Random graphs, social networks and the internet

Lecture 3

Mariana Olvera-Cravioto

UNC Chapel Hill

molvera@unc.edu

October 7th, 2021

The Albert-Barabási model

- All the random graph models we have seen so far are **static**.
- Static models do not explain how graphs grow.
- **Evolving** models propose a mechanism for choosing how a new vertex will connect to the existing graph.
- Vertices are labeled in the order in which they arrive to the graph.
- One of the most famous evolving random graph models is the **Albert-Barabási graph** or **preferential attachment model**.
- This model assumes that an incoming vertex will choose a vertex to connect to with probability proportional to its degree.
- In other words, newcomers "prefer" to attach to high degree vertices.

The Albert-Barabási model... cont.

- The model starts with one vertex that has a self-loop.
- At each time step, a new vertex arrives and connects by drawing one edge either to itself, or to an existing vertex.
- Let $D_i(k)$ be the degree of vertex i after k vertices have arrived.
- When vertex $k+1$ arrives it attaches to vertex i with probability:

$$p_i(k) = \begin{cases} \frac{D_i(k)}{2k+1}, & i = 1, \ldots, k, \\ \frac{1}{2k+1}, & i = k+1. \end{cases}$$

- This model produces **scale-free** graphs with degree distribution:

$$P_k(n) = \frac{1}{n} \sum_{i=1}^{n} 1(D_i(n) = k) \approx 4k^{-3}$$

for large n.

Preferential attachment models

- A generalization of the model allows each new vertex to attach using $m \geq 1$ edges, and attaches the jth edge of vertex $k+1$ to vertex i with probability:

$$p_i(k) = \frac{D_i(k, j-1) + \delta}{\sum_{v=1}^{k} (D_v(k, j-1) + \delta)}, \qquad i = 1, \ldots, k, k+1,$$

  where $\delta > -m$ and $D_i(t, j)$ is the degree of vertex i after t vertices have arrived and j edges of vertex $t+1$ have been attached.

- This model generates **scale-free** graphs with degree distribution

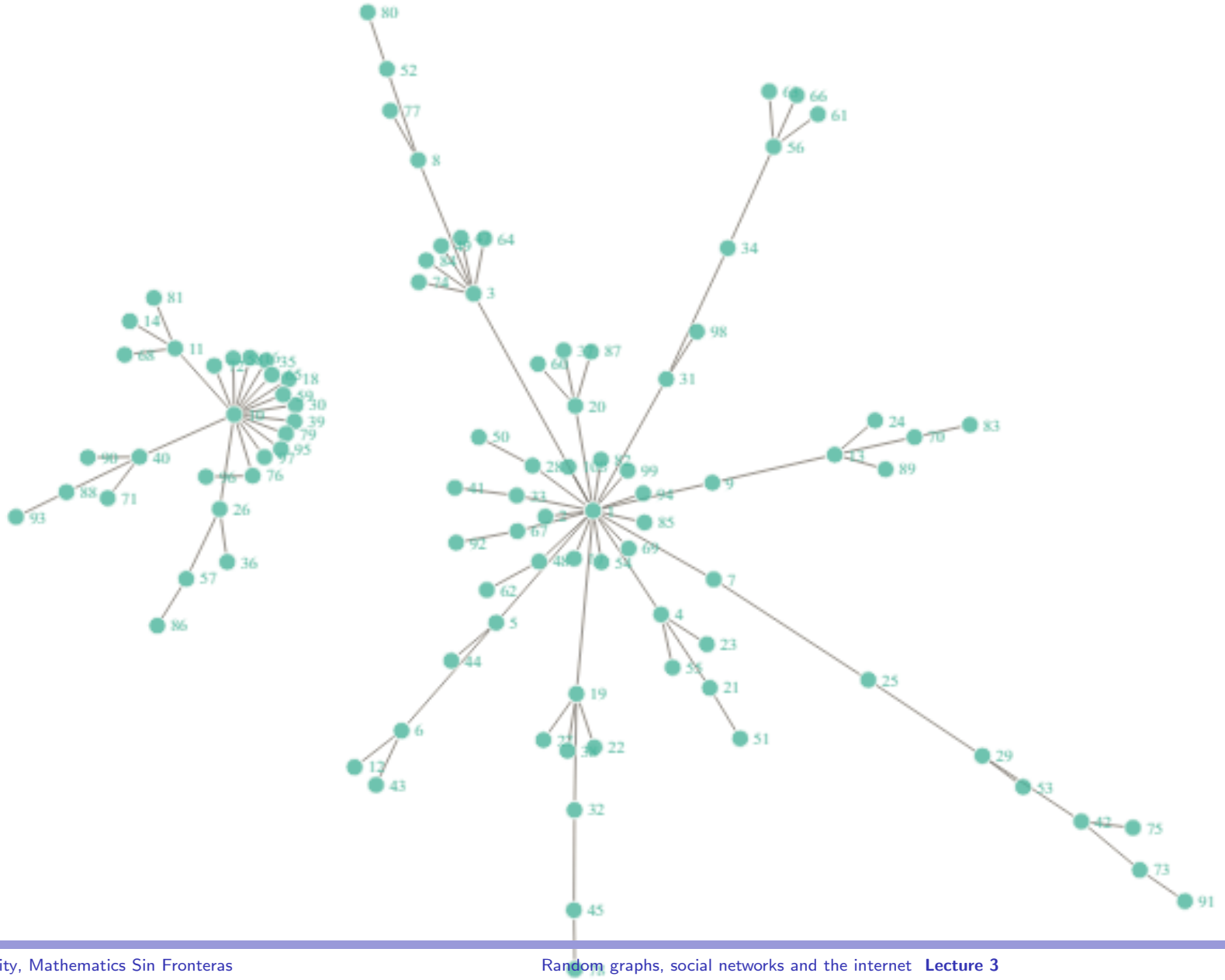
$$P_k(n) = \frac{1}{n} \sum_{i=1}^{n} 1(D_i(n, m) = k) \approx C_{m,\delta} k^{-\tau}$$

