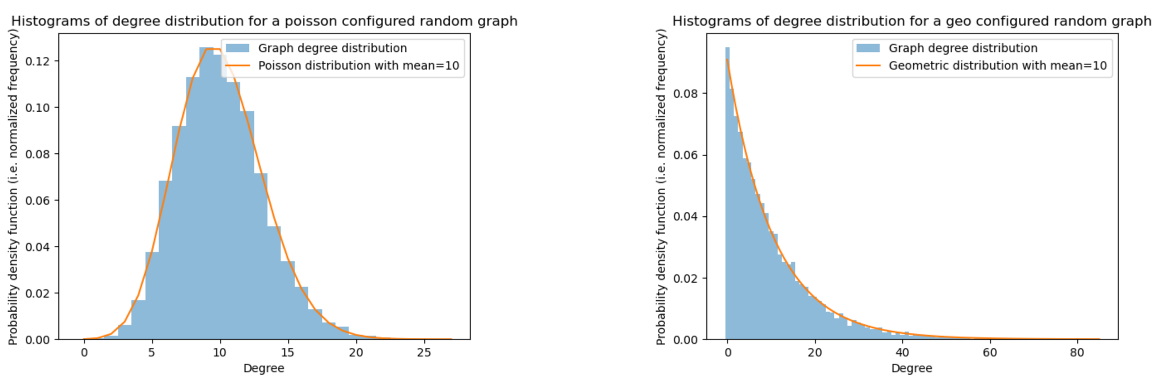
Engineering Tripos Part IIA
SF5 Networks, friendship, and disease: Interim Rerpot 2

Ruitong Sun [rs2177]

1 Configuration Model Sampling

Configuration model is a method to generate networks that (approximately) follow a given degree distribution. Using this method (Appendix A), a network with $n = 10{,}000$ nodes and mean degree $\mu = 10$ is sampled for degree distributions following a Poisson and a geometric ($p_k = p(1-p)^k, k \geq 0$) distribution, respectively. Figure 1 verifies that the degree distributions match their respective probability mass functions (pmf) [1].



(a) Poisson

(b) geometric

Figure 1: Histograms of graph degree distributions sampled from configuration model, in comparison to theoretical distributions.

2 Friendship Paradox

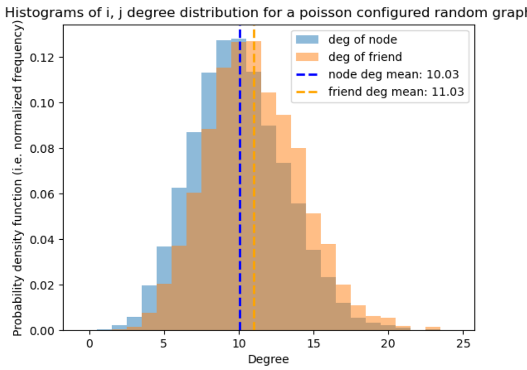
2.1 Empirical observations

Figure 2 illustrates the degree distributions of a randomly chosen friend of a randomly chosen node, for Poisson and geometric degree distributions respectively (Appendix B) [2]. It is observed in both cases that, friend degree distributions are more shifted rightwards than node degree, hence showing the "friendship paradox": on average, an individual's friends have more friends than that individual.

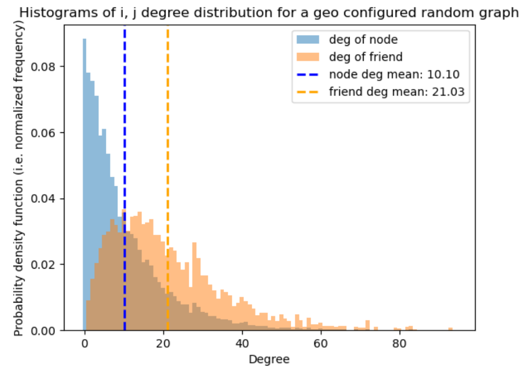
This can be further validated from histograms of $\Delta_i = \kappa_i - k_i$, where k_i is degree of node i and κ_i is average degree of node i's friends (Appendix C). As the histogram mean of Δ is larger than 0 for both cases in Figure 3.

[1] Note that in the algorithm, duplicate edges and self-loops are neglected.

[2] During friend degree sampling, node with degree 0 is neglected and passed to next sampling iteration.

