经济管理学院

课　程　报　告

（复杂网络与社会计算）

题　　目：week10 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 5 月 2 日

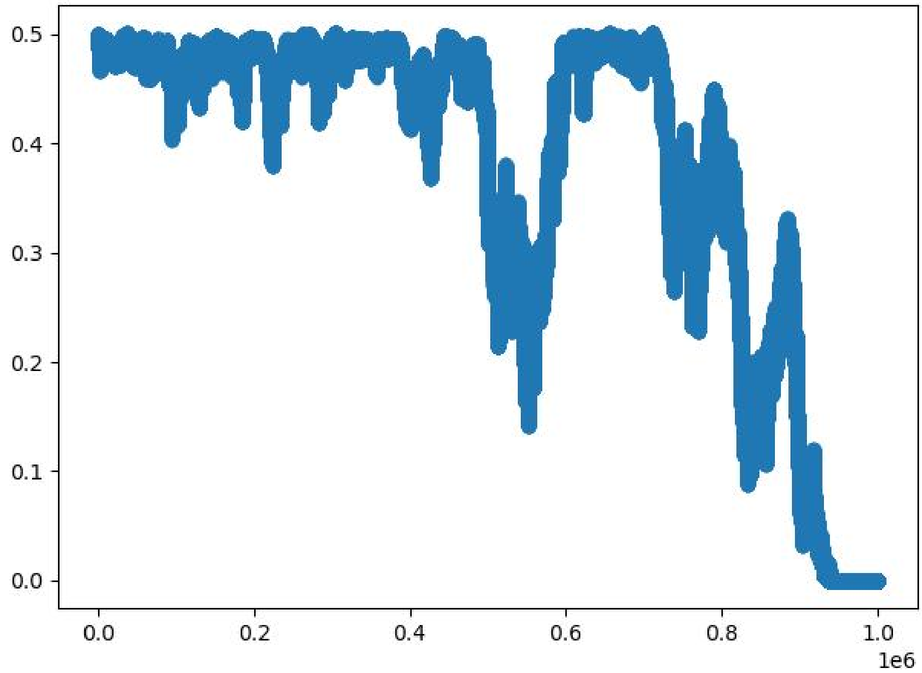
北京航空航天大學
BEIHANG UNIVERSITY

作业要求如下：

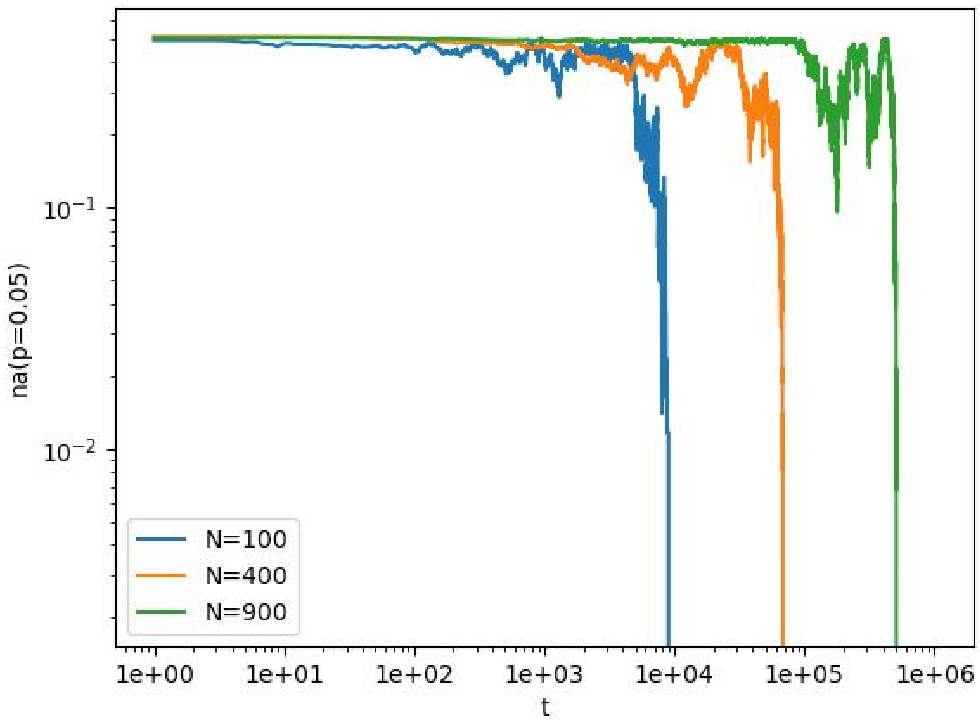
2. 根据既有研究，尝试在$d=2$的格子网络上随机增加（或随机重连）比例为p的长程边，观察Voter模型的$n_a(t)$随时间的变化是否会出现一段相对平坦的区间（称为plateau），此时系统处于亚稳态。请通过变化N和p来观察plateau长度的变化规律（亦即，亚稳态的持续时长）。另外，考虑到$d=2$的格子非常容易可视化(每个节点可以分配平面坐标)，请对随机的一次仿真（N建议大一些，p可以变化），观察不同p时，处于plateau时期的观点分布成什么形态（不同观点的节点有不同的颜色标记，可以不绘制边）

