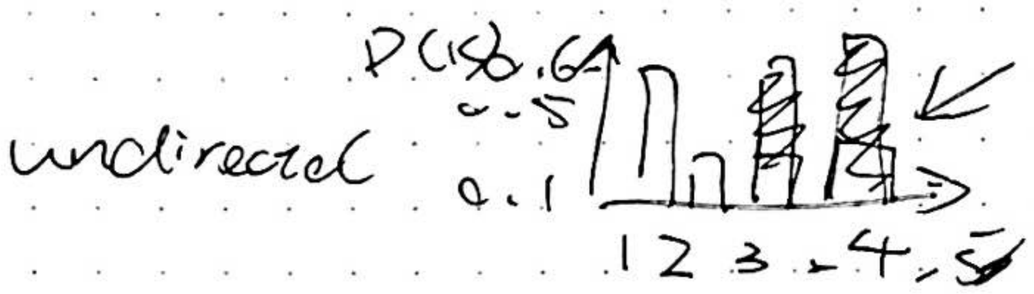
Properties of graphs.

(1) Degree distribution $P(k)$:

probability that a randomly chosen node has degree k.

N_k = # nodes with degree k.

$P(k) = N_k / N$ → normalized histogram.



directed: one for in-degree one for out degree.

(2) Paths in a graph

A path is a sequence of nodes in which each node is linked to the next one.

$P_n = \{i_0, i_1, i_2, \dots i_n\}$ $P_n = \{(i_0, i_1), (i_1, i_2), (i_2, i_3)$
$\dots (i_{n-1}, i_n)\}$

→ distance (shortest path, geodesic) between a pair of nodes: no. of edges along the shortest path connecting the nodes.

* if two nodes are unconnected, d is ∞

for directed graphs → paths need to follow the direction of arrows ⇒ distance is not symmetric

h_B - c the c - B.

→ network diameter: the maximum (shortest path) distance between any pair of nodes in a graph.

(Due this is a fragile definition as one string of nodes can skew this metric).

an alternative: average path length for a connected graph or a strongly connected direct. graph.

$$\bar{h} = \frac{1}{2E_{max}} \sum_{i,j \neq i} h_{ij}$$

- h_{ij} is the distance from node i to node j.
- E_{max} is the max. no. of edges (total no. of node pairs) $= n(n-1)/2$.

→ we compute the average only over the connected pairs of nodes. (ignore (infinite length paths)

↔ :

(3). clustering coefficient (non-directed → Undirected Graph → inspired social network)

↳ how connected are i's neighbours to each other?

node i with degree k_i

$C_i \in [0, 1]$.

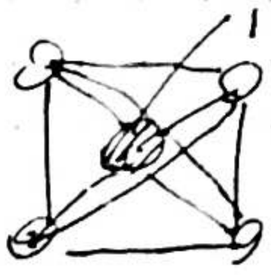
$$C_i = \frac{2e_i}{k_i(k_i-1)}$$

← note C_i is undefined for nodes with degree 0 or 1.

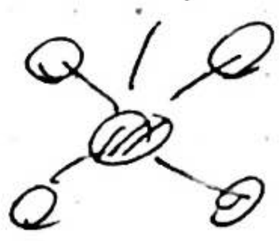
where e_i is the no. of edges between the neighbours of node i.

↑ all possible edges between neighbours

(e_i = max no of edges between the k_i neighbors.)



$C_i = 1$



$C_i \geq 0$.

average clustering coefficient:

$$C = \frac{1}{N} \sum_{i}^{N} C_i$$

(friends of mine - they tend to know each other

$C_i \rightarrow$ high).

(4.) Connectivity

Size of the largest connected component.

↳ largest set where any two ~~vertices~~ vertices can be joined by a path.

To find connected components

1. Start from random node and perform breath first search (BFS).
2. Label the nodes that BFS visits.
3. If all nodes are visited, the network is connected.

Otherwise find an unvisited node and repeat BFS.

Once we know these four metrics for a network, what next? Should we be surprised?

→ we need a model!

Simplest model of graphs.

Erdös-Renyi Random Graphs.

two variants.

- G_{np}: undirected graph on n nodes. where each edge (u,v) appears i.i.d with probability p
- G_{nm}: undirected graph with n nodes, and m edges picked uniformly at random.

