Mining Complex Networks

2023 CMS Summer Meeting

François Théberge
theberge@ieee.org

Tutte Institute for Mathematics and Computing
uOttawa adjunct professor (Mathematics and Statistics)

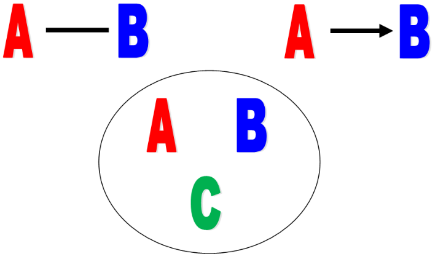
June 2023

Roadmap

1. 9:00-9:50
   - Background material
   - **Relational data and graphs**
   - Random graph models
2. 10:00-10:50
   - Measures of centrality
   - Degree correlation
   - Community detection
3. 11:00-12:00
   - Graph (vertex) embedding
   - Hypergraphs
   - Other applications

Relational Data

- not all data can be represented in a data frame
- data could be *relational*
- examples of relations between entities:
  - *A* and *B* are friends
  - *A* sends an email to *B*
  - *A*, *B* and *C* are in the same team
- the above are modelled as edges or hyperedges:



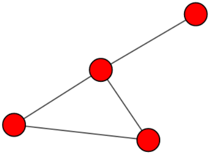
Graphs

For a graph $G = (V, E)$, let $n = |V|$ and $m = |E|$

We map the vertices to integers indices $v_1 \dots v_n$ for convenience

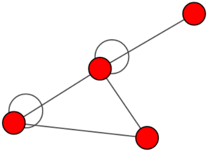
Let $A = (a_{ij})$, the adjacency matrix s.t. $a_{ij} > 0 \iff (v_i, v_j) \in E$

Undirected (unweighted) graph: $a_{ij} = a_{ji} \in \{0, 1\}, a_{ii} = 0$.

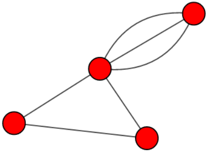


Graphs

Undirected graph with self-loops: some $a_{ii} = 1$.

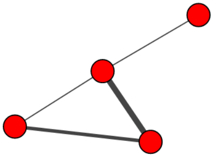


Multigraph: $a_{ij} \in \mathbb{N}$

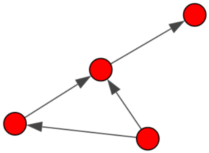


Graphs

Weighted graph: $a_{ij} \geq 0$



Directed graph: can have $a_{ij} \neq a_{ji}$



Graphs

$w = (v_{i_0}, v_{i_1}, ..., v_{i_k})$ is a **walk** between v_{i_0} and v_{i_k} if all $(v_{i_l}, v_{i_{l+1}}) \in E$. A **path** is a walk without repeated nodes.

For unweighted graphs, its **length** is the number of edges. For weighted graphs, the sum of edge weights.

A **connected component** for an undirected graph is a maximal subgraph for which any 2 nodes are connected by a path.

For directed graphs, we distinguish **strong** connected components (connections via directed paths) and **weak** ones.

The **distance** between two nodes (v_i, v_j) is the minimum path length from u to v.

The **diameter** of a (connected) graph is the maximum distance between two nodes.

Graphs

The **degree** of node *v* in the number of edges incident to it.

For directed graphs, we usually distinguish **in-degree** and **out-degree**.

The **degree distribution** describes the distribution of node degrees for a given graphs via statistics such as: minimum and maximum degree (δ, Δ), mean degree, median degree, etc.

Graphs

Many networks, in particular social networks, exhibit **homophily**:

Friends share common friends with higher than random probability.

We can measure this by looking at **triangles** in a graph

Consider an undirected graphs $G = (V, E)$:

A *triad* is a subgraph of 3 nodes forming a tree; let $n_\wedge$ the number of triads in a graph G;

A *triangle* is a fully-connected subgraph with 3 nodes; let n_Δ be the number of triangles in graph G;

Graphs

The **global clustering coefficient** or *graph transitivity* of G is defined as

$$C_{glob} = \frac{3n_\Delta}{n_\wedge}.$$

For a given node $v_i \in V$ of degree d_i, let $n_i(e)$ be the number of edges between neighbours of v_i;

