经济管理学院

课　程　报　告

（复杂网络与社会计算）

题　　目：week6 课程作业

课程教师：赵吉昌

学院/专业：信息管理与信息系统

学生姓名：范春

学　　号：21377061

2024 年 4 月 2 日

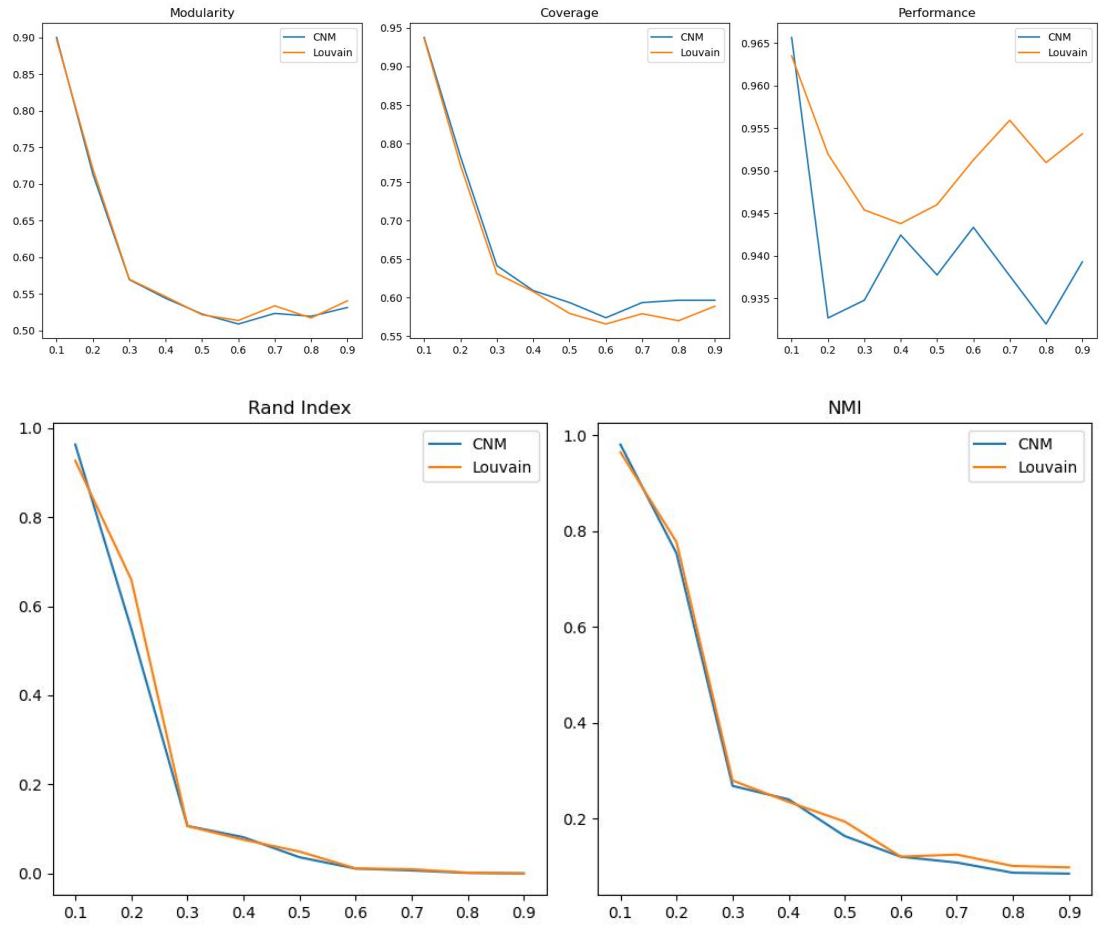
北京航空航天大學
BEIHANG UNIVERSITY

作业要求如下：

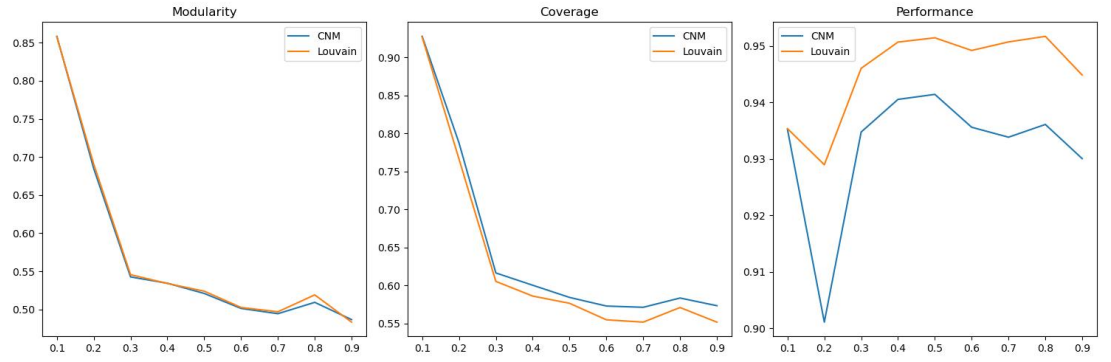
2. 参照1中的实验思路，利用LFR benchmark，生成一定规模的网络，并通过调整μ来测试不同社团发现算法性能的变化。具体地，发现算法应至少包括CNM和Louvain，评价指标则至少包括模块度，Coverage, Performance，Rand index和NMI。

