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.

IF YOU HAVE EXTENDED MODULARITY CODE, PLEASE SHARE!!!

CODE:

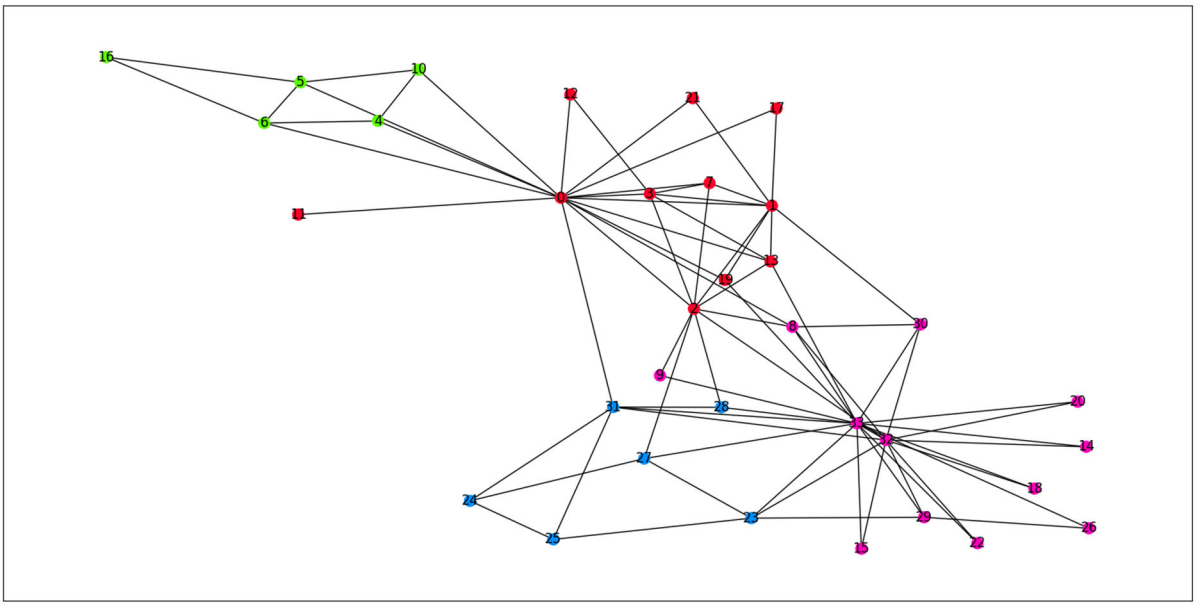
```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import community as community_louvain
import matplotlib.cm as cm
import math
import networkx as nx
from networkx.algorithms.community import LFR_benchmark_graph
import matplotlib.pyplot as plt

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)

    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 = 1000, tau1 = 2, tau2 = 1.1, mu = 0.1,
min_degree = 20, max_degree = 50, max_iters = 2500, seed = 10)
LFR_Graph = LFR_benchmark_graph (n = 250, tau1 = 3, tau2 = 1.5, mu = 0.1,
average_degree = 5, min_community = 20, 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")

# IF YOU HAVE EXTENDED MODALARITY CODE, PLEASE SHARE! WHATSAPP ME OR SHARE
THROUGH YOUR OWN MEDIUM!
```

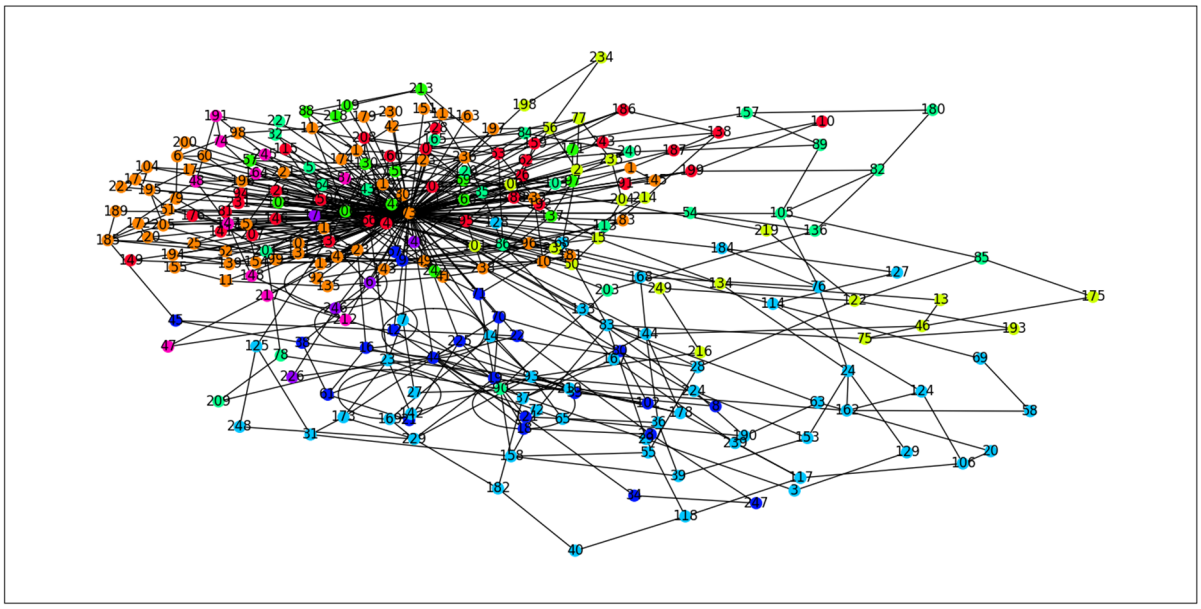
OUTPUT AND OBSERVATIONS (NETWORKX LIBRARY):

// KARATE CLUB




```
PS D:\Documents\NITS\Semester VI\(LAB) CS321 SNA> python -u "d:\Documents\NITS\Semester VI\(LAB) CS321 SNA\lab6_github.py"

karate club
AVU =  0.14741413916146298
AVI =  0.41488154348134487
AUI =  0.7819400955168019
ANUI =  0.39097004775840094
```

// LFR Graph



```
LFR graph
AVU =  0.1395627378263119
AVI =  0.38274291056923154
AUI =  0.7266695682483021
ANUI =  0.36333478412415104
PS D:\Documents\NITS\Semester VI\(LAB) CS321 SNA>
```