复杂网络可视化精华案例四【复杂网络的脆弱性分析：节点的随机攻击和蓄意攻击】

原创 单哥的科研日常 单哥的科研日常 2022-05-19 23:59 发表于湖北

收录于合集
#复杂网络建模 55 #python 47

前言

任务需求：这一份案例的目标是完成复杂网络的脆弱性分析，即抗毁性分析，研究网络在节点的随机攻击和蓄意攻击下，其最大连通子图的相对大小S和全局效率E的变化。

任务需求：这一份案例的目标是完成复杂网络的脆弱性分析，即抗毁性分析，研究网络在节点的随机攻击和度值蓄意攻击下，其最大连通子图的相对大小S和全局效率E的变化。如下图所示，左图表示网络在随机攻击和蓄意攻击下，剩余网络的最大连通子图的相对大小S与移除节点比例之间的依赖关系；右图表示网络在随机攻击和蓄意攻击下，剩余网络的全局效率E与移除节点比例之间的依赖关系。

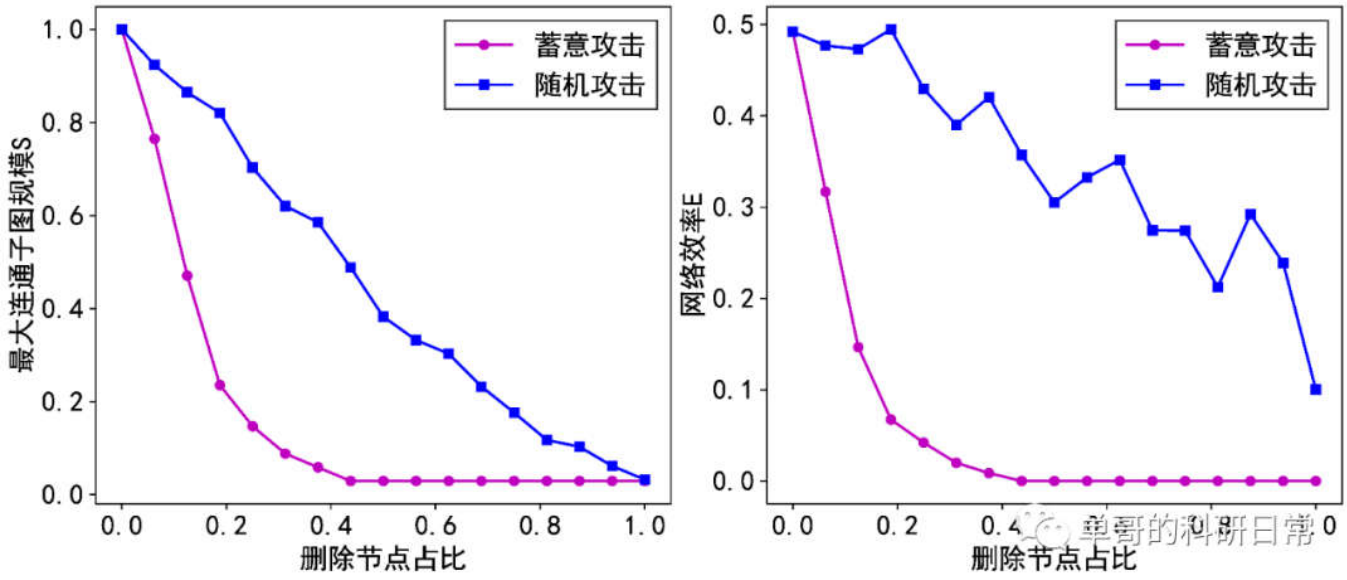
主程序如下

```
#!/usr/bin/python3
# -*- coding: utf-8 -*-
# Author :  单哥的科研日常
# Time : 2022/5/19 0:16

import pandas as pd
import numpy as np
import networkx as nx
import matplotlib.pyplot as plt
import random
plt.rc('font', family='SimHei')


if __name__ == '__main__':
    f_size = 16 # 字体大小
    df = pd.read_csv("edgelist.csv")
    G = nx.from_pandas_edgelist(df, 'source', 'target', create_using = nx.Graph())
    n = len(G.nodes())

    step = 2 # 每次移除两个节点，对于大规模网络，可以适当增加step的值
```

```
if n%step==0:
    nums = int(n/step)
else:
    nums = int(n/step) + 1

# print(nums)
# 失效节点比例
q = np.linspace(0, n, nums)/n
# print(q)


# 蓄意攻击节点：按度的大小删除节点
S_G1 = np.zeros(nums) # 最大连通子图的相对大小
E_G1 = np.zeros(nums) #最大连通子图的相对效率
rn1 = 0
c1 = 0
S_G1[0] = 1.0
eff0 = cal_eff(G)
E_G1[0] = eff0
while True:
    G1 = G.copy()
    rn1 = rn1 + step
    c1 = c1 + 1
    if c1 == nums:
        break
    IA_G1 = intentional_attack_degree(G1, rn1)
    Gcc1 = sorted(nx.connected_components(IA_G1), key=len, reverse=True)
    # 得到图IA_G1的最大连通组件
    largest_cc1 = IA_G1.subgraph(Gcc1[0])
    n_lcc1 = len(largest_cc1.nodes)
    E_G1[c1] = cal_eff(IA_G1)
    # 计算去点后最大连通集团占网络总节点的比例
    S_G1[c1] = n_lcc1/n

print("===============")


# 随机删除节点：随机删除节点结果会有波动，因此可以设置统计平均
samples = 10 # 设置统计平均次数
S2 = np.zeros(nums)
E2 = np.zeros(nums)
for i in range(samples):
    Smax2 = np.zeros(nums)
    E_G2 = np.zeros(nums)
    rn2 = 0
    c2 = 0
    Smax2[0] = 1.0 # 如果初始网络是连通网络
    E_G2[0] = eff0
    while True:
        G2 = G.copy()
        rn2 = rn2 + step
        c2 = c2 + 1
        if c2 == nums:
            break
        IA_G2 = random_attack(G2, rn2)
