经济管理学院

课　程　报　告

（复杂网络与社会计算）

题　　目：社团结构与社团发现

课程教师：赵吉昌

学院/专业：信息管理与信息系统

学生姓名：郭加璐

学　　号：21377225

作业内容：

1. 阅读本周参考资料。

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。

功能实现及分析：

1. LFR benchmark 生成网络

利用 LFR benchmark 生成网络，网络规模为 1000 个节点，μ的取值为 0.1-0.5（以 0.1 为步长），分别使用 CNM 和 Louvain 发现算法，通过模块度 modularity、Coverage、Performance、Rand index 和 NMI 等评价指标来测试算法性能随μ的变化。

```
def LFR_benchmark_graph():
    N = 1000  #节点数
    for mu in np.arange(0.1,0.6,0.1):
        #LFR benchmark 生成网络
        G = nx.LFR_benchmark_graph(n=N, tau1=3.0, tau2=1.5, mu=mu,
                                   average_degree=5, min_degree=None,
                                  max_degree=None,
                                   min_community=20, max_community=30,
                                   tol=1e-07, max_iters=500, seed=42)
        communities = {node: data["community"] for node, data in
                       G.nodes(data=True)}
        true_partitions = np.zeros(N)
        for node, comms in communities.items():
            true_partitions[node] = list(comms)[0]
        #CNM 算法
        cnm_communities = list(greedy_modularity_communities(G))
        cnm_partitions = {node: i for i, com in enumerate(cnm_communities)
                          for node in com}
        #Louvain 算法
        louvain_partitions = community_louvain.best_partition(G)
```

```
        louvain_communities = {}
        for node, community_id in louvain_partitions.items():
            louvain_communities.setdefault(community_id, []).append(node)
        louvain_communities = list(louvain_communities.values())
        #评价 CNM 算法性能
        cnm_results = evaluate_performance(G, cnm_communities,
                                           cnm_partitions)
        print("CNM Evaluation: Modularity: {}, Coverage: {}, Performance:
              {}, Rand Index: {}, NMI: {}".format(*cnm_results))
        #评价 Louvain 算法性能
        louvain_results = evaluate_performance(G, louvain_communities,
                                               louvain_partitions)
        print("Louvain Evaluation: Modularity: {}, Coverage: {},
              Performance: {}, Rand Index: {}, NMI:
              {}".format(*louvain_results))
        #保存结果
        save_result(mu, cnm_results, louvain_results)
        G_with_community = add_community_to_graph(G.copy(),
                                                  cnm_partitions)
        save_graph(G_with_community, mu, "cnm")
        G_with_community = add_community_to_graph(G.copy(),
                                                  louvain_partitions)
        save_graph(G_with_community, mu, "louvain")
```

2. 社团发现算法性能测试

计算评价指标：模块度 modularity、Coverage、Performance、Rand index 和 NMI。

```
def evaluate_performance(G, true_partitions, detected_partitions):
    #模块度
    modularity = community_louvain.modularity(detected_partitions, G)
    #Coverage and Performance
    coverage, performance = nx.algorithms.community.quality.
                            partition_quality(G, true_partitions)
    #Rand index
    node_to_community = {node: idx for idx, community in
                         enumerate(true_partitions) for node in community}
    true_labels = [node_to_community[node] for node in G.nodes()]
    predicted_labels = [G.nodes[node]['community'] for node in G.nodes()]
    rand_index = adjusted_rand_score(true_labels, predicted_labels)
    #NMI
    nmi = normalized_mutual_info_score(true_labels, predicted_labels)
    #返回评价指标
    return modularity, coverage, performance, rand_index, nmi
```

将计算结果保存到本地。

```
def save_result(mu, cnm_results, louvain_results):
    with open(f'./results_mu_{mu:.2f}.txt', 'w') as f:
        f.write(f"mu: {mu}\n")
        f.write("CNM Evaluation:\n")
        f.write(f"Modularity: {cnm_results[0]}, Coverage:
              {cnm_results[1]}, Performance: {cnm_results[2]}, Rand Index:
              {cnm_results[3]}, NMI: {cnm_results[4]}\n")
        f.write("Louvain Evaluation:\n")
        f.write(f"Modularity: {louvain_results[0]}, Coverage:
              {louvain_results[1]}, Performance: {louvain_results[2]},
              Rand Index: {louvain_results[3]}, NMI:
              {louvain_results[4]}\n")
```

