROP_TAIL approach is applied to the resulting COEUS community to separate the nodes that exhibit weak community ties and are removed. We compare the efficiency of COEUS with streams of edges comparing the $F1$ scores of DROP_TAIL and EDGE_QUALITY. We have three test cases:

- Case 1: SIMPLE, GROUND_TRUTH
- Case 2: EDGE_QUALITY, GROUND_TRUTH
- Case 3: EDGE_QUALITY, DROP_TAIL

The results of this approach for the Amazon dataset is listed in the 2nd column of Table III and the results of this approach for the DBLP dataset is listed in the 2nd column of Table IV. The results of this approach for the webpage dataset is listed in the 2nd column of Table V. Note that since for this later dataset we don't have the ground truth we use as ground truth a naïve one in which once two nodes are connected by an edge they belong to the same community.

TABLE III: F1-scores on Amazon dataset

Case	COEUS [5]	Motif-COEUS
1	0.76	0.83
2	0.85	0.83
3	0.80	0.81

TABLE IV: F1-scores on DBLP dataset

Case	COEUS [5]	Motif-COEUS
1	0.38	0.40
2	0.43	0.40
3	0.25	0.33

TABLE V: F1-scores on Webpage dataset

Case	COEUS [5]	Motif-COEUS
1	0.90	0.47
2	0.90	0.47
3	0.74	0.47

C. Motif-COEUS

In the Motif-COEUS we consider a stream of triangles instead of edges. So first we need to find all triangles in each dataset. We added the edges of the network to a MySQL database, where two nodes of the edges were considered as two columns and named $node1$ and $node2$. The database is named 'edge_data', then the following SQL returns all 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 VI.

TABLE VI: Graphs of our dataset with triangles

Dataset	Type	Nodes	Triangles
Amazon	Co-purchasing	334,863	667,129
DBLP	Co-authorship	317,080	2,224,385
Webpages	Link citation	1,977,975	1,269,112

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

The results of Motif-COEUS for the Amazon dataset is listed in the 3rd column of Table III. The results of the Motif-COEUS for the DBLP dataset is listed in the 3rd column of Table IV. The results of the Motif-COEUS for the webpage dataset created is listed in the 3rd column of Table V.

D. Comparing results of COEUS and Motif COEUS

Comparing the 2nd and 3rd column of Table III we see that for Case 1 the Motif-COEUS gives much better result than the existing COEUS for the Amazon dataset. Case 3 gives almost the same result for both COEUS implementations. However, test Case 2 does not provide better result for the Motif-COEUS compared to existing COEUS.

Similarly, comparing the 2nd and 3rd column of Table IV we see that for Case 1 and Case 3 the Motif-COEUS gives better result than the existing COEUS for the DBLP dataset. Again, Case 2 does not provide better result for the Motif-COEUS compared to existing COEUS.

Comparing the 2nd and 3rd column of Table V we see that Motif-COEUS fails to perform better than the existing COEUS for all 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 Motif-COEUS. The Motif-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. Note that the accuracy of existing COEUS and Motif COEUS is calculated by comparing the output of these methods against the ground-truth communities.

VII. Conclusion and Future Work

We propose Motif-COEUS method which detects online local overlapping graph communities by using streams of motifs, such as triangles of nodes, and seed nodes. We provide experimental results and compare it to another algorithm named COEUS, which was the motivation of our work. We use two public datasets, one of Amazon data and the other of DBLP data. Furthermore, we create and experiment on a new dataset that consists of web pages and their links by using the Internet Archive. This latter dataset provides insights to better understand how working with motifs is different than working with edges.

We experimented with three datasets, two publicly available from Amazon and DBLP, and one that we constructed based on web pages. We conclude that by considering triangles we get better communities structures, closer to the ones given in the ground truth or through the DROP_TAIL approach. Notably, we see that Motif-COEUS failed to perform well for any of the test cases on our new Webpage dataset. One of the reasons for this was that Motif-COEUS use triangles as the input stream and the Webpage dataset as a stream has fewer triangles as compared to nodes due to the generally dispersive nature of web crawling. Specifically, the triangles might be present, but the nodes and relevant edges are only witnessed much further in the crawl data stream, potentially in a different WARC file then the ones analyzed. We also do not have a ground-truth community for the Webpage dataset, and uses one that is simplistic which our algorithm uses to pick the initial seed-set.

In future work, we will try to improve the accuracy of our algorithms by incorporating higher order structures such as cliques of size 4 as well as lower order structures. Another future direction would be to refine our web data by taking the initial WARC files and filtering the whole data set on the first X many distinct nodes found to capture this graph structure in a smaller stream.

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*, pp. 223–232, 2006.

[4] Y.-Y. Ahn, J. P. Bagrow, and S. Lehmann, "Link communities reveal multiscale complexity in networks," *nature*, vol. 466, no. 7307, 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*, pp. 676–685, 2017.

[6] 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*, pp. 555–564, 2017.

[7] 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.

[8] 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.

[9] J. Yang and J. Leskovec, "Overlapping community detection at scale: a nonnegative matrix factorization approach," in *6th ACM WSDM 2013*, pp. 587–596, 2013.

[10] 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*, pp. 271–279, 2003.

[11] 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*, pp. 475–486, 2006.

[12] 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.

[13] M. E. J. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Phys. Rev. E*, vol. 69, p. 026113, Feb. 2004.

[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*, pp. 2099–2108, 2013.

[15] T. S. Evans and R. Lambiotte, "Line graphs, link partitions, and overlapping communities," *Phys. Rev. E*, vol. 80, p. 016105, Jul 2009.

[16] A. McGregor, "Graph stream algorithms: a survey," *SIGMOD Record*, vol. 43, no. 1, pp. 9–20, 2014.

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