        Gcc2 = sorted(nx.connected_components(IA_G2), key=len, reverse=True)
        #得到图IA_G2的最大连通组件
        largest_cc2 = IA_G2.subgraph(Gcc2[0])
```

```
            n_lcc2 = len(largest_cc2.nodes)
            E_G2[c2] = cal_eff(IA_G2)
            # 计算去点后最大连通集团占网络总节点的比例
            Smax2[c2] = n_lcc2/n
        S2 = S2 + Smax2
        E2 = E2 + E_G2

S_G2 = S2/samples
E_G2 = E2/samples


plt.figure(figsize=(12,4.8))
plt.subplot(121)
plt.plot(q, S_G1, "mo-", markersize=5, label=u'蓄意攻击')
plt.plot(q, S_G2, "bs-", markersize=5, label=u'随机攻击')
plt.legend(loc=1, edgecolor = "black", fontsize=f_size, fancybox=False)
plt.xlabel(u"删除节点占比", fontsize=f_size)
plt.ylabel(u"最大连通子图规模S", fontsize=f_size)
plt.tick_params(labelsize=f_size)


plt.subplot(122)
plt.plot(q, E_G1, "mo-", markersize=5, label=u'蓄意攻击')
plt.plot(q, E_G2, "bs-", markersize=5, label=u'随机攻击')
plt.legend(loc=1, edgecolor = "black", fontsize=f_size, fancybox=False)
plt.xlabel(u"删除节点占比", fontsize=f_size)
plt.ylabel(u"网络效率E", fontsize=f_size)
plt.tick_params(labelsize=f_size)

plt.tight_layout()
plt.show()
```

绘制的结果图如下：



其中子函数`random_attack()`和`intentional_attack_degree()`如下：

```
# 计算全局效率
def cal_eff(G):
    nodes = list(G.nodes())
```

```
    n = len(nodes)
    eff_list = [1./nx.shortest_path_length(G, nodes[i],nodes[j]) for i in range(n-1)
                for j in range(i+1,n) if nx.has_path(G, nodes[i],nodes[j])]
    av_eff = sum(eff_list)/(n*(n-1)/2)
    return av_eff

# 随机攻击：随机移除节点
def random_attack(G, rn):
    q = 0
    while q < rn:
        random_node = random.choice(list(G.nodes))
        if random_node in G.nodes():
            G.remove_node(random_node)
            q = q + 1

        if not G.nodes():
            break

    return G

# 蓄意攻击：按度的大小移除点
def intentional_attack_degree(G, rn):
    q = 0
    while q < rn:
        deg = dict(G.degree())  #获取键为节点序号，值为节点度的字典
        maxd_key = max(deg, key = deg.get) #获取加权度值最大的键，即节点序号
        if maxd_key in G.nodes():
            G.remove_node(maxd_key)
            q = q + 1

        if not G.nodes():
            break

    return G
```