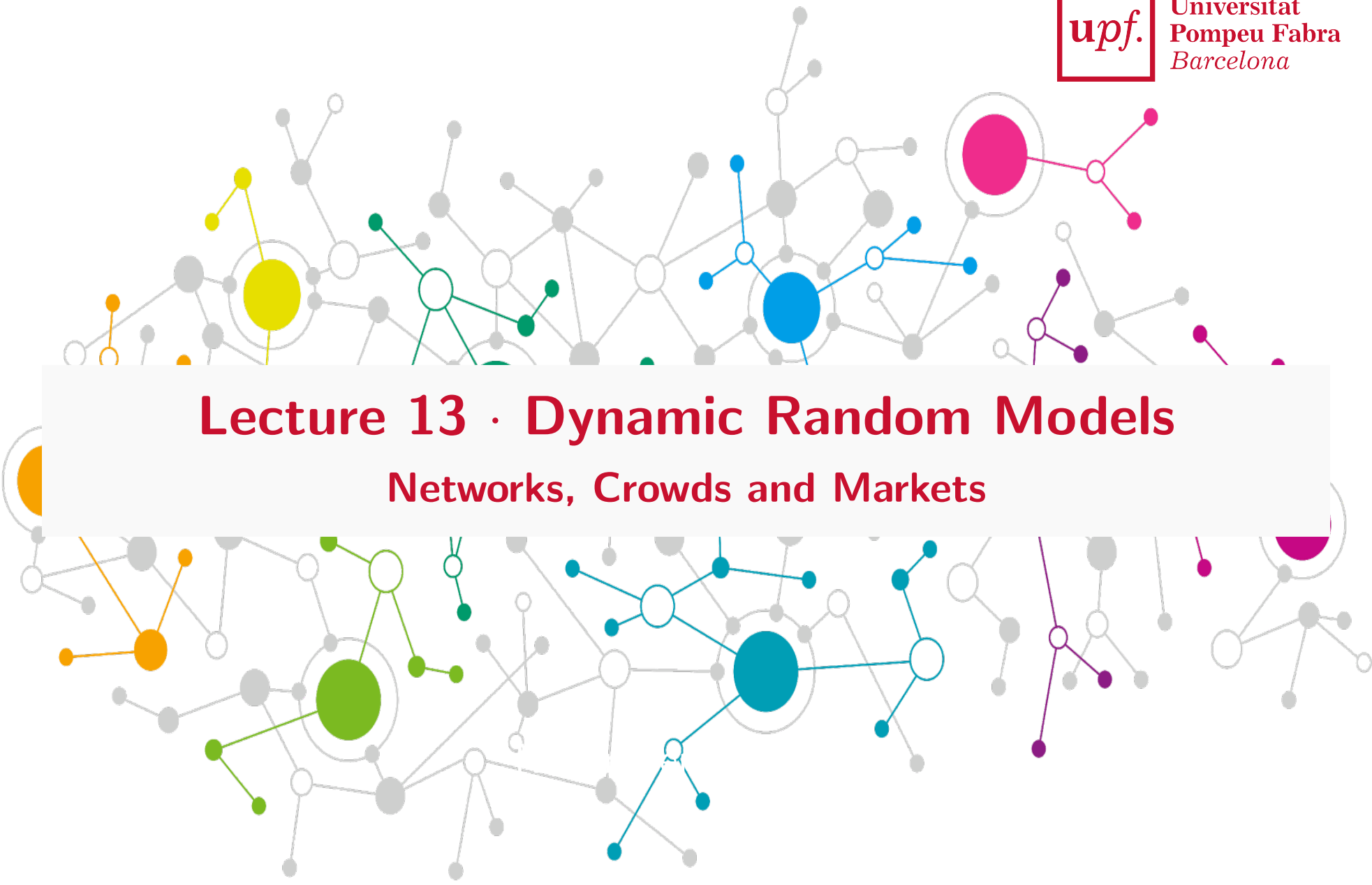


Universitat Pompeu Fabra Barcelona

Lecture 13 · Dynamic Random Models

Networks, Crowds and Markets

Quick recap

Erdős-Renyi model is a simple baseline model but it has problems:

- Not a good generative model for realistic networks.
- Degree distribution highly contrated around the mean.
- No community structure.

One problem with this model is that each node/edge is treated equally.

We introduced some static models giving heterogeneity in edges.

- Exponential Random Graph Models: e.g. p_2-model.
- Latent space model.

Today we study the preferential attachment and configuration models.

- Generating networks with arbitrary degree distribution

Configuration model

The Configuration Model

Goal: Generate graph with a given degree sequence $\{k_1, \ldots, k_n\}$.

Algorithm:

1. Give each node i exactly k_i *stubs* (half–edges).
2. Randomly pair all $2L = \sum_i k_i$ stubs to form L edges.
3. Optionally discard self–loops or multi–edges for a simple graph.

Key property:

- Every network with the same degrees has equal probability.

Expected adjacency:

$$E[A_{ij}] = \frac{k_i k_j}{2L-1} \approx \frac{k_i k_j}{2L}.$$

Configuration Model: intuition and limitations

Intuition:

- Each node keeps the degree, but partners are chosen uniformly at random.
- This gives a uniform distribution on graphs with given degree sequence.
- $E[A_{ij}] \propto k_i k_j$: nodes with many stubs have higher expected connectivity, even without any preference or dynamics.

Limitations:

- Can create self–loops or parallel edges (rare for large n).
- Produces no community structure or clustering.

A static, structureless baseline for networks with given degree sequence.

Preferential attachment

From Static to Growing Models

All previous models assumed a fixed number of nodes and edges.

But real networks *grow* over time: new users, new webpages, new firms.

Preferential attachment: New node attaches to existing node v with probability proportional to $\deg(v)$.

- "Rich get richer" $\rightarrow$ hubs emerge.

From Static to Growing Models

All previous models assumed a fixed number of nodes and edges.

But real networks *grow* over time: new users, new webpages, new firms.

Preferential attachment: New node attaches to existing node v with probability proportional to $\deg(v)$.

- "Rich get richer" $\rightarrow$ hubs emerge.

Result: degree distribution follows a *power law*.

- Few very large hubs.
- Many low-degree nodes.
- Matches data: web, citation networks, finance.

Preferential Attachment: Formal Definition

We construct a growing sequence of graphs $G_m, G_{m+1}, G_{m+2}, \ldots$.