控制台输出结果：

```
PS D:\大学\复杂网络与社会计算\data\week6\Community-Generate-and-Community-Structure> d:; cd 'd:\大学\复杂网络与社会计算\data\week6\Community-Generate-and-Community-S
orch1\python.exe' 'c:\Users\rita\.vscode\extensions\ms-python.debugpy-2024.4.0-win32-x64\bundled\libs\debugpy\adapter/../..\debugpy\launcher' '39252' '--' 'D:\大学\复
ty-Generate-and-Community-Structure\community_generate.py'
CNM Evaluation: Modularity: 0.9105604342393936, Coverage: 0.936919315403423, Performance: 0.978638638638386, Rand Index: 1.0, NMI: 1.0
Louvain Evaluation: Modularity: 0.9075280516017957, Coverage: 0.9364303178484108, Performance: 0.9762982982982983, Rand Index: 1.0, NMI: 1.0
d:\anaconda3\envs\pytorch1\Lib\site-packages\networkx\readwrite\pajek.py:75: UserWarning: Node attribute community is not processed. Non-string attribute.
  warnings.warn(
CNM Evaluation: Modularity: 0.7574398847406086, Coverage: 0.8047595920349684, Performance: 0.9577837837837838, Rand Index: 1.0, NMI: 1.0
Louvain Evaluation: Modularity: 0.7562812523514083, Coverage: 0.7969888295288975, Performance: 0.9633793793793793, Rand Index: 1.0, NMI: 1.0
d:\anaconda3\envs\pytorch1\Lib\site-packages\networkx\readwrite\pajek.py:75: UserWarning: Node attribute community is not processed. Non-string attribute.
  warnings.warn(
CNM Evaluation: Modularity: 0.5969943815495586, Coverage: 0.6677949709864603, Performance: 0.9374134134134134, Rand Index: 1.0, NMI: 1.0
Louvain Evaluation: Modularity: 0.6087935249860638, Coverage: 0.6561895551257253, Performance: 0.9553953953953954, Rand Index: 1.0, NMI: 1.0
d:\anaconda3\envs\pytorch1\Lib\site-packages\networkx\readwrite\pajek.py:75: UserWarning: Node attribute community is not processed. Non-string attribute.
  warnings.warn(
CNM Evaluation: Modularity: 0.5599129690974097, Coverage: 0.6303294573643411, Performance: 0.9348248248248249, Rand Index: 1.0, NMI: 1.0
Louvain Evaluation: Modularity: 0.5630949396445525, Coverage: 0.627906976744186, Performance: 0.9378278278278278, Rand Index: 1.0, NMI: 0.9999999999999999
d:\anaconda3\envs\pytorch1\Lib\site-packages\networkx\readwrite\pajek.py:75: UserWarning: Node attribute community is not processed. Non-string attribute.
  warnings.warn(
```

具体计算结果如下：

μ	Evaluation	Modularity	Coverage	Performance	Rand Index	NMI
0.1	CNM	0.910560434	0.936919315	0.978638639	0.006087276	0.698623425
	Louvain	0.908097632	0.935452323	0.977791792	0.005122098	0.682142676
0.2	CNM	0.757439885	0.804759592	0.957783784	0.002712322	0.630925479
	Louvain	0.760441313	0.801359883	0.961647648	0.0032781	0.642828376
0.3	CNM	0.596994382	0.667794971	0.937413413	0.000434767	0.579020589
	Louvain	0.599251887	0.650870406	0.950620621	0.000424989	0.602683591
0.4	CNM	0.559912969	0.630329457	0.934824825	5.31E-05	0.571417871
	Louvain	0.56151751	0.620155039	0.943189189	0.000356405	0.588340132
0.5	CNM	0.541230312	0.611781748	0.937905906	0.000125898	0.580681427
	Louvain	0.545305349	0.60791888	0.942924925	-5.90E-05	0.584628839

模块度可以衡量社团内部连接的紧密程度，是评价社团划分质量的指标，模块度越大说明社团结构越好。在μ=0.1 时，CNM 算法和 Louvain 算法的模块度均大于 0.9，说明两种算法都能识别出社团结构。随着μ的增加，模块度呈现下降趋势，当μ=0.5 时，两种算法的模块度均在 0.54 附近，社团结构划分逐渐模糊。

Coverage 是社团内部的边占所有边的比例。随着μ的增加，Coverage 下降，表明社区内部节点之间的联系减弱了。

归一化互信息 NMI 在μ较低时较高，随着μ的增加，NMI 有所下降，但是仍能维持在 0.6 左右，这表示尽管社区边界变得模糊，但是算法检测到的社区结构与实际的社区结构有一定程度的相似性。