3. 复杂网络分析工具Gephi支持基于社团的可视化。请利用其将2中某个μ下的发现结果进行可视化。

```
nx.LFR_benchmark_graph(n, tau1, tau2, mu, average_degree=5,min_community=20,max_community=100)
```

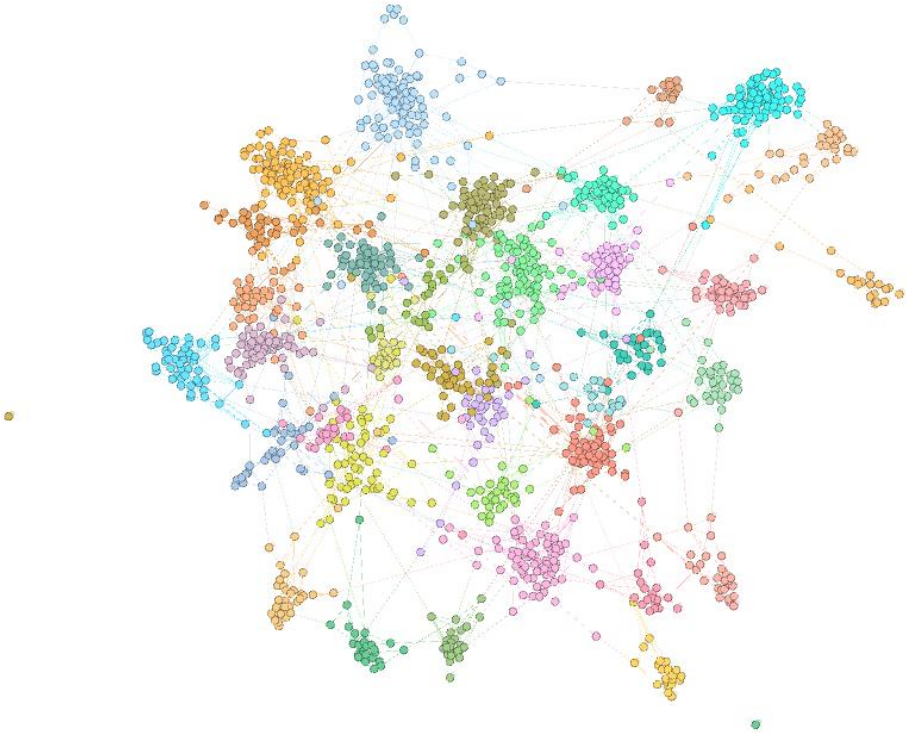



```
nx.LFR_benchmark_graph(n, tau1, tau2, mu, average_degree=5,min_community=50,max_community=200)
```

