Homework 3

专业：大数据001

学号：2184114639

姓名：郅啸淇

日期：2023年5月7日星期日

1. 利用课堂上提供的一个社交网络数据，利用三种以上的社团发掘算法挖掘其社团结构，比较结果的差异（如运行时间，社团数量）。

1.1 说明你使用的社交网络。

使用学生社交网络（课件中anodes和aedges），有65个节点和104条边

1.2 说明你使用的三种以上的社团挖掘算法。

首先是planted I-partition算法，分为4组，每组32个节点，同组任意两个节点连接概率为0.5，不同组任意两个节点连接概率为0.01

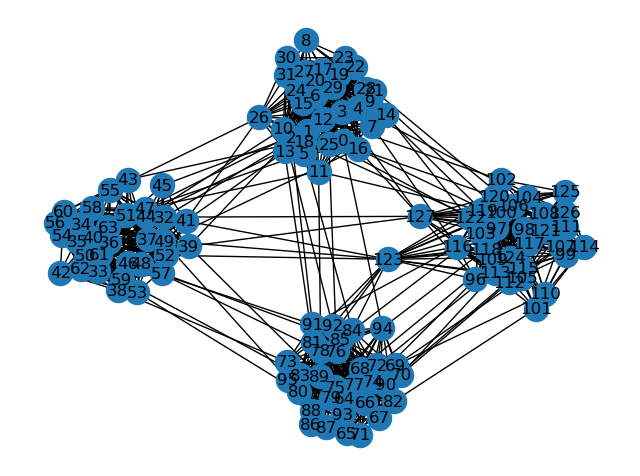
然后是kmeans对邻接矩阵进行聚类

然后是greedy\_modularity\_communities算法

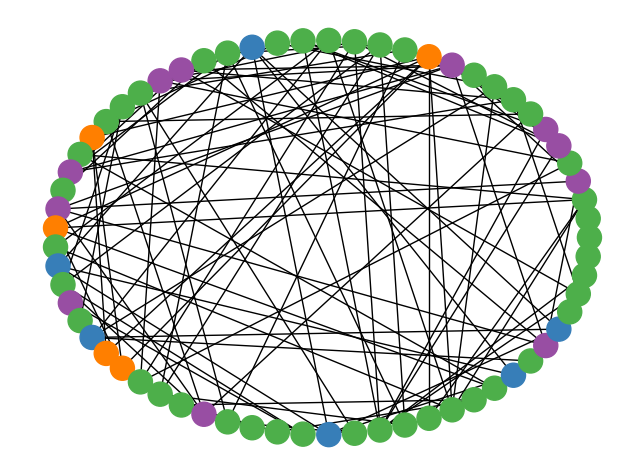
最后是Girvan Newman算法，采用第二步结果。

完整代码

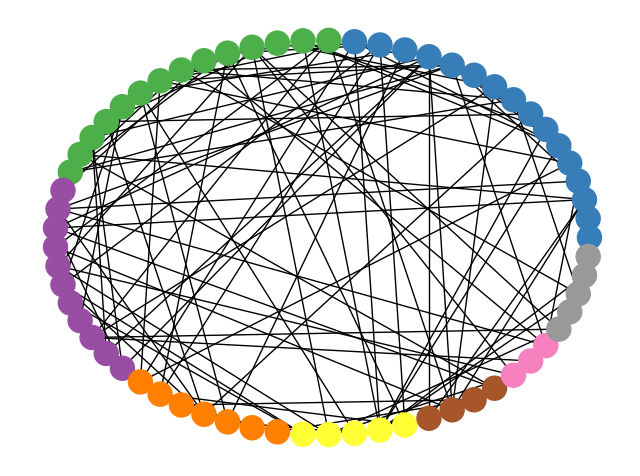
1. **import** networkx as nx
2. **import** matplotlib.pyplot as plt
3. **import** pandas as pd
4. **import** time
5. **from** matplotlib **import** cm
6. **from** sklearn.cluster **import** KMeans
7. **from** networkx.algorithms.community **import** greedy\_modularity\_communities
8. **from** networkx.algorithms.community **import** girvan\_newman
10. pd.set\_option('display.max\_columns', None)
11. pd.set\_option('display.max\_rows', None)
13. # 导入数据
14. data\_nodes = pd.read\_csv("anodes.csv")
15. data\_edges = pd.read\_csv("aedges.csv")
17. # 提取节点和链接
18. nodes = data\_nodes["Id"]
19. source = data\_edges["Source"]
20. target = data\_edges["Target"]
22. # 生成图并插入节点和链接
23. G = nx.Graph()
24. G.add\_nodes\_from(nodes)
25. edges = []
26. **for** i **in** range(len(source)):
27. edges.append((source[i], target[i]))
28. G.add\_edges\_from(edges)
29. pos = nx.circular\_layout(G)
30. # nx.draw(G,pos,with\_labels=True)
32. flag = int(input("1:planted I-partation 2:Kmeans 3:Greedy Modularity Communities 4:Girvan Newman Choose algorithm: "))
33. **if** flag == 1:
34. # planted I-partation graph (Number of groups,Number of vertices in each group,probability of connecting vertices within a group,probability of connected vertices between groups)
35. start\_time = time.time()