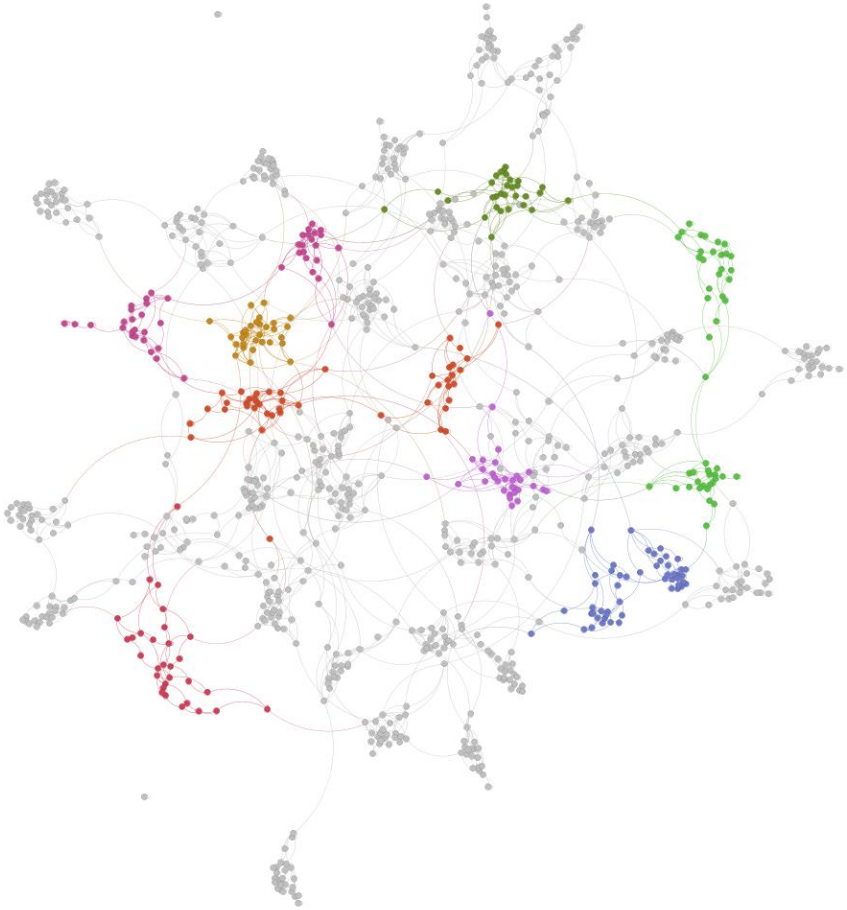
3. 社团可视化---Gephi

将生成的网络保存为 lfr_network_mu_μ.net 文件，使用 Gephi 进行可视化。

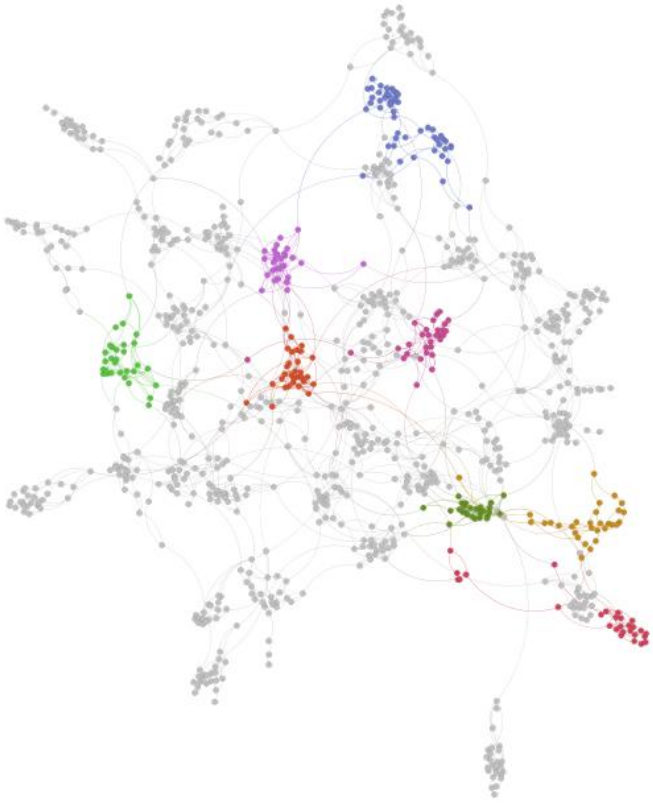
```
def add_community_to_graph(G, partition):
    #添加社团信息
    for node, community_id in partition.items():
        G.nodes[node]['community'] = str(community_id)
    return G

def save_graph(G, mu, evaluation):
    #保存生成网络的.gexf 文件
    nx.write_gexf(G,f"lfr_network_mu_{mu:.2f}_evaluation_{evaluation}.ge
                  xf")
```

