5-2018

Community Detection in Cyber Networks

Harsha Vithalrao Deshmukh
Purdue University

COMMUNITY DETECTION IN CYBER NETWORKS

by

Harsha Vithalrao Deshmukh

A Thesis

Submitted to the Faculty of Purdue University

In Partial Fulfillment of the Requirements for the degree of

Master of Science

Department of Computer & Information Technology

West Lafayette, Indiana

May 2018

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF COMMITTEE APPROVAL

Dr. John Springer, Chair

Department of Computer and Information Technology

Dr. Eric Matson

Department of Computer and Information Technology

Dr. Eric Dietz

Department of Computer and Information Technology

Approved by:

Dr. Eric Matson

Head of the Graduate Program

Dedicated to my loving parents

ACKNOWLEDGMENTS

First and foremost, I would like to express my sincere gratitude to my advisor Dr. Springer, who guided me through the entire process of research and thesis. I would like to thank him for supporting, encouraging, and mentoring me throughout my entire time as a graduate student at Purdue University. I would also like to thank my committee members Dr. Eric Dietz and Dr. Eric Matson for providing feedback and suggestions. Next, I would like to thank my parents and sisters for their unconditional love, affection, and support. I would especially like to thank my mom for motivating and inspiring me.

Lastly, I would like to thank Purdue University for providing me all the resources required for performing this research.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Author: Deshmukh, Harsha, V. MS
Institution: Purdue University
Degree Received: May 2018
Title: Community Detection in Cyber Networks
Major Professor: John Springer

Community detection has been widely studied and implemented across various research domains such as social networks, biological networks, neuroscience, and cybersecurity. In the context of cyber networks, it involves identifying the groups of network nodes such that the network connections are dense within the group and are sparser between the groups. Various community detection algorithms can be utilized to detect the underlying community structure of a given network. However, it is crucial to evaluate the quality of the detected communities as there are a number of ways that a particular network may be partitioned into communities, and thus, a quality evaluation metric needs to be used to determine the best partitioning. Modularity is one such measure, and when evaluating the modularity index, researchers have considered null models for graphs with specific structures or characteristics. However, most real-world complex networks as a whole do not exhibit one specific characteristic but instead consist of various identifiable subgraphs that do respectively exhibit particular characteristcs, and accordingly, formulating a null model for these individual subgraphs may improve the modularity value and thereby improve the quality of the partitioning otherwise known as the detected communities.

This research investigates the extent to which the modularity value increases when a bipartite subgraph is taken into consideration while performing community detection. This is accomplished by designing and developing an empirical setting that first identifies the presence of a bipartite subgraph and then utilizes it to perform community detection. Our empirical study

and results suggest that the quality of the detected communities is enhanced by leveraging the presence of bipartite subnetwork in the given real world complex network. Furthermore, we present the applicability of this research in cybersecurity domain to alleviate the consequences of any worm attack. We can achieve this by employing our technique to obtain a better underlying community structure for identifying the most vulnerable set of nodes in the compromised network.

CHAPTER 1. INTRODUCTION

This chapter provides an overview of this thesis followed by an initial introduction to community detection as it relates to the cyber networks. Furthermore, this chapter highlights the research questions, significance of this study, and statement of purpose. It also briefs about the scope, assumptions, and limitations of this research.

1.1. Thesis overview and organization

The central topic of this thesis is community detection in real word cyber- networks. Our research addresses various important questions related to community detection:

1. What is the extent to which the quality of the community structure increases when we leverage the presence of any subgraph and use its graph-specific null model while formulating the composite modularity?

   Chapters 3 and 4 cover the research framework, methodology, empirical setting, and execution workflow towards addressing this question for bipartite subgraph.

2. How does the bipartite subgraph size affect the overall quality of the detected community structure?

   This question has been addressed in Chapter 4.

Chapter 1 provides the required background study and highlights the significance of our research. Relevant literature has been reviewed in Chapter 2. It summarizes the related previous work done in this field and presents a strong basis for reasoning and justification of the author's work. Chapter 5 reports our research findings, discussion, conclusion, and directions for future work.

1.2. Structural and statistical properties of networks

A network is comprised of vertices and links, and based on the application domain, the links can be weighted/unweighted and/or directed/undirected. Some popular examples are the World Wide Web, the food web, language networks, social networks, gene regularity networks, biological metabolic networks, collaboration networks, cyber networks, and call graph networks. The structural and statistical properties of any given network include (but are not limited to) size (e.g., number of vertices and number of links), network degree distribution (in/out), average clustering coefficient, community structure, network diameter, and average path length.

1.3. Community detection

With the continued explosion of cyber traffic across billions of IP addresses across the globe, it has become extremely challenging to analyze the networks due to its growing size and complexity. One promising solution is to identify the communities in the network structure and perform analysis at a community level rather than at a network node level.

By taking this approach, one can thus achieve a substantial reduction in the number of nodes over which analysis is performed and thereby increase the efficiency and effectiveness of the analyses performed over any real-world complex network. A community is defined as "[t]he division of network nodes into groups within which the network connections are dense, but between which are sparser" (Newman & Girvan, 2004, p. 1). The process of identifying such communities in the network is referred to as community detection. Furthermore, Newman and Girvan (2004) derived a quality function that reflects the quality of community partitioning with reference to a null model.

1.4. Research questions

1. Given a complex cyber network, how can one identify a bipartite subgraph and utilize it to perform community detection by formulating a composite modularity metric of the partitioned network?

2. What is the extent to which this modularity index increases in comparison to the modularity index generated by Clauset-Newman-Moore algorithm?

1.5. Significance of this research

Let us now systematically understand the significance of this research.

We know that given a graph G with n vertices, the number of possible partitions in k clusters of G is given by $S\ (n, k)$ (Andrews, 1998). The total number of possible partitions is the nth Bell number given by Equation 1.

$$B_n = \sum_{k=0}^{n} S(n,k) \qquad \text{(Eqn. 1)}$$

$$B_n \sim \frac{1}{\sqrt{n}} [\lambda(\mathrm{n})]^{\frac{n+1}{2}} e^{\lambda(\mathrm{n})-\mathrm{n}-1} \qquad \text{(Eqn. 2)}$$

where

$$\lambda(\mathrm{n}) = e^{\mathrm{W(n)}} = \frac{n}{W(n)} \qquad \text{(Eqn. 3)}$$

W(n) is the Lambert's W function and λ(n) is given by the Equation 3. From the Equation 2, B(n) is observed to grow exponentially faster with respect to the graph size (Fortunato, 2010).

This indicates that enumeration and evaluation of all the partitions of a graph is not feasible. Moreover, these partitions are not all equally good. This argument raises an important concern: how does one quantify "goodness" of the partition?

As previously mentioned, methods for quantifying the goodness of a particular partition is through the use of community detection, and substantial research has been done on various community detection methods and algorithms (Barber, 2007; Xu, Wang, & Gu, 2014; Aiello, Kalmanek, McDaniel, Sen, Spatscheck, & Van der, 2005; Chen, Kuzmin, & Szymanski, 2014). Chapter 2 highlights some of the most popular community detection algorithms. However, they have not considered the possibility of the presence of more than one relevant substructure, which can potentially maximize the quality function.

According to Newman (2006), a good community structure for a network is characterized by both the presence of fewer edges between the groups and cases when the total count of edges that exist between the groups is less than expected. This expected set of edges is defined by the Null model (P_{ij}) and it would be inappropriate to use the same null model (usually it is Bernoulli random graph where P_{ij}= p for all i, j) (Barber, 2007) for all the graphs. The null model in consideration should be the most appropriate one for that graph to formulate the quality function that yields the maximum modularity as naturally it is possible that different null models may yield their corresponding different values for the modularity metric. Therefore, this calls for a wise decision over the choice of the null model because, in any real-world network, one can potentially observe various types of networks in a single complex network. Therefore, the researcher must identify the subgraphs and formulate the subgraphs' individual modularities; in other words, one must define a specific null model for each of the identified subgraphs and formulate a resulting composite quality function. Thus, this research will incorporate the presence of any specific subnetworks identifiable in the given network by utilizing it during the evaluation of composite modularity metric.

Let us consider a scenario in which worm impacts a computer node in a given network. This malware has the potential to harm its host network by consuming bandwidth and overloading the servers by propagating and affecting the other nodes present in the network. Without loss of generality, let us assume that if one identifies the communities within the network, then all the network nodes present in the community of the affected node are more vulnerable to worm propagation as compared to the other nodes present in the network. This can be justified using the nature of communities.

Thus, if we are successful in identifying the nodes that are the most vulnerable, one can potentially reduce the number of nodes taken under consideration for initial investigation (instead of checking and investigating all the nodes of the network) and thus eventually prevent worm propagation efficiently. Therefore, a high modularity community structure can help one identify the most vulnerable set of nodes in the network. Furthermore, the modularity of a network viewed as communities can be maximized by identifying the relevant substructures and formulating a quality function that best represents the overall network as communities of respective substructures/characteristics. This approach may even have implications for dynamic networks where nodes “come and go” with high velocity.

1.6. Statement of purpose

The principal objective of this research is to investigate the extent to which the consideration of bipartite subgraph increases the value of modularity index in comparison to the Clauset-Newman-Moore modularity.

1.7. Scope

A Community structure "is the division of network nodes into groups within which the network connections are dense, but between which they are sparser" (Newman & Girvan, 2004, p. 1). Some of the traditional methods of identifying the communities are graph partitioning, spectral clustering, partitional clustering, and hierarchical clustering (Fortunato, 2010). Moreover, the efficiency of the communities thus identified can be measured with the help of modularity metric (Q) and can be expressed as stated in Equation 4.

$$Q = \frac{1}{2m} \sum_{i,j}(A_{ij} - P_{ij})\delta(C_i, C_j) \qquad \text{(Eqn. 4)}$$

We can observe that the modularity metric depends on the null reference model P_{ij}, i.e., the expected number of edges between the vertex pairs i, j. Larger values of Q indicate stronger community structure. Thus, for a specific structure of a network (e.g., bipartite graph), the $P_{i,j}$ would be defined by considering the edge specifications for that structure. Based on the literature review, only single structure networks (for e.g., Bernoulli random graph, bipartite network, and Cyclic structure) have been explored and examined. However, a real-world complex network may consist of more than one identifiable subnetworks. Thus, consideration of these subnetworks while performing community detection has the potential of maximizing the modularity metric. The scope of this research is:

1. Identifying the presence of a bipartite subnetwork in a real-world cyber network.
2. Formulating a composite modularity metric towards defining the quality function for the network that has a bipartite subgraph.

3. Evaluation of extent to which the new composite modularity metric is increased in comparison to the modularity value evaluated by Clauset-Newman- Moore algorithm alone.

1.8. Assumptions

Following are the assumptions of this research:

1. The execution environment would remain the same for both the algorithms considered for comparison.
2. Every network node (IP Address) can be both a source node and destination node and the network will not be manipulated under any circumstances.
3. The network sample (dataset) obtained from Center for Applied Internet Data Analysis (CAIDA) is representative of a real-world cyber network.
4. The number and size of the communities are not known a priori.

1.9. Limitations

The limitations of this research are:

1. The only subgraph type taken into consideration is the bipartite graph.
2. This research focuses only on static community detection.
3. The traffic flow captured is limited to CAIDA only.
4. Only unweighted, undirected graphs are considered for this research.
5. Multiple edges from the same source to the same destination will be eliminated. In other words, redundant links from one network node to another network node are discarded and considered as a single link.

6. Modularity is the only metric considered for evaluating the quality of the detected community structure.
7. As the Clauset-Newman-Moore algorithm follows a greedy modularity optimization technique, it may fail to detect communities smaller than a scale (Fortunato & Barthelemy, 2006)

1.10. Definitions of key terms

Community – "The division of network nodes into groups within which the network connections are dense, but between which are sparser." (Newman & Girvan, 2004, p. 1).

Modularity – "The extent, relative to a null model network, to which edges are formed within the modules instead of between the modules." (Barber, 2007, p. 1)

1.11. Summary

This chapter presented a brief introduction to the research conducted by the author. It also underlined the significance, scope, limitations, and assumptions.

CHAPTER 2. REVIEW OF RELEVANT LITERATURE

This chapter provides an insight into the relevant literature the author reviewed for this research. It covers the approach utilized for literature review, various community detection algorithms, application of graph theory, and basis for evaluating the quality of the community structure.

2.1 Introduction

The examination and study of networks have become extremely popular and is indeed a ubiquitous topic across a plethora of branches in the broader fields of science and engineering (Francisco & Oliveira, 2011; Girvan & Newman, 2002; Pizzuti & Clara, 2008; Raghavan, Albert, & Kumara, 2007). This is because very often, systems of special interest can be represented in the form of network, e.g., Internet, food webs, neural networks, communication networks, social networks, etc. (Medus & Dorso, 2009).

Moreover, it is easy to imagine that certain elements of network interact with a specific group of network elements more frequently than other network components. This specificity of interaction of a network element with only certain network elements potentially hints toward some similarity of behavior, function, performance, and/or dependence. Thus, such subgroups of network elements that interact within the group more than they do with the rest of the network can be referred to as communities.