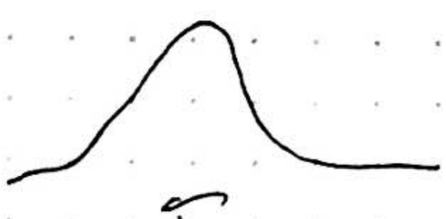
for the Gnp model we can get:

degree distribution: binomial for Gnp.

Let $P(k)$ denote fraction of nodes with degree k

$$P(k) = \binom{n-1}{k} p^k (1-p)^{n-1-k}$$

- k: select k nodes out of the rest of the nodes
- p^k: p of having k edges
- $(1-p)^{n-1-k}$: prob. of missing the rest of the $n-1-k$ edges



bell curve like Gaussian ~~Mean~~

mean $\bar{k} = p(n-1)$

variance $\sigma^2 = p(1-p)(n-1)$

$$\frac{\sigma}{\bar{k}} = \left[\frac{1-p}{p}\frac{1}{n-1}\right]^{\frac{1}{2}} = \frac{1}{(n-1)^{1/2}}$$

Π

as the network size increases, the distribution becomes increasingly narrow
↳ degree of a node is in the vicinity of $\bar{k}$.

clustering coefficient

$$C_i = \frac{2e_i}{k_i(k_i-1)}$$

e_i: no. of edges between k_i's neighbours

Edges in Gnp appear i.i.d. with p.

$$E[e_i] = p\,\frac{k_i(k_i-1)}{2}$$

Then $E[C_i] = \frac{p \cdot k_i(k-1)}{k_i(k_i-1)} = p = \frac{\bar{k}}{n-1} \approx \frac{\bar{k}}{n}$.

clustering coefficient of a random graph is small

$n \to$ Large $E[C_i] \to 0$.

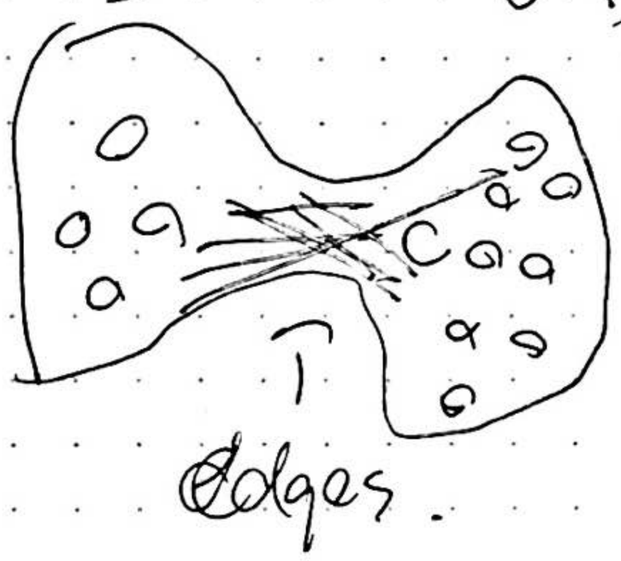
Def Expansion

Graph $G(V, E)$ has expansion α if

$\forall S \subseteq V$ # of edges leaving $S \geq \alpha \cdot \min$

$(|S|, |V \setminus S|)$

↳ $\alpha = \min_{S \subseteq V} \frac{\#\text{ edges leaving } S}{\min(|S|, |V \setminus S|)}$



Expansion

In a graph on n nodes where expansion, for all pairs of nodes, there is a path of length $O(\log n)/\alpha)$

Random graph G_{np}:

For $\log n > np > c$, $diam(G_{np}) = O(\log n / \log(np))$.

random graphs have good expansions so it takes a logarithmic no. of steps for BFS to visit all nodes.

↳ if things / properties are different ~~then~~ to the GNP model. we should be surprised.

The small-world model:

high clustering + ~~low~~ small diameter ?

↓ ↙

the 'controversy'.

consequence of expansion.

Short paths : $O(\log n)$ ← the smallest diameter we can get if we keep the degree constant. But the clustering is low.

