经济管理学院

课　程　报　告

（复杂网络与社会计算）

题　　目：　week9 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

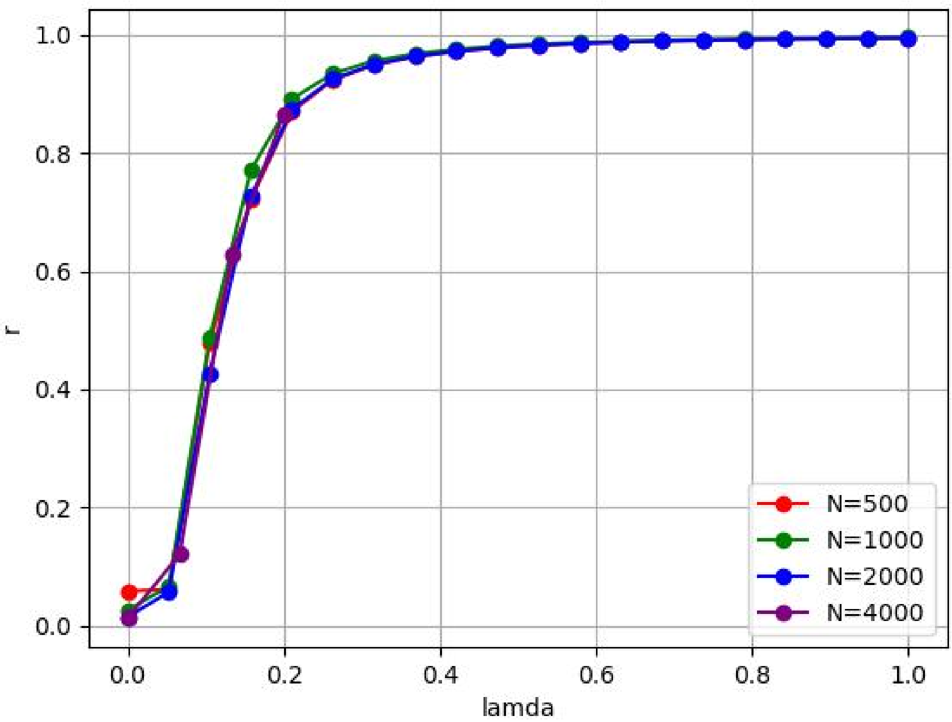
2024 年 4 月 10 日

北京航空航天大學
BEIHANG UNIVERSITY

作业要求如下：

参考 kuramoto 库的实现代码（利用积分实现），对 1 中的 2 篇论文的模型进行复现，观察对于不同规模的 BA 网络来讲，是否存在临界的耦合强度，并与理论的临界值进行比较，以进一步讨论临界值是否存在，还是因为有限尺度效应（finite size effect）。需要注意序参量的定义。

Model1:

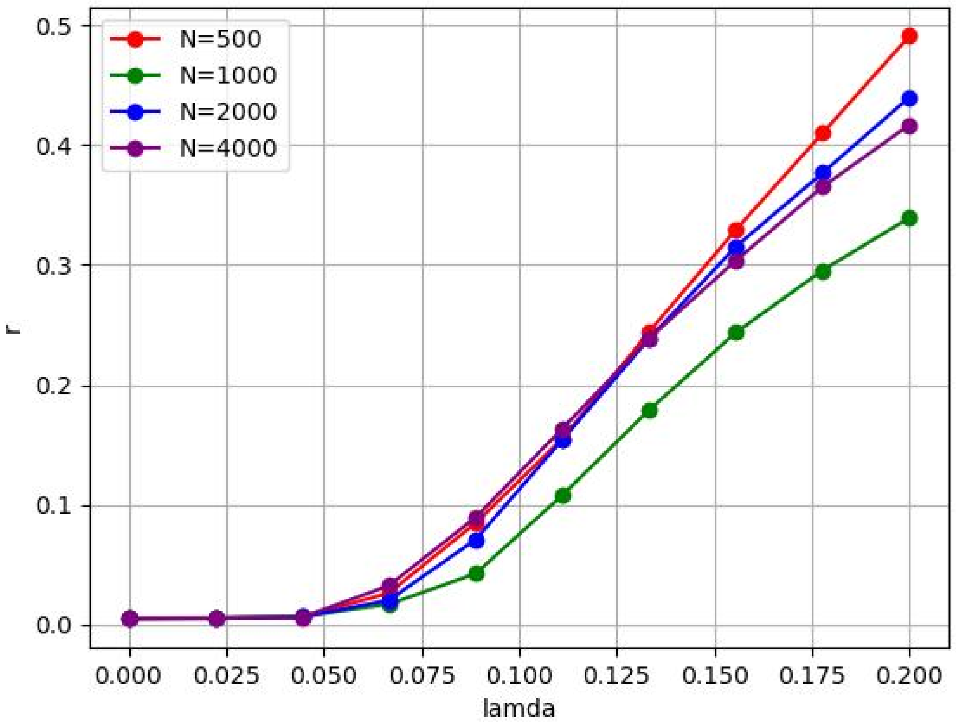


代码如下：

```
import os
import numpy as np
import networkx as nx
import random
from kuramoto import Kuramoto
from multiprocessing import Pool

def run_simulation(N):
    random.seed(3407)
    m = 3
    G = nx.barabasi_albert_graph(N, m)
    G_mat = nx.to_numpy_array(G)
    natfreqs = np.random.uniform(-0.5, 0.5, N)
    coupling_vals = np.linspace(0, 1, 5)
    angles_vec = np.random.uniform(-np.pi, np.pi, N)
    runs = []
    for coupling in coupling_vals:
        model = Kuramoto(coupling=coupling, dt=0.01, T=100, n_nodes=N, natfreqs=natfreqs)
        act_mat = model.run(adj_mat=G_mat, angles_vec=angles_vec)
```

```
        runs.append(act_mat)
    runs_array = np.array(runs)
    results = []
    for i, coupling in enumerate(coupling_vals):
        r_mean = np.mean([model.phase_coherence(vec) for vec in runs_array[i, :, -1000:].T])
        results.append((coupling, r_mean))
    data_filename = f'result/{N}_100_3.txt'
    np.savetxt(data_filename, results, fmt='%.5f', header='Coupling, Order Parameter')

    return N, results

if __name__ == "__main__":
    N_values = [500, 1000, 2000, 4000]
    pool = Pool(processes=len(N_values))
    results = pool.map(run_simulation, N_values)
    pool.close()
    pool.join()
```

Model2:



代码如下：

```
import os
import numpy as np
import networkx as nx
import random
from my_kuramoto import Kuramoto
from multiprocessing import Pool
```

```python
def run_simulation(N):
    random.seed(3407)
    m = 8
    G = nx.barabasi_albert_graph(N, m)
    G_mat = nx.to_numpy_array(G)
    natfreqs = np.random.normal(0, 1, N)
    coupling_vals = np.linspace(0, 0.2, 10)
    angles_vec = np.random.uniform(-np.pi, np.pi, N)

    runs = []
    for coupling in coupling_vals:
        model = Kuramoto(coupling=coupling, dt=0.01, T=100, n_nodes=N, natfreqs=natfreqs)
        act_mat = model.run(adj_mat=G_mat, angles_vec=angles_vec)
        runs.append(act_mat)
    runs_array = np.array(runs)
    results = []
    for i, coupling in enumerate(coupling_vals):
        r_mean = np.mean([model.phase_coherence(vec) for vec in runs_array[i, :, -1000:].T])
        results.append((coupling, r_mean))
    # 保存数据到文件
    data_filename = f'result/{N}_100_3.txt'
    np.savetxt(data_filename, results, fmt='%.5f', header='Coupling, Order Parameter')

    return N, results
```

不同之处在于将 natfreqs 的分布由均匀分布改为正态分布。且根据序参量将 Kuramoto 类中的 phase_coherence 方法修改为:

```python
    @staticmethod
    def phase_coherence(angles_vec):
        #suma = sum([(np.e ** (1j * i)) for i in angles_vec])
        #return abs(suma/len(angles_vec))
        suma = 0
        N = len(angles_vec)
        for j in range(1,N):
            for i in range(j):
                suma += np.e**(-(angles_vec[i]-angles_vec[j])**2)

        return (2*suma) / (N*(N-1))
```