The word "community" has a myriad of definitions varying based on the context. In the context of social networks, it refers to the "group of entities closer to each other in comparison to other entities of the dataset" (Bedi & Sharma, 2016, p. 116). In other words, "[a] community is formed by individuals such that those within a group interact with each other more frequently than with those outside the group" (Bedi & Sharma, 2016, p. 116). In biological context,

Barabási (2016) defines community as a group of molecules in a metabolic network that carries out a specific cellular function.

As this research focuses on complex networks of computer nodes otherwise known as cyber networks, a community in this context can be defined as "[t]he division of network nodes into groups within which the network connections are dense, but between which are sparser" (Newman & Girvan, 2004, p. 1).

2.2 Approach to this Literature Review

This literature review primarily provides a holistic view of the entire process of community detection, significance, and its applicability in the field of cyber networks. Furthermore, graph theory is an extremely powerful mathematical theory and tool to understand, visualize, comprehend, and manipulate networks, and as community detection extensively utilizes graph theory in its approach, this manuscript will also provide a glimpse into the aspects of graph theory pertaining to cyber networks.

To further appreciate the community detection techniques, the author has summarized limitations of various popular clustering and partition algorithms as it relates to community detection in cyber networks. Chapter 2 summarizes most widely employed community detection techniques, their classification, and algorithms.

Furthermore, after having identified the possible community structures by using one or more algorithms, it is also crucial to evaluate the quality of the communities identified. Various metrics can be employed to evaluate the quality. Thus, this literature review also sheds light on the various metrics and established the most relevant approach in the context of cyber networks.

2.3 Introduction to community detection

As discussed earlier in chapter 1, community detection is the process of identifying "[t]he division of network nodes into groups within which the network connections are dense, but between which are sparser" (Newman & Girvan, 2004, p. 1). Alternatively, it can be defined as "a locally dense connected subgraph in a network" (Barabási, 2016, p. 6), and moreover, communities can be classified as strong and weak communities (Barabási, 2016). Consider a subnetwork *C* of a large complex network and let k_i^{int} denote the internal degree of the node *i*, that is, the total number of links that connect node *i* to the other nodes present in *C*. Similarly, k_i^{ext} denotes the external degree of a node *i* representing the total number of links that connect node *i* to the other nodes (that do not belong to *C*) present in the network.

A community C is a strong community if ∀ node i ∈ *C*, it satisfies Equation 5. A weak community can be expressed as Equation 6; that is, the total internal degree of all the nodes in *C* exceeds the total external degree of all the nodes present in the same *C*.

$$k_i^{int}(C) > k_i^{ext}(C) \quad \text{(Eqn. 5)}$$

$$\sum_{i \in C} k_i^{int}(C) > \sum_{i \in C} k_i^{ext}(C) \quad \text{(Eqn. 6)}$$

Thus, detecting and characterizing such community structures in a network is referred to as community detection (Chen, Kuzmin, & Szymanski, 2014). Moreover, "[t]he ability to find and analyze such groups can provide invaluable help in understanding and visualizing the structure of network" (Newman & Girvan, 2004, p. 1).

2.4 Community detection techniques and algorithms

Research over community structures in networks has a long and rich history (Newman & Girvan, 2004). It is based on similar "ideas of graph partitioning in graph theory and computer science, and hierarchical clustering in sociology" (Newman & Girvan, 2004, p. 1). This section summarizes various traditional partitioning and clustering algorithms.

2.4.1 Graph partitioning

Graph partitioning divides the vertices into c groups of a predetermined size such that edges lying between the group are minimized (Fortunato, 2010). Graph Bisection is a special form of graph partitioning that involves partitioning the graph into just two subgraphs such that the number of edges between the two pieces is minimized (Boppana, 1987). In fact, the number of links between the nodes in the two subgroups is called the cut- size and an effective graph partitioning algorithm would be the one that is able to minimize the cut size to a large extent. Boppana (1987) provides an efficient algorithm that evaluates graph partitions based on the eigenvalues and eigenvectors associated the graphs. However, there is a major concern in using graph partitioning as a method for community detection. The number and size of the communities are predefined in case of graph partitioning; however, this is not the case in community detection where both the parameters (i.e., number and size) are unknown. Moreover, the number of possible bisections increases exponentially with the size of the cluster; this can be expressed as stated in Equation 7.

$$e^{-(N+1)\ln 2-\frac{1}{2}\ln N} \qquad \text{(Eqn. 7)}$$

where N is the number of vertices in a graph. Barabási (2016) provides an elegant proof for the above-represented count using the Stirling's formula.

To generalize from graph bisection to graph partitioning, Equation 8 provides the number of possible partitions of a network of N vertices.

$$B_N = \frac{1}{e}\sum_{j=0}^{\infty}\frac{j^N}{j!} \qquad \text{(Eqn. 8)}$$

According to Barabási (2016), it is impossible to examine all the partitions of any large network because the number of possible ways a network can be partitioned grows exponentially or faster with the network size. Furthermore, according to Fortunato (2010), algorithms for graph partitioning are not suitable for community detection because the algorithms for community detection should be capable of revealing information about the community structure – such as the number of communities – instead of expecting these characteristics a priori as inputs.

2.4.2 Partitional clustering

This technique involves identifying clusters in a network. Here, the number of network clusters is predefined. The measure of dissimilarity is the distance between the pair of vertices where some of the possible considerations for the distances are Euclidian distance, sum of squared distance, or Manhattan distance. Essentially, it involves minimizing a loss function based on the distances between the points and/or seeds (alternatively, clusters) (Fortunato, 2010). Some of the classical algorithms utilizing this approach are minimum-k clustering, k-means clustering, and k-medoids to name a few. One major limitation of this technique is that it requires the number of clusters to be specified as an input, which may not be known a priori in the real world complex network applications.

In the preceding, the author summarized various traditional methods for graph partitioning and clustering along with their respective concerns as it relates to their applicability for community detection. All the aforementioned methods have two common limitations: 1) the algorithms expect an a priori knowledge about the number and size of the clusters and 2) they fail to determine a metric that expresses the quality of the partitions obtained. To overcome these limitations, researchers developed a new class of algorithms, hierarchical clustering. This technique aims at identifying groups of nodes with high similarity present in a network (Fortunato, 2010). The two most popular classes of algorithms for hierarchical clustering are:

a) Agglomerative algorithms- The subgroups are recursively merged if there exists a high similarity.
b) Divisive algorithms- The clusters are recursively split by removing the links that connect vertices with low similarity.

This family of algorithms overcomes the first limitation. However, the second limitation was only overcome in a true sense when modularity based hierarchical clustering techniques were developed because a) the aforementioned common hierarchical clustering approaches yield more than one community structure (i.e., a hierarchy of community structures) and b) as a result, it is essential to determine a metric that expresses the quality of the partitions for obtaining the best community structure. Section 2.6 details the process of quantifying the quality of community structure. Before we dive deeper into our literature review, let us discuss how real-world computer networks are modeled in terms of graphs as our research focuses on community detection in cyber networks. In the next section, we shall see some key aspects of graph theory pertaining to computer networks.

2.5 Application of graph theory in computer networks

A graph G is defined as an ordered pair of sets {V, E}, where V is a finite non-empty set of vertices in the network and E is the set of edges/links between the vertices. In set theory notation, E can be represented as $E \subseteq \{(u,v)|u,v \in \mathrm{V}\}$ (Silva & Zhao, 2016). A network of computer nodes can be represented as a graph with vertices as the computer nodes and a link denotes an exchange of a data packet between any two incident nodes. Next, we discuss different types of graphs.

1. Bipartite graph: Figure 1 is an example of a bipartite graph. Silva and Zhao (2016) provide the definition of a bipartite graph as follows:

> A bipartite graph is a graph whose set of vertices V can be split into two disjoint non- empty subsets $V1$ and $V2$, $V = V1 \cup V2$, in such a way that $(u,v) \in \mathrm{E} \implies \mathrm{u} \in \mathrm{V1}\ , \mathrm{v} \in \mathrm{V2}$. Therefore, no edge exists between pairs of vertices in the same subsets $V1$ & $V2$.

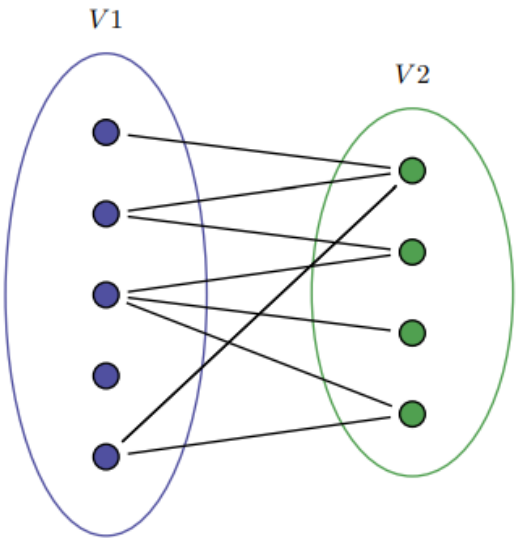

Figure 1: Bipartite graph

2. Cyclic graph: Figure 2 is an example of a cyclic graph. Weisstein (n.d.) defined a cyclic graph as a graph that contains at least one graph cycle.

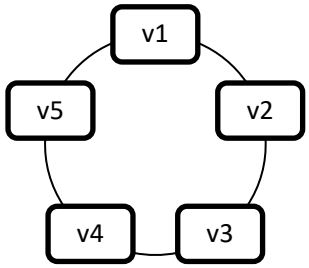

Figure 2: Cyclic graph

In general, any network can be examined and analyzed by modeling it as a graph; this thesis focuses on examining the community structure of a cyber network. In the next section, we shall study various community structure quality indicators.

2.6. Quantifying the quality of the community structure

Newman and Girvan (2004) first introduced the concept of evaluation of the quality of the community structure. As mentioned earlier in Section 1.5, they coined the term modularity, which is a measure of goodness of the partitioned network. (Fortunato, 2010) refers to this measure as the quality function Q. According to the researchers (Newman & Girvan, 2004), Q can be using Equation 9.

$$Q = \frac{1}{2m} \sum_{i,j}(A_{ij} - P_{ij})\delta(C_i, C_j) \quad \text{(Eqn. 9)}$$

where,

m: the total number of edges in the actual network

A_{ij}- adjacency matrix elements of the actual network

P_{ij}- Expected number of edges between vertices i and j in the null model.

The indicator δ function yields one if vertices i and j belong to the same community and otherwise is zero. Also, considering that $-1 \leq Q \leq 1$, larger values of Q indicate strong community structure. Thus, this quality function gave rise to another class of algorithms referred to as modularity optimization-based algorithms. Algorithms under this category exploit the technique of modularity maximization to detect community structures. Fortunato (2010) and Barabási (2016) provided an exhaustive list and detailed explanation of modularity-based algorithms.

Please recall that "[a] good division of a network into communities is not merely one in which the number of edges running between the groups is small. Rather, it is one, in which the number of edges between groups is smaller than expected" (Newman, 2006, p. 5). This expected set of edges is defined by the null model (P_{ij}) and it would be inappropriate to use the same null model (usually it is Bernoulli random graph where P_{ij}= p for all i, j) for most real-world networks. The null model in consideration should be the most appropriate one for that graph to formulate the quality function that yields the maximum modularity as naturally it is possible that different null models may yield their corresponding different values for the modularity metric.

Barber (2007) thus formulated a new P_{ij} for graphs that are inherently bipartite in nature. P_{ij} for bipartite graph can be expressed as Equation 10 and the resulting quality function (Q) can be expressed as Equation 11.

$$P_{ij} = \frac{(k_i d_j)}{2m} \quad \text{(Eqn. 10)}$$

$$Q' = \frac{1}{m} \sum_{i=1}^{p} \sum_{j=1}^{q} (A_{ij} - P_{ij}) \delta(C_i, C_j) \quad \text{(Eqn. 11)}$$

where,

p and q are the counts of vertices belonging to the two disjoint sets respectively

$$k_i = \sum_{i=1}^{q} P_{ij} \quad \text{(Eqn. 12)}$$

$$d_j = \sum_{j=1}^{p} P_{ij} \qquad \text{(Eqn. 13)}$$

Equations 12 and 13 provide an expression for k_i and d_j respectively. Thus, to formulate the quality function that yields maximum modularity, the null model in consideration should be the most appropriate one for that graph type as discussed in Section 1.5.

Apart from modularity, Leskovec, Lang, and Mahoney (2010) elucidated a list of various criteria for measuring the quality of the community structure. They have categorized the quality/score functions as multi-criterion scores and single-criterion scores.

- Multi-criterion: Conductance, expansion, internal density, cut ratio, normalized cut, maximum out-degree fraction, and average out-degree fraction.
- Single-criterion: Modularity ratio, volume, and edges cut.