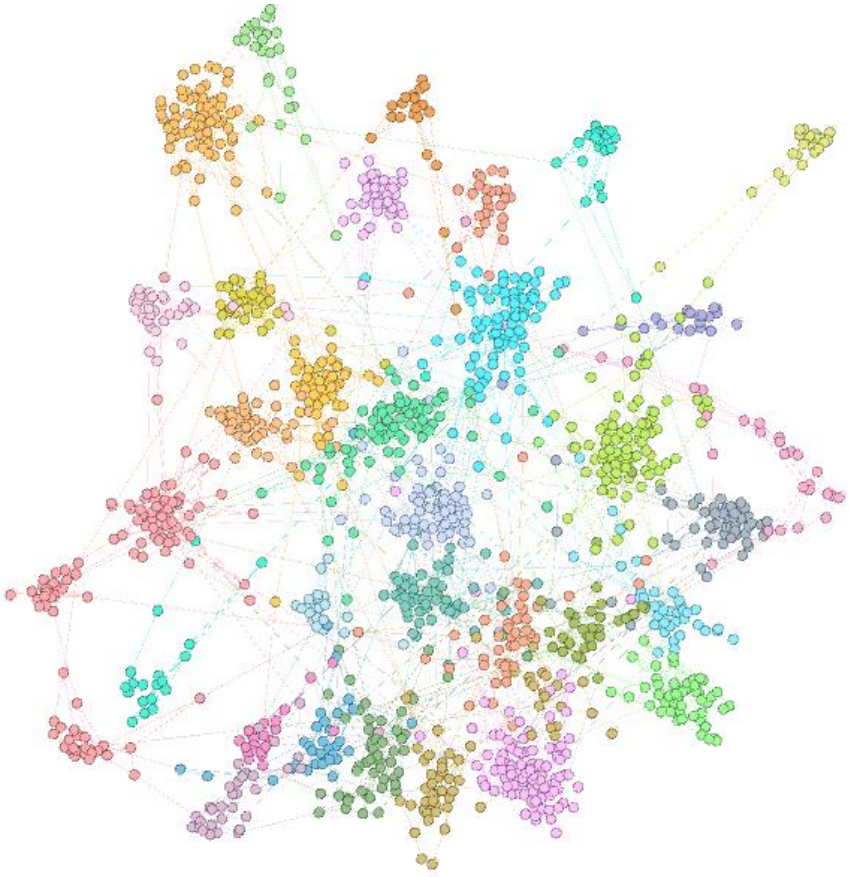


当参数 `min_community` 和 `max_community` 不同时，评价指标 performance 的变化趋势不同。

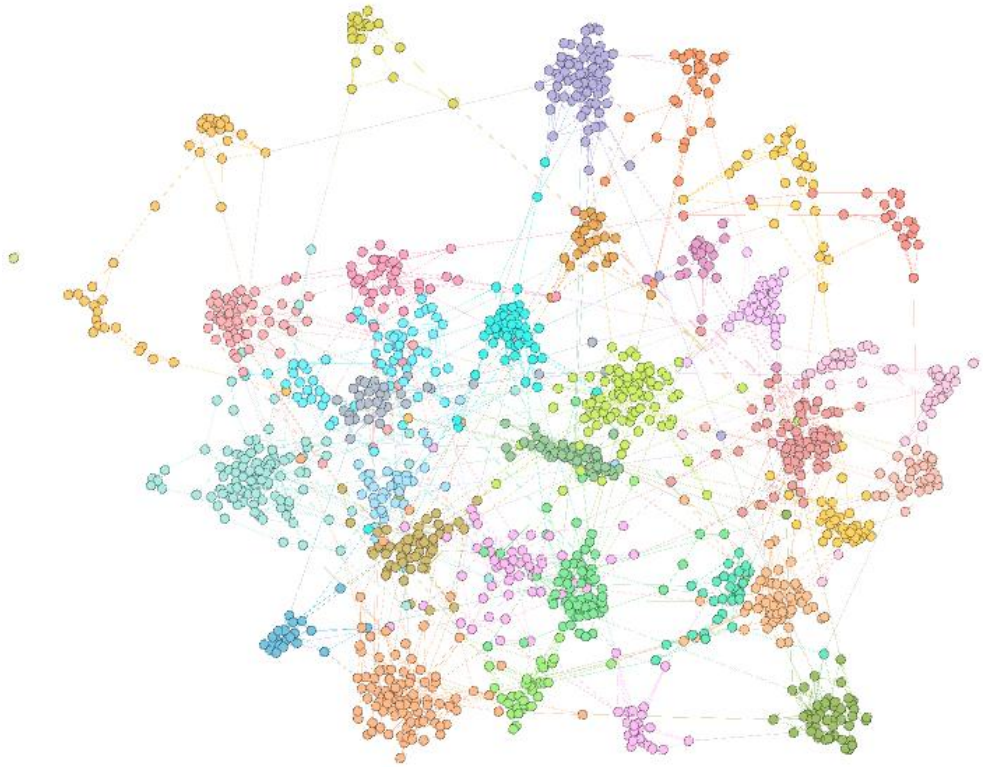
原网络（μ=0.1）：35 个社团



CNM（μ=0.1）：35 个社团



Louvain（μ=0.1）：33 个社团



完整代码如下：

```
import networkx as nx
import community as community_louvain
from sklearn.metrics import normalized_mutual_info_score, adjusted_rand_score
import pandas as pd
from networkx.readwrite import gexf

#生成一定规模的网络
def generateG(n, tau1, tau2, mu):
    return nx.LFR_benchmark_graph(n, tau1, tau2, mu,
average_degree=5,min_community=10,max_community=50)

def CNM(G):
    partition = nx.community.greedy_modularity_communities(G)
    communities_list = [set(comm) for comm in partition]
    return communities_list

def Louvain(G):
    partition = nx.community.louvain_communities(G)
    return partition

def evaluationIndicators(G, communities):
    modularity = nx.community.modularity(G, communities)
```

```
    coverage, performance = nx.community.partition_quality(G, communities)

    labels_true = [None]*len(G)
    i=0
    for node, data in G.nodes(data=True):
        node_communities = data['community']
        for item in node_communities:
            labels_true[item] = i
        i+=1

    labels_pred = [None] * len(G)
    for idx, community in enumerate(communities):
        for node in community:
            labels_pred[node] = idx

    rand_index = adjusted_rand_score(labels_true,labels_pred)

    nmi = normalized_mutual_info_score(labels_true,labels_pred)
    return modularity, coverage, performance, rand_index, nmi

def add_communities_to_graph(G, communities):
    for community_id, community in enumerate(communities):
        for node in community:
            G.nodes[node]['community'] = community_id

def main():
    n = 1000
    tau1 = 3
    tau2 = 1.5
    # 初始化列表来保存结果
    results = []

    for i in range(1,10):
        mu = 0.1*i