Node v_i's **local clustering coefficient** is defined as:

$$c_i = \frac{n_i(e)}{\binom{d_i}{2}}.$$

Clustering

The **(average local) clustering coefficient** for graph G is:

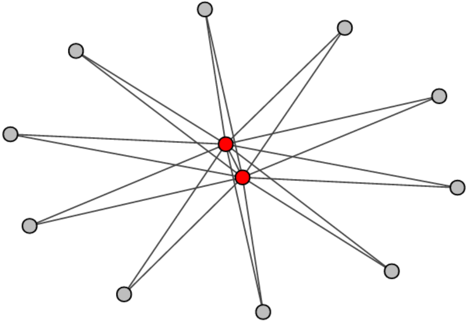
$$C_{loc} = \frac{1}{n} \sum_i c_i$$

the average over all c_i's.

Quantities C_{glob} and C_{loc} are often similar, but this can be misleading;

Clustering

Consider the following graph with $n+2$ nodes (n grey nodes):



Clustering

In this graph, $n_\Delta = n$ and $n_\wedge = 3n + n(n-1) = n^2 + 2n$, so

$$C_{glob} = \frac{3}{n+2} \to 0$$

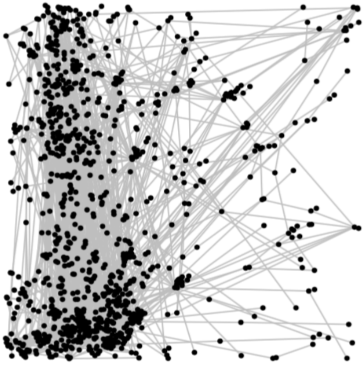
with the limit as $n \to \infty$.

For the 2 red nodes, we get $c_i = 2/(n+1)$ while for the grey nodes, $c_i = 1$. The (average local) clustering coefficient is therefore:

$$C_{loc} = \frac{n^2 + n + 4}{n^2 + 3n + 2} \to 1.$$

Dataset – GitHub developers

Nodes: web and ml developers (37.7k)
Edges: starring common repositories (289k)



(b) GitHub ml subgraph

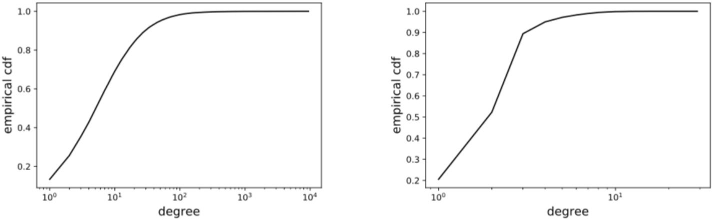
Datasets – Europe electric grid

Nodes: high voltage stations (13.8k)
Edges: physical lines (17.2k)

(c) European Grid network

Comparing Github and Grid graphs

The first difference is in the **degree distribution**
GitHub graph has many low degree nodes and some very large degree nodes, typical of social graphs

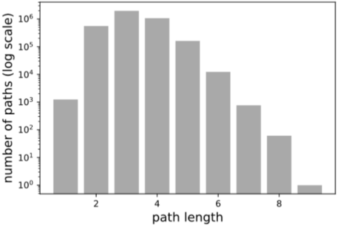


(a) GitHub graph (b) European Grid network

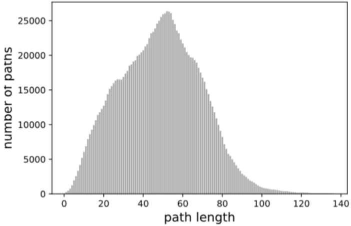
FIGURE 1.3
Empirical cumulative degree distribution (CDF) for the two graphs under study.

Comparing Github and Grid graphs

Shortest path lengths (distances between nodes) are also very different.



(a) GitHub graph



(b) European Grid network

GitHub graph exhibits the **small-world** phenomenon.

Comparing Github and Grid graphs

graph	GitHub	GitHub (ml)	GitHub (web)	Grid
# nodes	37,700	9,739	27,961	13,844
# edges	289,003	19,684	224,623	17,277
δ	1	0	0	1
$\langle k \rangle$	15.332	4.042	16.067	2.496
median degree	6	2	6	2
$d_{quant_{99}}$	138	39	145	8
Δ	9,458	482	8,194	16
diameter	11	13	9	147
# components	1	2,466	297	59
the largest component	37,700	7,083	27,653	13,478
# isolates	0	2,308	285	0
C_{glob}	0.012	0.034	0.014	0.100
C_{loc}	0.193	0.141	0.207	0.113

TABLE 1.1
Basic descriptive statistics for the GitHub and the Grid graphs. The GitHub subgraphs built with the two types of developers (ml and web) are also included. $d_{quant_{99}}$ refers to the 99^{th} quantile for the degree distribution.