2.7. Bipartite subgraph identification

Mubayi and Turán (2010) proposed an algorithm that can identify a bipartite subgraph in the given bipartite graph/network. Following is the algorithm:

Input: $G = \{V, E\}$ with $|V| = n$, $|E| = m$; s and t are parameters

if $(0 < m < 8n^{\frac{3}{2}})$ then return any $(\{u\}, \{v\})$ with $(u, v) \in E$

else

$R :=$ s vertices having highest degree

for all subsets $C \subseteq R$ with $|C| = t$ do

$\boldsymbol{D} := \cap\{N(v) - R : v \in \boldsymbol{C}\}$

if $|D| \geq t$ then $D' :=$ any set of t elements of D, return (C, D')

Output: (C, D') Bipartite graph.

The time complexity of this algorithm is $O(n^{2.42})$.

However, this algorithm requires a bipartite graph as an input for returning a dense bipartite subgraph. The focus of this research is to identify a bipartite subgraph from the given real-world network, irrespective of the underlying structure of the network. Therefore, the author developed the following algorithm for obtaining a bipartite subgraph from any given network.

Algorithm: Find_Bipartite (V, E)

Input: $\boldsymbol{G = (V, E), t}$

Output: $\boldsymbol{G_B = (R, S)}$

Begin

repeat

Initialize: $R, S, e \leftarrow \emptyset$;
$t' \leftarrow random(t)$;

for $e = 1\ to\ t'$ **do**

$R \coloneqq R \cup E_e[1]$;

$S \coloneqq S \cup E_e[2]$;

if $\exists \{ (p,q) \in E\ such\ that\ (p,q) \in R\ or\ (p,q) \in S \}$ **then**

break;

end if

end for

until $\boldsymbol{e = t}$

return $\boldsymbol{(R, S)}$

End

This algorithm takes as input the graph (G) represented as a set of vertices (V) and links (E) and the bipartite subgraph size (t) in terms of edges. We know that a bipartite graph is a set of two disjoint subsets such that there exists no link between the vertices belonging to the same subset. The same definition is used to generate a bipartite subgraph. First, vertices of a randomly selected link from the original graph are placed in the two disjoint subsets respectively. Next, another edge from the graph is selected and the two incident vertices are added respectively to

the two disjoint sets. During every subsequent addition of nodes in the disjoint sets, it is verified that there exists no link between the noes belonging to the same sub set. These two steps are repeated iteratively until the bipartite subgraph size limit (t) is reached. The run time complexity of this algorithm is $O(n^2)$.

2.8. Community detection in bipartite graphs

Pesantez-Cabrera and Kalyanaraman (2016) proposed an algorithm that performs community detection in bipartite networks to which they refer as biLouvain algorithm. This algorithm extends the Louvain algorithm proposed by Blondel, Guillaume, Lambiotte, and Lefebvre (2008). The general scheme of the algorithm is as follows:

1. Given a bipartite graph, initialize a set of $n_1 + n_2$ communities, where $n_1 = |V_1|$ and $n_2 = |V_2|$. Here, each vertex is placed in its own community.
2. At every iteration, both the set of vertices are scanned linearly. For each vertex i:
    a. Obtain a list of candidate communities to which i can move.
    b. Evaluate the modularity increase resulted from moving i from its current community to each of the candidate communities.
    c. Move vertex i to the candidate community that maximizes the net modularity gain (condition on only if the gain is positive).
3. A phase terminates when the modularity gain converges.
4. A new graph is generated through a compaction step and this newly generated graph is given as an input to the next phase (step 1). The algorithm terminates when any two consecutive phases yield a negligible modularity gain.

2.9. Fast algorithm for community detection

Newman-Girvan algorithm demands substantial amount of computational resources with a running time of $O(n^3)$. For instance, in our experimental setup for 10, 48,575 nodes, the time we observed for successful execution of the algorithm with the resource characteristics mentioned in experiment evaluation (Section 4.4) was 138.31 hours. This clearly indicates that the Newman Girvan algorithm for community detection does not scale well for extremely large real-world networks.

Later, Newman (2004) proposed a fast algorithm community detection that is an agglomerative hierarchical clustering method. Initially, each vertex is considered to be the only member of one of n communities. The communities are iteratively merged in pairs while choosing at each step the group that yields the highest increase in Q (modularity). Newman observed that Q can never be increased by joining the pair of communities between which there exists no edges at all. Thus, Newman considered only those pairs between which edges were present. The change in Q when joining two communities is given by Equation 14 and can be calculated in a constant time.

$$\Delta Q = e_{ij} + e_{ji} - 2a_i a_j \quad \text{(Eqn. 14)}$$

where, e_{ij} is the percentage of edges in the network connecting vertices in group i to those in group j (Eqn. 15) and a_i represents the percentage of edge endpoints attached to vertices in community i (Eqn. 16).

$$e_{ij} = \frac{1}{2m}\sum_{v,w} A_{vw}\,\delta(C_v, i)\delta(C_w, j) \quad \text{(Eqn. 15)}$$

$$a_i = \frac{1}{2m}\sum_{v} k_v\,\delta(C_v, i) \quad \text{(Eqn. 16)}$$

This algorithm runs in $O(n^2)$ time.

2.10. Community detection in very large networks: Clauset- Newman- Moore algorithm

Continuing with the discussion from Section 2.9, Newman (2004) performed community detection by maintaining an adjacency matrix A_{vw} and evaluating ΔQ_{ij} followed by finding the pair i, j with the largest ΔQ_{ij}. Clauset, Newman, and Moore (2004) claimed that this calculation of ΔQ_{ij} and finding the pair i, j possessing the largest ΔQ_{ij} is time consuming. Thus, they formulated an algorithm that focuses on maintaining and updating a matrix of value of ΔQ_{ij}, instead of tracking the adjacency matrix and calculating ΔQ_{ij} every time. In addition, they employ a max-heap that contains the largest element of each row of the ΔQ_{ij} matrix, and as a result, the running time of this algorithm is $O(n\ log^2 n)$. The overall scheme of the algorithm proposed by Clauset et al. (2008) is as follows:

1. Evaluate the initial values of ΔQ_{ij} and a_i using Equations 17 and 18. Next, the max-heap is populated with the largest element of each row of ΔQ_{ij}.
2. The largest ΔQ_{ij} is selected from heap H, the corresponding communities are joined, and the matrix $\Delta Q, a_i, and\ H$ are updated. Q is then incremented by ΔQ_{ij}.
3. Repeat step 2 until only one community remains.

$$\Delta Q_{ij} = \begin{cases} \frac{1}{2m} - \frac{k_i k_j}{(2m)^2} & if\ i,j\ are\ connected, \\ 0 & otherwise \end{cases} \quad \text{(Eqn. 17)}$$

$$a_i = \frac{k_i}{2m} \quad \text{(Eqn. 18)}$$

2.11. Additional community detection algorithms

Very often community detection algorithms can be devised as unsupervised learning based clustering/partitioning techniques (Leskovec, 2008). This section highlights some of the most popular community detection algorithms that are extensively used in numerous spheres of network science.

2.11.1 Louvain algorithm

The Louvain algorithm was formulated by Blondel et al. (2008). This algorithm is based on a heuristic technique constructed using modularity optimization. It can discover the high modularity clusters of very large networks and unrolls an entire hierarchical community structure of the network. Louvain algorithm works in two phases. The two phases are iteratively repeated until either only one node is left, or the modularity no longer increases. Following are the two steps:

1. Every node in the network is assigned a different community, then the decision of movement of a node to its adjacent community is made based on largest modularity gain. The modularity change is evaluated using Equation 19.

$$\Delta Q = \left[\frac{\sum_{in} + 2\,k_{i,in}}{2m} - \left(\frac{\sum_{in} + 2\,k_{i,in}}{2m}\right)^2\right] - \left[\frac{\Sigma_{in}}{2m} - \left(\frac{\Sigma_{tot}}{2m}\right)^2 - \left(\frac{k_i}{2m}\right)^2\right] \qquad \text{(Eqn. 19)}$$

where the following hold:

Σ_{in} represents the summation of the weights of edges inside the community,

k_i is the total weight of edges incident to i,

Σ_{tot} is the total weight of the edges incident to all the nodes in the community,

m is sum of weights of all the edges present in the network, and

$k_{i,in}$ is the addition of the weights of edges from node i to nodes in the community.

2. Each of the communities obtained in step 1 is now considered as one node.

The run time complexity of this algorithm is $O(n\ logn)$.

2.11.2 Walktrap algorithm

The Walktrap algorithm was formulated by Pons and Latapy (2006). The basic intuition of this algorithm is that, the random walks on any graph tend to remain in the same cluster. It utilizes a hierarchical clustering approach. First, every vertex is considered as a partition, then distances between all neighboring vertices are computed. Next, two adjacent communities are chosen according to the distance-based criterion and are merged to form a single community. Finally, the distances between the communities are updated. This is repeated for $(n-1)$ times, where n denotes the number of nodes present in the network. The overall run time complexity of this algorithm is $O(n^2 logn)$.

2.11.3 Infomap algorithm

The Infomap algorithm was formulated by Rosvall and Bergstrom (2007). This algorithm is based on analysis of information flow through the network and employs random walks on the network to unroll its community structure. The basic intuition is that, the algorithm maximizes a minimum description length objective function and the communities are identified by obtaining an optimal compression of its network structure

2.11.4 Label propagation algorithm

The label propagation algorithm was formulated by Raghavan et al. (2007). The idea of this algorithm is that every node is assigned a unique label depicting the community to which it belongs and any given node determines its community based on the community labels of its

neighbors. Next, the vertices of the graph are arranged in a random order and later sequencially, each node chooses to be a part of the community to which a majority of its neighbors belong.

2.12. Summary

Literature review of this manuscript provided a strong basis and justification for addressing the proposed research questions. Various community detection techniques were discussed and the need for specifying an evaluation criterion was explained systematically.

CHAPTER 3. FRAMEWORK AND METHODOLOGY

This chapter details about the overall research framework and the proposed methodology. We then discuss the experimental setup used for implementing the proposed technique. Lastly, the data sources, variables, and the execution environment have been provided for replicability.

3.1. Research framework: an overview

The final deliverable of this research is a technique that evaluates a composite modularity metric for the given network by considering the presence of any bipartite subgraph in the same network. Essentially, it addresses the following research questions.

1) Given a complex cyber network, how can one identify a bipartite subgraph and utilize it to perform community detection by formulating a composite modularity metric of the partitioned network?
2) What is the extent to which the modularity index increases in comparison to the modularity index generated by Clauset-Newman-Moore algorithm?

In other words, this research investigates whether leveraging the presence of a bipartite subgraph in the given network helps us detect a better underlying community structure of the network. Here, the quality metric used to evaluate the community structure is modularity. In our research, we have used modularity as the quality indicator because the baseline algorithm that we use, Clauset-Newman-Moore, is a modularity based agglomerative algorithm.

3.2. General methodology and empirical setting

This section outlines the methodology used to address the research questions. Let us first review the variables used in this study. Table 1 provides a comprehensive list of variables used in our study.

Table 1: Variables

Variable	Description
$e\_bipartite$	The total number of edges/links in the bipartite graph.
e_total	The total number of unique edges/links in the original graph.
$nodes$	The total number of unique nodes/vertices present in the graph.
Q_{CNM}	The modularity value of the community structure of the entire network as evaluated by utilizing the Clauset-Newman-Moore algorithm. A detailed explanation of the working functionality of the algorithm can be found in the Section 2.10.
$Q_{CNM:Remainder}$	The modularity value of the community structure of the remainder network as detected by utilizing the Clauset-Newman-Moore algorithm. Here, the remainder network is the original network, i.e., a bipartite network. Considering the original graph $G = (V, E)$ and the identified bipartite network in the graph $G_B = (V', E')$, the remainder graph is $G_R = (V'', E'')$, where the following holds: $V'' = V - V'$ and $E'' = E - E'$
Q_B	The modularity value of the community structure of the Bipartite subgraph as evaluated by utilizing the BiLouvain algorithm. A detailed explanation of the working functionality of this algorithm can be found in the Section 2.8
$Q_{Composite}$	The composite modularity evaluated using Equation 20.
Q_{diff}	The magnitude of the extent to which the quality of the communities detected using the proposed empirical setting is greater than the one evaluated by Q_{CNM} alone. Essentially, $Q_{diff} = Q_{Composite} - Q_{CNM}$.

Explanatory/Independent variables: $e\_bipartite,\ e\_total, nodes$

Dependent variables: Q_{diff} , $Q_{Composite}$, Q_{CNM}, $Q_{CNM:Remainder}$, Q_B

Most important dependent variable for this study: Q_{diff}

Figure 3 is a flowchart that delineates the overall procedure for performing the community detection and evaluation of the modularity index after having identified the bipartite subgraph.