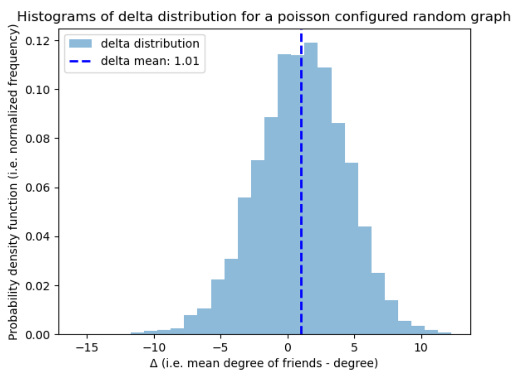


(a) Poisson

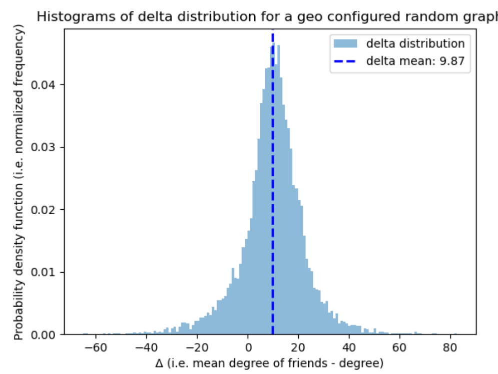


(b) geometric

Figure 2: Histograms of node degree distributions and friend degree distributions.

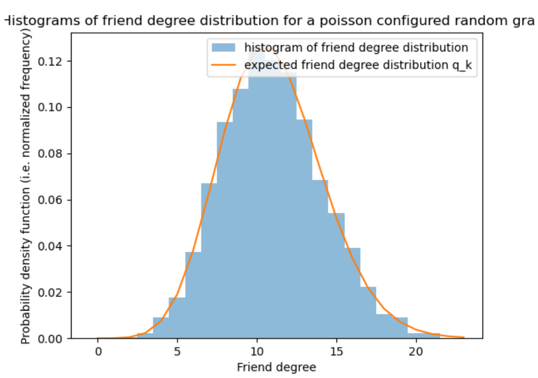


(a) Poisson

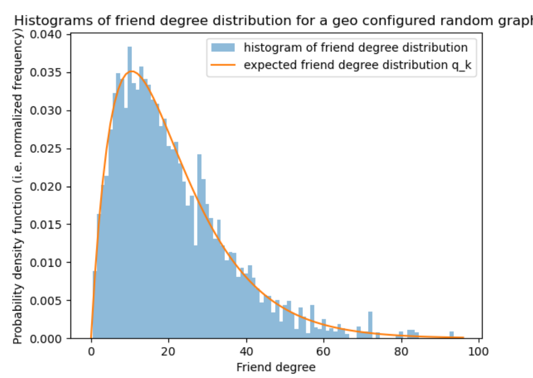


(b) geometric

Figure 3: Histograms of node degree distributions and friend degree distributions.



(a) Poisson



(b) geometric

Figure 4: Histograms of friend degree distributions in comparison to theoretical q_k in Equation1.

2.2 Friend degree distribution

Let q_k denote the distribution for degree of a random friend. For the configuration model, the probability that a friend of degree k gets randomly selected is proportional to k weighted degree distribution

p_k (as there are k edges leading to a friend of degree k). To make it a valid probability distribution, normalization by expected degree $\langle k \rangle = \sum_{k'} k' p_{k'}$ is required. This gives:

$$q_k = \frac{kp_k}{\sum_{k'} k' p_{k'}} \tag{1}$$

An alternative way to argue is that sampling a random friend of a random node is equivalent as sampling a random 'tug' (N.B. each edge consists of 2 'tugs', each leading to 1 node) among all 'tugs'. Hence the proportion of 'tugs' connected to a node of degree k is given by the ratio $q_k = kp_k / \langle k \rangle$, which is same as above.

This can be verified by Figure 4 where friend degree histograms match with the computed plots of q_k for both Poisson and geometric p_k.