Power law

In a **power law** function, one quantity varies as a power of the other.

This is often observed in real, social-type graphs, in particular for the **degree distribution**.

Let p_k the proportion of nodes with degree k, then

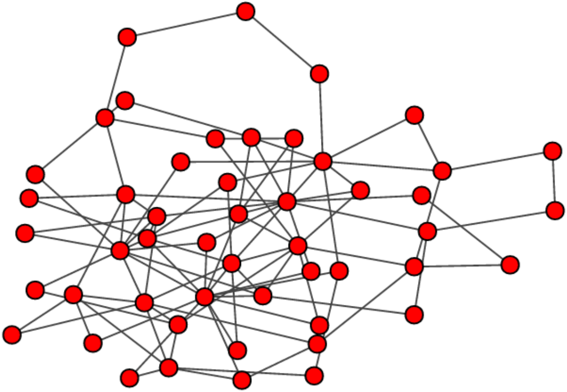
$$p_k = ck^{-\gamma}$$

is an example of a power-law distribution, where c is the normalizing constant.

For degree distribution in social networks, the values typically observed are $2 \leq \gamma \leq 3$.

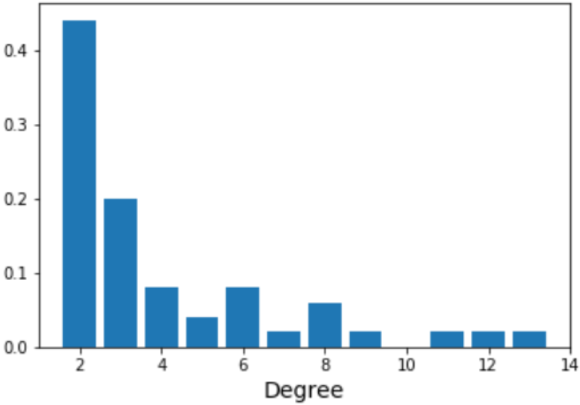
Power law

Here is an example of a graph with power-law degree distribution where $\gamma = 2$:



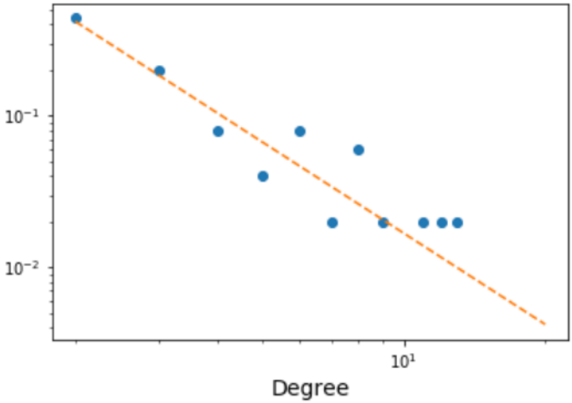
Power law

and it's degree distribution:



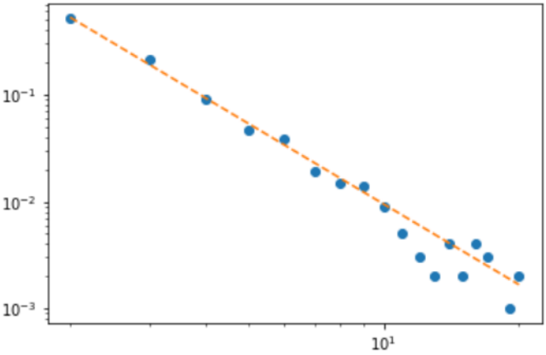
Power law

On a log-log plot, we see a roughly linear relation (with log binning of degrees):



Power law

Here is another log-log plot for a much larger graph with $\gamma = 2.5$:



Power law

Therefore, power-law graphs exhibit a degree distribution with:

- a fairly large number of small degree nodes, and
- a ”long tail” which amounts to the presence of high degree nodes.

Such high degree nodes are called hubs (hubs and authorities for directed graphs).