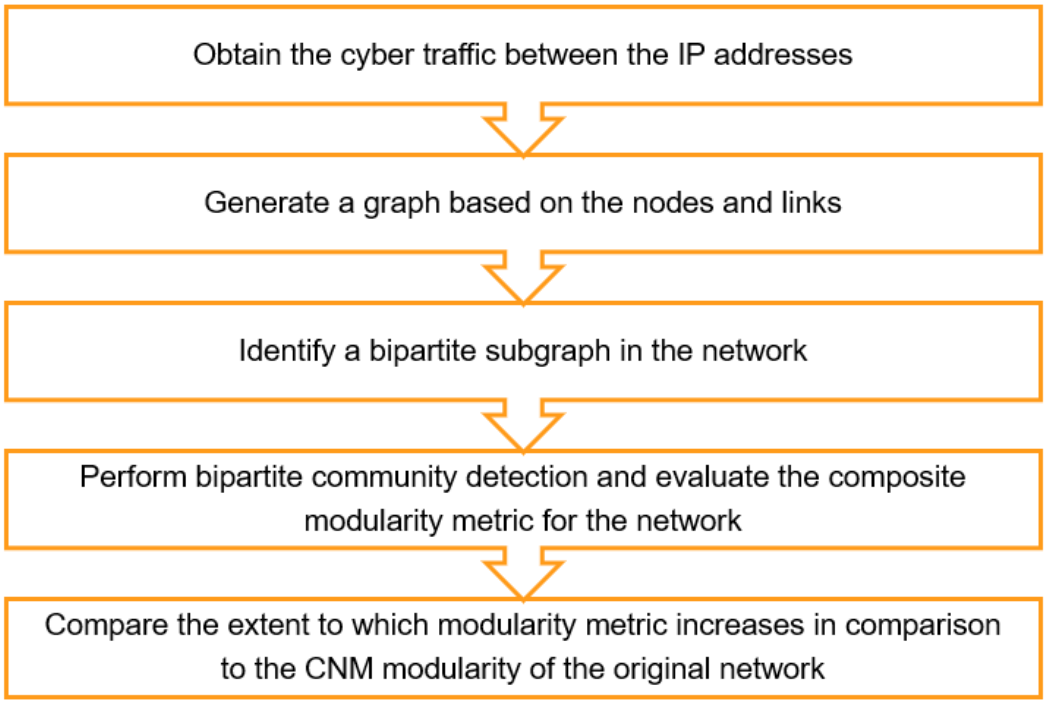

Figure 3: Workflow

1. Obtain the cyber traffic between the IP addresses:
    a. Extract the source and destination IP addresses from the packet capture files. This step has been discussed comprehensively in the Section 4.2.
2. Identification of bipartite subgraph in the network:
    a. Build an edge list file based on the source-destination pairs obtained from step 1.
    b. Build an adjacency matrix for the network.
    c. Identify a bipartite subgraph using the Find_ Bipartite algorithm proposed in this thesis. This algorithm was discussed in Section 2.7.

3. Perform bipartite community detection by using the biLouvain algorithm proposed by Pesantez-Cabrera and Kalyanaraman (2016) and evaluate Q_B. This algorithm was discussed in Section 2.8. Next, the overall modularity of the graph can be calculated by using Equation 19 (Liu, Liu, Murata, & Wakita, 2014).

$$Q_{composite} = Q(L) = \sum_{y=1}^{s} \frac{m^{[y]}}{m} Q^{[y]}(L) \quad \text{(Eqn. 19)}$$

where

$$G = G^{[1]} \cup G^{[2]} \cup \ldots \cup G^{[s]}$$

and it follows that

$$V = V^{[1]} \cup V^{[2]} \cup \ldots \cup V^{[s]}$$

$$E = E^{[1]} \cup E^{[2]} \cup \ldots \cup E^{[s]}$$

$m = \sum_{y=1}^{s} m^{[y]}$ (total number of edges)

$Q^{[y]}$: $modularity\ of\ \ G^{[y]}$

In this research, we have the following two types of networks:

a) Identified Bipartite subgraph and

b) Remainder network.

Based on the two types of networks we use, Equation 19 can now be expressed as Equation 20.

$$\sum_{y=1}^{s} \frac{m^{[y]}}{m} Q^{[y]}(L) = \left(\frac{Number\ of\ edges\ in\ Bipartite\ network}{Total\ number\ of\ edges\ in\ the\ network} \times Q_B\right) +$$

$$\left(\frac{Number\ of\ edges\ in\ the\ remainder\ network}{Total\ number\ of\ edges\ in\ the\ network} \times Q_{CNM:Remainder}\right) \text{(Eqn. 20)}$$

4. Evaluate Q_{CNM} for the entire graph without considering the bipartite subgraph

5. Evaluate $Q_{diff} = Q_{Composite} - Q_{CNM}$.

The aforementioned steps are summarized in Figure 4.

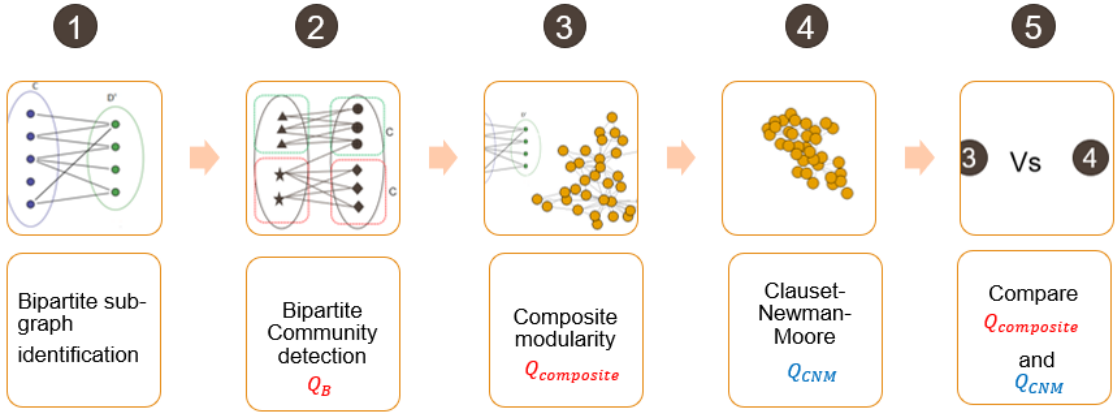

Figure 4: Framework

3.3. Threat to validity

This research focuses on evaluating modularity metric for a static cyber network. The results thus obtained may not directly be generalized to any dynamic cyber network. Also, one needs to be wary about the resolution limit while performing modularity optimization.

3.4. Summary

This chapter outlined the overall research framework, empirical setting, and methodology used in this research. Next, chapter 4 will emphasize on the execution aspects of the proposed methodology.

CHAPTER 4. EXPERIMENT EVALUATION AND RESULTS

This chapter provides an insight into the data source, variables, experimental setup, and parameter configuration. Furthermore, it highlights the research experiment execution workflow that will enable reproducibility and replicability of the results.

4.1. Data source

Our research focuses on community detection as it relates to the cyber networks, and thus we employ a data set consisting of IP addresses. Additionally, we know that a graph is defined by a set of unique vertices and the links connecting the vertices. In our case, the unique IP addresses represent the nodes/vertices of a network where the edges are characterized by the presence of a communication link between the two incident IP addresses. The IP addresses are extracted from the data packet information obtained from Center for Applied Internet Data Analysis (CAIDA). The original data set consisted traffic traces in the form of packet capture (pcap) files. The pcap files were read using the tcpdump packet analyzer tool.

4.2. Data preprocessing

The IP addresses were extracted from the data packet information obtained from tcpdump. The dot-decimal octet notation of the IP addresses was converted to an integer format for the ease of node representation in the graph. For example, let us consider the following IP address: 128.210.105.48. The IP address is broken down into a set of 4 octets with the following integer representation:

$$= 256^3 * first\ octet + 256^2 * second\ octet + 256^1 * third\ octet + fourth\ octet$$

$$= 256^3 * 128 + 256^2 * 210 + 256^1 * 105 + 48$$

$$= 2161273136$$

4.3. Experimental setup

Figure 5 demonstrates the steps undertaken to realize and apply the research framework discussed in the previous section to a real world complex network. The complete code is provided in Appendix A. For the ease of demonstration, each step has been partitioned into four segments: functionality, description, input, and output.

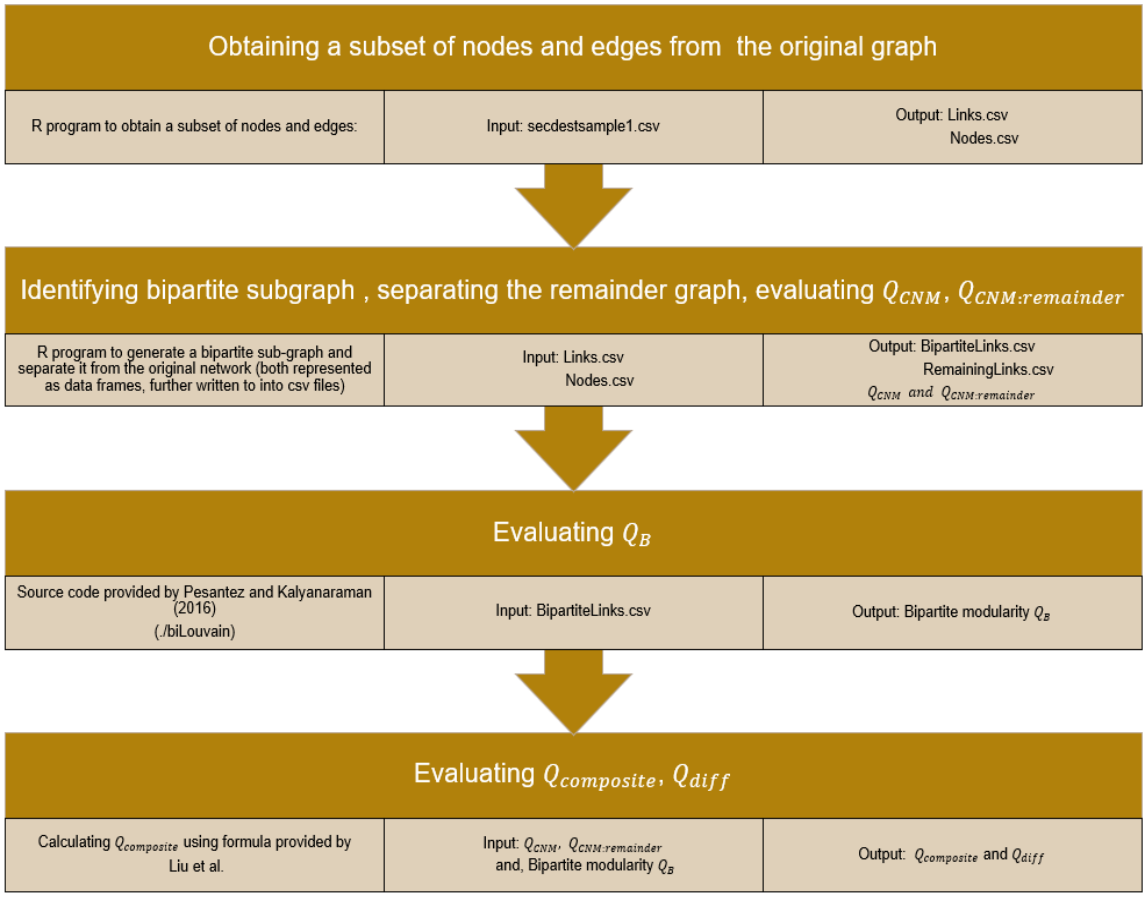

Figure 5: Experimental setup

The following section provides an insight into implementation aspect of the experimental setup.

Modularity evaluation and comparison

Input: $\boldsymbol{G = (V, E)}$

Output: $\boldsymbol{Q_{diff}}$

$e_{total} \leftarrow unique(E)$;

$nodes \leftarrow unique(V)$;

$G_{initial} \leftarrow graph(e_{total}, nodes\ )$

$graph\ edge\ list \leftarrow list\ of\ edges\ for\ the\ entire\ graph$

$bipartite\ edge\ list \leftarrow list\ of\ bipartite\ edges\ for\ the\ sub\ graph$

for $e = 1\ to\ length(graph\ edge\ list)$ **do**

$edges \leftarrow graph\ edge\ list[e]$;

$G_{entire} \leftarrow graph(graph\ edge\ list[e]\ )$;

$Community_{entire} \leftarrow CNM(G_{entire}\ )$;

$Q_{CNM} \leftarrow modularity(Community_{entire}\ )$;

for $e' = 1\ to\ length(bipartite\ edge\ list)$ **do**

$e_{bipartite} \leftarrow e'[bipartite\ edge\ list]$**;**

$e_{remainder} \leftarrow\ \ edges - e_{bipartite}$;

$G_{bipartite} \leftarrow FindBipartite(G_{entire}, e_{bipartite}\ )$;

$G_{remainder} \leftarrow graph(G_{entire} - G_{bipartite}\ , e_{remainder})$;

$Community_{bipartite} \leftarrow biLouvain(G_{bipartite}\ )$;

$\boldsymbol{Q_B} \leftarrow modularity(Community_{bipartite}\ )$;

$Community_{remainder} \leftarrow CNM(G_{remainder})$;

$\boldsymbol{Q_{CNM:Remainder}} \leftarrow modularity(Community_{remainder}\ )$;

$Q_{composite} \leftarrow \left(\frac{e\_bipartite}{edges} \times Q_B\right) + \left(\frac{e_{remainder}}{edges}\ \times Q_{CNM:Remainder}\ \right)$;

$\boldsymbol{Q_{diff}} \leftarrow (Q_{composite} - Q_{CNM})$;

end for

end for

4.4. Execution workflow

Figure 6 provides an insight into the execution workflow. All the programs were executed on Purdue Rice community cluster execution environment. Table 2 details the specifications of the Rice HPC community cluster.