BUT networks (like social networks) have 'local' structure:

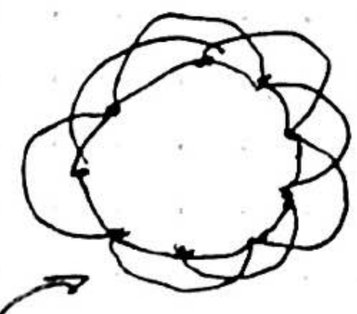
triadic closure : friend of a friend is my friend.

High clustering but diameter — is also high.

Can we have both ?

Small world model

(1) Start with a lower-dimensional regular lattice which has high clustering coefficients.



(2) Rewire introduce randomness ('shortcuts')

add/remove edges to create shortcuts to join remote parts of the lattice. For each edge - with prob. p move the other end point to a random node.

Kronecker Graph model: generating large realistic graphs.

idea: recursive graph generation.

Self-similarity: Object is similar to a part of itself: the whole has the same shape as one or more of the parts.

Kronecker product of matrixes A and B is given:

$$C = A \otimes B = \begin{pmatrix} a_{1,1}B & a_{1,2}B & \cdots & a_{1,m}B \\ a_{2,1}B & a_{2,2}B & \cdots & a_{2,m}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1}B & a_{n,2}B & & a_{n,m}B \end{pmatrix}$$

$N \times M \quad K \times L \qquad \nearrow N*K \times M*L$

each B is expanded in place.

Kronecker product of two graphs → Kronecker product of their adjacency matrices.

Kronecker graph is obtained by growing sequence of graphs by iterating the Kronecker product over the initiator matrix K_1.

$K_1^{[m]} = K_m = K_1 \otimes K_1 \otimes \cdots K_1 = K_{m-1} \otimes K_1$.

Can only generate graphs with n^2 nodes is not much a shortcoming when the graph is large enough.

Stochastic Kronecker graph : (bring stochastic aspect to the generative model)

Create $|V_1 \times N_1|$ probability matrix Θ_1. compute the k^{th} Kronecker power Θ_k. for each entry P_{uv} of Θ_k include an edge (u,v) in K_k with probability P_{uv}

$\downarrow$

this is an expensive operation when there are 1 million nodes.

$O(n^2)$ $<$ no. of nodes

$\swarrow$

the fix : exploit the sparsity of adj. matrix
↳ exploit the recursive structure of Kronecker graphs.
drop edges onto the graph one by one.

Fast Kronecker ~~graph~~ generator algorithm.
for generating directed graphs.

Insert 1 edge on graph G on $n = 2^m$ nodes:
Create normalised matrix $L_{uv} = \Theta_{uv} / (\sum_{ij} \Theta_{ij})$
for $i = 1 \dots m$.
- start with $x=0, y=0$
- pick a row/column (u,v) with prob. L_{uv}.
- Descend into quadrant (u,v) at level i of G.

$x+ = u \cdot 2^{m-i}$, $y+ = v \cdot 2^{m-i}$

Add an edge (x,y) to G

~~SNAP - snap.py~~

Motifs and structural roles in network

Subgraphs - break graphs down into non-isomorphic subparts and "count" how often they occur in the graph.

↳ network significance profile : a feature vector with values for all subgraph types.

+ : over represented

− : underrepresented.

insights : networks from same domain have similar significance profiles.

Network motifs : recurring, significant, patterns of interconnections

- help us understand how networks work
- help us predict operation of nets. and reaction of the network in a given situation.

found in many networks

more frequent than expected

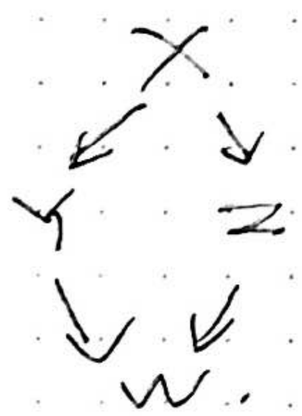
compared to randomly generated network.

small induced subgraph

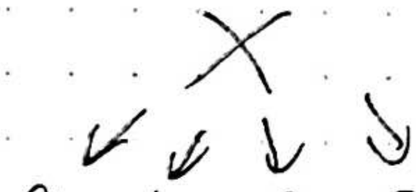
→ feed forward loops found in networks

of neurons where they neutralize biological noise

→ parallel loops : found in food webs.