36. G\_LP = nx.planted\_partition\_graph(4, 32, 0.5, 0.01, seed=114514)
37. nx.draw(G\_LP, with\_labels=True)
38. end\_time = time.time()
39. **print**(end\_time - start\_time)
40. plt.show()
42. **elif** flag == 2:
43. # Kmeans
44. start\_time = time.time()
45. A = pd.DataFrame(nx.adjacency\_matrix(G).todense())
46. kmeans = KMeans(n\_clusters=4, random\_state=114514)
47. kmeans.fit(A)
48. y = kmeans.predict(A)
49. colors = []
50. **for** i **in** range(len(G.nodes)):
51. colors.append(cm.Set1(y[i] + 1))
52. nx.draw(G, pos=nx.circular\_layout(G), node\_color=colors)
53. end\_time = time.time()
54. **print**(end\_time - start\_time)
55. plt.show()
57. **elif** flag == 3:
58. # Greedy Modularity Communities
59. start\_time = time.time()
60. c = greedy\_modularity\_communities(G)
61. a = dict(enumerate(c))
62. b = [0] \* len(G.nodes)
63. j = 0
64. **for** k, v **in** a.items():
65. **for** v1 **in** range(len(v)):
66. b[j + v1] = k
67. j += len(v)
68. colors = []
69. **for** i **in** range(len(G.nodes)):
70. colors.append(cm.Set1(b[i] + 1))
71. nx.draw(G, pos, node\_color=colors)
72. end\_time = time.time()
73. **print**(end\_time - start\_time)
74. plt.show()
76. **elif** flag == 4:
77. # Girvan Newman
78. start\_time = time.time()
79. comp = girvan\_newman(G)
80. # a = dict(enumerate(next(comp)))
81. # a = dict(enumerate(next(comp)))
82. a = dict(enumerate(next(comp)))
83. a = dict(enumerate(next(comp)))
84. b = [0] \* len(G.nodes)
85. j = 0
86. **for** k, v **in** a.items():
87. **for** v1 **in** range(len(v)):
88. b[j + v1] = k
89. j += len(v)
90. colors = []
91. **for** i **in** range(len(G.nodes)):
92. colors.append(cm.Set1(b[i] + 1))
93. nx.draw(G, pos, node\_color=colors)
94. end\_time = time.time()
95. **print**(end\_time - start\_time)
96. plt.show()



