NATIONAL INSTITUTE OF TECHNOLOGY SILCHAR

Cachar, Assam

B.Tech. VIth Sem

Subject Code: CS-321

Subject Name: Social Network Analysis Lab

Submitted By:

Name : Subhojit Ghimire

Sch. Id. : 1912160

Branch : CSE – B

AIM: TO ANALYSE THE QUALITY OF COMMUNITIES DETECTED WITH WLC METHOD WITH FOLLOWING EXPERIMENTAL SETUP:

QUALITY MEASURES: ANUI, EXTENDED MODULARITY

THEORY:

1. **ANUI (Average Normalised Unifiability and Isolability)**

$$Q_{ANUI}\ (G,C) = \frac{Q_{AUI}\ (G,C)}{2} = \frac{Q_{AVI}\ (G,C)}{1 + \ Q_{AVU}(G,C) \times \ Q_{AVI}(G,C)}$$

$$where, Q_{AVI}(G,C) = \frac{1}{k}\sum_{i=1}^{k} Isolability\ \ (C_i)$$

$$and, Q_{AVI}(G,C) = \frac{1}{k}\sum_{i=1}^{k} Unifiability\ \ (C_i)$$

$$Unifiability\ (C_i) = \sum_{j=1}^{k} Unifiability\ (C_i, C_j)$$

$$Isolability\ (C_i) = \frac{\sum_{u \in C_i v} \delta\ (u,v)}{\sum_{u \in C_i v} \delta\ (u,v) + \ \sum_{u \in C_i, v \notin C_i} \delta\ (u,v)}$$

2. **Extended Modularity:**

$$EQ = \frac{1}{2m}\sum_{i}\sum_{v \in C_i, w \in C_i} \frac{1}{O_v O_w}\left[A_{vw} - \frac{k_v k_w}{2m}\right]$$

$where, O_v\ is\ the\ number\ of\ communities\ to\ which\ the\ node\ v\ belongs$
$k_i\ is\ the\ degree\ of\ this\ node$
$and, m\ is\ the\ total\ number\ of\ edges$

DATASETS: Zachary's Karate Club, LFR Graphs for Overlapping Communities.

CODE:

```
from tkinter import E
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import community as community_louvain
import matplotlib.cm as cm
import math
import networkx as nx
from networkx.algorithms.community import LFR_benchmark_graph
import matplotlib.pyplot as plt
from itertools import combinations

alpha = {}

def f (x, pr=30):
    return 2. * pr * x - pr

def logistic (x):
    b = 1 + np.exp(-f(x))
    return 1.0 / b

def logweight (i, j):
    return logistic (alpha[i])*logistic(alpha[j])

def EQ (graph, communities, weight='weight', p=30, func=logweight):
    q = 0.0
    degrees = dict(graph.degree(weight=weight))
    m = sum(degrees.values())
    n = graph.number_of_nodes()

    for nd in graph.nodes:
        alpha[nd] = 0

    for community in communities:
        for nd in community:
            alpha[int(nd)] = alpha[int(nd)] + 1

    for k in alpha:
        alpha[k] = 1./alpha[k]

    for nd1, nd2 in combinations(graph.nodes, 2):
        if graph.has_edge(nd1, nd2):
            e = graph[nd1][nd2]
            wt = e.get(weight, 1)
        else:
            wt = 0
        for community in communities:
```