Summary

We saw several simple descriptive statistics to characterize graphs such as:

- degree distribution
- shortest path distribution
- clustering coefficients

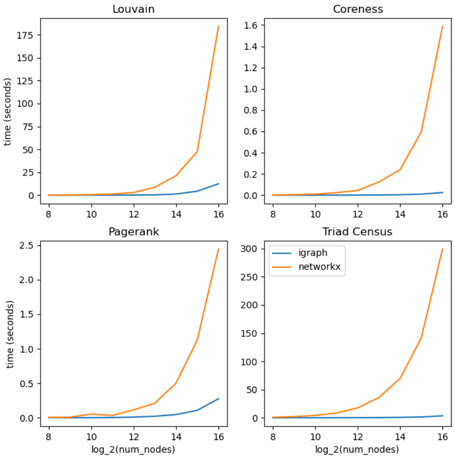
In practice, **EDA** (exploratory data analysis) is an important part of a data mining task.

When possible, **visualization** is also useful.

There are several good tools for handling graphs. For Python users, we recommend **python-igraph**, which is used in our notebooks.

Summary

igraph has good scalability



Summary

links

Slides and notebooks:
github.com/ftheberge/CMS2023

CRC textbook:
www.torontomu.ca/mining-complex-networks

companion notebooks:
github.com/ftheberge/GraphMiningNotebooks

Roadmap

1. 11:00-11:50
   - Background material
   - Relational data and graphs
   - **Random graph models**
2. 12:00-12:50
   - Measures of centrality
   - Degree correlation
   - Community detection
3. 13:00-14:00
   - Graph (vertex) embedding
   - Hypergraphs
   - Other applications

Random Graph Models

Theory of random graphs is at the intersection of graph theory and probability.

It is an active area of research:

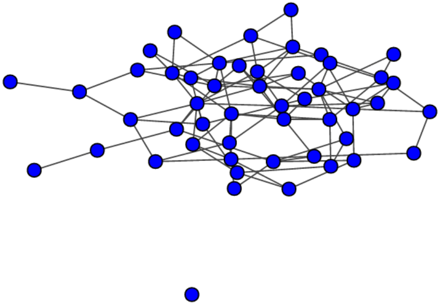
- uncovering properties of typical graphs
- find “surprising” objects
- help understand network formation (eg. preferential attachment models explain power-law degree distribution)
- for data science:
  - create flexible synthetic yet realistic graphs to test various scenarios
  - benchmark algorithms

We review a few commonly used models.

Erdős-Rényi Models

The $G(n, m)$ model:

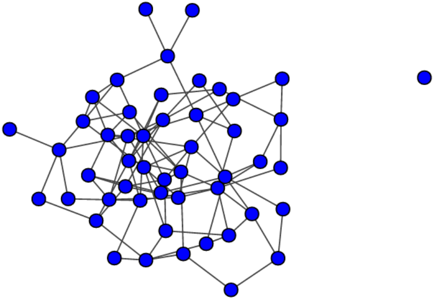
- n nodes
- m edges chosen at random from $N = \binom{n}{2}$ pairs
- average degree $k = 2m/n$



Erdős-Rényi Models

The $G(n, p)$ model:

- n nodes
- each possible $N = \binom{n}{2}$ node pair is connected with probability p
- expected number of edges is Np
- expected average degree $k = p(n-1)$
- allows for easy calculation of graph statistics



Number of edges:

Let P_m the probability of getting m edges with the $G(n, p)$ model, and let $N = \binom{n}{2}$.

$$P_m = \binom{N}{m} p^m (1-p)^{N-m}$$

Given that N is large and p is (typically) small, we can use the Poisson approximation to the binomial with $\lambda = Np$:

$$P_m \approx \frac{e^{-\lambda}\lambda^m}{m!}$$

Erdős-Rényi Models

Degree distribution:

Let p_k, the probability that some node has degree k in $G(n, p)$.

