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 ACCURACY OF COMMUNITIES DETECTED WITH WLC METHOD WITH FOLLOWING EXPERIMENTAL SETUP:

ACCURACY MEASURES: GENERALISED EXTERNAL INDEX

THEORY:

1. **Generalised External Index (GEI):** These generalised external indexes are used as general measures for comparing two partitions of the same data set into overlapping categories. If falls under the concept of Rand Index (RI), which further falls under the concept of ground-truth-based validation measures for overlapping community structure. GEI is calculated as follows:

$$GEI\ (\Psi, C) = \frac{a_G}{a_G + d_G}$$

$$where, \Psi = \{\Psi_1, \Psi_2, \Psi_3, \dots, \Psi_k\}\ is\ the\ set\ of\ detected\ communities$$

$$C = \{c_1, c_2, c_3, \dots, c_j\} is\ the\ set\ of\ ground - truth\ communities.$$

$$a_G\ is\ agreement\ associated\ to\ pair\ (i,j)\ calculated\ as:$$

$$a_G\ (i,j) = \min\ (\alpha_\Psi\ (i,j),\ \alpha_C\ (i,j)\} + \min\ \{\beta_\Psi\ (i),\ \beta_C\ (i)\} + \min\ \{\beta_\Psi\ (j),\ \beta_C\ (j)\}$$

$$d_G\ (i,j) = \text{abs}\ [\alpha_\Psi\ (i,j) - \alpha_C\ (i,j)] + \text{abs}\ [\beta_\Psi\ (i) - \beta_C\ (i)] + \text{abs}\ [\beta_\Psi\ (j) - \beta_C\ (j)]$$

Here, $\alpha_\Psi\ (i,j)$ represents number of communities shared by nodes i and j in partition Ψ
$\alpha_C\ (i,j)$ represents number of communities shared by nodes i and j in partition C
$\beta_\Psi\ (i)$ represents number of communities in Ψ that node i is a part of, minus 1
$\beta_C\ (i)$ represents number of communities in C that node i is a part of, minus 1
$\beta_\Psi\ (j)$ represents number of communities in Ψ that node j is a part of, minus 1
$\beta_C\ (j)$ represents number of communities in C that node j is a part of, minus 1