  for large n, where $\tau = 3 + \delta/m$.

Preferential attachment models... cont.

- In preferential attachment models, the degrees of **older** vertices are very different from those of **younger** ones.
- The "time-stamp" of a vertex, i.e., its time of arrival, gives us a lot of information about its properties.
- Older vertices tend to have larger degrees.
- The **largest degree** grows as $O(n^{-1/(2+\delta/m)})$ as $n \to \infty$.

An Albert-Barabási graph



Stochastic simulation

- Suppose we want to run experiments using one of the random graph models we have seen.
- For example, we can change the number of vertices and the parameters of the model to explore its properties.
- We may want to use the graph to evaluate other phenomena on graphs.
- **Question:** How do we do it?

Stochastic simulation

- Suppose we want to run experiments using one of the random graph models we have seen.
- For example, we can change the number of vertices and the parameters of the model to explore its properties.
- We may want to use the graph to evaluate other phenomena on graphs.
- **Question:** How do we do it?
- **Answer:** We can use a computer to generate random numbers that we can then use to create the graph.
- The technique of generating random numbers or processes using a computer is the focus of **stochastic simulation**.

Uniform random numbers

- Any programming language (e.g. Python, C, R, Matlab), and even Excel, can generate random numbers.
- We will use U to denote a uniform random variable on the interval $[0, 1]$.
- For any $0 \leq a < b \leq 1$,

$$P(a < U < b) = b - a$$

- Informally, U is equally likely to take any value in $[0, 1]$.
- Computers can generate long sequences of independent uniform random numbers in $[0, 1]$.
- **Uniform** random numbers can be used to generate any other kind of random numbers.

Virtual coin flips

- Suppose we want to simulate a coin flip with head probability p.
- We will generate a random number X such that

$$X = \begin{cases} 1, & \text{with probability } p, \\ 0, & \text{with probability } 1-p. \end{cases}$$