Table 2: Purdue Rice Community Cluster Specifications

Operating System	Red Hat Enterprise Linux 6
Workload manager	Moab
Resource manager	TORQUE
# of nodes	576
Processors per node	Two 10-Core Intel Xeon-E5
Cores per node	20
Memory per node	64GB

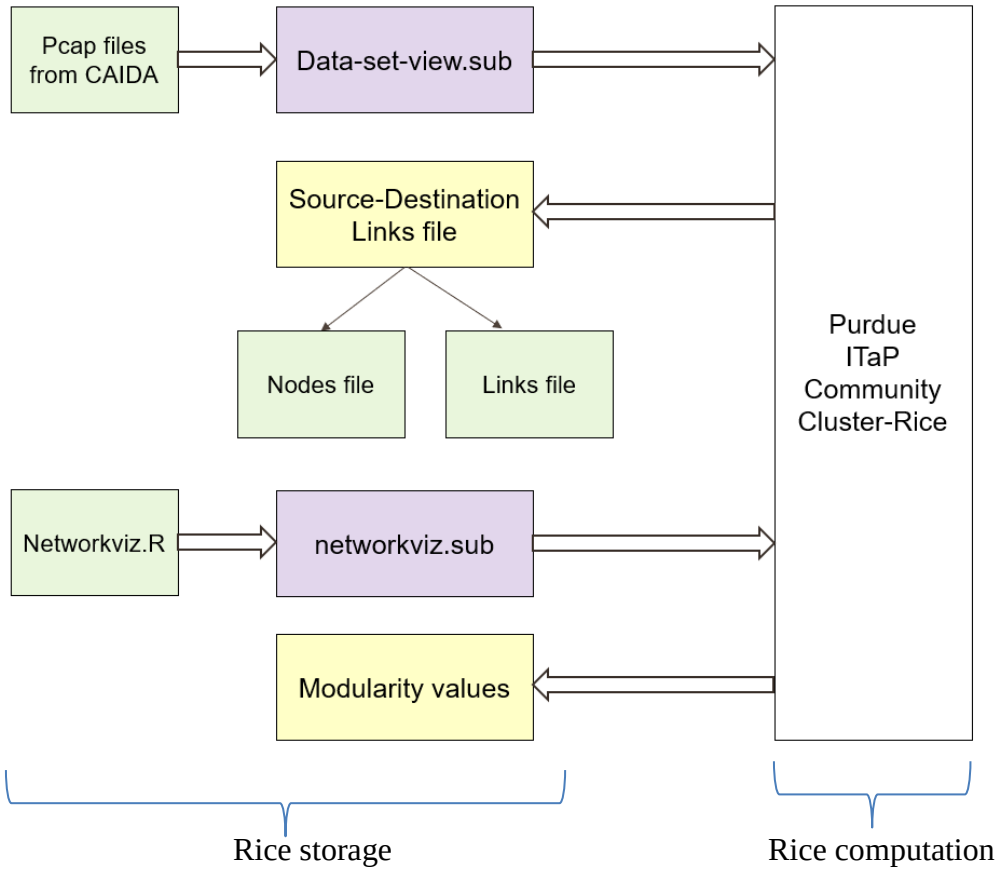

Figure 6: Execution Workflow

Recall that the IP traces were obtained from the CAIDA data set. Jobs are submitted to Rice using. sub file. For example, networkviz.sub contains the instructions to execute Networkviz.R file that in turn yields the output in the modularity values file. Figure 6 is intended for illustrative purpose and does not accommodate the entire execution workflow. The exhaustive list of tasks performed were discussed in the Section 4.3.

4.5. Parameter configuration

The explanatory variables were tuned to observe the effect of the bipartite graph size on the composite modularity. For a small scale bipartite graph size of 10-100 vertices, our empirical setting was not advantageous because such relatively small bipartite subgraph did not contribute much to the bipartite modularity thereby decreasing the overall magnitude of the first term in

Equation 20. Thus, the expected bipartite subgraph size was tuned to 100 and above. Next, the total nodes in a graph were obtained from the original list of non-unique vertices ranging from 40,000 to 100,000 nodes. In this research, we performed multiple experiments by varying the nodes count of the original graph.

4.6. Results and analysis

This chapter provides experimental observations and the results obtained. We also examine the effect of variation of some of the crucial parameters we discussed in Section 4.5 on the quality of the detected communities. Thus, this section demonstrates the applicability of our proposed research framework to the real world complex network.

4.6.1. Empirical observations and results

This section presents the experimental observations. As discussed in the experimental setup in Section 4.3, we are primarily interested in the following variables:

1. Size of the bipartite subgraph ($e\_bipartite$)
2. Size of the original graph ($nodes/e\_total$)
3. Increase in the modularity value by employing our method (Q_{diff})

Here, the Q_{diff} values are averaged over four iterations. Table 3 presents the results obtained by performing experiments using the following values of the input variables:

1. $graph\_edge\_list = \{500000, 600000, 700000, 800000, 900000, 1000000\}$

   Here, the $graph\_edge\_list$ represents the set of number of edges considered from the original graph. For each count of the number of edges mentioned in the $graph\_edge\_list$, we consider only unique edges while building a graph.
2. $bipartite\_edge\_list = \{100, 300, 500, 700, 900\}$

$bipartite\_edge\_list$ represents the set of the bipartite subgraph edges.

Please note that the variables e_B and $Q_{CNM:R}$ mentioned in Table 3 correspond to $e\_bipartite$ and $Q_{CNM:Remainder}$ variables specified in Table 1 respectively.

Table 3: Experimental Observations

e_B	e_total	$nodes$	Q_{CNM}	Q_B	$Q_{CNM:R}$	$Q_{composite}$	Q_{diff}
100	45835	53000	0.989022	0.994461	0.99113	0.991137	**0.002115**
300	45835	53000	0.989022	0.997986	0.994358	0.994382	**0.00536**
500	45835	53000	0.989022	0.992558	0.995469	0.995437	**0.006415**
700	45835	53000	0.989022	0.995906	0.995796	0.995798	**0.006776**
900	45835	53000	0.989022	0.987961	0.996762	0.996589	**0.007567**
100	50377	58056	0.990038	0.994853	0.991556	0.991563	**0.001524**
300	50377	58056	0.990038	0.997997	0.994131	0.994154	**0.004116**
500	50377	58056	0.990038	0.996576	0.995761	0.995769	**0.005731**
700	50377	58056	0.990038	0.990406	0.996131	0.996051	**0.006013**
900	50377	58056	0.990038	0.994565	0.996546	0.996511	**0.006472**
100	54972	63161	0.989859	0.994657	0.991111	0.991117	**0.001259**
300	54972	63161	0.989859	0.998063	0.993637	0.993661	**0.003802**
500	54972	63161	0.989859	0.998624	0.994691	0.994727	**0.004869**
700	54972	63161	0.989859	0.99743	0.995521	0.995546	**0.005687**
900	54972	63161	0.989859	0.998113	0.996366	0.996395	**0.006536**
100	59556	68163	0.989938	0.99505	0.991372	0.991379	**0.00144**
300	59556	68163	0.989938	0.99793	0.993332	0.993355	**0.003417**
500	59556	68163	0.989938	0.998616	0.994708	0.994741	**0.004802**
700	59556	68163	0.989938	0.999061	0.995476	0.995518	**0.00558**
900	59556	68163	0.989938	0.993583	0.995901	0.995866	**0.005928**
100	64210	73186	0.990027	0.994265	0.991048	0.991053	**0.001026**
300	64210	73186	0.990027	0.998063	0.992651	0.992676	**0.002649**
500	64210	73186	0.990027	0.998723	0.993695	0.993734	**0.003708**
700	64210	73186	0.990027	0.999095	0.995121	0.995165	**0.005138**
900	64210	73186	0.990027	0.992383	0.995883	0.995834	**0.005807**
100	68637	77924	0.989764	0.994853	0.990833	0.990839	**0.001075**
300	68637	77924	0.989764	0.998052	0.992563	0.992587	**0.002824**
500	68637	77924	0.989764	0.994675	0.993594	0.993602	**0.003838**
700	68637	77924	0.989764	0.994696	0.994045	0.994052	**0.004288**
900	68637	77924	0.989764	0.994584	0.995342	0.995332	**0.005568**

Based on the observations, following are the key results:

We observe that the evaluated Q_{diff} values are all positive. Furthermore, we know that $Q_{diff} = Q_{Composite} - Q_{CNM}$. This indicates that our proposed technique of identifying and using the specific subgraph for community detection was advantageous. It is primarily due to the improvement of modularity value of the community structure associated with the real world graph under consideration.

It can also be observed that Q_{diff} has a positive relationship with the bipartite subgraph size (e_B). One plausible reasoning for this observation is that with the increase in bipartite subgraph size, we are essentially expecting an increase in the edges and/or nodes contributing towards the bipartite modularity. Moreover, as the bipartite modularity (Q_B) is evaluated by considering the null model specific to the bipartite graph, it yields a better community structure.

In the next section, we will discuss the effect of bipartite subgraph size on Q_{diff}.

4.6.2. Effect of varying the size of bipartite subgraph

As discussed earlier in the previous section, Q_{diff} and e_B are positively related. Let us now visually inspect the variation of Q_{diff} with respect to the bipartite subgraph size. Figure 7 is a line plot generated from the observations mentioned in Table 3 (Section 4.6.1) for the original graph size (in terms of e_total) of 45835 unique edges. Here, we plot the Q_{diff} values corresponding to the bipartite subgraph sizes ranging from 100 to 900 (with an increment of 200).

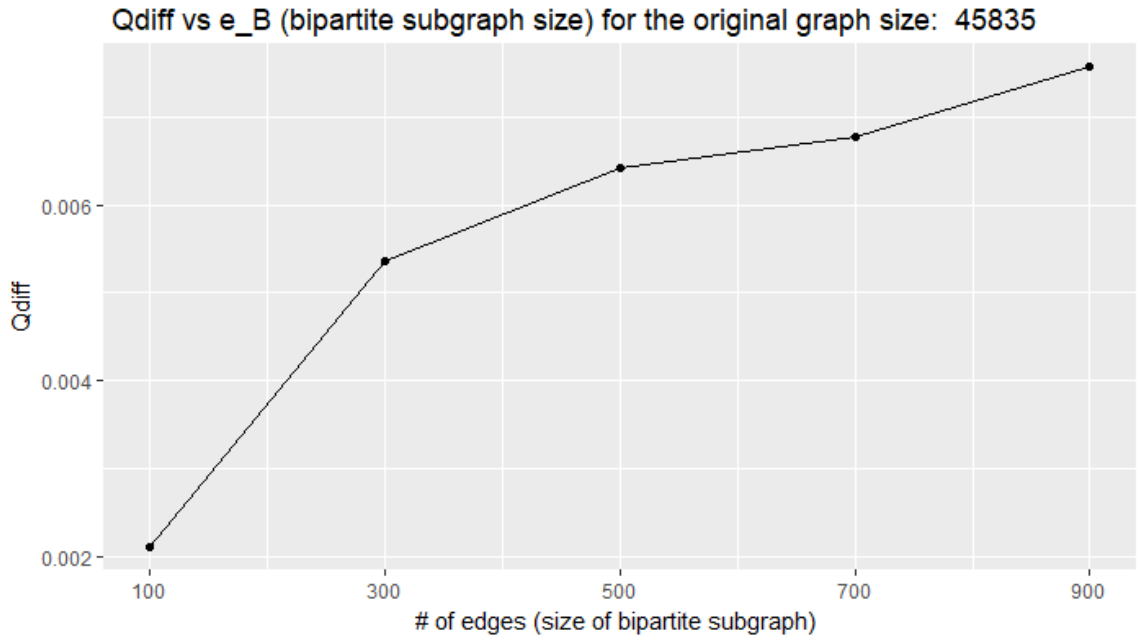

Figure 7: Line plot for $Qdiff$ vs e_B for $e\_total = 45835$

We observe that $Qdiff$ increases with the increase in bipartite subgraph size (e_B). Recall from Section 3.2. that we had employed biLouvain algorithm to evaluate the bipartite modularity (i.e., modularity value of the bipartite subgraph community structure) and because biLouvain considers null model specific to the bipartite graph, we expect an increase in the overall composite modularity. Furthermore, with an increase in the bipartite subgraph size, we are essentially increasing the total number of edges contributing towards a graph where we utilize the specific (most appropriate) null model (e.g., our bipartite subgraph model) instead of a generic model (e.g., our remainder graph model).