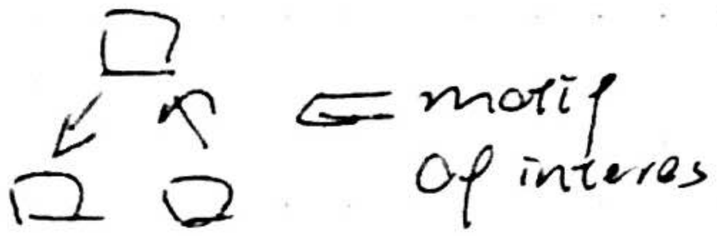
→ Single-input modules : found in gene control networks



Single input module.

Induced subgraphs

a graph formed from a subset X of the vertices of graph G and all of the edges connecting pairs of vertices in subset X.



recurrence

Allow overlapping of motifs.

significance.

subgraphs that occur in a real network much more often than in a random network have functional significance.

Z-score

$$Z_i = (N_i^{real} - \bar{N}_i^{rand}) / std(N_i^{rand})$$

network significance profile (SP)

$$SP_i = Z_i / \sqrt{\sum_j Z_j^2}$$ ← a vector of normalized Z-scores.

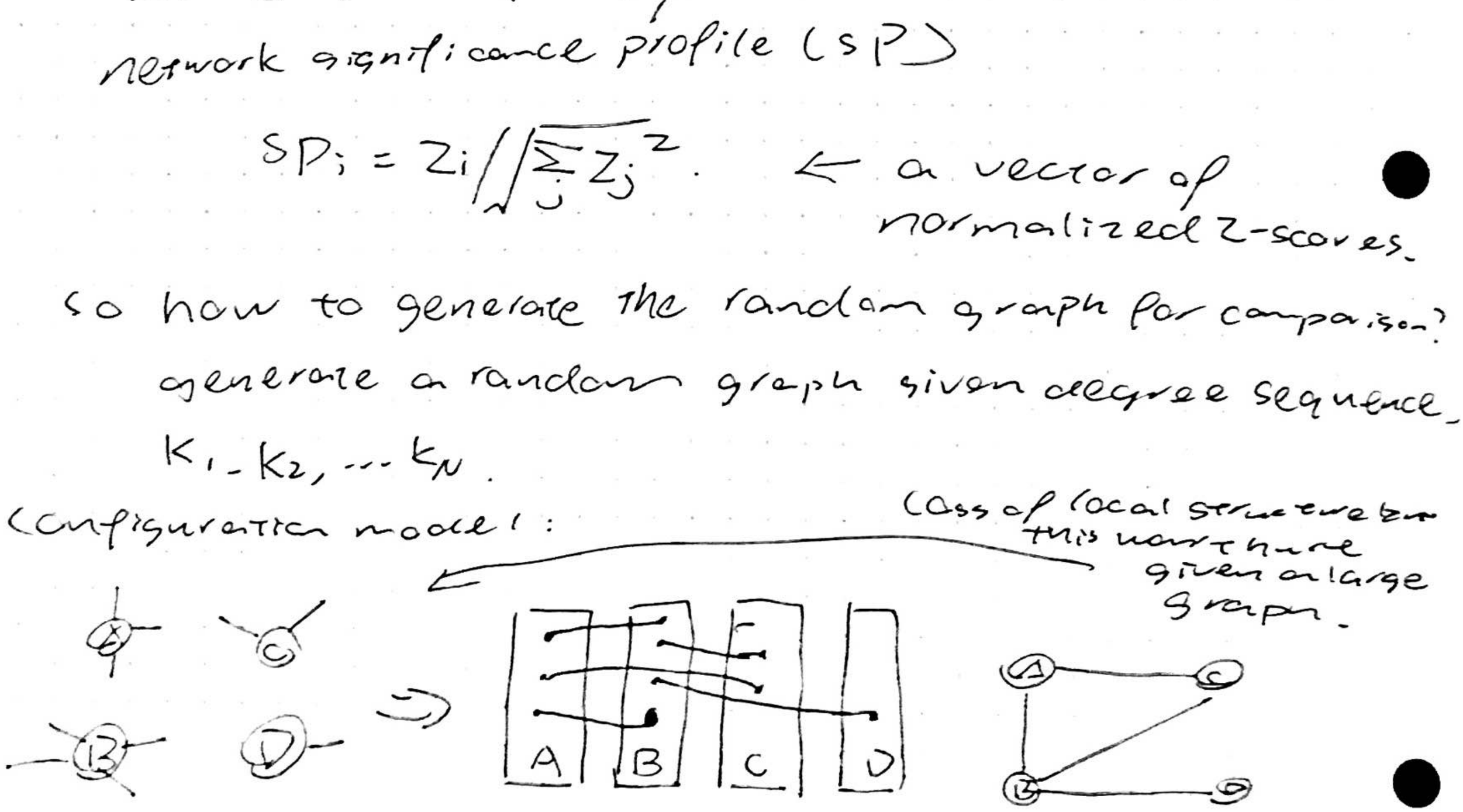
so how to generate the random graph for comparison?