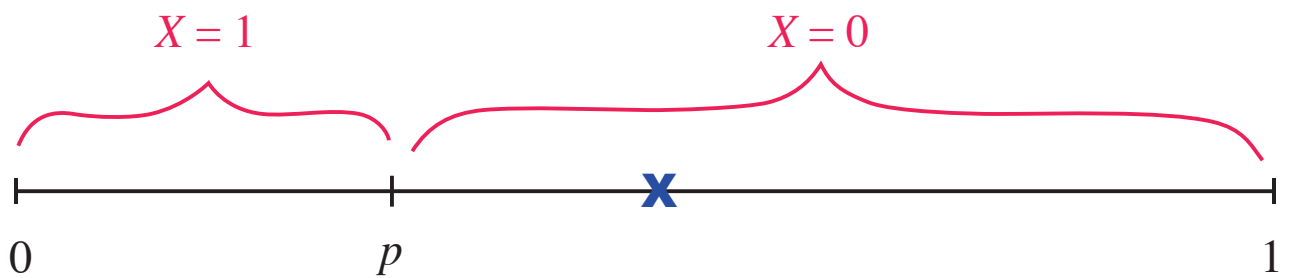
- We associate $X = 1$ with the event that the coin flip was a **head**, and $X = 0$ with the event that the coin flip was a **tail**.
- **Question:** How can we use a uniform random number U to generate X?

Virtual coin flips

- Suppose we want to simulate a coin flip with head probability p.
- We will generate a random number X such that

$$X = \begin{cases} 1, & \text{with probability } p, \\ 0, & \text{with probability } 1-p. \end{cases}$$

- We associate $X = 1$ with the event that the coin flip was a **head**, and $X = 0$ with the event that the coin flip was a **tail**.
- **Question:** How can we use a uniform random number U to generate X?



Virtual coin flips... cont.

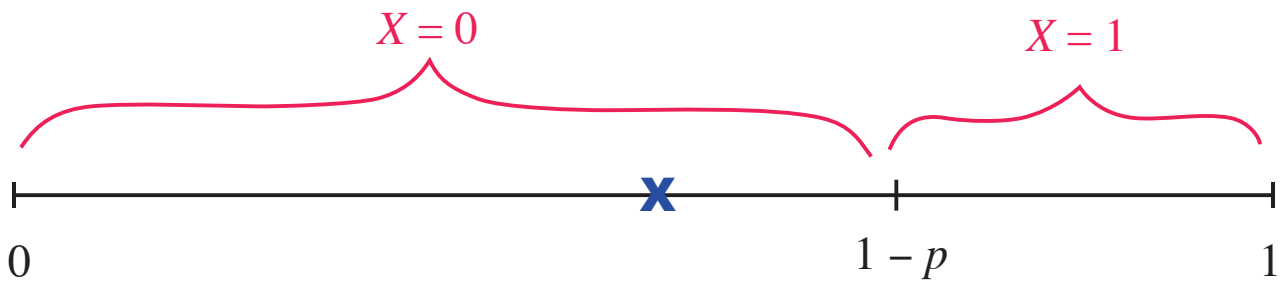
- This is equivalent to setting:

$$X = 1(U \leq p),$$

where $1(A) = 1$ if A happens and $1(A) = 0$ if A^c happens.

- **Note:** We could also have taken

$$X = 1(U > 1 - p)$$



Generating discrete random numbers

- Suppose we want to simulate a random variable X that can take the values $\{x_1, x_2, \ldots, x_k\}$.
- Let $\{p_i : 1 \leq i \leq k\}$ be the probability mass function of X, i.e.,

$$p_i = P(X = x_i), \qquad 1 \leq i \leq k$$

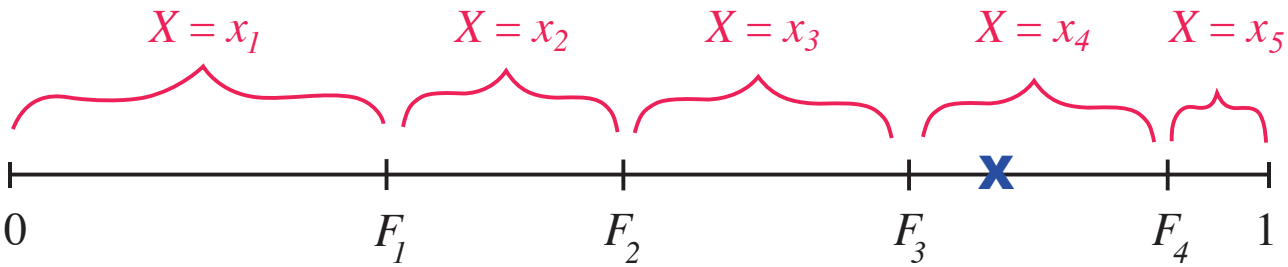
- Now compute the distribution function of X by setting $F_0 = 0$ and:

$$F_i = p_1 + \cdots + p_i, \qquad 1 \leq i \leq k$$

- **Question:** How can we use a uniform random number U to generate X?