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

CODE:

```
import networkx as nx
import numpy as np
import time
import matplotlib.pyplot as plt
from collections import defaultdict
from networkx.algorithms.community import LFR_benchmark_graph

def modu1 (G,N,res):
    m=0
    for  U in res:
        n=len(U)
        S=G.subgraph(U)
        rr=[]
        for kk in res:
            if not kk==U:
                rr.extend(kk)
        ov=list(set(U).intersection(set(rr)))
        sum1= 0
        i=0
        while i<len(U):
            j=i+1
            while j<len(U):
                if U[i] in ov :
                    o=S.degree(U[i])
                    o1=0
                    for ll in res:
                        if U[i] in ll:
                            S1=G.subgraph(ll)
                            o1=o1+S1.degree(U[i])
                    al1=o/o1
                else :
                    al1=1
                if  U[j] in ov  :
                    oo=S.degree(U[j])
                    oo1=0
                    for ll in res:

                        if U[j]in ll:
                            S1=G.subgraph(ll)
                            oo1=oo1+S1.degree(U[j])
                    al2=oo/oo1
                else :
                    al2=1
                if G.has_edge(U[j],U[i]) :
                    x=((1-((G.degree(U[i])*G.degree(U[j]))/(2*N)))*al1*al2)
                    sum1= sum1+2*x
                else :
```

```
                    sum1= sum1+2*((0-
((G.degree(U[i])*G.degree(U[j]))/(2*N)))*al1*al2)
                j=j+1
            i=i+1
        m=m+sum1
    m=m/(2*N)
    return(m)

def WLC(path,sep):
    t=[]
    tri=[]
    G=nx.read_edgelist(path, comments='#', delimiter=sep,
nodetype=int,encoding='utf-8')#txt file
    ns=len(G.nodes())
    N=G.number_of_edges()
    t=[]
    den=nx.density(G)
    re=[]
    res=[]
    res1=[]
    res2=[]
    rr=[]
    w1=[]
    tps1= time.time()
    T11=list(G.nodes())
    i=0
    while i<len(T11):
        cpt1=0
        xx=list (G.neighbors(T11[i]))
        a=len(xx)
        j=0
        while j < a-1:
            j1=j+1
            while j1<a:
                if  G.has_edge(xx[j],xx[j1]):
                   cpt1=cpt1+1
                j1=j1+1
            j=j+1
        if a>1:
            w1.append(2*cpt1/(a*(a-1)))
        else:
            w1.append(0)
        i=i+1
    T=G.nodes()
    while len(T)>0:
        nst=[]
        S=G.subgraph(T)
        for k in T:
```

```
    nst.append([S.degree(k),k])
nst.sort(reverse=True)
l=nst[0][1]
ini=list(set(S.neighbors(l)))
ini.append(l)
n=len(ini)
n1=len(ini)
b=True
while b==True:
    m1=[]
    temp=-1
    for r in ini:
        a=w1[T11.index (r)]
        x=list(S.neighbors(r))
        ww1=0
        ww2=0
        if len(x)>0:
            for rr1 in x:
                d1=w1[T11.index (rr1)]
                d=(d1+len(sorted(nx.common_neighbors(G, r, rr1))))
                ww1=ww1+d
                if rr1 in ini:
                    ww2=ww2+d
            if ww1>0:
                bl=ww2/ww1
                if bl<0.5:
                    ini.remove(r)
    n1=len(ini)
    if  n1<n:
        n=n1
        b=True
    else:
        b=False
b=1
while b==1:
    x=[]
    for k in ini:
        x.extend(G.neighbors(k))
        x=list(set(x)-set(ini))
    n=len(ini)
    m1=[]
    for r in x:

        x1=list(G.neighbors(r))
        ww1=0
        ww2=0
        if len(x1)>0:
            for rr1 in x1:
```

```
                        d1=w1[T11.index(rr1)]
                        d=(d1+len(sorted(nx.common_neighbors(G, r, rr1))))#
                        ww1=ww1+d
                        if rr1 in ini:
                            ww2=ww2+d
                    if ww1>0:
                        bl=ww2/ww1
                        if bl>=0.4:
                            m1.append(r)
            ini.extend(m1)
            n1=len(ini)
            if n1>n:
                b=1
            else:
                    b=0
                    break
        res.append(ini)
        rr.extend(ini)
        T=list(set(T)-set(ini))
        if (len(ini)==0):
            T.remove(l)
    tps2= time.time()
    m=0
    fichier = open("results.txt", "w")
    for res1 in res:
        for k in res1:
            fichier.write(str(k-1))
            fichier.write(' ')
        fichier.write('\n')
    fichier.close()

def GEI_calc(graph,attr):
    n = graph.number_of_nodes()
    gt_membership = [graph.nodes[v][attr] for v in graph.nodes()]
    # print(gt_membership)
    gt_communities = {}
    for i in range(0,n):
        if gt_membership[i] not in gt_communities:
            gt_communities[gt_membership[i]] = []

        gt_communities[gt_membership[i]].append(i)
    communities={}
    i=0
    f = open('results.txt', 'r')
    Lines = f.readlines()
    for line in Lines:
        a = list(map(int,line.split()))
        if len(a):
```

```
            communities[i]=a
            i=i+1
    num = len(communities)
    alpha_psi = []
    for i in range(0, n+1):
        alpha_psi.append(list(np.zeros(n+1, dtype=int)))

    for i in range(0,n):
        for j in range(i+1,n):
            for c in range(0,num):
                if i in communities[c] and j in communities[c]:
                    alpha_psi[i][j] += 1
    alpha_gt = []
    for i in range(0, n+1):
        alpha_gt.append(list(np.zeros(n+1, dtype=int)))

    for i in range(0,n):
        for j in range(i+1,n):
            for c in gt_communities.keys():
                if i in gt_communities[c] and j in gt_communities[c]:
                    alpha_gt[i][j] += 1
    beta_psi = [0 for i in range(0,n)]
    for i in range(0,n):
        for c in range(0,num):
            if i in communities[c]:
                beta_psi[i] += 1
        beta_psi[i] -= 1
    beta_gt = [0 for i in range(0,n)]
    for i in range(0,n):
        for c in gt_communities.keys():
            if i in gt_communities[c]:
                beta_gt[i] += 1
        beta_gt[i] -= 1
    a_G = []
    for i in range(0, n+1):
        a_G.append(list(np.zeros(n+1, dtype=int)))
    for i in range(0,n):
        for j in range(i+1,n):
            a_G[i][j] = min(alpha_psi[i][j], alpha_gt[i][j]) +
min(beta_psi[i], beta_gt[i]) + min(beta_psi[j], beta_gt[j])
    d_G = []
    for i in range(0, n+1):
        d_G.append(list(np.zeros(n+1, dtype=int)))
    for i in range(0,n):
        for j in range(i+1,n):
            d_G[i][j] = abs(alpha_psi[i][j] - alpha_gt[i][j]) +
abs(beta_psi[i] - beta_gt[i]) + abs(beta_psi[j] - beta_gt[j])
    a = np.sum(a_G)
```