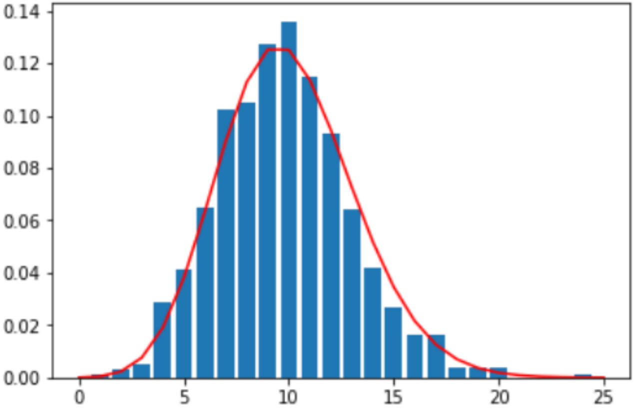
$$p_k = \binom{n-1}{k} p^k (1-p)^{n-1-k} \approx \frac{e^{-\lambda}\lambda^k}{k!}$$

with $\lambda = (n-1)p$, the expected average degree.

In view of the Poisson distribution, such models do not generate *hubs*, the high-degree nodes typically seen in social-type networks.

Erdős-Rényi Models

Degree distribution ($n = 1000, \ p = .01$):



Erdős-Rényi Models

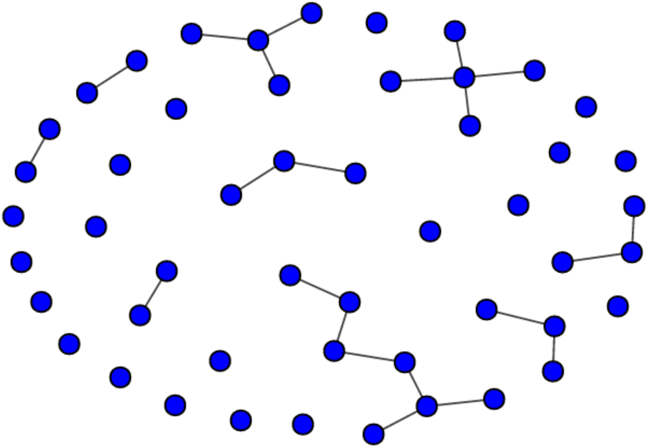
Giant connected component

For $G(n, p)$ with expected average degree $k = p(n - 1)$, we distinguish 3 regimes:

1. $k < 1$: subcritical regime; no giant component, connected components are mostly trees.
2. $k > 1$: supercritical regime; single giant component, small components are mostly trees.
3. $k >> \log(n)$: connected regime; no isolated nodes or small components.

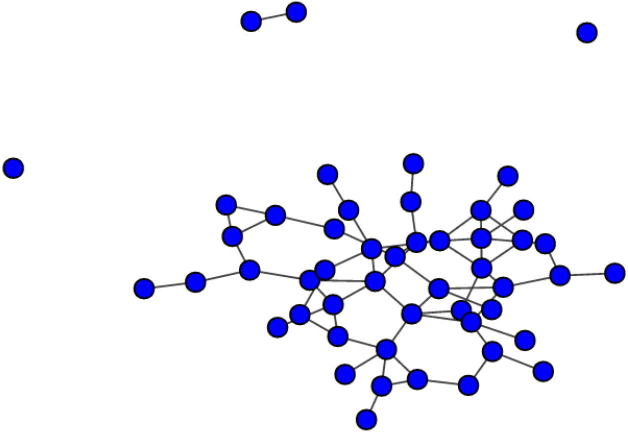
Erdős-Rényi Models

Subcritical regime:



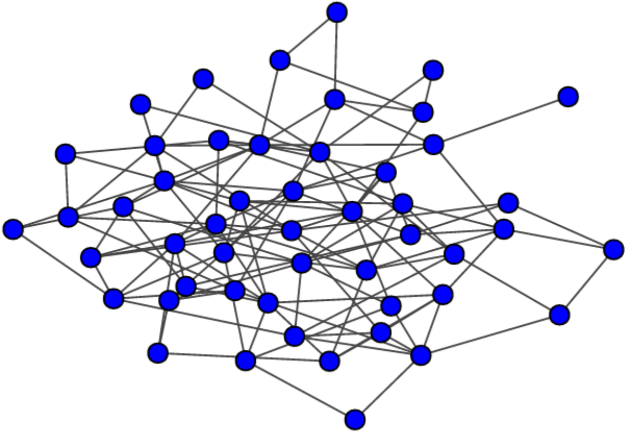
Erdős-Rényi Models

Supercritical regime:



Erdős-Rényi Models

Connected regime:



Chung-Lu Models

Degree distribution in Erdős-Rényi Models is likely non-realistic

Let G be a graph with vertices $V = \{v_1, \ldots, v_n\}$.