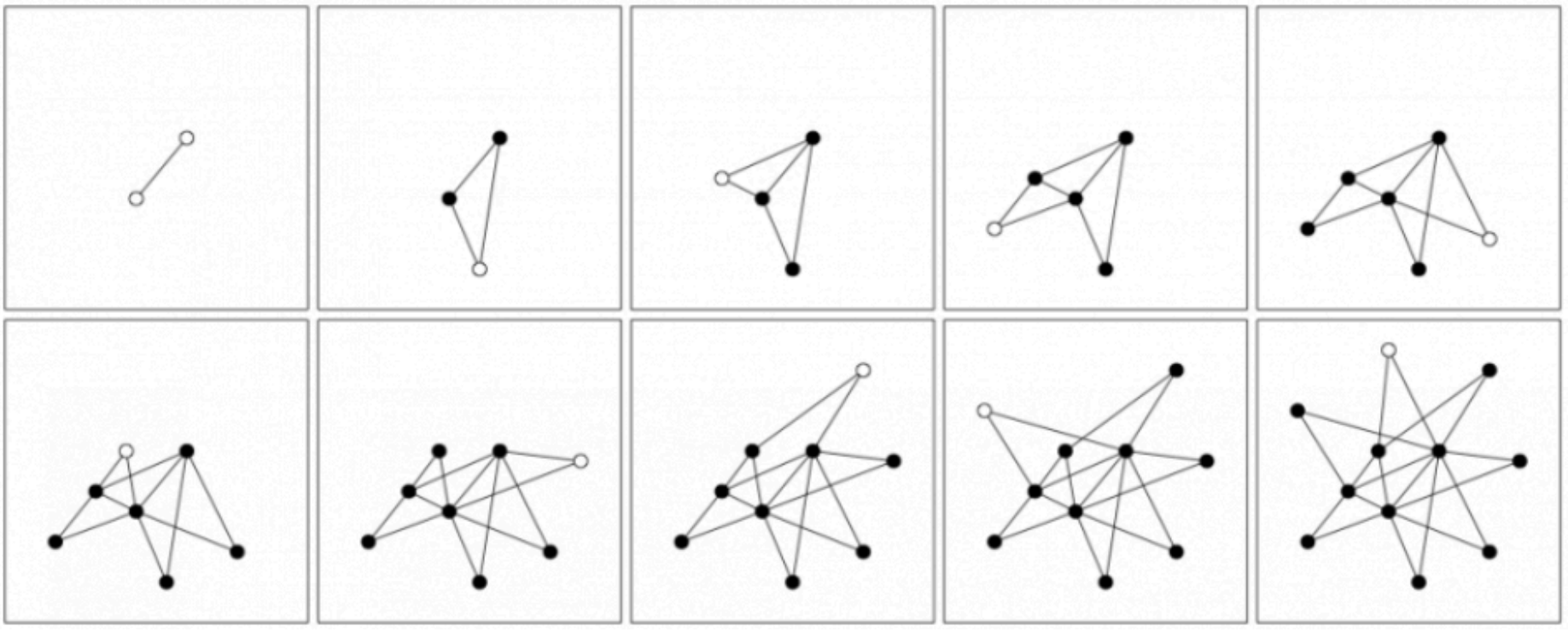
1. **Initialization:** Start from a complete graph G_m on m nodes (so each node initially has degree $m-1$).
2. **Growth rule:** For each step $t = m+1, m+2, \ldots$:
   - Add a new node v_t and m edges sticking out of it.
   - Connect each edge to a node u with probability

$$\mathbb{P}(v_t \to u) = \frac{\deg(u, t-1)}{\sum_w \deg(w, t-1)}.$$

   Thus, high-degree nodes are more likely to receive new links.

This process defines the **Barabási–Albert (BA) model**.

Evolution of the Barabási-Albert model



Expected degree growth in the BA model

If t_u is the time when u appears, we can show

$$\mathbb{E}[d_t] \approx m\sqrt{\frac{t}{t_u}}.$$

Derivation: At time $t \geq m$, the network has $L_t = \binom{m}{2} + m(t-m)$ edges. Up to constants depending only on m, we may write $L_t \approx mt$ (think large t).

Expected degree growth in the BA model

If t_u is the time when u appears, we can show

$$\mathbb{E}[d_t] \approx m\sqrt{\frac{t}{t_u}}.$$

Derivation: At time $t \geq m$, the network has $L_t = \binom{m}{2} + m(t-m)$ edges. Up to constants depending only on m, we may write $L_t \approx mt$ (think large t).

Fix a node u, $d_t := \deg(u, t)$. When a new node arrives, it creates m new edges, each connecting to an existing node with probability proportional to its degree:

$$\mathbb{P}(\text{edge connects to } u) = \frac{d_t}{\sum_v \deg(v,t)} = \frac{d_t}{2L_t} \approx \frac{d_t}{2mt}.$$

Expected degree growth in the BA model

If t_u is the time when u appears, we can show

$$\mathbb{E}[d_t] \approx m\sqrt{\frac{t}{t_u}}.$$

Derivation: At time $t \geq m$, the network has $L_t = \binom{m}{2} + m(t-m)$ edges. Up to constants depending only on m, we may write $L_t \approx mt$ (think large t).

Fix a node u, $d_t := \deg(u,t)$. When a new node arrives, it creates m new edges, each connecting to an existing node with probability proportional to its degree:

$$\mathbb{P}(\text{edge connects to } u) = \frac{d_t}{\sum_v \deg(v,t)} = \frac{d_t}{2L_t} \approx \frac{d_t}{2mt}.$$

Expected increment:

$$\mathbb{E}[d_{t+1} - d_t \mid d_t] \approx m \cdot \frac{d_t}{2mt} = \frac{d_t}{2t}.$$

Expected degree growth in the BA model

If t_u is the time when u appears, we can show

$$\mathbb{E}[d_t] \approx m\sqrt{\frac{t}{t_u}}.$$

Derivation: At time $t \geq m$, the network has $L_t = \binom{m}{2} + m(t-m)$ edges. Up to constants depending only on m, we may write $L_t \approx mt$ (think large t).

Fix a node u, $d_t := \deg(u,t)$. When a new node arrives, it creates m new edges, each connecting to an existing node with probability proportional to its degree:

$$\mathbb{P}(\text{edge connects to } u) = \frac{d_t}{\sum_v \deg(v,t)} = \frac{d_t}{2L_t} \approx \frac{d_t}{2mt}.$$

Expected increment:

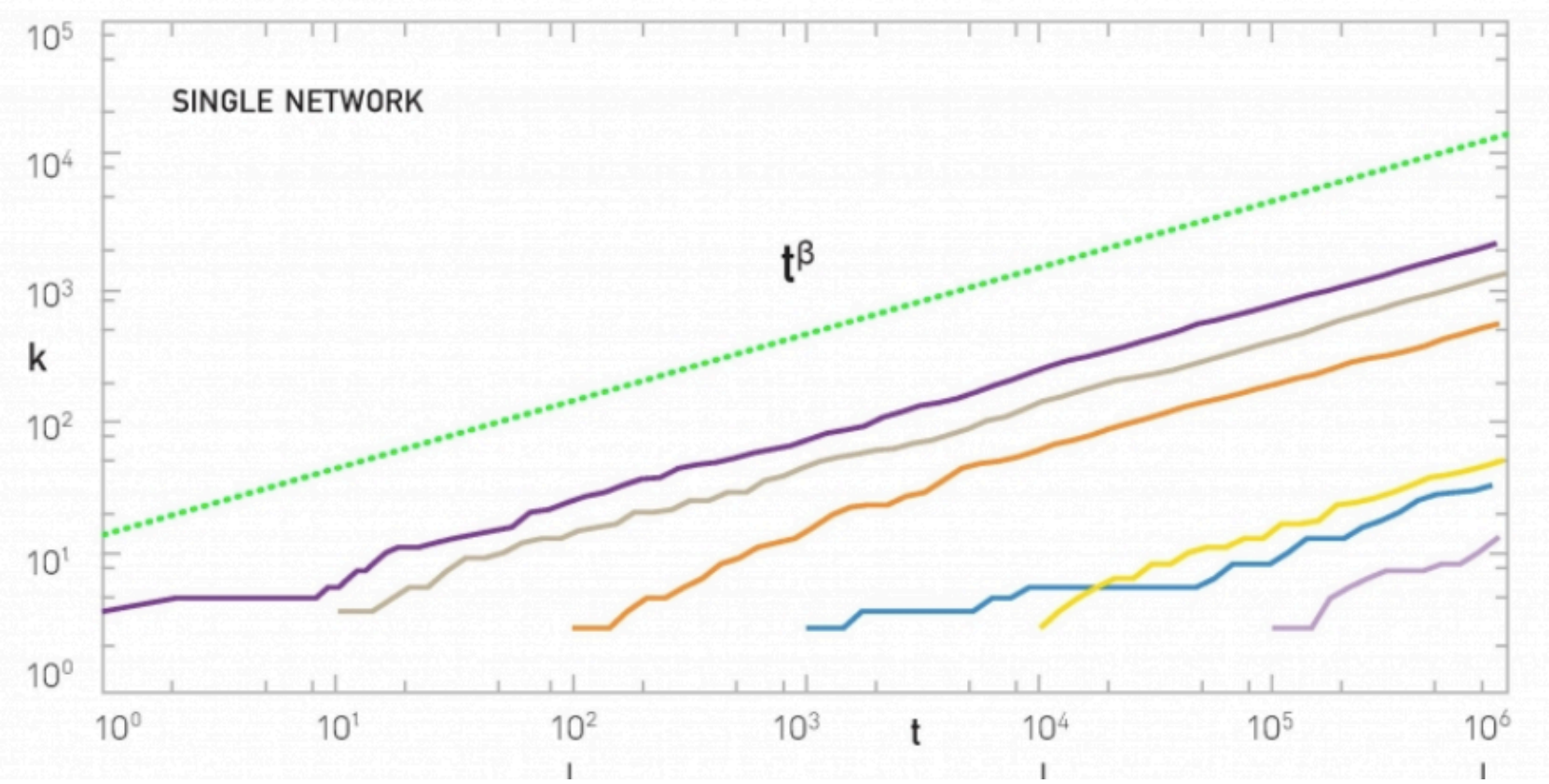
$$\mathbb{E}[d_{t+1} - d_t \mid d_t] \approx m \cdot \frac{d_t}{2mt} = \frac{d_t}{2t}.$$