generate a random graph given degree sequence

$k_1, k_2, \ldots k_N$.

Configuration model:

Class of local structure that this network have given a large graph.



nodes with spokes

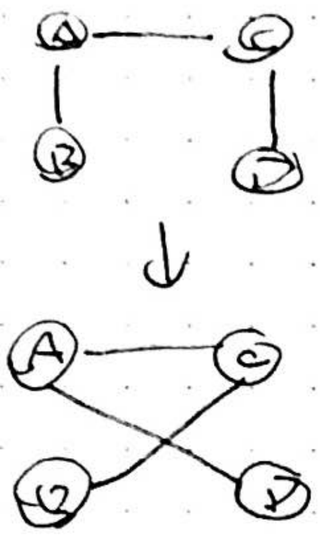
randomly pair up 'mini'-nodes

Resulting graph.

Alternative for spokes: swirdning.

start from a given Graph G.

Repeating the swirching step Q · |E| times:



- select a pair of edges A→B, C→D at random.
- exchange the end points to give A→D, C→B.

Exchange edges, only if no multiple edges or self-edges are generated.

result: a randomly rewired graph

* Q needs to be large enough (e.g. Q = 100) for the process to converge.

Graphlets: node feature vectors.

connected non-isomorphic subgraphs:

induced subgraphs of any frequency.

used to obtain a node-level subgraph metric.

Automorphism Orbits.

↳ takes into account the symmetries of a subgraph.

Graphlet Degree Vector (GDV): a vector with the frequency of the node in each orbit position (counts #(graphlets) that a node touches at a particular orbit).

↳ provides a measure of a node's local network topology.

Finding size-k motifs/graphlets requires solving two challenges:

1) enumerating all size-k connected subgraphs

2) counting # (occurances of each subgraph type).

NP-complete!

ESU algorithm:

Two sets: Vsubgraph: Coherently constructed subgraph (motif)

Vextension: set of candidate nodes to extend the motif.

Idea: Starting with a node v, add those nodes u to Vextension set that have two properties.

→ u's node_id must be large than that of v.

→ u may only be neighbored to some newly added node w but not of any node already in Vsubgraph.

ESU is implemented a recursive function; it can be displayed as a tree-structure of depth called the ESU-tree.

use ESU-tree to count subgraphs.

↳ classify subgraphs placed in the ESU-Tree Leaves into non-isomorphic size-k classes.

Graph Isomorphism.

Graphs G and H are isomorphic if there exists a bijection $f: V(G) \to V(H)$ such that:

any two nodes u and v of G are adjacent in G iff f(u) and f(v) are adjacent in H.

Structure roles in network.

Roles are 'functions' of nodes in a network and they are measured by structural behaviors.

Roles vs. Groups in Networks.