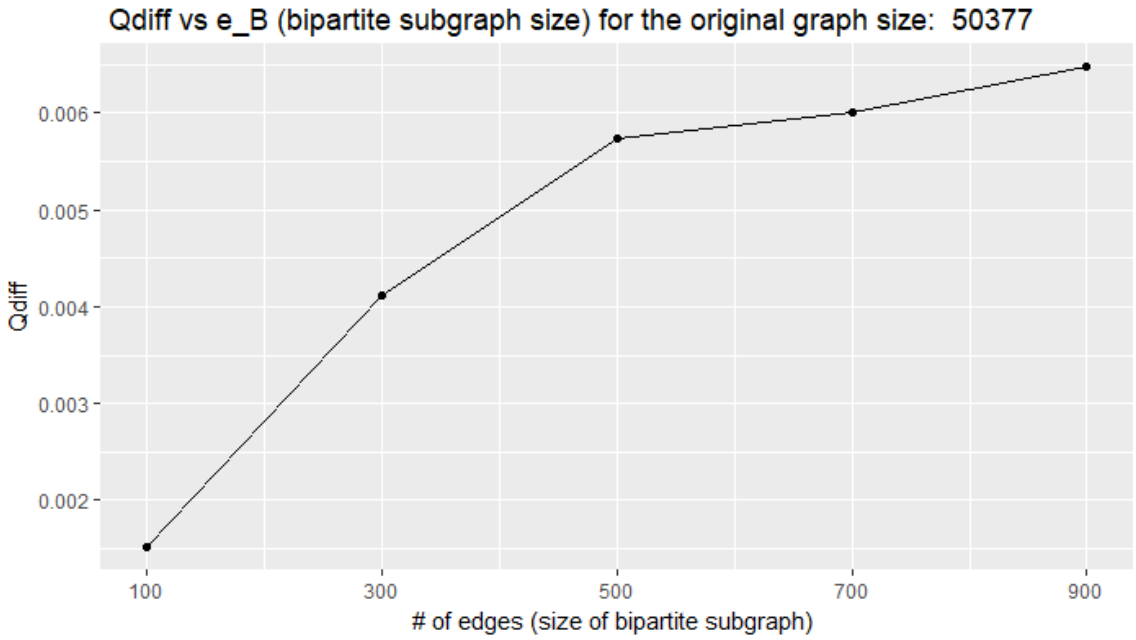

Figure 8: Line plot for $Qdiff$ vs e_B for $e_{total} = 50377$

Here; like the previous line plot (Figure 7), we observe that the $Qdiff$ values exhibit a positive relationship with the bipartite subgraph size (e_B).

Similarly, we generate line plots based on the observations mentioned in Table 3 for all the remaining e_total values. Figures 9, 10, 11, and 12 illustrate the variation of $Qdiff$ with respect to the bipartite subgraph size for the original graphs consisting of 54972, 59556, 64210, and 68637 unique edges respectively.

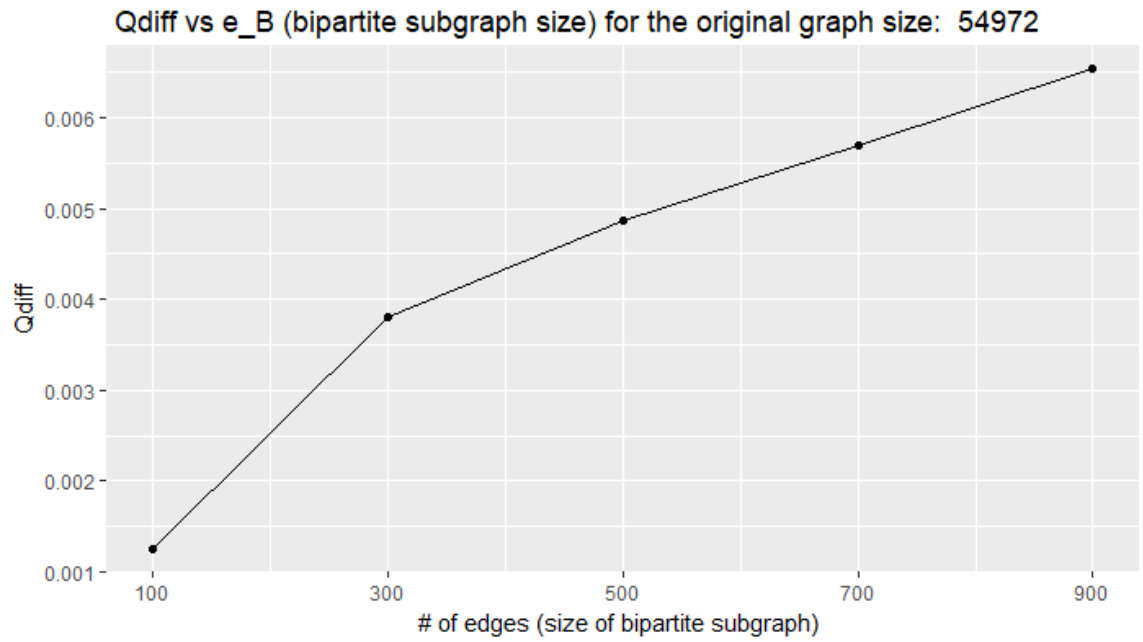

Figure 9: Line plot for $Qdiff$ vs e_B for $e\_total = 54972$

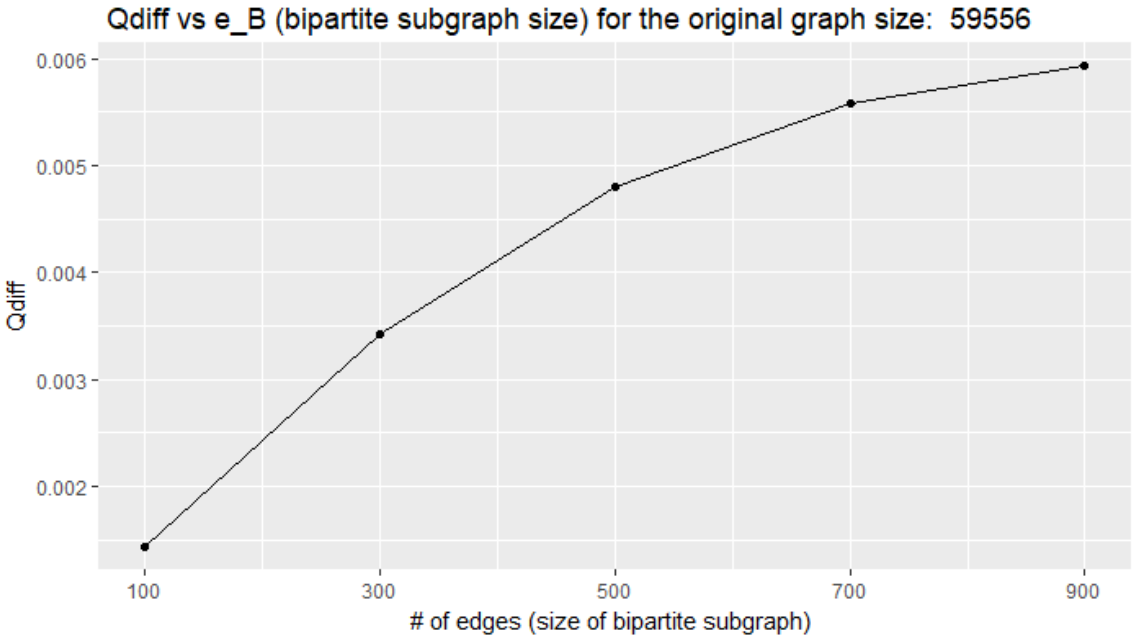

Figure 10: Line plot for $Qdiff$ vs e_B for $e\_total = 59556$

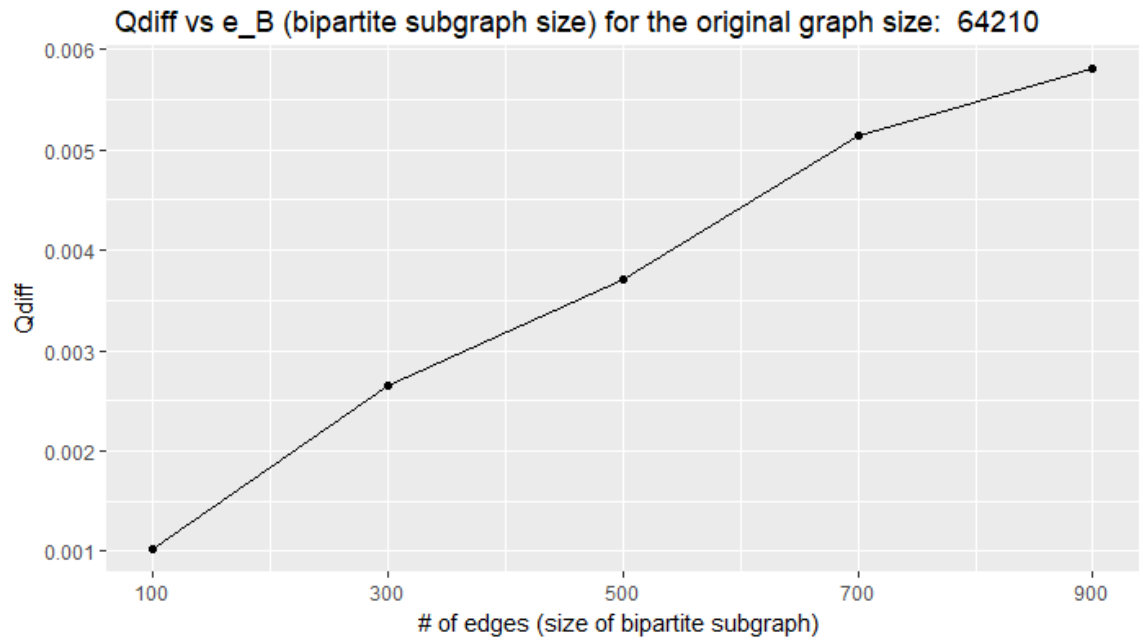

Figure 11: Line plot for $Qdiff$ vs e_B for $e\_total = 64210$

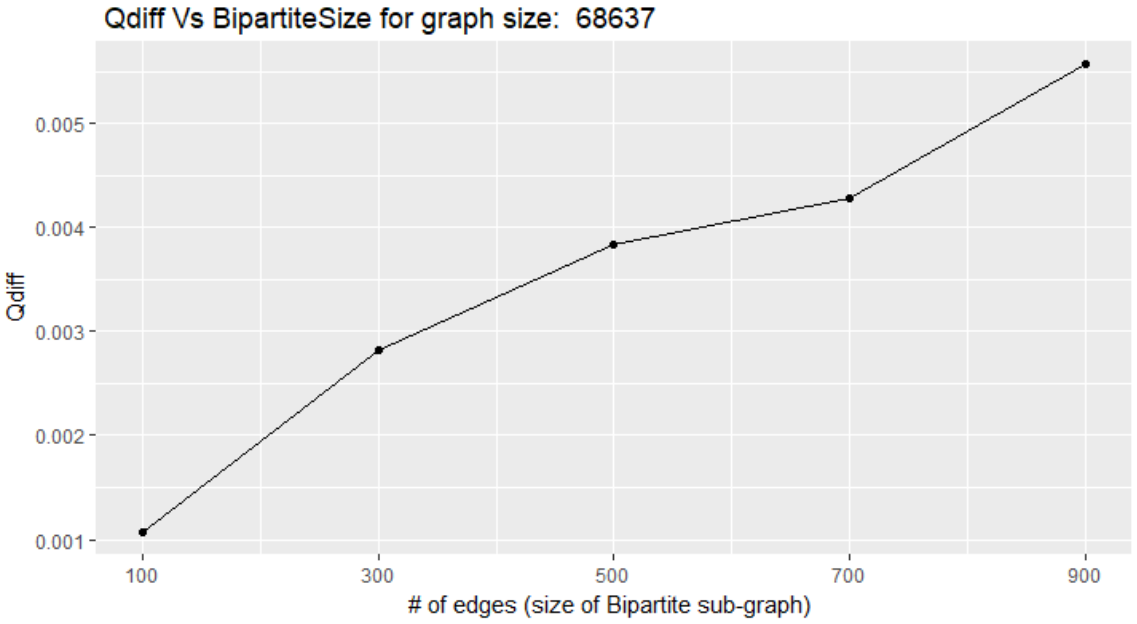

Figure 12: Line plot for $Qdiff$ vs e_B for $e\_total = 68637$

It is evident from Figures 9, 10, 11, and 12 that $Qdiff$ increases with the increase in bipartite subgraph size. It follows the same argument stated earlier (Section 4.6.1.) that increasing the bipartite subgraph size subsequently increases the composite modularity $Q_{composite}$, thereby enhancing the $Qdiff$ value. Figure 13 presents a comprehensive picture of the line plots we saw earlier in this section.