3. （附加）保存某次仿真的每一步可视化图片，进而合成为视频，可以观察voter模型下观点的连续演化过程

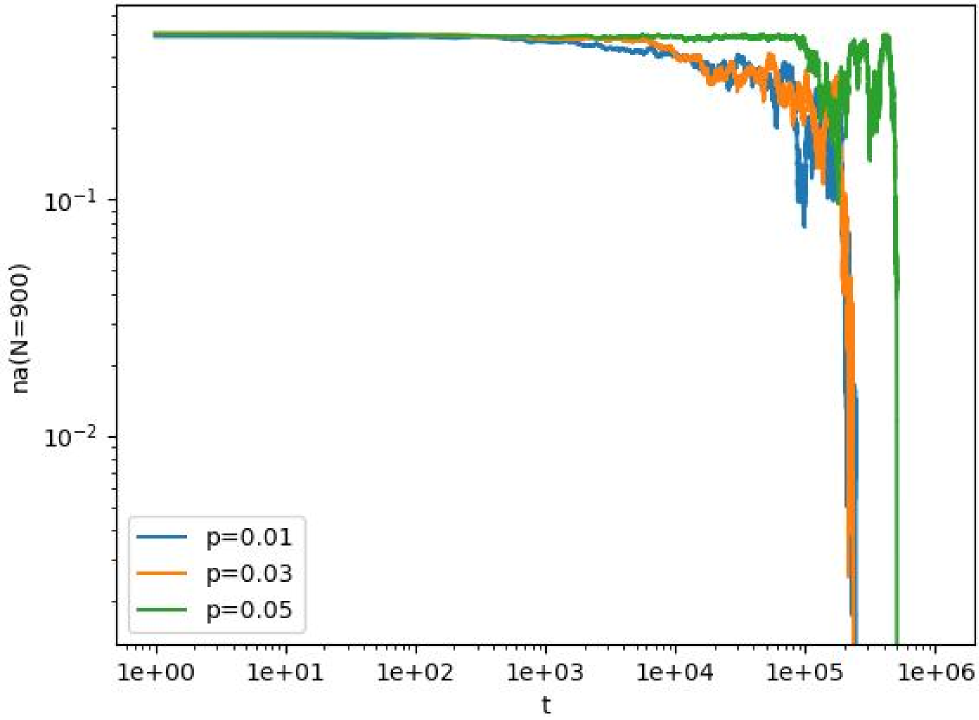
1、下图为 900 个节点，p=0.05 时$n_a t$随时间的变化，可以发现确实存在了一段相对平坦的区间。