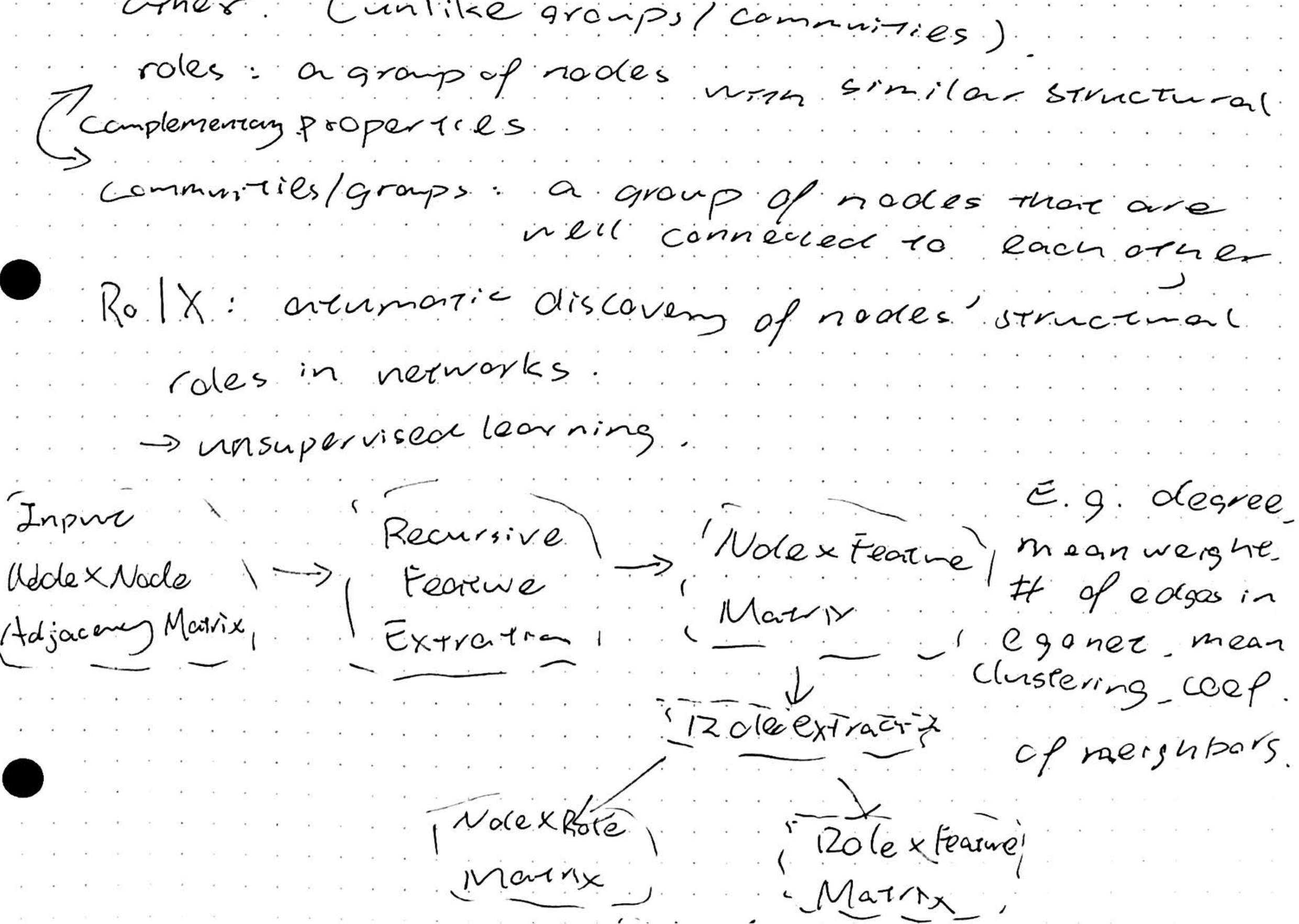
Nodes with the same role need not be in direct, or even indirect interaction with each other. (unlike groups/communities).

roles: a group of nodes with similar structural complementary properties

communities/groups: a group of nodes that are well connected to each other

RolX: automatic discovery of nodes' structural roles in networks.

→ unsupervised learning.



Recursive Feature Extraction. turns network connectivity into structural features.

idea: aggregate features of a node and use them to generate new recursive features.

Base set of a node's neighborhood features.

→ local features

→ ego net features

↓

includes the node, its neighbors and any edges in the induced subgraph on these nodes.

Community Structure in Network.

Granovetter's theory suggests that networks are composed of tightly connected sets of nodes.

Network communities: (clusters, groups, modules)

sets of nodes with lots of internal connections and few external ones (to the rest of the network).

sets of tightly connected nodes.

Modularity Q

- a measure of how well a network is partitioned into communities. Given a partitioning of the network into groups disjoint $s \in S$.