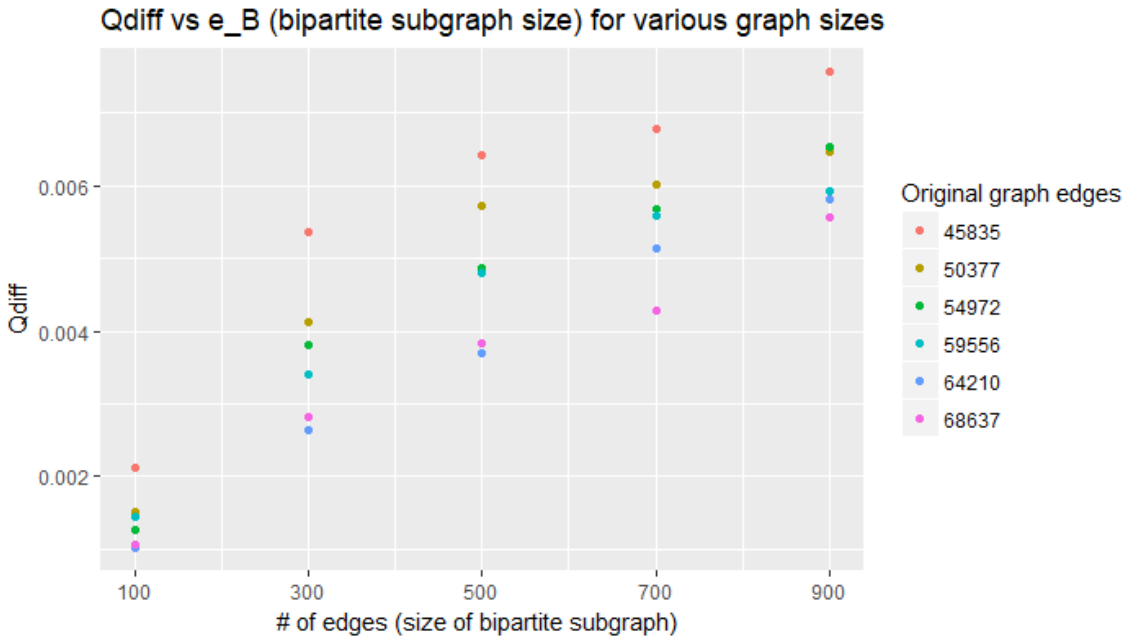

Figure 13: Line plot for $Qdiff$ vs e_B

The next chapter highlights the research findings and describes how our results help address the research questions we posed in Section 1.4.

CHAPTER 5. CONCLUSION AND FUTURE DIRECTION

This chapter presents the major research findings from this work. Next, conclusion and a vision for the future have been discussed.

5.1. Research findings and contribution

The principal goal of this thesis was to investigate whether considering the presence of a bipartite subgraph results in an increase in the overall composite modularity. However, along the way, various other interesting observations were made. First, we identified the need to develop an algorithm that identifies a bipartite subgraph in the given graph. This algorithm was discussed in Section 2.7. Recall that we referred to this algorithm as Find_Bipartite algorithm. The development of the Find_Bipartite algorithm addresses the first segment of our research question 1, that is, obtaining a bipartite subgraph from the given network. Second, we proposed an empirical methodology to evaluate bipartite modularity, composite modularity and Q_{diff} (please recall that $Q_{diff} = Q_{Composite} - Q_{CNM}$). This experimental setup was discussed in Chapter 3. Chapter 3 systematically explained how we can use the identified bipartite subgraph in community detection by formulating a composite modularity metric. It also highlighted the method to evaluate the extent to which this composite modularity increased in comparison to the modularity value obtained by employing just the Clauset-Newman-Moore algorithm; this addressed our second research question. Revisiting the applicability of our research in the cybersecurity domain (Section 1.5), we can potentially alleviate the consequence of worm attack by employing our empirical setting to the compromised network for identifying the most vulnerable (to worm infection) set of nodes in the network. For instance; given the first compromised node (say node X) in the network, the most vulnerable set of nodes is characterized

by the nodes belonging to node X's community. Thus, it is essential to identify and exploit the underlying community structure that possesses a high modularity. Our results suggest that it is advantageous to identify the presence of a bipartite subgraph and incorporate the composite modularity for performing community detection to obtain a high modularity community structure. Third; for a relatively large network (50,000 or more nodes), the bipartite subgraph size and Q_{diff} exhibit a positive relationship. The effect of variation of bipartite subgraph size on $Qdiff$ was discussed in Section 4.6.

5.2. Discussion and conclusion

This thesis primarily focuses on designing and developing an empirical setting that will enable us to investigate whether considering the presence of a bipartite subnetwork aids towards obtaining a better community structure of the network. As discussed in the previous section, we observe an overall enhancement in the quality of the detected communities when the presence of bipartite subgraph is considered while performing community detection. Furthermore, our results corroborate with the initial proposed idea that using an appropriate null model for the specific underlying subnetwork enhances the quality of the community structure. However, one limitation of using this approach is the overall increased time complexity. This is primarily due to the empirical setup (a five-fold process of identifying the bipartite graph, separating the remainder graph, computing the CNM modularity of the remainder graph and the original graph, evaluating composite modularity, and calculating the modularity difference). Here, we can notice a tradeoff between the quality of the detected community structure and computational complexity (considering both time and space hierarchy) required. Thus, this calls for a wise decision over the choice of one constraint over the other based on the underlying application specification and requirements.

5.3. Future work

One potential future direction would be generalization of the experimental setup presented in this research to weighted and/or directed networks. We wish that this initial effort of achieving a better community structure by using the appropriate null reference model for the bipartite subnetwork will provide an encouragement to explore and use other null reference models corresponding to any other identifiable subgraphs present in the network. To exemplify, it would be interesting to observe the results of incorporating null models for cyclic or k-partite subnetworks.

Yet another direction for future research is to investigate the effect of accommodating more than two null models for a single network. This research emphasized on modularity as the quality criterion. Apart from modularity, it will be interesting to study the effect of employing our proposed methodology on various other quality metrics such as conductance, cut-ratio, etc.

REFERENCES

Aiello, W., Kalmanek, C., McDaniel, P., Sen, S., Spatscheck, O., & Van der Merwe, J. (2005, March). Analysis of communities of interest in data networks. In *International Workshop on Passive and Active Network Measurement* (pp. 83-96). Springer Berlin Heidelberg.

Andrews, G. E. (1998). *The theory of partitions* (No. 2). Cambridge university press.

Barabási, A. L. (2016). *Network science*. Cambridge university press.

Barber, M. J. (2007). Modularity and community detection in bipartite networks. *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, *76*(6). https://doi.org/10.1103/PhysRevE.76.066102

Bedi, P., & Sharma, C. (2016a). Community detection in social networks. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*. https://doi.org/10.1002/widm.1178

Bedi, P., & Sharma, C. (2016b). Community detection in social networks. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *6*(3), 115–135. https://doi.org/10.1002/widm.1178

Blondel, V. D., Guillaume, J.-L., Lambiotte, R., & Lefebvre, E. (2008). Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, *2008*(10), P10008. https://doi.org/10.1088/1742-5468/2008/10/P10008

Boppana, R. B. (1987). Eigenvalues and graph bisection: An average-case analysis. In *28th Annual Symposium on Foundations of Computer Science (sfcs 1987)* (pp. 280–285). IEEE. https://doi.org/10.1109/SFCS.1987.22

Chen, M., Kuzmin, K., & Szymanski, B. K. (2014). Community detection via maximization of modularity and its variants. *IEEE Transactions on Computational Social Systems*, *1*(1), 46–65. https://doi.org/10.1109/TCSS.2014.2307458

Clauset, A., Newman, M. E. J., & Moore, C. (2004). Finding community structure in very large networks. Retrieved from https://arxiv.org/pdf/cond-mat/0408187.pdf

Fortunato, S. (2010). Community detection in graphs. *Physics Reports*. https://doi.org/10.1016/j.physrep.2009.11.002

Fortunato, S., & Barthelemy, M. (2006). Resolution limit in community detection, 1–8. https://doi.org/10.1073/pnas.0605965104

Francisco, A. P., & Oliveira, A. L. (2011). On community detection in very large networks.

Communications in Computer and Information Science.

Girvan, M., & Newman, M. E. J. (2002). Community structure in social and biological networks. *Proceedings of the National Academy of Sciences of the United States of America*, *99*(12), 7821–6. https://doi.org/10.1073/pnas.122653799

Leskovec, J. (2008). Dynamics of Large Networks. Retrieved from http://reports-archive.adm.cs.cmu.edu/anon/anon/ml2008/CMU-ML-08-111.pdf

Leskovec, J., Lang, K. J., & Mahoney, M. (2010). Empirical comparison of algorithms for network community detection. *Conference on World Wide Web {WWW}*, 631–640. https://doi.org/http://doi.acm.org/10.1145/1772690.1772755

Liu, X., Liu, W., Murata, T., & Wakita, K. (2014). A framework for community detection in heterogeneous multi-relational networks. Retrieved from https://arxiv.org/pdf/1407.4989.pdf

Medus, A. D., & Dorso, C. O. (2009). Alternative approach to community detection in networks. *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*. https://doi.org/10.1103/PhysRevE.79.066111

Mubayi, D., & Turán, G. (2010). Finding bipartite subgraphs efficiently. *Information Processing Letters*, *110*(5), 174–177. https://doi.org/10.1016/j.ipl.2009.11.015

Newman, M. E. J. (2004). Fast algorithm for detecting community structure in networks. *Physical Review E*, *69*(6), 66133. https://doi.org/10.1103/PhysRevE.69.066133

Newman, M. E. J. (2006). Finding community structure in networks using the eigenvectors of matrices. *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, *74*(3). https://doi.org/10.1103/PhysRevE.74.036104

Newman, M., & Girvan, M. (2004). Finding and evaluating community structure in networksNewman, M., & Girvan, M. (2004). Finding and evaluating community structure in networks. Physical Review E, 69, 1–16. https://doi.org/10.1103/PhysRevE.69.026113. *Physical Review E*, *69*, 1–16. https://doi.org/10.1103/PhysRevE.69.026113

Pesantez-Cabrera, P., & Kalyanaraman, A. (2016). Detecting Communities in Biological Bipartite Networks. https://doi.org/10.1145/2975167.2975177

Pizzuti, C., & Clara. (2008). Community detection in social networks with genetic algorithms. In *Proceedings of the 10th annual conference on Genetic and evolutionary computation - GECCO '08* (p. 1137). New York, New York, USA: ACM Press. https://doi.org/10.1145/1389095.1389316

Pons, P., & Latapy, M. (2006). Computing Communities in Large Networks Using Random Walks. *Journal of Graph Algorithms and Applications*, *10*(2), 191–218. Retrieved from http://www.liafa.jussieu.fr/

Raghavan, U. N., Albert, R., & Kumara, S. (2007). Near linear time algorithm to detect community structures in large-scale networks. *Physical Review E*, *76*(3), 36106. https://doi.org/10.1103/PhysRevE.76.036106

Silva, T. C., & Zhao, L. (2016). Complex Networks. In *Machine Learning in Complex Networks* (pp. 15–70). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-17290-3_2

Weisstein, E. W. (n.d.). Cyclic Graph. Retrieved from http://mathworld.wolfram.com/CyclicGraph.html

Xu, K., Wang, F., & Gu, L. (2014). Behavior Analysis of Internet Traffic via Bipartite Graphs and One-Mode Projections. In *IEEE/ACM Transactions on Networking* (Vol. 22, pp. 931–942). https://doi.org/10.1109/TNET.2013.2264634

APPENDIX A. CODE

1. Job submission file: This is the main file that invokes the R code written in the two R files described later and calls for execution of biLouvain algorithm.

```
#!/bin/sh -l
module load r/3.4.3
cd /scratch/rice/h/hdeshmuk/Bipartite/
echo "Hello Harsha!!"


#Declaring the graph edges list (Please refer Chapter 3 for notations)
declare  links_set=(500000 600000 700000 800000 900000 1000000)

#Declaring the bipartite edges list (Please refer Chapter 3 for notations)
declare  bipartite_links_set=(100 300 500 700 900)


for i in ${links_set[@]}
do
  echo "$i">>links_size5.csv
  echo "$i"
  for j in ${bipartite_links_set[@]}
  do
      echo "$j">>bipartite_size5.csv
      echo "$j"

      #Here BipartiteImplementation8ver13.R is the first R file
      R --vanilla --no-save < BipartiteImplementation8ver13.R
      echo "Done evaluating the Qcnm and Qcnm-remaining"
      echo "Now evaluating Qbipartite"

      ./biLouvain -i srcdestsampleBipartiteLinks$i$j.csv -d "," -ci 0.01 -cp
0.00 -initial Harsha145 -o Harsha145

      echo "Done  evaluating Qbipartite"
      echo "$(grep "Murata+" Harsha145_ResultsModularity.txt | sed 's/.*://')
">>test2.csv

     #Here BipartiteImplementation8ver13.R is the second R file
      R --vanilla --no-save < BipartiteImplementation8ver14.R
      echo "Done second R file"
      echo "Number $i"

  done

done
```