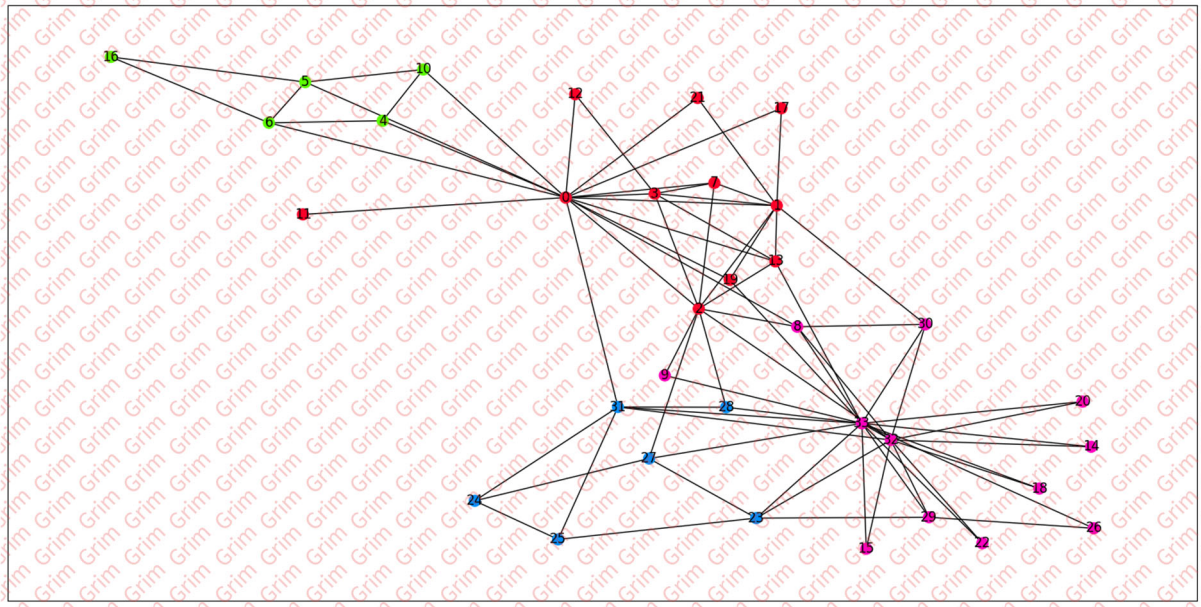
```
            beta_out = 0.0
            for j in graph.nodes:
                if j in community: continue
                if j == nd1: continue
                if j == nd2: continue
                beta_out = beta_out + func(nd1, j)
            beta_out = beta_out / m

            beta_in = 0.0
            for i in graph.nodes:
                if i in community: continue
                if i == nd1: continue
                if i == nd2: continue
                beta_in = beta_in + func(i, nd2)
            beta_in = beta_in / m

            q = q + func(nd1, nd2)*wt - float(beta_in * beta_out *
degrees[nd1] * degrees[nd2] / m)
    return q / m

def delta(u,v):
  return math.mat[u][v]

def Unifiability(Graph, Ci, Cj, mat):
    sum1, sum2, sum3 = 0, 0, 0
    for i in Ci:
        for j in Cj:
            sum1 += int (mat [[i], [j]])
    for i in Ci:
        for j in Graph:
            sum2 += int (mat [[i], [j]])
        for j in Cj:
            sum2 -= int (mat [[i], [j]])
    for i in Cj:
        for j in Graph:
            sum3 += int (mat [[i], [j]])
        for j in Ci:
            sum3 -= int (mat [[i], [j]])
    return sum1 / (sum2 + sum3 - sum1)

def AVU (Graph, cluster, mat):
    sum_Unifiability = 0
    for i in cluster:
        for j in cluster:
            if i != j:
                sum_Unifiability += Unifiability (Graph, i, j, mat)
    return sum_Unifiability / len (cluster)
```

```
def isolability (Graph, Ci, mat):
    sum1, sum2 = 0, 0
    for i in Ci:
        for j in Ci:
            sum1 += int (mat [[i], [j]])
    for i in Ci:
        for j in Graph:
            if i != j:
                sum2 += int (mat [[i], [j]])
    return sum1 / (sum1 + sum2)

def AVI (Graph, cluster, mat):
    sum = 0
    for i in cluster:
        sum += isolability (Graph, i, mat)
    return sum / len (cluster)

def AUI (Graph, cluster, mat):
    AVI_G = AVI (Graph, cluster, mat)
    AVU_G = AVU (Graph, cluster, mat)
    return (2 * AVI_G) / (1 + AVU_G * AVI_G)

def ANUI (Graph,cluster,mat):
    return AUI(Graph,cluster,mat)/2

def readGraph (graphName):
    Graph = nx.read_gml (graphName, label = 'id')
    partition = community_louvain.best_partition (Graph)
    pos = nx.spring_layout (Graph)
    cmap = cm.get_cmap ('rainbow', max (partition.values()) + 1)

    mat = nx.to_numpy_matrix (Graph)

    cluster = [[]]
    maxPartitionVal = 0
    for i in partition:
        if partition[i] > maxPartitionVal:
            maxPartitionVal = partition[i]
    for i in range (maxPartitionVal):
        cluster += [[]]
    for i in partition:
        cluster [partition[i]].append (i)

    var = 1
    for i in cluster:
        var += 1

    AVU_G = AVU (Graph, cluster, mat)
```

```
    print ("AVU = ", AVU_G)
    AVI_G = AVI (Graph, cluster, mat)
    print ("AVI = ", AVI_G)
    AUI_G = AUI (Graph, cluster, mat)
    print ("AUI = ", AUI_G)
    ANUI_G = ANUI (Graph, cluster, mat)
    print ("ANUI = ", ANUI_G)
    EQ_G = EQ (Graph, cluster)
    print ("EXTENDED MODULARITY = ", EQ_G)

    plt.figure (figsize = (15, 15))
    nx.draw_networkx (Graph, with_labels = True, node_size = 100, node_color =
list (partition.values()), cmap = plt.get_cmap ('gist_rainbow'))
    plt.show ()

# KARATE CLUB
print ('\nkarate club')
karate_Graph = nx.karate_club_graph ()
nx.write_gml (karate_Graph, "karate.gml")
readGraph ("karate.gml")

# LFR BENCHMARK
print ('\nLFR graph')
LFR_Graph = LFR_benchmark_graph (n = 100, tau1 = 3, tau2 = 1.5, mu = 0.25,
average_degree = 10, min_community = 5, seed = 10)
nx.set_node_attributes (LFR_Graph, {n: ','.join (map (str,
LFR_Graph.nodes[n]['community'])) for n in LFR_Graph.nodes()}, 'community')
nx.write_gml (LFR_Graph, "lfrgraph.gml")
readGraph ("lfrgraph.gml")
```