2、下图为 p 不变，N 变化时的仿真结果。可以得出：N 越大，plateau 长度越长。

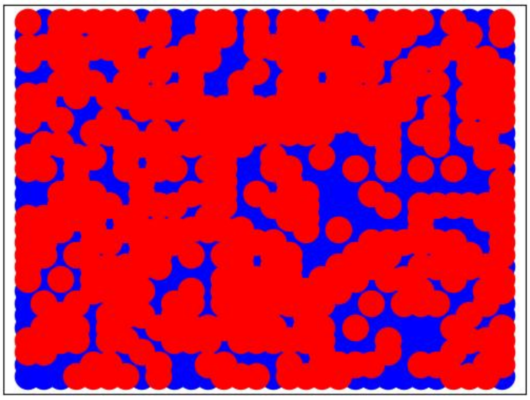
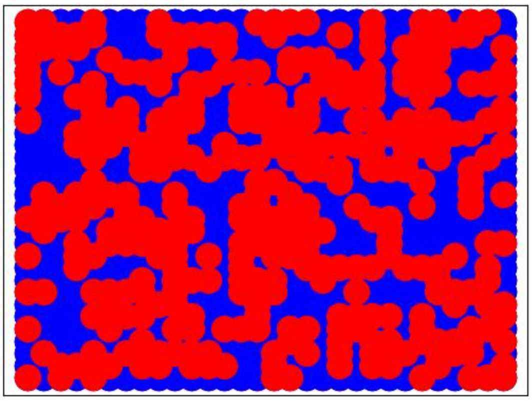


3、下图为 N 不变，p 变化时的仿真结果。

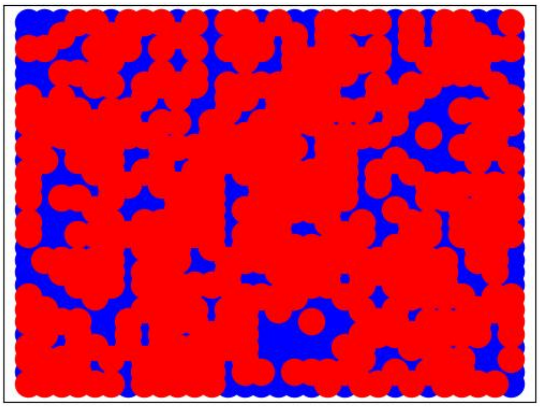
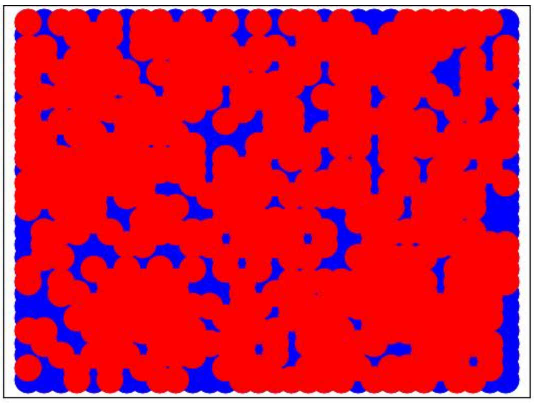