Let $\delta = \langle \kappa \rangle - \langle k \rangle$ denote the difference between expected friend degree $\langle \kappa \rangle = \sum_k kq_k$ and expected node degree $\langle k \rangle = \sum_k kp_k$. Substituting expressions for q_k in Equation 1[3]:

$$\delta = \sum_k kq_k - \sum_k kp_k = \sum_k \left(\frac{kp_k}{\langle k \rangle}\right) - \langle k \rangle = \sum_k k^2 p_k / \langle k \rangle - \langle k \rangle$$

$$\delta = \langle k^2 \rangle / \langle k \rangle - \langle k \rangle = \text{var}[k] / \langle k \rangle \tag{2}$$

Equation 2 shows that the quantity Δ is ratio of variance of degree distribution, to its mean. Specific to the 2 distributions:

- Poisson: $\delta = \lambda \div \lambda = 1$
- geometric: $\delta = \frac{1-p}{p^2} \div \frac{1-p}{p} = 1/p$

Setting degree mean $\langle k \rangle = 10$, $\lambda = 10$ for Poisson distribution and $p = \frac{1}{1+\langle k \rangle} = \frac{1}{11}$ for geometric distributions. This gives value $\delta = 1$ and 11 respectively, which is consistent to the mean values differences from histograms in Figure 2 by subtracting friend degree mean and node degree mean.

2.3 Generating function

The generating function for p_k is defined

$$G(z) = \sum_k z^k p_k \tag{3}$$

Through manipulations:

$$G'(z) = \sum_k z^{k-1} kp_k = \frac{1}{z} \sum_k z^k kp_k$$

$$G'(1) = \sum_k kp_k$$

$$\sum_k z^k q_k = \sum_k z^k \frac{kp_k}{\sum_{k'} k' p_{k'}} = \frac{\sum_k z^k kp_k}{G'(1)} = \frac{zG'(z)}{G'(1)}$$

Thus generating function for q_k is

$$Q(z) = \frac{zG'(z)}{G'(1)} \tag{4}$$

Mean of q_k can then be computed via $Q'(1)$. Specific expressions, which analytically validates friendship paradox for the 2 distributions:

- $Q(z) = ze^{\lambda(z-1)}$, hence friend mean degree is greater than node: $Q'(1) = \lambda + 1 > \lambda$
- $Q(z) = \frac{p^2 z}{(1-(1-p)z)^2}$, hence friend mean degree is greater than node: $Q'(1) = \frac{2}{p} - 1 > \frac{1-p}{p}$

[3] $\langle k^2 \rangle = \text{var}[k] + \langle k \rangle^2$

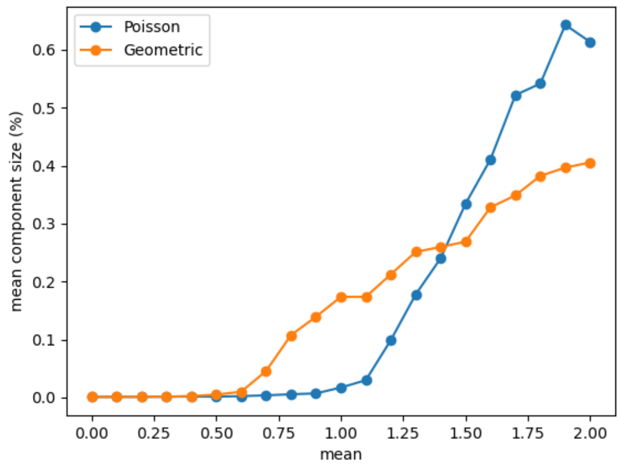


Figure 5: Average component size for Poisson/geometric configured graphs over $\langle k \rangle$.

3 Component

Analytically, it is expected that average size of component containing a given node start to grow once $\langle k \rangle > 1$. However, Figure 5 shows that growth starts consistently earlier for geometric degree distribution than Poisson, before the critical point $\langle k \rangle = 1$. This is because a heavier tail of geometric distribution [4], lead to more presence of high-degree nodes, which disproportionately increase the connectivity of the network. As a result, there observes an early formation of giant component.

A mathematical argument can be developed by considering the excess degree of a node reached by following one of its random edge (i.e. friend degree -1). Formation of giant component requires mean of excess degree > 1:

$$\langle k_{excess} \rangle = \sum_k (k-1) q_k = \frac{\sum_k (k^2 - k) p_k}{\langle k \rangle} = \frac{\langle k^2 \rangle - \langle k \rangle}{\langle k \rangle} > 1$$

By re-arranging, this leads to the giant component condition: $\langle k^2 \rangle - 2\langle k \rangle > 0$.