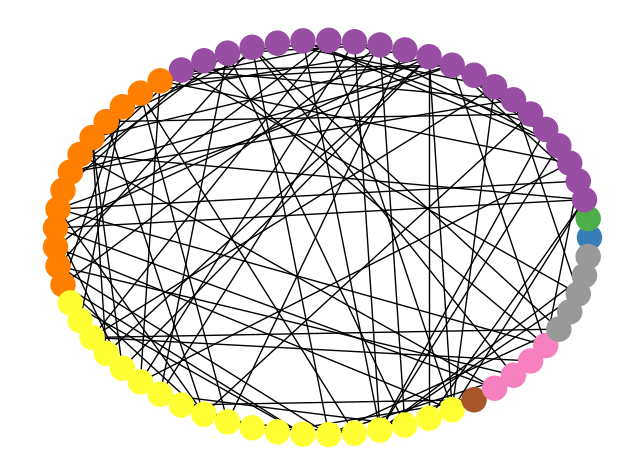
Planted I-partition



Kmeans



Greedy Modularity Communities



Girvan Newman

1.3 三种算法结果上的差异有哪些？

Kmeasn对于社团的识别较差（社团数量较少），其他算法运行结果基本相同

1.4 三种算法运行时间的差异是什么？

planted I-partition: 0.08482861518859863秒

kmeans: 0.11710596084594727秒

greedy\_modularity\_communities: 0.05505037307739258秒

Girvan\_Newman: 0.058053016662597656秒

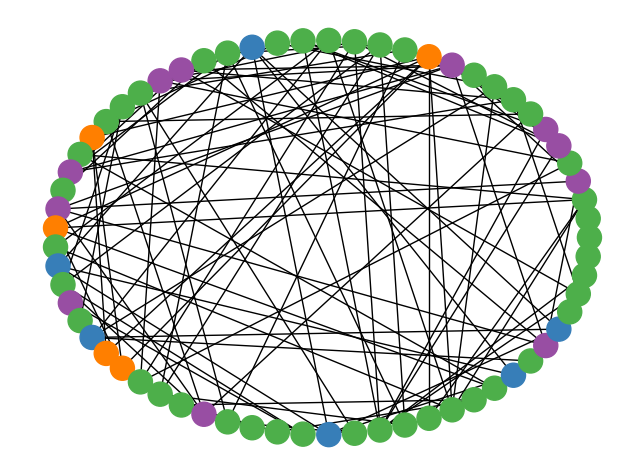
2. 设计一个社团识别算法，用Python代码实现，并和上述三种社团发掘算法做比较。

2.1 说明你设计的社团识别算法的思路。

首先构建节点之间的CN指标矩阵（类似邻接矩阵），然后使用Kmeans进行聚类分析，聚类结果即为识别社团

完整代码

1. **import** networkx as nx
2. **import** matplotlib.pyplot as plt
3. **import** pandas as pd
4. **import** time
5. **from** matplotlib **import** cm
6. **from** sklearn.cluster **import** KMeans
7. **import** numpy as np
8. **from** matplotlib.colors **import** ListedColormap
9. pd.set\_option('display.max\_columns', None)
10. pd.set\_option('display.max\_rows', None)
12. # 导入数据
13. data\_nodes = pd.read\_csv("anodes.csv")
14. data\_edges = pd.read\_csv("aedges.csv")
16. # 提取节点和链接
17. nodes = data\_nodes["Id"]
18. source = data\_edges["Source"]
19. target = data\_edges["Target"]
21. # 生成图并插入节点和链接
22. G = nx.Graph()
23. G.add\_nodes\_from(nodes)
24. edges = []
25. **for** i **in** range(len(source)):
26. edges.append((source[i], target[i]))
27. G.add\_edges\_from(edges)
28. pos = nx.circular\_layout(G)
30. start\_time = time.time()
31. # 构建图并计算CN指标矩阵
32. **def** build\_cn\_matrix(G):
33. nodes = list(G.nodes())
34. n = len(nodes)
35. cn\_matrix = np.zeros((n, n))
36. **for** i, u **in** enumerate(nodes):
37. **for** j, v **in** enumerate(nodes):
38. **if** j > i:
39. common\_neighbors = set(G[u]) & set(G[v])
40. cn\_matrix[i, j] = cn\_matrix[j, i] = len(common\_neighbors)
41. **return** cn\_matrix
43. # 对节点进行聚类
44. # k是簇的数量
45. **def** cluster\_nodes(cn\_matrix, k):
46. kmeans = KMeans(n\_clusters=k)
47. kmeans.fit(cn\_matrix)
48. labels = kmeans.labels\_
49. **return** labels
51. # 进行社团识别
52. cn\_matrix = build\_cn\_matrix(G)
53. labels = cluster\_nodes(cn\_matrix, 3)
54. colors = ['r', 'g', 'b']
55. node\_colors = [colors[label] **for** label **in** labels]
56. cmap = ListedColormap(colors)
57. nx.draw(G, pos, node\_color=node\_colors)
58. end\_time = time.time()
59. **print**(end\_time-start\_time)
60. plt.show()
61. **for** i, node **in** enumerate(G.nodes()):
62. **print**(f"Node '{node}' belongs to cluster {labels[i]}")



Zxq算法运行结果

2.2 你的社团识别算法是哪种类型？基于相似度的计算、基于拓扑的计算、还是基于Q函数的计算、还是其它？

是基于节点相似度CN指标的聚类算法

2.3 你的社团识别算法和第1题中的社团识别算法相比，差别是什么？有什么优势（或劣势）？

区别是使用聚类算法的指标不同，优点是考虑了节点的共同好友数，缺点是CN指标对节点相似度的刻画不够准确，可以引入别的变量进行修正。

Zxq算法运行时间: 0.07306528091430664秒