$Q \propto \sum_{s \in S} [(\#\text{edges within groups}) - (\text{expected } \#\text{ edges within groups})]$.

↑

need a null model

Null model: Given real G on n nodes and m edges. Construct rewired network G'
- same degree distribution but uniformly random connections

Different to before → consider G' as a multigraph.

The expected no. of edges between nodes i and j of degrees k_i and k_j equals:

$$k_i \cdot \frac{k_j}{2m} = \frac{k_i \cdot k_j}{2m} \quad \left(\sum_{u \in N} k_u = 2m\right)$$

$$Q(G,S) = \frac{1}{2m} \sum_{s \in S} \sum_{i \in s} \sum_{j \in s} \left(A_{ij} - \frac{k_i k_j}{2m}\right)$$

↑ normalizing

$A_j = 1$ if $i \to j$

0 otherwise

Const.: $-1 \leq Q \leq 1$

↑ Modularity values take range

if positive → community structure (0.3 - 0.7)

Idea: We can identify communities by maximizing modularity.

How do we find them?

Louvain Algorithm (greedy algorithm.)

High level overview:

Phase 1: Partition: modularity is optimized by allowing only local changes to node-communities memberships.

Phase 2: Aggregation: the identified communities are aggregated into super nodes to build a new network.

Go to Phase 1. (The passes are repeated iteratively until no increase of modularity is possible)

Louvain: 1st phase (Partitioning).

→ put each node in a graph into a distinct community (one node per community).

For each node i, the algorithm performs two calculations

① compute the modularity delta (ΔQ) when putting node i into the community of same neighbor j.

② Move i to a community of node j that yields the largest gain in ΔQ.

Phase 1 runs until no movement yields a gain (a local maxima of the modularity is attained).

* note that the output of the algorithm depends on the order in which the nodes are considered. However, research has shown that this has no direct impact in the overall modularity.

$$\Delta Q = \Delta Q(i \to C) + \Delta Q(D \to i)$$

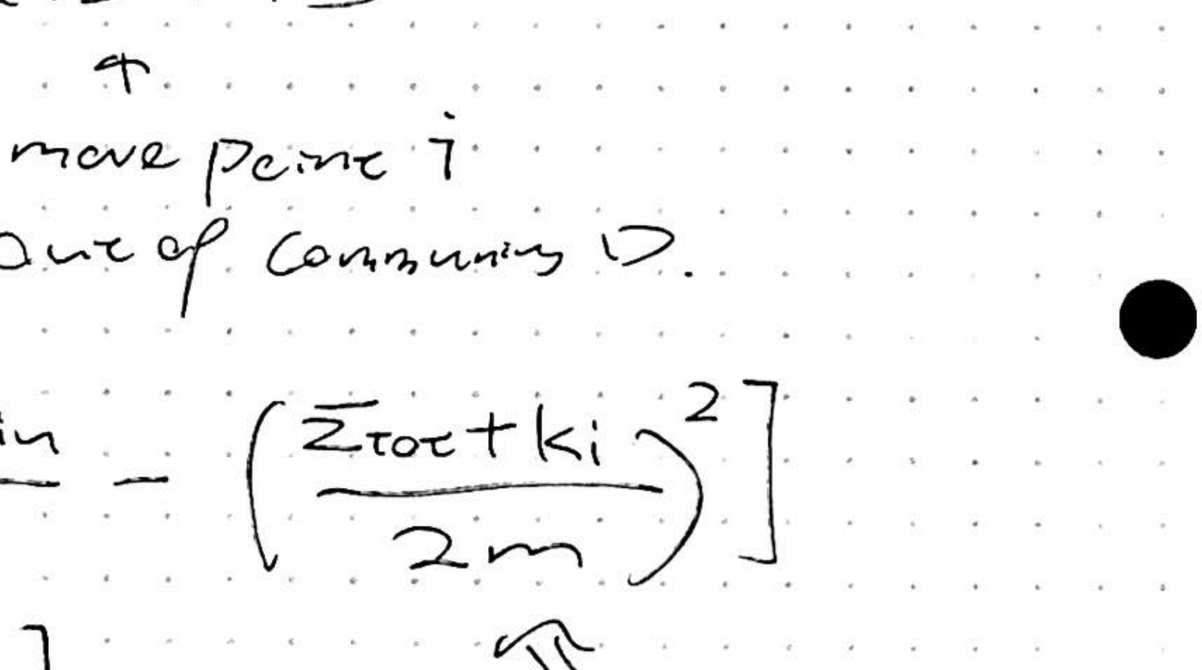
↑ move point i into community C

↑ move point i out of community D.

