Quick Review

Community detection:

identifying relatively dense subgraphs in a network

→ ratio of internal to external edges (cut)

{Issue: global optimal solution is just a single comm.

Algs: agglomerative
divisive

hierarchical

→ Modularity

- One of the most widely-used approaches for C.D. problems

approaches for C.D. problems

- Explicity captures an objective measure of "how" modular a network is by comparing to a random network (null model)

→ used directly for optimization or relative comparisons

Basic Hypothesis: random networks lack any real community structure

↳ so, we can explicitly measure how clustered our network is relative to what's randomly expected (null hypothesis)

How do we calculate modularity

modularity Q

$$M = \frac{1}{2m} \sum_{\forall u,v \in C} \left(A_{uv} - \frac{d(u)\,d(v)}{2m} \right)$$

$\frac{d(u)\,d(v)}{2m}$ → random graph attachment probabilities

M = modularity

$m = |E(G)|$

$\forall u,v \in C$ = all vertex pairs in all comms.

A_{uv} = # edges between u, v or sum of weights

$d(u)$ = degree of u

$\frac{d(u)\,d(v)}{2m}$ = probability vertex u and vertex v would attach in a random graph (or # edges expected between u and v)

→ good approximation for loopy multigraphs

Issue: many graphs that we consider for C.D. are simple

→ NOT a good approximation

→ NOT a good approximation for simple graphs (more later)

Modularity Maximization

We can maximize modularity directly as a C.D. algorithm

→ Usually: greedy agglomerative

Newman Algorithm

- Greedy agglomerative algo.
- Initially: all vertices in unique community
- Iterate while # coms > 1: (or other stopping criteria)

  Merge community pairs that maximize modularity gain

Pros: Captures hierachy of comms.

good quality in practice

good theoretical basis

Cons: $O(n^3)$ = slow

potential issues w/ modularity

Faster algorithm: Louvain

- same as the above, but we do explicit coarsening

  → create a new graph where each community is a single vertex

→ can be faster and outputs better results

Note: these will not find the "optimal" solution

Note 2: most graphs do not have

Note 2: most graphs do not have a modularity peak

↳ usually a plateau with solutions +/- a few percent of same modularity

---

Issues with modularity

Resolution Limit: we can't resolve small communities

↳ Q: how small?

Change in modularity by combining communities A and B

$$\Delta M = \frac{l_{AB}}{m} - \frac{k_A k_B}{2m^2}$$

l_{AB} = # of edges between A and B

$k_{A/B}$ = sum of degrees of vertices in community A/B

Consider when $\Delta M = 0$ (threshold to merge)

Consider when $\Delta M > 0$
(threshold to merge)