- Poisson: $\langle k^2 \rangle - 2\langle k \rangle = \lambda(\lambda - 1) > 0 \Rightarrow \lambda = 1$ thus growth only starts at $\langle k \rangle = 1$.
- geometric: $\langle k^2 \rangle - 2\langle k \rangle = 1/p > 0$ which is always satisfied, hence allowing greater connectivity even at lower $\langle k \rangle = 1$.

4 Correlation of Neighbouring Degrees

A higher degree correlation (i.e. high degree assortativity) means nodes with similar degrees tend to connect to each other. $\Delta_i = \kappa_i - k_i$ tends to become smaller towards 0 (i.e. node i's friends are mostly as popular as itself, on average). The friendship paradox may still hold, but less pronounced across the entire network, as few people have significantly more friends than they do (i.e. fewer cases of high-degree node to low-degree node connection, which skew the friend degree average). Instead, friendship paradox may become more localized within clusters of similar-degree nodes.

High degree correlation lead to strong local clustering, hence fragmentation of the network into multiple smaller components. This may inhibit the formation of a giant component. As a result, the critical threshold for growth of component size from 0 may be higher in assortative networks, requiring larger $\langle k \rangle$ to form a giant component.

[4] For geometric distribution, degree variance $\frac{1-p}{p^2}$ can be significantly larger than mean $\frac{1-p}{p}$, making the distribution broader and more skewed.

A Appendix: Configuration Model

```
def config_graph_gen(n, k_ary, m_only=False):
    """Generate a configuration model network with n nodes and degree array k_ary."""
    S = np.array([i for i in range(n) for _ in range(k_ary[i])])
    S = np.random.permutation(S)
    if len(S) % 2 == 1:
        S = S[:-1]

    # S gives list of edges, with each row representing the index of the two
    nodes connected by the edge
    S = S.reshape(-1, 2)
    adj_m = np.zeros((n, n), dtype=int)
    for idx in S:
        if idx[0] != idx[1]:
            adj_m[idx[0]][idx[1]] = 1
            adj_m[idx[1]][idx[0]] = 1

    if m_only:
        return adj_m
    else:
        rm_graph = Network(adj_m=adj_m)
        return rm_graph

def deg_dist_poisson(n, mean):
    """Generate a degree array k_ary for n nodes from Poisson distribution
    with mean = lam."""
    lam = float(mean)
    k_ary = np.random.poisson(lam, n)
    return k_ary


def deg_dist_geo(n,mean):
    """Generate a degree array k_ary for n nodes from geometric distribution
    (# failures before success) with mean = (1-p)/p."""
    p = 1/(1+mean)
    k_ary = np.random.geometric(p, n)-1 # subtract 1 to account for k>=0 not 1
    return k_ary
```

B Appendix: Friend Degree Sampling

```
def friend_deg_dist(graph, num_trials=10000):
    """sample the degree distribution of friends of a random node"""
    n = graph.n
    friend_degs = np.zeros(num_trials)
    for idx in range(num_trials):
        # Randomly select a node i with at least one friend
        friends_i = None
        while not friends_i:
            i = np.random.randint(n)
            friends_i = graph.neighbors(i)

        # Randomly select a friend j of the node i and compute degree of j
        j = np.random.choice(list(friends_i))
        friend_degs[idx] = graph.deg(j)
```

```
    if 0 in friend_degs:
        raise ValueError("Zero degree found in friend degree distribution")
    return friend_degs
```

C Appendix: Δ Sampling

```
def delta_hist(graph):
    """Plotting histograms of difference between degree of random node i
    and average of its friends' degrees"""
    plt.figure()
    n = graph.n
    delta_dist = []
    for i in range(n):
        friends_i = graph.neighbors(i)
        if friends_i: # only consider nodes with at least one friend
            k_i = graph.deg(i)
            kappa_i = np.mean([graph.deg(j) for j in friends_i])
            delta_i = kappa_i - k_i
            delta_dist.append(delta_i)

    bins = np.arange(np.min(delta_dist), np.max(delta_dist))-0.5
    plt.hist(delta_dist, bins=bins, density=True, alpha=0.5, label="delta distribution")
```