        G = generateG(n, tau1, tau2, mu)
        communitiesCNM = CNM(G)
        communitiesLouvain = Louvain(G)

        modularity1, coverage1, performance1, rand_index1, nmi1 = evaluationIndicators(G,
communitiesCNM)
        modularity2, coverage2, performance2, rand_index2, nmi2 = evaluationIndicators(G,
communitiesLouvain)

        for node, data in G.nodes(data=True):
```

```
            if 'community' in data:
                data['community'] = ','.join(str(community) for community in data['community'])
        gexf_path = f'C:\\Users\\范春\\Desktop\\week6\\{mu:.2f}.gexf'
        nx.write_gexf(G,gexf_path)

        G1 = G.copy()
        add_communities_to_graph(G1, communitiesCNM)
        gexf_path = f'C:\\Users\\范春\\Desktop\\week6\\CNM_{mu:.2f}.gexf'
        nx.write_gexf(G1, gexf_path)

        G2 = G.copy()
        add_communities_to_graph(G2, communitiesLouvain)
        gexf_path = f'C:\\Users\\范春\\Desktop\\week6\\Louvain_{mu:.2f}.gexf'
        nx.write_gexf(G2, gexf_path)

         result_row = {
            'modularity1': modularity1,
            'coverage1': coverage1,
            'performance1': performance1,
            'rand_index1': rand_index1,
            'nmi1': nmi1,
            'modularity2': modularity2,
            'coverage2': coverage2,
            'performance2': performance2,
            'rand_index2': rand_index2,
            'nmi2': nmi2,
            'mu': mu
        }
        results.append(result_row)

    df = pd.DataFrame(results)

    column_order = ['mu'] + ['modularity1', 'coverage1', 'performance1', 'rand_index1', 'nmi1']
+ ['modularity2', 'coverage2', 'performance2', 'rand_index2', 'nmi2']
    df = df[column_order]

    excel_file_path = "C:\\Users\\范春\\Desktop\\week6\\result.xlsx"
    df.to_excel(excel_file_path, index=False)
    print(f"Results have been saved to {excel_file_path}.")

if __name__=="__main__":
    main()
```