$$\frac{l_{AB}}{2m} = \frac{k_A k_B}{2m^2}$$

$$l_{AB} = \frac{k_A k_B}{2m}$$

if $l_{AB} > \frac{k_A k_B}{2m}$ $\Rightarrow$ we gain modularity from the merge

assume $k_A = k_B = k$

$$l_{AB} = 1$$

$$1 > \frac{k^2}{2m}$$

$$2m > k^2$$

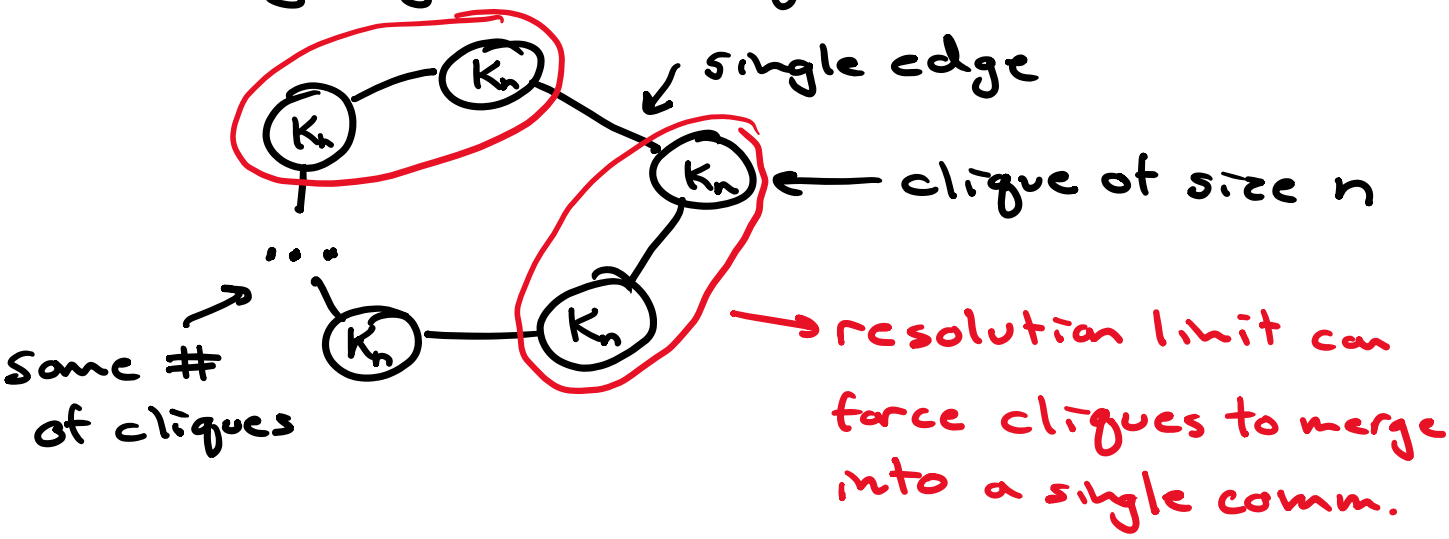
$$\boxed{\sqrt{2m} > k}$$

$\hookrightarrow$ we will always merge A and B if $k \leq \sqrt{2m}$

$\Rightarrow$ so this is a lower bound on the "size" of the community

the size of the community that modularity maximization will find

The "ring of cliques" graph highlights this quite well



Issue: this goes against our understanding of communities

Why this isn't a huge deal:

- Most algos are hierarchical
  - ↳ we have small communities to observe before they merge
- However: a large distribution in comm. sizes can be problematic (multi-resolution approaches)

The other problem

↳ $\frac{d(u)\,d(v)}{2m}$

- Most social networks are simple
- The above approximation can be bad for simple graphs
  * especially for dense, skewed graphs (subgraphs)

What happens when $2m << d(u)\,d(v)$

multigraphs → $\frac{d(u)\,d(v)}{2m}$ expected # edges

simple graphs → nonsense

Takeaway: modularity can be meaningless in skewed or dense graphs

Fix: we could use actual attachment probabilities for simple graphs

T

probabilities in simple graphs

Issue: no closed-form way to calculate simple graph attachment probabilities

(100,000 pt. bonus: figure the above out)

Conductance

Generally: a measure of how quickly a random walk converges to a steady state

Well-defined comms.: very quick

poorly-defined comms: very slow

We generally measure conductance relative to some edge cut $[S, \bar{S}]$

S: vertices in comm. $\bar{S}$: all other verts

$$\text{Conductance}(S) = \frac{\text{cut}(S)}{}$$

$$\text{Conductance}(S) = \frac{\text{cut}(S)}{\min(K_S, K_{\bar{S}})}$$

$\text{Cut}(S) = |[S, \bar{S}]| = \#$ edges in cut (edges leaving comm.)

K_S = sum of degrees in S

How to apply to C.D.:

Issue: we define condutance relative to a single community

↳ tough to directly use as a global measure

We can still use it as a local measure for optmization

Challenge: we need good seed vertices

Challenge 2: we would need to determine comm. sizes (or # of comms)

Issue overall: choices made above are going to bias our results