With Chung-Lu models, we also consider a degree sequence: $d_i = deg_G(v_i)$, with $vol(V) = \sum_i d_i = 2|E|$.

Degree distribution could be **power-law**.

Model I: probability of an edge between vertices v_i and v_j is:

$$p_{ij} = \frac{d_i d_j}{vol(V)}, \ i \neq j \text{ and } p_{ii} = \frac{d_i^2}{2vol(V)}.$$

This is also known as the **Bernoulli Chung-Lu** model.

Chung-Lu Models

If we define $\mathcal{CL}_1(G)$ to be the distribution of graphs obtained with Model I, then for $G' = (V, E') \sim \mathcal{CL}_1(G)$:

- $\mathbb{E}_{G' \sim \mathcal{CL}_1(G)}(deg_{G'}(v_i)) = d_i, 1 \leq i \leq n$,
- $\mathbb{E}(|E'|) = |E|$
- there are no multi-edges, and
- there can be self-edges.

Generating graphs with Model I is not scalable, as it requires $O(n^2)$ Bernoulli experiments.

Chung-Lu Models

Model II is generated from a scalable algorithm with $O(|E|)$ steps only.

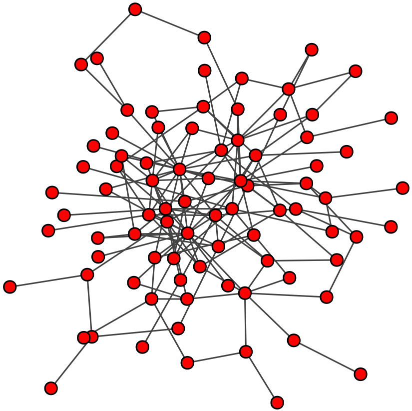
Generate a graph over vertices V by selecting $|E|$ edges $e = (u_1, u_2)$ where each u_i is independently sampled from V according to the multinomial distribution where $p(v_i) = d_i / vol(V)$.

If we define $\mathcal{CL}_2(G)$ to be the distribution of graphs obtained with Model II, then for $G' = (V, E') \sim \mathcal{CL}_2(G)$:

- $\mathbb{E}_{G' \sim \mathcal{CL}_2(G)}(deg_{G'}(v_i)) = d_i, 1 \leq i \leq n$,
- there can be multi-edges, and
- there can be self-edges.

Chung-Lu Models

Chung-Lu graph with power-law degree distribution:



Configuration Model

The Chung-Lu models are probabilistic models for edge generation, with **expected** degree sequence.

With the **configuration model**, we specify an exact degree sequence for the nodes: $d_1, ..., d_n$.

The sequence needs to be **graphic.**

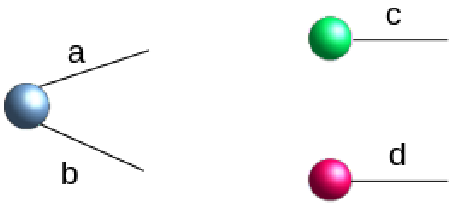
Each graph with n nodes and this exact degree sequence is assigned the same probability.

This is fast to generate ... but graphs may have loops and multi-edges.

Configuration Model

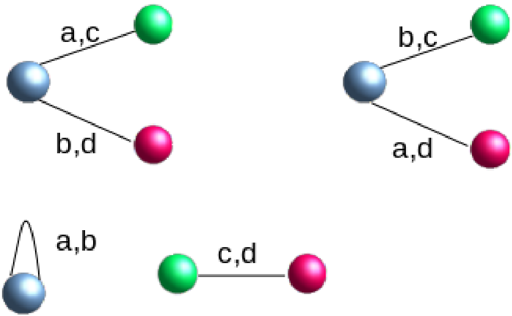
For each node i, we assign d_i stubs (half-edges);

Stubs are connected at random.



Configuration Model

This can generate self-edges and multi-edges



Configuration Model

For this model, the expected number of edges between nodes $i \neq j$ is:

$$e_{ij} = \frac{d_i d_j}{2m - 1}$$

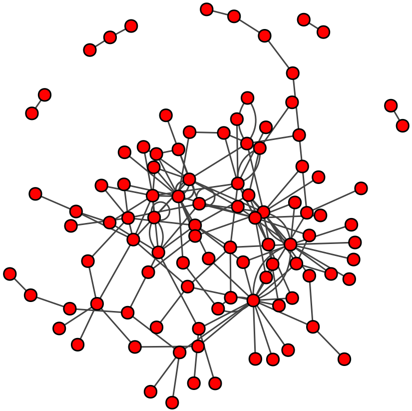
and:

$$e_{ii} = \frac{d_i(d_i - 1)}{2(2m - 1)}$$

Several algorithms to generate such graphs are available in `igraph`.