This gives a recursion that gives the expected degree growth:

$$\mathbb{E}[d_{t+1}] \approx \mathbb{E}[d_t]\Big(1 + \frac{1}{2t}\Big), \qquad t \geq m.$$

Evolution of the degree

Simulated degrees of a few nodes in the log-log scale:



Quick comparison (who becomes a hub?)

Recall: $\mathbb{E}[d_t] \approx m\sqrt{\frac{t}{t_u}}$.

Two nodes joined at $t_u = 10$ and $t_v = 100$. After $t = 1000$ with $m = 3$:

$$\frac{\deg(u, 1000)}{\deg(v, 1000)} = \sqrt{\frac{1000/10}{1000/100}} = \sqrt{10} \approx 3.16,$$

$$\deg(u, 1000) = 3\sqrt{100} = 30, \qquad \deg(v, 1000) = 3\sqrt{10} \approx 9.48.$$

Earlier arrival systematically advantages degree.

The ratio of expected degrees depends on t_u, t_v but not on m.

Heuristic derivation of the degree distribution

From the recursion we found:

$$\mathbb{E}[\deg(u,t)] \;\approx\; m\left(\frac{t}{t_u}\right)^{1/2}.$$

To find the degree distribution at time t, note that

$$\deg(u,t) \approx m\left(\frac{t}{t_u}\right)^{1/2} \quad\Longleftrightarrow\quad t_u \approx t\,\frac{m^2}{\deg(u,t)^2}.$$

Heuristic derivation of the degree distribution

From the recursion we found:

$$\mathbb{E}[\deg(u,t)] \;\approx\; m\left(\frac{t}{t_u}\right)^{1/2}.$$

To find the degree distribution at time t, note that

$$\deg(u,t) \approx m\left(\frac{t}{t_u}\right)^{1/2} \quad\Longleftrightarrow\quad t_u \approx t\,\frac{m^2}{\deg(u,t)^2}.$$

Since arrival times t_u are roughly *uniform* on $\{1,2,\ldots,t\}$, we can compute

$$\mathbb{P}(\deg(u,t)\geq k) \;\approx\; \mathbb{P}\left(t_u \leq t\,\frac{m^2}{k^2}\right) \;\approx\; \frac{m^2}{k^2}.$$

Asymptotic tail

The prob. that a node has degree $\geq k$ decreases quadratically in k:

$$\mathbb{P}(\deg \geq k) \;\propto\; k^{-2} \quad\Longrightarrow\quad p_k = \mathbb{P}(\deg = k) \;\propto\; k^{-3}.$$

(at least for large k)

Hence: the Barabási–Albert model produces a *power-law* degree distribution with

$$\boxed{\gamma = 3.}$$

Asymptotic tail

The prob. that a node has degree $\geq k$ decreases quadratically in k:

$$\mathbb{P}(\deg \geq k) \;\propto\; k^{-2} \quad\Longrightarrow\quad p_k = \mathbb{P}(\deg = k) \;\propto\; k^{-3}.$$

(at least for large k)

Hence: the Barabási–Albert model produces a *power-law* degree distribution with

$$\boxed{\gamma = 3.}$$

Takeaways

- The tail exponent $\gamma = 3$ is universal for the BA model (all m).
- This formula matches simulations closely.

Exercise (preferential attachment probabilities)

Consider the preferential attachment model with $m = 1$. Given the degree multiset $\{1, 1, 1, 1, 2, 3, 3, 4, 4, 5, 5, 8\}$, let a new node add *one* link using BA attachment $\Pr\{u\} = \deg(u)/(2L)$.

a) Probability it attaches to the highest–degree node:

$$\Pr\{\text{choose } k = 8\} \;=\; \frac{8}{\sum \deg} \;=\; \frac{8}{38}.$$

b) Probability it attaches to a node of degree 1: there are four such nodes, each with probability $1/38$: $4 \times \frac{1}{38} = \frac{4}{38}$.

What is coming next

1. What do we mean by a community?
2. Zachary's Karate Club
3. Hypotheses and definitions (H1–H4)
4. The Girvan–Newman algorithm
5. The Stochastic Block Model (SBM)

Communities in graphs

What are communities?

Definition (Informal)

Groups of nodes with more connections inside than outside.

Examples:

- Social networks: circles of friends, political communities.
- Scientific collaboration: fields or subdisciplines.
- Biology: protein complexes in interaction networks.
- Infrastructure: airline networks with hubs and regional groups.

There is no single formal definition.

Communities in Economics

- **Trade Blocs:** Countries cluster into EU, NAFTA, ASEAN.
- **Political Polarization:** Twitter users cluster into left vs. right.
- **Firms:** Industry supply chains reveal modular communities.

Detecting communities = identifying hidden structure in markets.

A famous example: Communication structure in Belgium.

Zachary's Karate Club

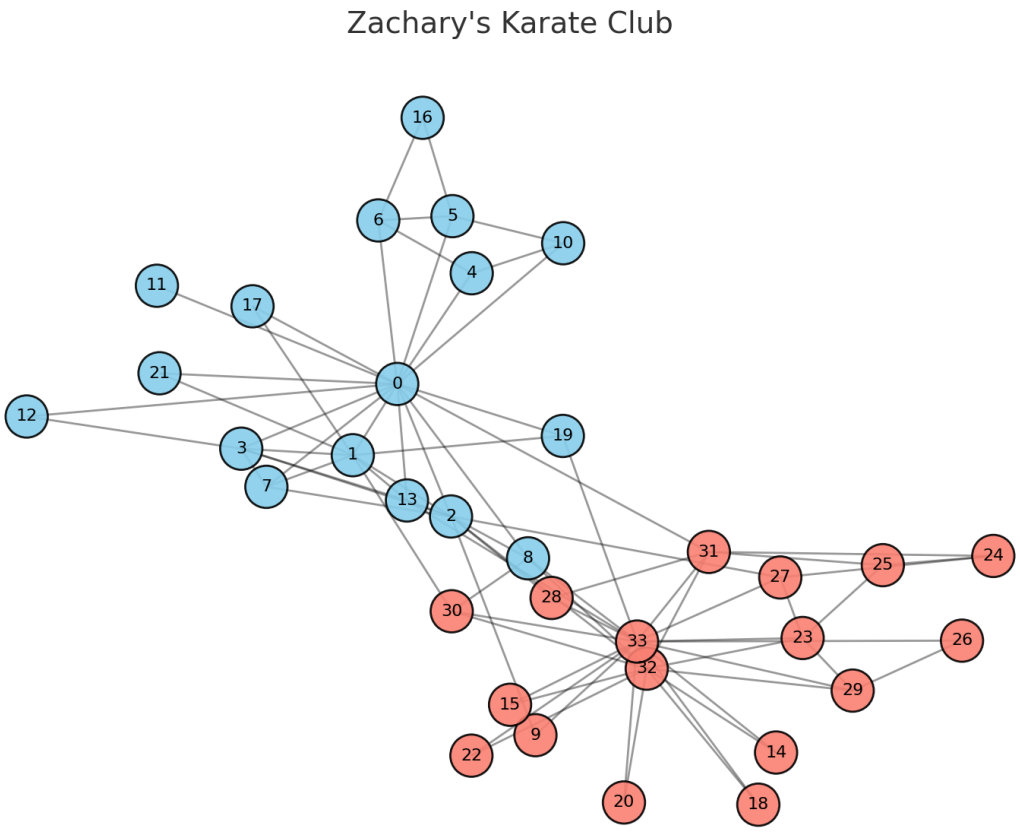
This is the first famous example of community structure in a network.