Generating discrete random numbers... cont.

- Let U be a uniform random number in $[0, 1]$.



- This is equivalent to setting:

$$X = \sum_{i=1}^{k} x_i 1(F_{i-1} \leq U < F_i)$$

Simulating a continuous distribution

- Suppose we want to generate a random variable X such that:
  - X has a distribution function $F(x) = P(X \leq x)$
  - $F(x)$ is continuous and strictly increasing when $0 < F(x) < 1$ (it can be just nondecreasing for $F(x) = 0$ or $F(x) = 1$)
  - $F(x)$ has an inverse $F^{-1}(x)$.

- **Algorithm:**
  1. Generate $U \sim$ Uniform[0,1]
  2. Return $X = F^{-1}(U)$

- This is called the **inverse transform method**.

Example: Pareto random numbers

- Generating scale-free graphs using some of the models we saw requires simulating Pareto random numbers.
- A Pareto distribution function takes the form:

$$F(x) = 1 - (x/b)^{-\alpha}, \qquad x \geq b$$

- The parameter $\alpha > 0$ controls the **shape** of the distribution (how heavy its tail is), and the parameter $b > 0$ controls its **scale** (and its support).
- We can compute its inverse as follows:

Example: Pareto random numbers

- Generating scale-free graphs using some of the models we saw requires simulating Pareto random numbers.
- A Pareto distribution function takes the form:

$$F(x) = 1 - (x/b)^{-\alpha}, \qquad x \geq b$$

- The parameter $\alpha > 0$ controls the **shape** of the distribution (how heavy its tail is), and the parameter $b > 0$ controls its **scale** (and its support).
- We can compute its inverse as follows:

$$u = 1 - (x/b)^{-\alpha} \iff (x/b)^{-\alpha} = 1 - u \iff x = b(1-u)^{-1/\alpha}$$

- Hence, $F^{-1}(u) = b(1-u)^{-1/\alpha}$, and $X = F^{-1}(U)$ has a Pareto(α, b) distribution.

Generating random graphs

- Generating coin flips, discrete random numbers, and continuous random variables via the inverse transform method is all we need to simulate any of the random graph models we have seen.
- The models we have seen are:
  - Erdős-Rényi graph
  - Chung-Lu model
  - Stochastic block model
  - Random intersection graph
  - Albert-Barabási model
- We will give an algorithm in **pseudocode** to generate each of the models.
- In all cases, the goal is to obtain the adjacency matrix A for the generated graph.

Erdős-Rényi graph

- This algorithm generates an undirected Erdős-Rényi graph with n vertices and edge probabilities p.
- Set $p \in (0,1)$ and n; initialize matrix $A \in \mathbb{R}^n \times \mathbb{R}^n$.
- For $i = 1 : n$
  - For $j = 1 : n$
    - If $i = j$ set
    $$a_{i,j} = 0$$
    - If $i \neq j$ generate a uniform $[0,1]$ random number $U_{i,j}$ and set
    $$a_{i,j} = a_{j,i} = 1(U_{i,j} \leq p)$$
- Return adjacency matrix A.

Chung-Lu model

- This algorithm generates a Chung-Lu model with n vertices and weight distribution $F(x) = 1 - (x/b)^{-\alpha}$.
- Set n and initialize matrix $A \in \mathbb{R}^n \times \mathbb{R}^n$ and weight vector $W \in \mathbb{R}^n$.
- For $i = 1 : n$
  - Generate a uniform $[0, 1]$ random number U_i and set

$$W_i = b(1 - U_i)^{-1/\alpha}$$

- Compute $L_n = \sum_{i=1}^{n} W_i$.
- For $i = 1 : n$
  - For $j = 1 : n$
    - If $i = j$ set

$$a_{i,j} = 0$$

    - If $i \neq j$ generate a uniform $[0, 1]$ random number $U_{i,j}$ and set

$$p_{i,j} = \frac{W_i W_j}{L_n} \wedge 1 \qquad \text{and} \qquad a_{i,j} = a_{j,i} = 1\,(U_{i,j} \leq p_{i,j})$$

- Return adjacency matrix A.