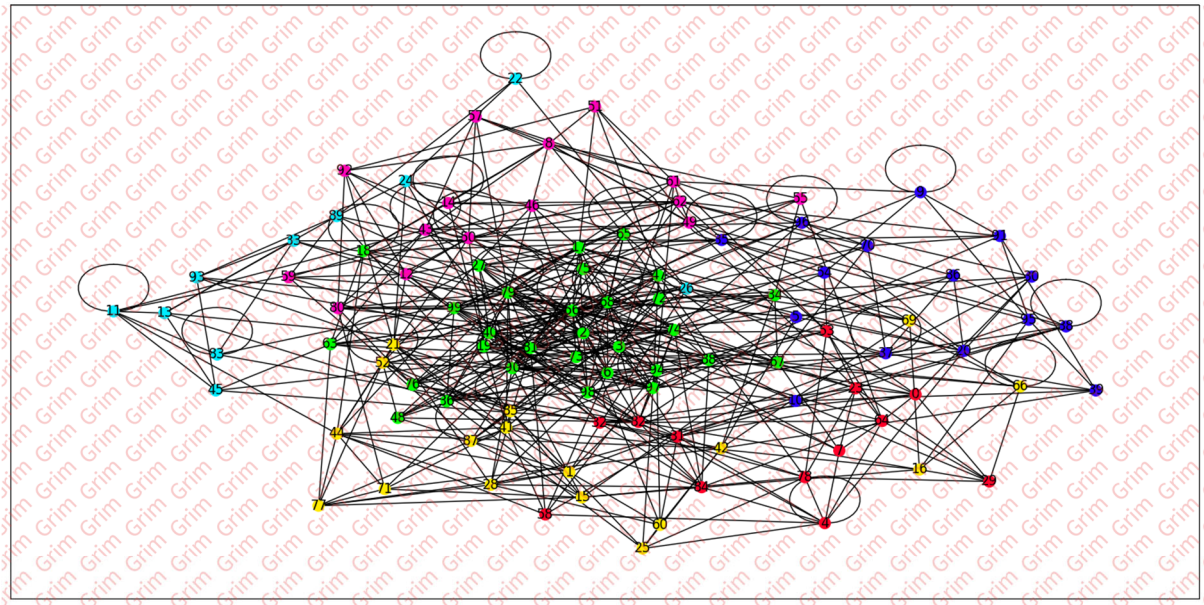
OUTPUT AND OBSERVATIONS (NETWORKX LIBRARY):

// KARATE CLUB



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL                                  Code

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS D:\Documents\NITS\Semester VI\(LAB) CS321 SNA> python -u "d:\Documents\NITS\Semester VI\(LAB) CS321 SNA\References and Ma
terials\Github\lab6.py"

karate club
AVU =  0.12367403742386311
AVI =  0.41380975699944533
AUI =  0.7873261134264353
ANUI =  0.39366305671321766
EXTENDED MODULARITY =  1.9539051960366318
```

// LFR Graph



```
LFR graph
AVU =  0.18931462744395922
AVI =  0.3692054025049771
AUI =  0.6901706489109252
ANUI =  0.3450853244554626
EXTENDED MODULARITY =  2.8427846862286015
PS D:\Documents\NITS\Semester VI\(LAB) CS321 SNA>
```