This network was initially analysed by Wayne W. Zachary, 1977.

This became the most classic network for community analysis.

- 34 members of a karate club, edges = friendships outside the club.
- Conflict between instructor ("Mr. Hi") and administrator ("John A") led to a split of the club.
- Zachary's analysis correctly predicted all but one member's side.

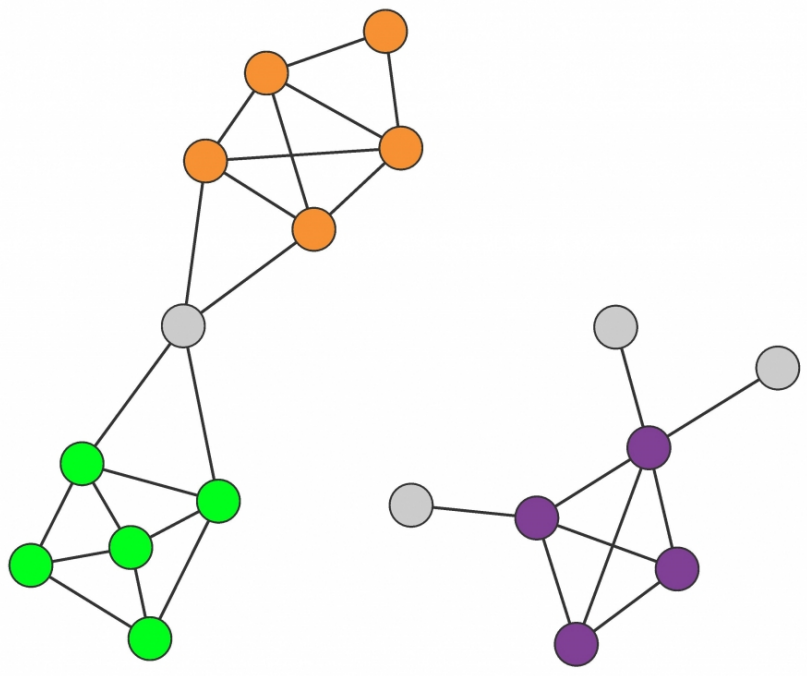
Karate Club Network



Communities: Defining principles

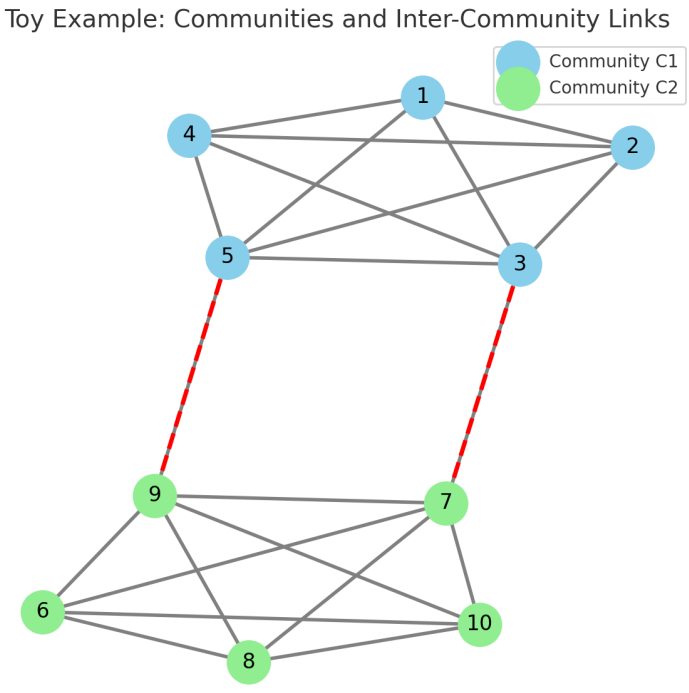
Principle 1 – Fundamental

A network's community structure is encoded in its wiring diagram. That is, communities can in principle be discovered by looking only at the graph structure.



Principle 2 – Connectedness & Density

A **community** should be a connected subgraph. Links inside a community should be denser than links going outside.



Strong vs Weak Communities

$$\deg_C(v) = \#\{\text{edges from } v \text{ to nodes in } C\}$$

$$\deg_{\overline{C}}(v) = \#\{\text{edges from } v \text{ to nodes outside } C\}.$$

Strong vs Weak Communities

$$\deg_C(v) = \#\{\text{edges from } v \text{ to nodes in } C\}$$

$$\deg_{\overline{C}}(v) = \#\{\text{edges from } v \text{ to nodes outside } C\}.$$

Strong community (restrictive): every node $v \in C$ satisfies

$$\deg_C(v) > \deg_{\overline{C}}(v).$$

Strong vs Weak Communities

$$\deg_C(v) = \#\{\text{edges from } v \text{ to nodes in } C\}$$
$$\deg_{\overline{C}}(v) = \#\{\text{edges from } v \text{ to nodes outside } C\}.$$

Strong community (restrictive): every node $v \in C$ satisfies

$$\deg_C(v) > \deg_{\overline{C}}(v).$$

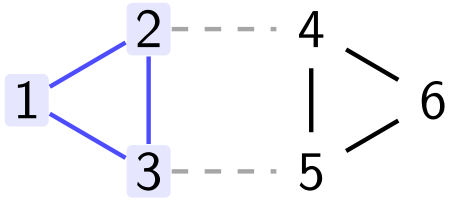
Weak community: total internal degree exceeds external degree:

$$\sum_{v \in C} \deg_C(v) \; > \; \sum_{v \in C} \deg_{\overline{C}}(v).$$

(the average in-community degree larger than out-community degree)

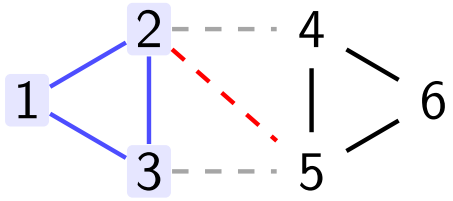
Example · Strong vs Weak Community

Original graph (strong)



$C = \{1, 2, 3\}$ $\overline{C} = \{4, 5, 6\}$

After adding edge $(2, 5)$



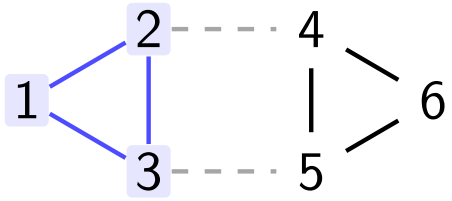
$C = \{1, 2, 3\}$ $\overline{C} = \{4, 5, 6\}$

v	$\deg_C(v)$	$\deg_{\overline{C}}(v)$	Strong?
1	2	0	✓
2	2	1	✓
3	2	1	✓

$\Rightarrow$ C is a strong community.

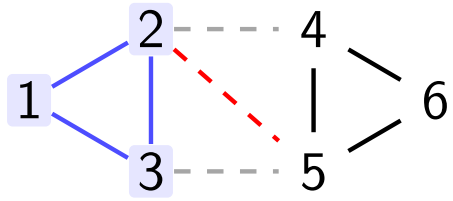
Example · Strong vs Weak Community

Original graph (strong)



$C = \{1, 2, 3\}$ $\overline{C} = \{4, 5, 6\}$

After adding edge $(2, 5)$



$C = \{1, 2, 3\}$ $\overline{C} = \{4, 5, 6\}$

v	$\deg_C(v)$	$\deg_{\overline{C}}(v)$	Strong?
1	2	0	✓
2	2	1	✓
3	2	1	✓

$\Rightarrow$ C is a strong community.