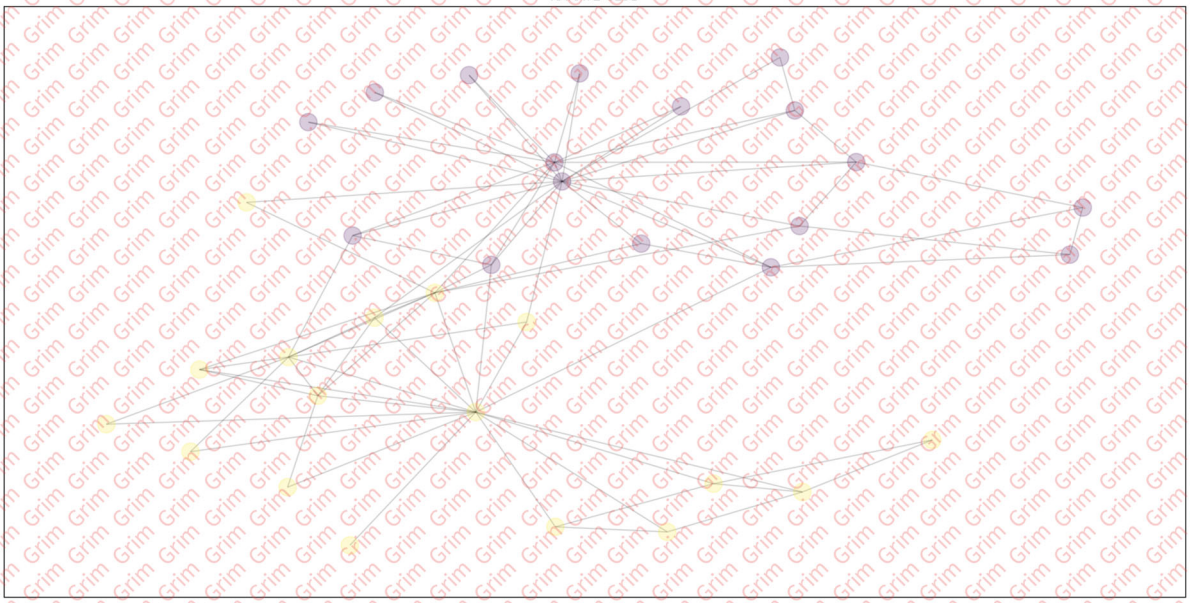
```
    d = np.sum(d_G)
    GEI = a/(a+d)
    return GEI

def graphPlot (path, title):
    GG = nx.read_gml (path, label = 'id')
    community = {}
    openResult = open ('results.txt', 'r')
    readLine = openResult.readlines ()
    ii = 0
    for line in readLine:
        aa = list (map (int, line.split ()))
        for xx in range(0, len(aa)):
            aa [xx] = aa [xx] + 1
        community [ii] = aa
        ii = ii + 1
    comDict = defaultdict (lambda: 0)
    comColour = dict ()
    for ii, com in community.items ():
        comColour |= {node: ii + 10 for node in com}
        for node in com:
            comDict [node] = comDict [node] + 1
    pos = nx.spring_layout (GG, k = 0.2, seed = 4572321)
    overlappedNodes = {node for node, n_comm in comDict.items() if n_comm > 1}
    nodeColour = [0 if nn in overlappedNodes else comColour [nn] for nn in GG]
    options = {
        "pos" : pos,
        "with_labels" : False,
        "node_color" : nodeColour,
        "node_size" : 250,
        "alpha" : 0.2
    }
    plt.figure (figsize = (15, 15))
    plt.title (title)
    nx.draw_networkx (GG, **options)
    plt.show ()

# KARATE CLUB
print ('\nKARATE CLUB')
karate_Graph = nx.karate_club_graph ()
nx.write_edgelist (karate_Graph, "karateedgelist.txt", delimiter = ',')
nx.write_gml (karate_Graph, "karate.gml")
f = 'karateedgelist.txt'
WLC (f, ',')
print ("GEI Value = ", GEI_calc (karate_Graph, 'club'))
graphPlot ('karate.gml', 'KARATE CLUB')
```

```
# LFR BENCHMARK
print ('\n\n\nLFR BENCHMARK')
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_edgelist (LFR_Graph, "lfredgelist.txt", delimiter = ',')
nx.write_gml (LFR_Graph, "LFR_Overlapping.gml")
f = 'lfredgelist.txt'
WLC (f, ',')
print ("GEI Value = ", GEI_calc (LFR_Graph, 'community'))
graphPlot ('LFR_Overlapping.gml', 'LFR BENCHMARK')
```

OUTPUT AND OBSERVATIONS:

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

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 Materials\Github\lab7.py"

KARATE CLUB
GEI Value =  0.3026315789473684


LFR BENCHMARK
GEI Value =  0.09017298490982702
PS D:\Documents\NITS\Semester VI\(LAB) CS321 SNA>
```

KARATE CLUB



LFR BENCHMARK