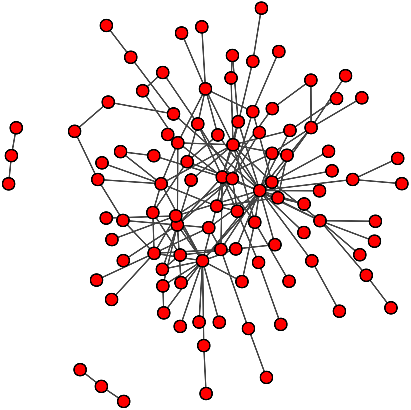
Configuration Model

Simple method (can generate loops, multiedges):



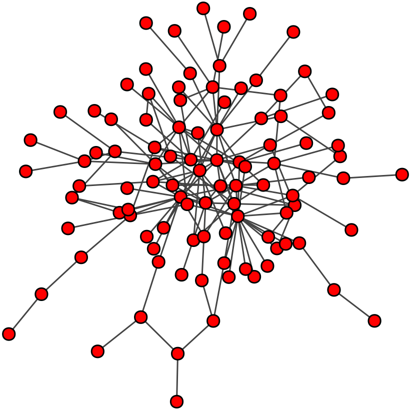
Configuration Model

Avoiding loops and multiedges (restart if stuck):



Configuration Model

Viger's algorithm (returns a connected graph):



Example - GitHub Subgraph

Modelling the GitHub graph:

Graph	Base	Binomial	Chung-Lu	Config.	Config.(V)
nodes	7,083	7,083	7,083	7,083	7,083
edges	19,491	19,491	19,491	19,491	19,491
$\delta = d_{min}$	1	0	0	1	1
d_{mean}	5.504	5.504	5.504	5.504	5.504
d_{median}	2	5	3	2	2
$\Delta = d_{max}$	482	18	406	482	482
diameter	13	10	11	10	11
components	1	25	1,063	70	1
largest	7,083	7,058	5,992	6,940	7,083
isolates	0	23	1,035	0	0
C_{glob}	0.0338	0.0007	0.0198	0.0185	0.0171
C_{loc}	0.1412	0.0006	0.0258	0.0242	0.0319

TABLE 2.8
Comparison of a few descriptive statistics for the base graph (a subgraph of the GitHub graph) with 4 random graph models.

Small world

Real graph do not typically look like random graphs.

Some observed characteristics include:

- relatively short paths between nodes (small diameter)
- local density behaviour (triangles, communities)
- large number of low degree nodes
- presence of high degree nodes (hubs, authorities)
- power law degree distribution

Barabasi-Albert

This is an example of a **preferential attachment** model, also referred to as *the rich gets richer*;

A graph is generated node by node:

- start from some initial graph (ex: empty graph, small clique) at step $t = 0$
- at step $t > 0$, add a new node with (up to) m edges to existing nodes
- the probability that the new node has an edge with (existing) node v_i is proportional to $d_i(t)$, the degree of node v_i before adding this new node.
- continue until n nodes are generated.

Barabasi-Albert

The model favours attaching to high degree nodes, thus forming hubs;

Asympotically, the proportion of degree k nodes is:

$$p_k \approx 2m(m+1)k^{-3}$$

There are several variations on that theme.

Roadmap

1. 11:00-11:50
   - Background material
   - Relational data and graphs
   - Random graph models
2. 12:00-12:50
   - Measures of centrality
   - Degree correlation
   - Community detection
3. 13:00-14:00
   - Graph (vertex) embedding
   - Hypergraphs
   - **Other applications**

Other applications

There are several topics we did not touch such as:

- overlapping community detection
- embedding edges, whole graphs
- semi-supervised learning
- anomaly detection
- graph robustness
- road networks
- dynamic graphs

Thanks for your attention!

theberge@ieee.org

Slides and notebooks: github.com/ftheberge/CMS2023

CRC textbook: www.torontomu.ca/mining-complex-networks

companion notebooks: github.com/ftheberge/GraphMiningNotebooks