Now add edge $(2, 5)$: $\deg_C(2) = 2$, $\deg_{\overline{C}}(2) = 2 \Rightarrow$ not strong. But

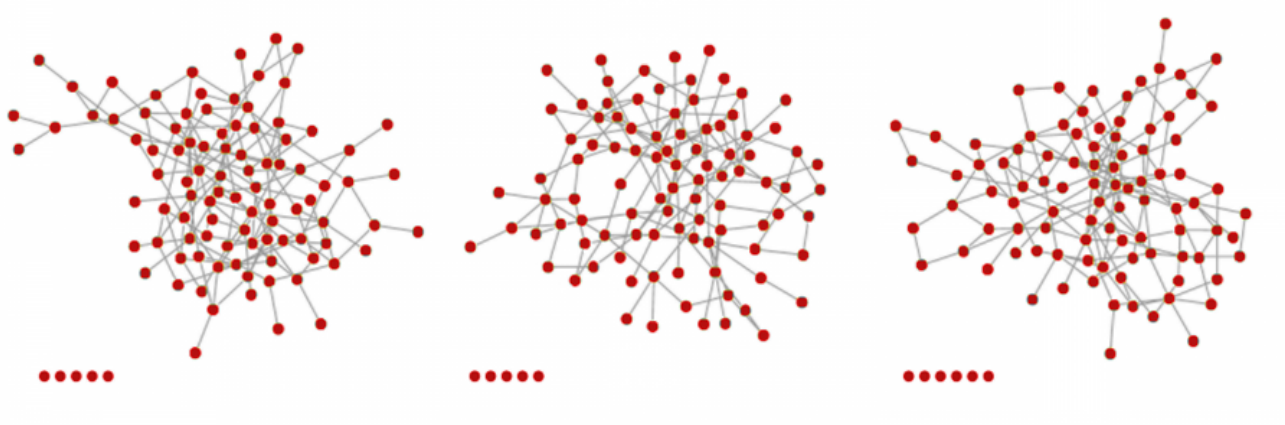
$$\sum_{v \in C} \deg_C(v) = 6, \qquad \sum_{v \in C} \deg_{\overline{C}}(v) = 4,$$

so C is still a weak community.

Principle 3 – Random Baseline

Erdős–Rényi graphs do not have meaningful community structure.

- Communities are detected when the observed structure **deviates significantly from total randomness**.
- Useful for benchmarking algorithms for community detection.



Principle 4 - Modularity Maximization

Connections *within communities* denser than expected by random chance.

Measured by modularity (Newman-Girvan 2004):

$$M = \frac{1}{2L} \sum_{i,j} \left(A_{ij} - \frac{k_i k_j}{2L} \right) \delta(c_i, c_j),$$

where

- A_{ij} - adjacency matrix (1 if edge i–j exists, else 0),
- k_i - degree of node i, $2L = \sum_i k_i$ (total edge ends),
- $\delta(c_i, c_j) = 1$ if i, j lie in the same community, 0 otherwise.

M compares the *observed* edges (A_{ij}) to what we would *expect at random* ($k_i k_j / 2L$), summing this excess over all within-community pairs.

Meaning of the expected term in modularity

In the random baseline (the *configuration model*), each node i has k_i edge ends (stubs), and all $2L$ stubs are paired uniformly at random.

Expected number of edges between i and j:

$$E[A_{ij}] = \frac{k_i k_j}{2L}.$$

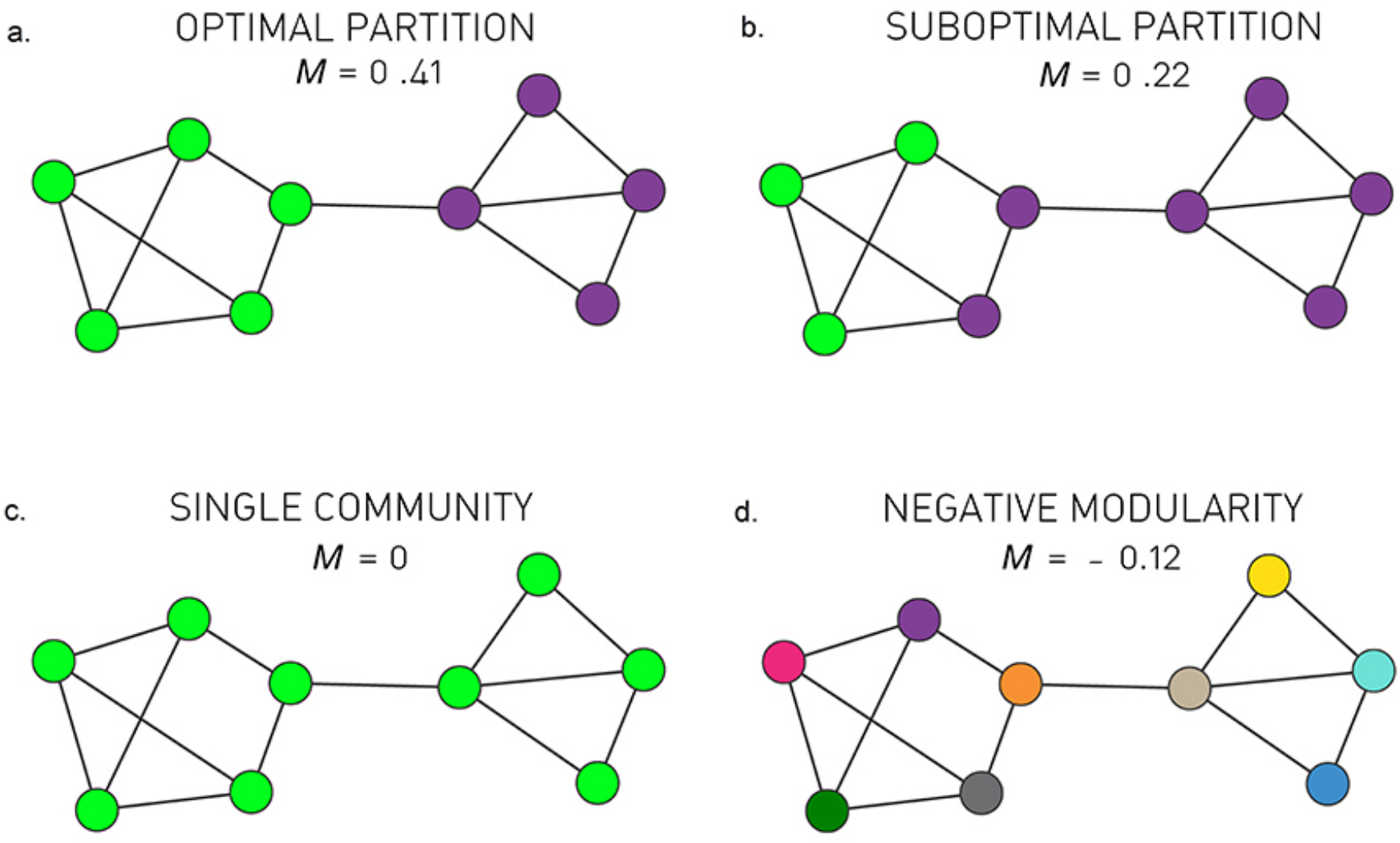
- This is not a probability but an *expected edge count* in that random multigraph.
- It can exceed 1 if both nodes have very high degree.
- It defines the "no community structure" baseline against which modularity compares the observed edges.

Hence:

$$M = \frac{1}{2L} \sum_{i,j} \Big(A_{ij} - E[A_{ij}] \Big) \delta(c_i, c_j)$$

measures how much more connected communities are than in that random reference network.

The Girvan-Newman Algorithm



Identifying communities

Community detection via hierarchical clustering

Find a computationally efficient community detection procedure.