2. First R file- This is the first R file mentioned in the job submission file

```
library(network)
library(igraph)
library(dplyr)

#Reading the csv file containing the list of source and destination IP addres
ses. This is without any modification to the links dataset. It contains dupli
cate edges.
#Would be removed in subequent steps
#Already cd'd to the location where srcedestsample1.csv is present.
links <- read.csv("srcdestsample1.csv", header=T, as.is=T)

#Obtaining the initial graph size in terms of edges from the links_size5.csv
file. This file is written during the execution of .sub file
size1<-read.csv("links_size5.csv",as.is=FALSE,header = FALSE)
uu<-tail(size1,n=1)
uu<-uu[[1]]


#Declaring Modularity Matrix
Modularity_matrix1<-matrix(nrow=1, ncol=8)

#Converting it into a data frame to accomodate all the data types
Modularity_matrix1 <- as.data.frame(Modularity_matrix1)

#Specifying the column names, these are our variables of interest
Column_names <- c("e_bipartite", "e_total", "nodes","Qcnm", "Qb", "RQcnm","Qc
omposite","Qdiff")
colnames(Modularity_matrix1) <- Column_names

#Specify counter for the matrix
counter_for_matrix<-1

#Print Modularity values- initially NA's
Modularity_matrix1

#Declaring and defining variable xx to provide unique file names for the bipa
rtite graph and remainder graph
xx<-uu+108

#Subsetting the data for only a specific number of links
print("The total number links are: ")
print(nrow(links))

#Creating a data frame of links and writing it to a csv file
dfrm_all_links <- data.frame(links)

dfrm_all_links<-dfrm_all_links[1:uu,]
```

```
filename_links <- paste('Har_links_',xx,'.csv',sep='')
write.table(dfrm_all_links, file=filename_links, sep=",", row.names=FALSE, co
l.names=TRUE, append = FALSE)

#Creating nodes list
dfrm_all_links1 <- data.frame(source = dfrm_all_links[1:uu,1])
filename_nodes <- paste('Har_nodes_',xx,'.csv',sep='')
write.table(dfrm_all_links1, file=filename_nodes, sep=",", row.names=FALSE, c
ol.names=TRUE, append = TRUE)

dfrm_all_links2 <- data.frame(source = dfrm_all_links[1:uu,2])
#filename_nodes <- paste('Har_nodes_',uu,'.csv',sep='')
write.table(dfrm_all_links2, file=filename_nodes, sep=",", row.names=FALSE, c
ol.names=TRUE, append = TRUE)


#Reading the data frames intonodes and links
nodes <- read.csv(filename_nodes, header=T, as.is=T)
links <- read.csv(filename_links, header=T, as.is=T)

# Prelimainary examination of the data:
head(nodes)
head(links)
nrow(nodes)


#Obtaining unique links : to remove multiple edges from the graph. Essentiall
y obtainig a simple graph
links<-(unique(links[,c("source", "destination")]))
#links<-(unique(links))
nodes<-(unique(nodes[,c("source")]))
#Total number of unique nodes in the graph
length(nodes)
print("The total number of unique links are: ")
print(nrow(links))


#Populating the e_total in the Final Comparison data frame i.e.: Modularity_m
atrix1
Modularity_matrix1[counter_for_matrix,2]<- nrow(links)

#Populating the nodes in the Final Comparison data frame i.e.: Modularity_mat
rix1
Modularity_matrix1[counter_for_matrix,3]<- length(nodes)


#Creating a data frame of links and writing it to a csv file
dfrm_all_links <- data.frame(links)
write.table(dfrm_all_links, file="srcdestsample22UniqueLinks.csv", sep=",", r
ow.names=FALSE, col.names=TRUE, append = FALSE)
```

```
#edge_matrix
edge_matrix<-as.matrix(links)


#Removing the duplicate edges
edge_matrix<-edge_matrix[!duplicated(t(apply(edge_matrix, 1, sort))),]

#Creating a graph data frame from the matrix
net <- graph_from_data_frame(d=edge_matrix, vertices=nodes, directed=F)

# Examine the resulting object:
class(net)


#Counting the total number of nodes
Number_of_edges<-floor(nrow(links))
Number_of_edges

# Function for generating Kt,t Bipartite graph from the given graph represent
ed by {nodes, links}
# This function generates a bipartite graph from the given original graph.
cnt<-1
Find_Bipartite<-function(nodes, links, s,t){


  repeat{
    i<-0
    random_links<-sample(1:nrow(links))
    edges_id<-random_links[1:t]
    Bipartite_set1_final<-c()
    Bipartite_set2_final<-c()
    for(i in 1:t){
      first_random<-links[edges_id[i],1]
      Bipartite_set1<-links[edges_id[i],1]
      Bipartite_set1_final<-c(Bipartite_set1_final, Bipartite_set1)

      second_random<-links[edges_id[i],2]
      Bipartite_set2<-links[edges_id[i],2]
      Bipartite_set2_final<-c(Bipartite_set2_final, Bipartite_set2)

      Characterized_version_Bipartite_set1_final<-as.character( Bipartite_set
1_final)
      Characterized_version_Bipartite_set2_final<-as.character( Bipartite_set
2_final)

      Characterized_first_random<-as.character(first_random)
      Characterized_second_random<-as.character(second_random)
```

```
      matrix_set1<-net[Characterized_first_random,Characterized_version_Bipar
tite_set1_final]
      matrix_set2<-net[Characterized_second_random,Characterized_version_Bipa
rtite_set2_final]
      # print(sum(matrix_set1))
      # print(sum(matrix_set2))

      confirmer<- as.integer(sum(matrix_set1)>0||sum(matrix_set2)>0)

      if(confirmer==1){
        print("The graph is not a Bipartite graph")
        break

      }

    }
    print(Bipartite_set1_final)
    print(Bipartite_set2_final)

    if(i==t){
      print("Done generating a bipartite graph")
      return(c(Bipartite_set1_final, Bipartite_set2_final))
      break
    }
  }



}




# Graph G is expressed as {V,E} where nodes=V, links=E, s: vertices having hi
ghest degree, t=user defined interger for Kt,t bipartite graph
#Reading the input from the bipartite_size5.csv for the bipartite graph size

bipartite_size1<-read.csv("bipartite_size5.csv",as.is=FALSE,header = FALSE)
ww<-tail(bipartite_size1,n=1)
ww<-ww[[1]]

#Calling the Find_Bipartite function
Bipartite_set1_final<-Find_Bipartite(nodes, links, 3,ww)

#This is the first disjoint set of the biparite graph
Bipartite_set1_final[1:ww]
m<-ww+1
n<-2*ww
#This is the second disjoint set of the biparite graph
Bipartite_set1_final[m:n]
```

```
#Populating the e_bipartite in the Final Comparison data frame ie: Modularity
_matrix1
Modularity_matrix1[counter_for_matrix,1]<- ww

#Creating a data frame for bipartite graph and writing in a csv file where so
urce is the first disjoint set and destination is the second disjoint set.
dfrm_bipartite <- data.frame(source = Bipartite_set1_final[1:ww], destination
= Bipartite_set1_final[m:n])
filename <- paste('srcdestsampleBipartiteLinks',uu,ww,'.csv',sep='')
write.table(dfrm_bipartite, file=filename, sep=",", row.names=FALSE, col.name
s=TRUE, append = FALSE)


#Evaluating the bipartite modularity
#The job submission script will evaluate the ./biLouvain after this file is d
one executing

#Creating a data frame for the remainining graph and writing in a csv file
dfrm_remaining<-anti_join(dfrm_all_links, dfrm_bipartite, by=c("source","dest
ination"))
dfrm_remaining<-dfrm_remaining[!(dfrm_remaining$source %in% dfrm_bipartite$so
urce),]
dfrm_remaining<-dfrm_remaining[!(dfrm_remaining$destination %in% dfrm_biparti
te$destination),]
write.table(dfrm_remaining, file="srcdestsample22RemainingLinks.csv", sep=","
, row.names=FALSE, col.names=TRUE, append = FALSE)



#Evaluating Cluset-Newman-Moore modularity for the entire graph. This is eval
uated by considering just the unique links
links <- read.csv(filename_links, header=T, as.is=T)
links_entire<-(unique(links[,c("source", "destination")]))

net_entire<-graph.data.frame(d=links_entire,directed=F)

class(net_entire)
simplify(net_entire, remove.multiple = TRUE, remove.loops = TRUE)
is_simple(net_entire)

#Community detection using Clauset-Newman-Moore Algorithm from igraph package
ceb_fast_entire<-cluster_fast_greedy(net_entire)

#Evaluting the modularity of the communities formed by the CNM algorithm
print("The modularity for the entire graph is:")
complete_graph_modularity<-modularity(ceb_fast_entire)
print(complete_graph_modularity)
```

```
#Populating the Qcnm in the Final Comparison data frame ie: Modularity_matrix
1
Modularity_matrix1[counter_for_matrix,4]<- complete_graph_modularity

#Evaluating Cluset-Newman-Moore modularity for the remainder graph. This is e
valuated by considering by removing the edges that constituted the Bipartite
graph.
links <- read.csv("srcdestsample22RemainingLinks.csv", header=T, as.is=T)
links_remaining<-(unique(links[,c("source", "destination")]))

net_remaining<-graph.data.frame(d=links,directed=F)

class(net_remaining)
simplify(net_remaining, remove.multiple = TRUE, remove.loops = TRUE)
is_simple(net_remaining)

#Community detection using Clauset-Newman-Moore Algorithm from igraph package
ceb_fast_remaining<-cluster_fast_greedy(net_remaining)

#Evaluting the modularity of the communities formed by the CNM algorithm
print("The modularity for the remaining graph is:")
ceb_modularity<-modularity(ceb_fast_remaining)
print(ceb_modularity)

#Populating the Qcnm in the Final Comparison data frame ie: Modularity_matrix
1
Modularity_matrix1[counter_for_matrix,6]<- modularity(ceb_fast_remaining)

#Printing the MODULARITY MATRIX
Modularity_matrix1

#Writing the modularity matrix to the csv file. This file is read by the Bipa
rtiteImplementation8ver14.csv to evaluate the composite modularity.
write.table(Modularity_matrix1, file="Modularity_matrix11.csv", sep=",", row.
names=FALSE, col.names=TRUE, append = FALSE)
```

3. Second R file- This is the second R file mentioned in the job submission file

```
library(network)
library(igraph)
library(dplyr)


#Declaring Modularity Matrix
Modularity_matrix2<-matrix(nrow=1, ncol=8)

#Converting it into a data frame to accomodate all the data types
```

```
Modularity_matrix2 <- as.data.frame(Modularity_matrix2)

#Specifying the column names
Column_names <- c("e_bipartite", "e_total", "nodes","Qcnm", "Qb", "RQcnm","Qc
omposite","Qdiff")
colnames(Modularity_matrix2) <- Column_names



#Reading the partially poplulated data frame from the previous execution of B
ipartiteImplementation8ver13.R
Modularity_matrix1 <- read.csv("Modularity_matrix11.csv", header=T, as.is=T)
Modularity_matrix2 <- read.csv("Modularity_matrix11.csv", header=T, as.is=T)
Modularity_matrix2

#Reading the bipartite modularity from the test2.csv file. This file was writ
ten when the job submission file was executed

aa<-read.csv("test2.csv",as.is=FALSE,header = FALSE)
#Consider the last entry from the test1.csv file
aa<-tail(aa,n=1)

#Insert the bipartite modularity value in the Modularity_matrix2
Modularity_matrix2$Qb<-aa[[1]]


#Total number of unique edges in the graph
e_total<-Modularity_matrix2$e_total

#Total number of bipartite edges in the graph
e_bipartite<-Modularity_matrix2$e_bipartite

#Total number of unique edges in the remainder graph
e_remaining<-e_total-e_bipartite

Modularity_matrix2

#Now evaluating the composite modularity

#Modularity of the bipartite graph as obtained from Pesantez and Kalyanaraman
(2016)
Q_b<-Modularity_matrix2$Qb


#Modularity of the remaining graph using CNM algorithm
Q_cnm_remaining<-Modularity_matrix2$RQcnm

#Modularity of the complete graph using CNM algorithm
Q_cnm<- Modularity_matrix2$Qcnm
```

```
#Composite modulrity first component
Q_B<-(e_bipartite/e_total)*Q_b


#Composite modulrity second component
Q_R<-(e_remaining/e_total)*Q_cnm_remaining


#Composite modularity
Q_composite<-(Q_B+Q_R)
Q_composite

#Populating the Qcomposite in the final Comparison data frame i.e.: Modularit
y_matrix2
Modularity_matrix2$Qcomposite<- Q_composite

#Difference between composite modularity and complete CNM modularity for the
entire graph
Q_diff<-Q_composite-Q_cnm
print("The difference in the modularity is:")
Q_diff

#Populating the Qdiff in the final Comparison data frame ie: Modularity_matri
x2
Modularity_matrix2$Qdiff<- Q_diff

Modularity_matrix2

#Writing the Modularity_matrix2 to the file. This is the final results file!
write.table(Modularity_matrix2, file="Modularity_matrix21.csv", sep=",", row.
names=FALSE, col.names=!file.exists("Modularity_matrix21.csv"), append = TRUE
)
```