Stochastic block model

- This algorithm generates a SBM model with n vertices, K communities of sizes $\{\pi_{1,n}, \ldots, \pi_{K,n}\}$, and kernel κ.
- Set n and initialize matrix $A \in \mathbb{R}^n \times \mathbb{R}^n$.
- Assign to each vertex its community label, say in a vector $(C_1, \ldots, C_n)$, e.g., by giving the first $\pi_{1,n}$ vertices label 1, the next $\pi_{2,n}$ label 2, etc.
- For $i = 1 : n$
  - For $j = 1 : n$
    - If $i = j$ set
    $$a_{i,j} = 0$$
    - If $i \neq j$ generate a uniform $[0, 1]$ random number $U_{i,j}$ and set
    $$p_{i,j} = \frac{\kappa(C_i, C_j)}{n} \qquad \text{and} \qquad a_{i,j} = a_{j,i} = 1\,(U_{i,j} \leq p_{i,j})$$
- Return adjacency matrix A.

Random intersection graph

- This algorithm generates an intersection graph with tunable clustering coefficient and weight distribution $F(x) = 1 - (x/b)^{-\alpha}$.
- Fix $\beta, \gamma > 0$, set n, initialize matrix $A \in \mathbb{R}^n \times \mathbb{R}^n$ and weight vector $W \in \mathbb{R}^n$.
- **Generating the bipartite graph:**
- Set $m = \lfloor \beta n \rfloor$ and initialize matrix $B \in \mathbb{R}^n \times \mathbb{R}^m$.
- For $i = 1 : n$
  - Generate a uniform $[0, 1]$ random number U_i' and set

$$W_i = b(1 - U_i')^{-1/\alpha}$$

  - Set $p_i = \frac{\gamma W_i}{n} \wedge 1$.
  - For $j = 1 : m$
    - Generate a uniform $[0, 1]$ random number $U_{i,j}$ and set

$$b_{i,j} = 1(U_{i,j} \leq p_i)$$

- Return adjacency matrix B for the bipartite graph.

Random intersection graph... cont.

- **Generating the intersection graph:**
- For $i = 1 : n$
  - For $j = 1 : m$
    - Let $N_i = B_{i\bullet}$, where $B_{i\bullet}$ is the ith row of matrix B (neighbors of i in the bipartite graph).
    - Let $N_j = B_{j\bullet}$ (neighbors of j in the bipartite graph).
    - Let $J = N_i - N_j$.
    - If $\min |J| = 0$ and $i \neq j$, then set $a_{i,j} = 1$; otherwise $a_{i,j} = 0$.
- Return adjacency matrix A for the random ntersection graph.
- **Note:** Above, for any vector $X = (x_1, \ldots, x_m)$ we write $|X| = (|x_1|, \ldots, |x_m|)$.

Albert-Barabási model

- This algorithm generates an Albert-Barabási model with n vertices.
- Initialize degree vector $D \in \mathbb{R}^n$ and adjacency matrix $A \in \mathbb{R}^n \times \mathbb{R}^n$.
- Set $D(1) = 2$ and $a_{1,1} = 1$.
- For $k = 2 : n$
  - Set $D(k) = 1$ and construct distribution vector F_k as follows:
  - Set $F_k(1) = D(1)$, and for $j = 2 : k$
    - Set $F_k(j) = F_k(j-1) + D(j)$
  - Normalize F_k by updating $F_k \to F_k / \sum_{j=1}^{k} F_k(j)$
  - Sample a uniform $[0, 1]$ random number U and compute

$$J = \sum_{j=1}^{k} j 1(F_k(j-1) < U \leq F_k(j))$$

  - Update:

$$a_{k,J} = a_{J,k} = 1, \qquad D(J) = D(J) + 1$$

- Return adjacency matrix A.

Thank you for your attention.