Let $S = [\delta_{ij}]$ be a similarity matrix:

- S symmetric;
- $s_{ij} \geq 0$ for all $i \neq j$.

If i, j are "close", s_{ij} is higher.

- Build a *similarity matrix* s_{ij} from the network.
- Iteratively group (or split) nodes using s_{ij}.
- Output is a dendrogram; cutting it gives a partition.

Ravasz agglomerative algorithm

Let $B_i := \{j : d(i,j) \leq 1\}$. Let A be the adjacency matrix.

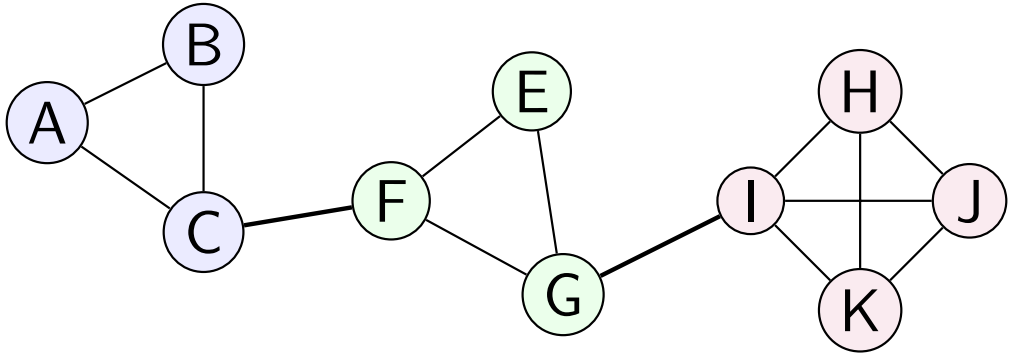
We define node similarity using the *topological overlap*:

$$s_{ij} = \frac{|B_i \cap B_j|}{\min\{|B_i|, |B_j|\}} \in [0,1], \qquad (i \neq j).$$

- $s_{ij} = 0$ iff i, j are not connected and they share no neighbors.
- $s_{ij} = 1$ iff $B_i \subseteq B_j$ or $B_j \subseteq B_j$.

Many authors subtract A_{ij} in both the nominator and denominator to slightly down-weight the direct edge when i and j are linked ($\tilde{s}_{ij}$).

Toy network and local overlap (schematic)



Pair (i,j)	$\lvert B_i\rvert$	$\lvert B_j\rvert$	$\lvert B_i \cap B_j\rvert$	s_{ij}	$\tilde{s}_{ij}$	A_{ij}
A–B	3	3	3	1.00	1.00	1
A–C	3	4	3	1.00	1.00	1
E–F	3	4	3	1.00	1.00	1
F–G	4	4	3	0.75	0.67	1
H–I	4	5	4	1.00	1.00	1
H–K	4	4	4	1.00	1.00	1
C–F (bridge)	4	4	2	0.50	0.33	1
G–I (bridge)	4	5	2	0.50	0.25	1
A–E	3	3	0	0.00	0.00	0
A–F	3	4	1	0.33	0.33	0

Ravasz agglomerative algorithm: hierarchical clustering

We apply standard hierarchical clustering to S.

Algorithm.

1. Compute the similarity matrix $s_{ij} = \frac{|B_i \cap B_j|}{\min(|B_i|,|B_j|)}$ for i, j.
2. Treat each node as a separate cluster.
3. Find the two clusters with the largest average pairwise s_{ij}.
4. Merge them into a new cluster. Compute similarities between this new cluster and every other cluster using a chosen linkage rule:

$$\delta_{(AB),C} = \begin{cases} \max_{i \in A, j \in B} s_{ij}, & \text{(single linkage)}, \\ \min_{i \in A, j \in B} s_{ij}, & \text{(complete linkage)}, \\ \text{average}_{i \in A, j \in B} s_{ij}, & \text{(average linkage)}. \end{cases}$$

5. Repeat Step 3–4 until all nodes are merged into one cluster.

Output. The sequence of merges defines a *dendrogram*. Cutting it at a chosen similarity threshold yields the community partition.

Constructing dendrogram (single linkage)

Single linkage between clusters A, B:

$$s(A, B) = \max_{i \in A,\, j \in B} s_{ij}.$$

Within-block merges (heights = 1).

$$\{A\} + \{B\} \xrightarrow{1} \{AB\}, \quad \{AB\} + \{C\} \xrightarrow{1} \{ABC\};$$

$$\{E\} + \{F\} \xrightarrow{1} \{EF\}, \quad \{EF\} + \{G\} \xrightarrow{\max(1,0.75)=1} \{EFG\};$$

$$\{H\} + \{I\} \xrightarrow{1} \{HI\}, \ \{J\} + \{K\} \xrightarrow{1} \{JK\}, \ \{HI\} + \{JK\} \xrightarrow{1} \{HIJK\}.$$

Between-block similarities (single linkage).

$$s(\{ABC\}, \{EFG\}) = \max s_{ij} = s_{CF} = \mathbf{0.5},$$

$$s(\{EFG\}, \{HIJK\}) = \max s_{ij} = s_{GI} = \mathbf{0.5},$$

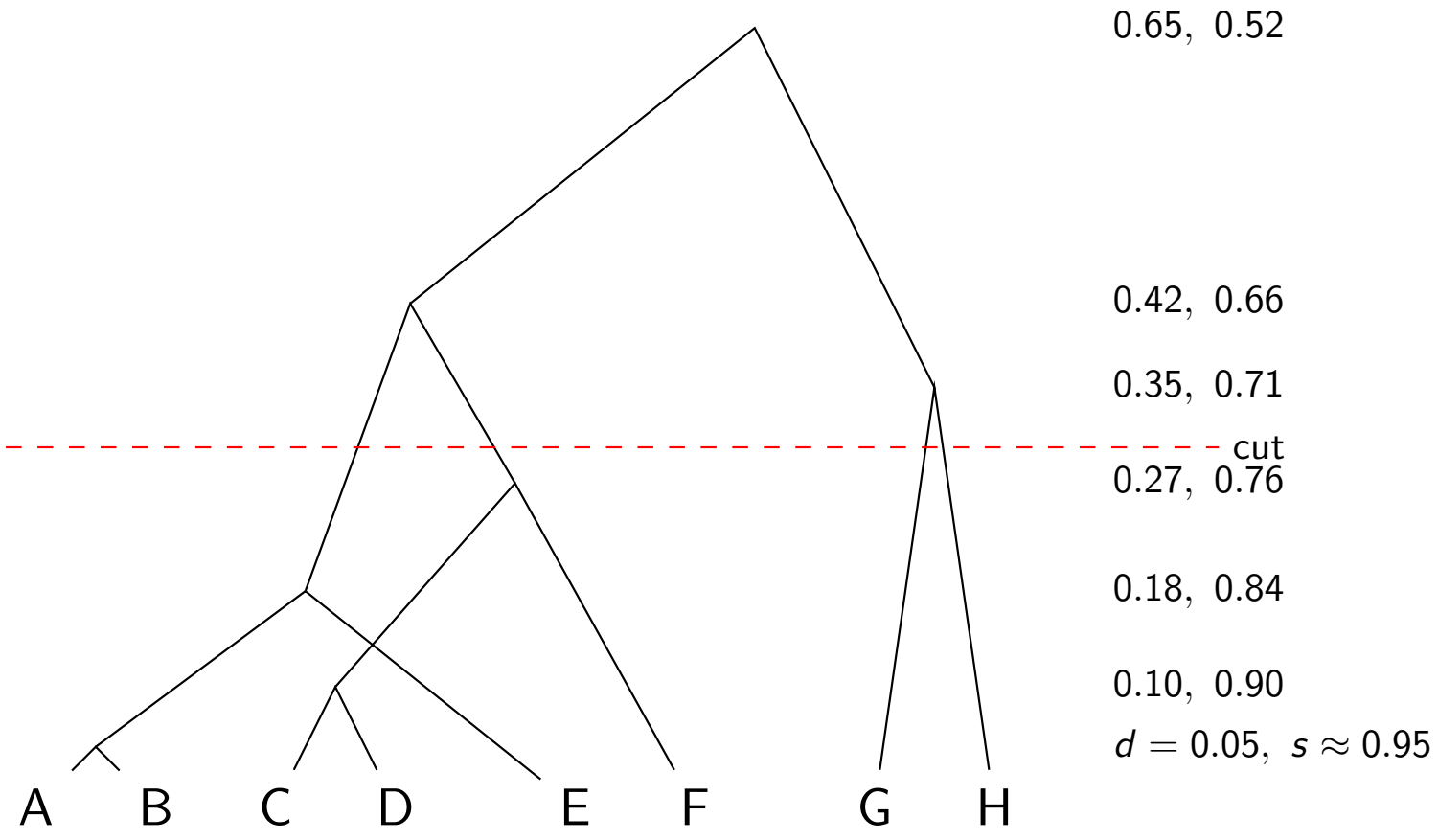
$$s(\{ABC\}, \{HIJK\}) = \mathbf{0}.$$

Merge order (single linkage).