$$\Delta Q(i \to C) = \left[\frac{\Sigma_{in} + k_{i,in}}{2m} - \left(\frac{\Sigma_{tot} + k_i}{2m}\right)^2\right] - \left[\frac{\Sigma_{in}}{2m} - \left(\frac{\Sigma_{tot}}{2m}\right)^2 - \left(\frac{k_i}{2m}\right)^2\right]$$

⇑ modularity contribution before merging node i.

⇑ modularity contribution after merging node i



edge weigh of the resulting super-node from merging C and i

rese of the graph modeled as a single node

Louvain: 2nd phase (Restructuring).

The communities obtained in the first phase are contracted into super-nodes. and the network is created accordingly.

→ super-nodes are connected if there is at least one edge between the nodes of the corresponding communities.

→ The weight of the edge between two super nodes is the sum of the weights from all edges between their corresponding communities.

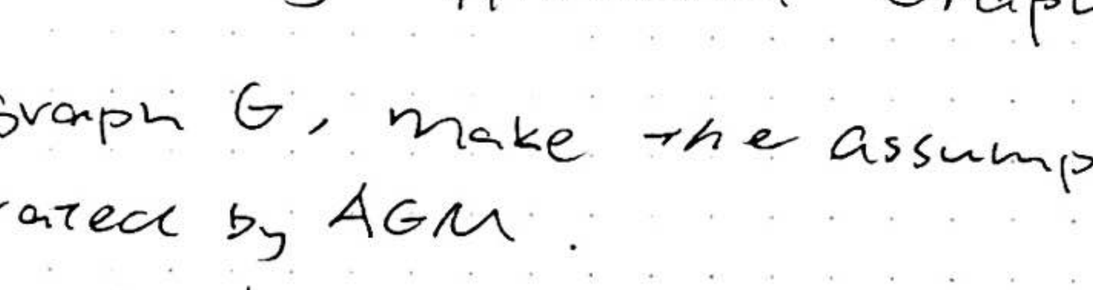
Phase 1 is then run on the super-node network.

Overlapping communities ---.

1) Define a generative model for graphs that is based on node communities affiliations.
   - Communities Affiliation Graph Model (AGM)

2. Given graph G, make the assumption that G was generated by AGM.
   - Find the best AGM that could have generated G. And this way we discover overlapped communities.

AGM. Generative process: from bipartite graph to a graph network.



Communities Affiliation.

Network.

- Given parameters $(V, C, M, \{p_c\})$.
  - Nodes in community c connect to each other by flipping a coin with probability p_c

- Nodes that belong to multiple communities have multiple coin flips.

- If they 'miss' the first time, they get another chance through the next community

$$P(u,v) = 1 - \prod_{c \in M_u \cap M_v} (1 - p_c)$$

Detecting communities:

given a graph, find the model $\bar{F}$

1) Affiliation graph M
2) Nbr. of communities C
3) Parameters p_c.

Graph fitting → how to estimate F given G.

$$MLE: \operatorname{argmax}_F P(G|F)$$

$$P(G|F) = \prod_{(u,v) \in G} P(u,v) \prod_{(u,v) \notin G} (1 - P(u,v))$$

Likelihood of edges in the graph — Likelihood of edges not in the graph.

in light of probabilities

we should relax hard membership assignments in AGM

↳ BigCLAM model

prob. of nodes u,v linking is $\propto$ to the strength of shared memberships:

$$P(u,v) = 1 - \exp(-f_u \cdot f_v^T)$$

$$l(F) = \sum_{(u,v) \in E} \log(1 - \exp(-f_u f_v^T)) - \sum_{(u,v) \notin E} f_u f_v^T$$

Optimization:

1 Start with random $\bar{F}$

2 update f_{uc} for node u while fixing the memberships of all other nodes. Update one community at a time