4、plateau 时期观点分布形态（N=900）

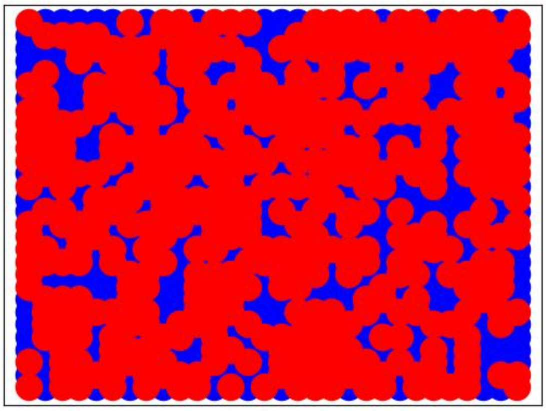
P=0.01

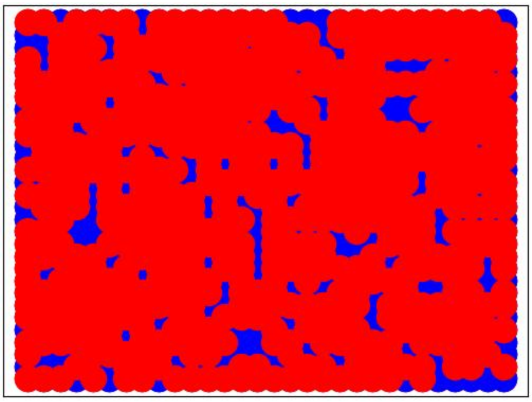



P=0.03

P=0.05





关键代码如下：

```
import networkx as nx
import numpy as np
import random
import pandas as pd
import matplotlib.pyplot as plt

# 创建二维格子网络并添加长程边
def create_grid_with_long_range_edges(N, p):
    # 创建 N*N 的二维格子网络
    G = nx.grid_2d_graph(N, N, periodic=False)

    # 添加长程边
    n = N*N
    num_long_range_edges = int(p * (n*(n-1)/2))
    existing_edges = set(G.edges())
    nodes = list(G.nodes())

    while num_long_range_edges > 0:
        # 随机选择两个不同的节点
        node1, node2 = random.sample(nodes, 2)
        # 如果这对节点之间没有边并且不是自环，则添加边
        if (node1, node2) not in existing_edges and node1 != node2:
            G.add_edge(node1, node2)
            existing_edges.add((node1, node2))
            num_long_range_edges -= 1
    states = np.array([1] * (n // 2) + [-1] * (n // 2))
    np.random.shuffle(states)
    state_dict = dict(zip(nodes, states))
    nx.set_node_attributes(G, state_dict, 'state')
    return G, existing_edges
# Voter 模型模拟函数
def voter_model_simulation(G, T, existing_edges):
```

```
    results = []
    nodes = list(G.nodes())
    # 获取节点位置
    pos = {node: (node[0], node[1]) for node in G.nodes()}
    for i in range(T):
        #可视化
        visualize_opinion_distribution(G, pos, i)
        # 随机选择一个节点
        node = random.sample(nodes, 1)[0]
        # 随机选择一个邻居
        neighbors = list(G.neighbors(node))
        if neighbors:
            neighbor = random.sample(neighbors, 1)[0]
            # 更新节点状态
            G.nodes[node]['state'] = G.nodes[neighbor]['state']
            results.append(calculate_na(G, existing_edges))

    return results
def calculate_na(G, existing_edges):
    result = 0
    for edges in existing_edges:
        node1 = edges[0]
        node2 = edges[1]
        if G.nodes[node1]['state'] != G.nodes[node2]['state']:
            result+=1

    return result/len(existing_edges)
def run(N, p):
    G, existing_edges = create_grid_with_long_range_edges(N, p)
    results = voter_model_simulation(G,1000000,existing_edges)
    return results
def visualize_opinion_distribution(G, pos, t):
    # 根据节点状态分组
    positive_nodes = [node for node, data in G.nodes(data=True) if data['state'] == 1]
    negative_nodes = [node for node, data in G.nodes(data=True) if data['state'] == -1]
    # 绘制正观点的节点
    nx.draw_networkx_nodes(G, pos, nodelist=positive_nodes, node_color='blue', label='Positive 
Opinion')
    # 绘制负观点的节点
    nx.draw_networkx_nodes(G, pos, nodelist=negative_nodes, node_color='red', label='Negative 
Opinion')
    figPath = f"C:\\Users\\范春\\Desktop\\week10\\img\\{t}.png"
    plt.savefig(figPath)
```