$$\{ABC\} \xrightarrow[0.5]{} \{EFG\} \longrightarrow \{ABC{+}EFG\} \quad \text{then} \quad \{ABC{+}EFG\} \xrightarrow[0.5]{} \{HIJK\}.$$

Dendrogram (single linkage; generic example)

Heights show dissimilarity $d = -\log s$ (so $s = e^{-d}$).



Larger d means weaker linkage.

Divisive community detection: Girvan–Newman algorithm

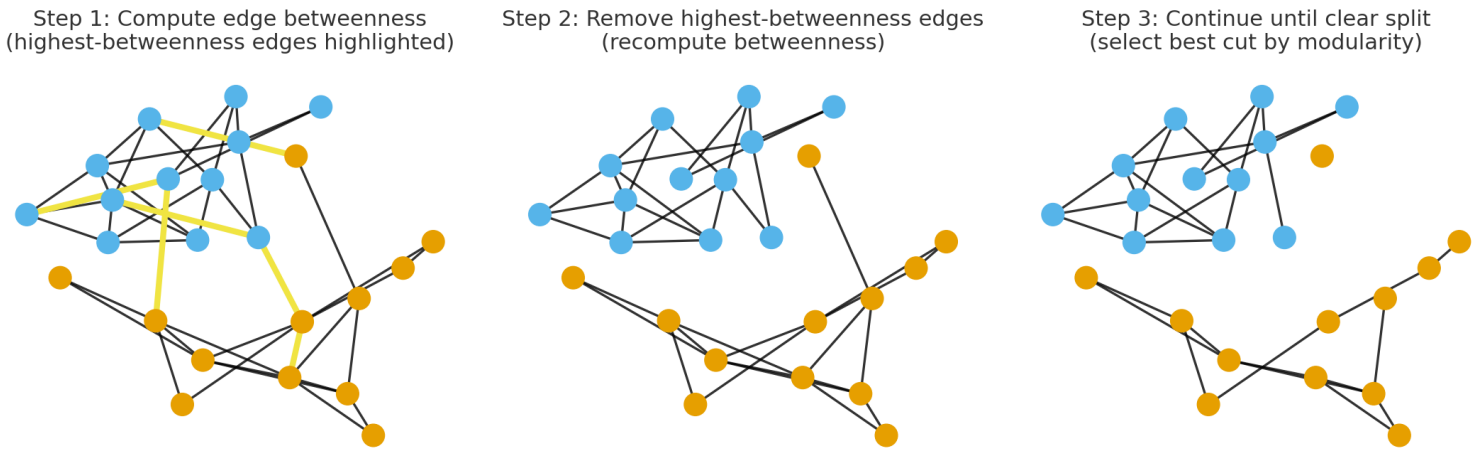
Goal: detect communities by removing *bridges* between them.

Key concept: edge betweenness.

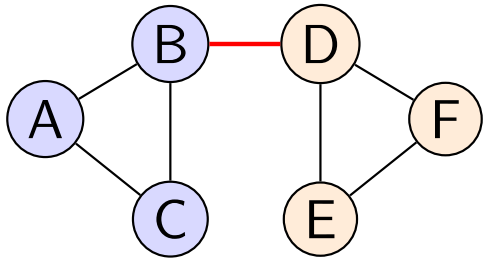
- For every edge, count **shortest paths** between all node pairs that use it.
- Bridges between communities lie on many such paths.

Algorithm (top–down / divisive):

1. Compute betweenness for all edges.
2. Remove edge with the highest value.
3. Recompute betweenness on the updated graph.
4. Repeat until the graph splits.



Girvan–Newman: toy example (manual computation)



Step 1. Compute edge betweenness

- Shortest paths inside each triangle use only local edges.
- All shortest paths between left and right triangles must go through (B, D).
- $\Rightarrow$ edge (B, D) has the highest betweenness.

Step 2. Remove (B, D)
Graph splits into two dense components:

$$\{A, B, C\}, \qquad \{D, E, F\}.$$

Communities recovered!

Girvan–Newman: selecting the best split

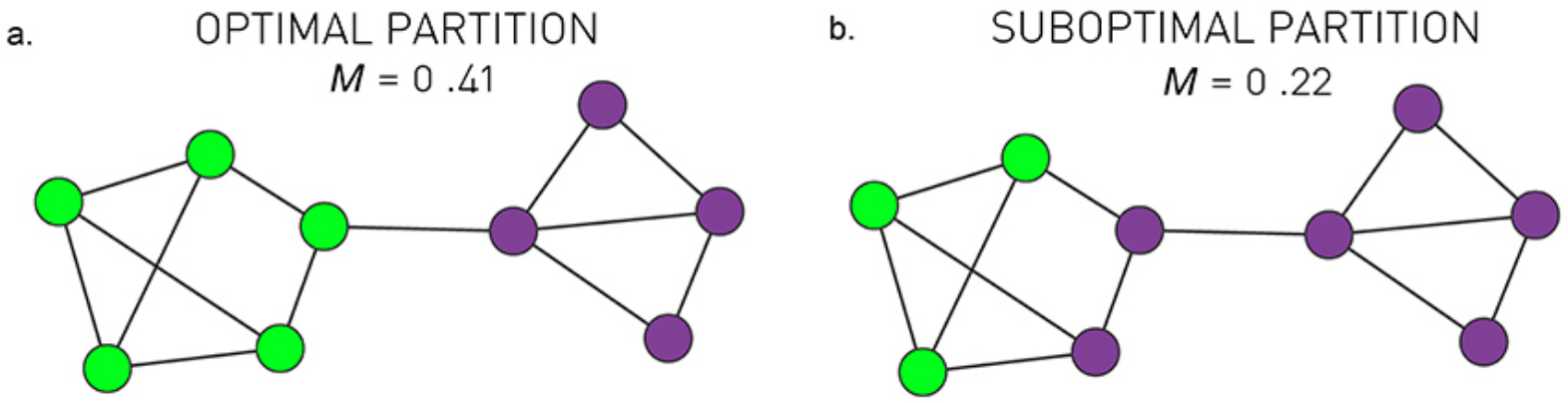
Which split is best?

Use modularity M:

$$M = \frac{1}{2m} \sum_{ij} \left(A_{ij} - \frac{k_i k_j}{2m} \right) \delta(c_i, c_j),$$

where m is #edges, A_{ij} adjacency, k_i degree, $\delta(c_i, c_j) = 1$ if i, j are in the same community.

Interpretation: M measures excess internal edges relative to a random network with the same degrees.



Agglomerative vs. Divisive (summary)

	Agglomerative (Ravasz)	Divisive (Girvan–Newman)
Direction	Bottom–up merges	Top–down edge removals
Input	Node similarity s_{ij}	Edge centrality s_{ij}
Linkage	Single / complete / average	Not applicable
Output	Dendrogram of merges	Dendrogram of splits
Cost	$O(N^2)$ typical	$O(LN)$–$O(N^3)$ (implementation)

Both yield a dendrogram; choose the cut to get communities.