对μ=0.1 和μ=0.5 生成的网络进行可视化，结果如下：

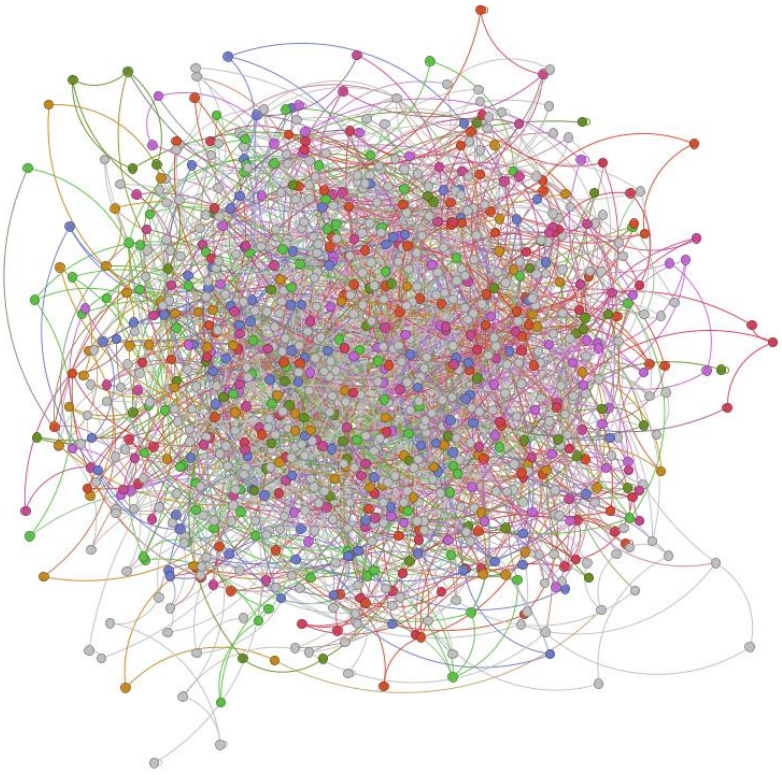
0.1_louvain：



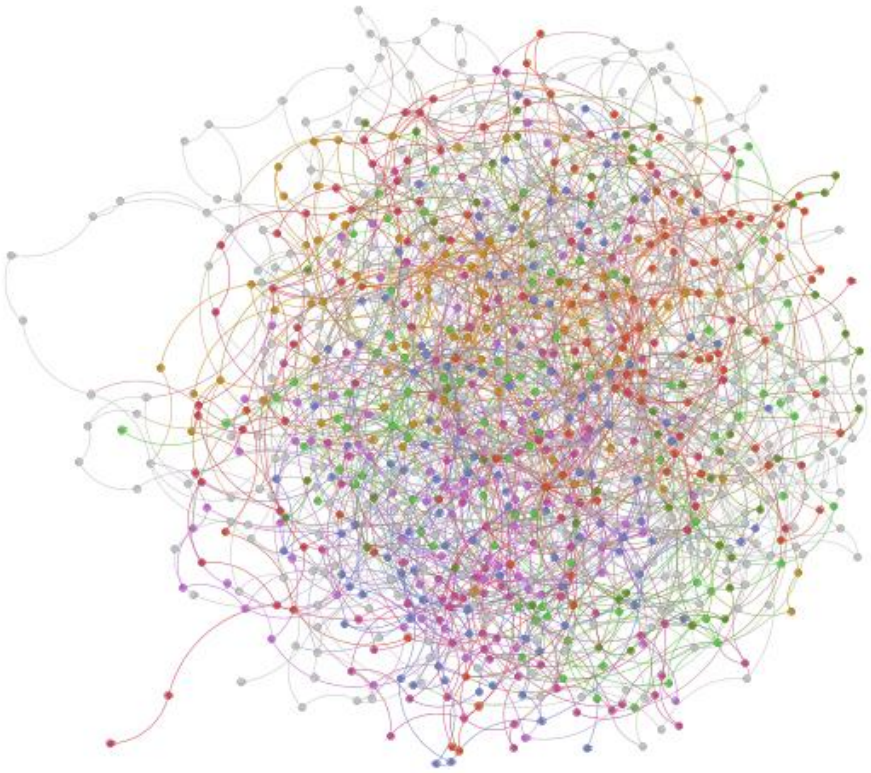
0.1_cnm：



0.5_louvain：



0.5_cnm：



当μ = 0.1 时，在两种算法下，网络中社团结构划分明确，具有紧密的节点簇。此时的模块度大于 0.9，即社区内连接紧密，而社区间连接相对稀疏，可以在图中观察到这一特点。

当μ = 0.5 时，在两种算法下，网络的社团结构都几乎无法辨认，社团内和社团间的连接情况相当，算法难以区分社团结构。

对比μ的两个取值下的可视化结果，可以发现：随着混合参数μ的增长，社团结构由清晰变得模糊，在低μ水平下算法发现社团的效率降低，无法实现良好的社团划分。