Stochastic Block Model

The Stochastic Block Model (SBM)

The SBM provides a simple generative model for networks with community structure.

Definition (Stochastic Block Model)

- N nodes, each assigned to one of K groups: $g_i \in \{1, \ldots, K\}$.
- Edge between i and j appears independently with probability

$$\Pr[(i,j) \in E] = p_{g_i, g_j}.$$

- The $K \times K$ matrix $P = (p_{ab})$ specifies connection probabilities between groups.

The Stochastic Block Model (SBM)

The SBM provides a simple generative model for networks with community structure.

Definition (Stochastic Block Model)

- N nodes, each assigned to one of K groups: $g_i \in \{1, \ldots, K\}$.
- Edge between i and j appears independently with probability

$$\Pr[(i,j) \in E] = p_{g_i, g_j}.$$

- The $K \times K$ matrix $P = (p_{ab})$ specifies connection probabilities between groups.

Applications:

- Benchmark for testing community detection algorithms.
- Statistical inference: given the observed network, estimate group labels $\{g_i\}$ and/or P.

SBM (2 groups): signal in the spectrum

Model. Two communities of sizes N_1, N_2 ($N = N_1 + N_2$).

$$\Pr[(i,j) \in E] = \begin{cases} p, & g_i = g_j, \\ q, & g_i \neq g_j. \end{cases} \qquad \text{with } p > q.$$

Let A be the adjacency matrix (random) and $B = \mathbb{E}[A]$ its expectation.

Block structure of B:

$$B = \begin{pmatrix} p\mathbf{1}_{N_1 \times N_1} & q\mathbf{1}_{N_1 \times N_2} \\ q\mathbf{1}_{N_2 \times N_1} & p\mathbf{1}_{N_2 \times N_2} \end{pmatrix},$$

a rank-2 matrix.

Key eigenvectors:

- $\mathbf{1}$ (all-ones) $\Rightarrow$ "size/degree" direction.
- Community indicator s ($s_i = +1$ if $g_i = 1$, $s_i = -1$ if $g_i = 2$).

Eigenvalues (balanced case $N_1 = N_2 = N/2$):

$$\lambda_1(B) \approx \tfrac{N}{2}\big(p + q\big),$$

$$\lambda_2(B) \approx \tfrac{N}{2}\big(p - q\big).$$

$\lambda_2(B)$ carries the community signal:

$$\underbrace{p - q}_{\text{separation}} \quad \uparrow \Rightarrow \text{ easier recovery.}$$

Idea: If we compute the leading eigenvectors of the *observed* A, they should align with those of B and reveal the partition.

Note

Spectral clustering works because A concentrates around its block-structured mean and the "community" eigenvector survives the noise when separation is large enough.

Spectral Clustering for Communities (practical)

Input: Graph $G = (V, E)$ with $|V| = N$ and number of groups K.

Matrix choice:

- *Adjacency* A for reasonably dense graphs.
- *(Normalized) Laplacian* $L_{\text{sym}} = I - D^{-1/2} A D^{-1/2}$ for sparse graphs or degree heterogeneity.

Algorithm:

1. Compute K eigenvectors:
   - top-K of A, or
   - bottom-K (smallest) of L_{sym}.
2. Stack them into $V \in \mathbb{R}^{N \times K}$ (row i = embedding of node i). (Option: row-normalize V.)
3. Run k-means on rows of V to obtain labels $\hat{g}_1, \ldots, \hat{g}_N$.

Model selection tips:

- Choose K via eigengap heuristic or cross-validation on modularity.
- Use degree-corrected variants (e.g., regularized Laplacian) if degrees are very skewed.

Hands-on: SBM + Spectral Clustering in NetworkX

```
import networkx as nx
import numpy as np
from sklearn.cluster import KMeans
from scipy.sparse.linalg import eigsh

# 1) Generate a 2-block SBM
N1, N2 = 80, 70
sizes = [N1, N2]
pin, pout = 0.12, 0.02
P = [[pin, pout], [pout, pin]]
G = nx.stochastic_block_model(sizes, P, seed=7)
A = nx.to_scipy_sparse_array(G, format="csr", dtype=float)
N = A.shape[0]

# 2) Use normalized Laplacian for robustness on sparse graphs
L = nx.normalized_laplacian_matrix(G)      # scipy sparse CSR
# take K=2 smallest eigenpairs of L (skip the zero vector if graph disconn
K = 2
vals, vecs = eigsh(L, k=K, which='SM')     # smallest magnitude eigenvalue
```

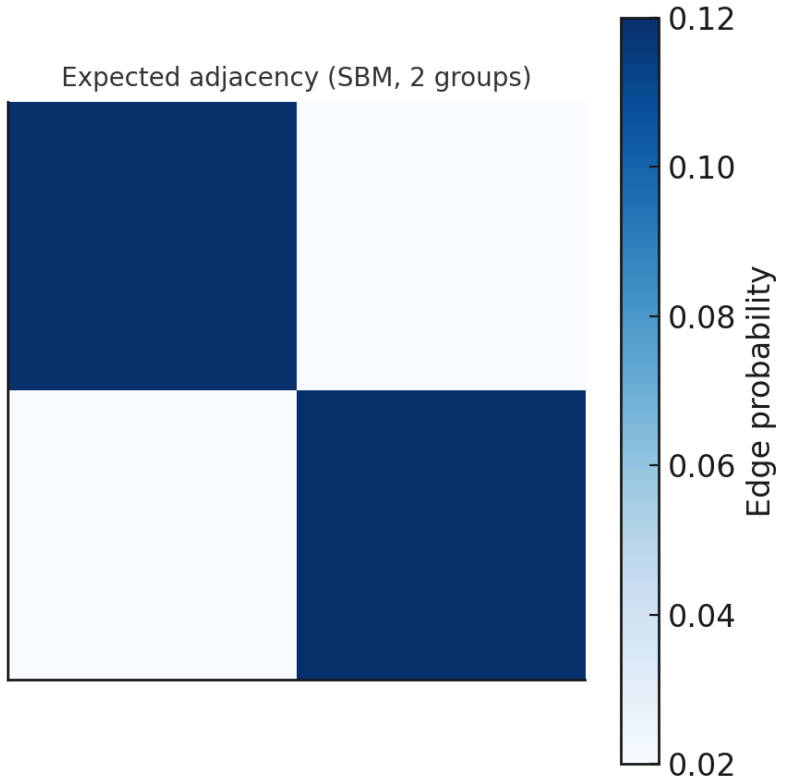
```
# 3) Row-embed nodes and k-means
X = vecs                                          # N x K embedding
X = X / (np.linalg.norm(X, axis=1, keepdims=True) + 1e-12)  # row-normaliz
labels = KMeans(n_clusters=2, n_init=20, random_state=7).fit_predict(X)

# 4) Compare to planted labels (0 for first block, 1 for second)
true = np.array([0]*N1 + [1]*N2)
acc = max((labels == true).mean(), (1-labels == true).mean())
print(f"Accuracy vs planted partition: {acc:.3f}")
```

Notes: For very sparse graphs, prefer Laplacian; for denser ones, adjacency often suffices. Replace K by eigengap-based choice when unknown.

What the eigenvectors "see": a geometric picture

- $B = \mathbb{E}[A]$ has two constant blocks.
- Its top eigenvector is roughly constant on all nodes (size/degree axis).
- Its second eigenvector is roughly $+c$ on group 1 and $-c$ on group 2.
- The observed A is a noisy version of B; eigenvectors of A wobble around those of B.



Expected adjacency B (2 blocks)

Consequence. Embedding nodes by the top eigenvectors separates communities along the "$+/-$" direction; k-means then recovers the groups.

Open Questions

- Do all networks really have communities?
- Are “communities” well-defined or context-dependent?
- Must all nodes belong to some community?
- How do communities influence dynamics (diffusion, epidemics)?