Lecture 19 - More subgraph mining

Monday, April 11, 2022 7:53 AM

Last class: motifs / anomalies

Note: only considering counts

Today: more subgraph analytics

Graph similarity measures

We can actually compare the general topology of two separate graphs using subgraph counts

Similar to motif finding:

→ considering a number of possible subgraph topologies

→ count them up

→ we can use these counts to compare separate networks

Subgraph frequency distance

- given graphs G and H

- given graphs G and H
- define $N_i(G)$ as counts for subgraph i on G
- define $T(G)$ as total counts over all subgraphs $T(G) = \sum_i N_i(G)$
- define $F_i(G) = -\log(N_i(G)/T(G))$

minimizing impact of a single subgraph

- define $D(G,H) = \sum_i |F_i(G) - F_i(H)|$

↳ uses: graph classification
null model selection

Q: what subgraphs?

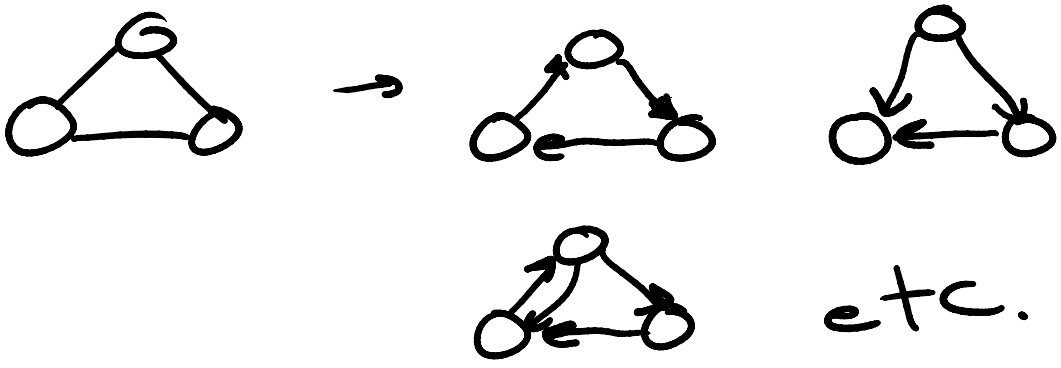
A: graphlets and other stuff

Undirected graphs: graphlets
aka all 3-5 induced vertex subgraphs

aka all 3-5 vertex [illegible]

- small enough to be countable at small → large scale
- big enough to be useful
  - can be counted efficiently via dynamic programming
  - capture topology within a few hops of a given vertex

Directed: a lot more possible subgraphs



etc.

Another option: treelets

- tree-structured subgraphs from 3-9 vertices
- Non-induced: more counts [illegible] can be minimized

- Non-induced
  so noise can be minimized
- Bigger, so captures more of the local topology around some v

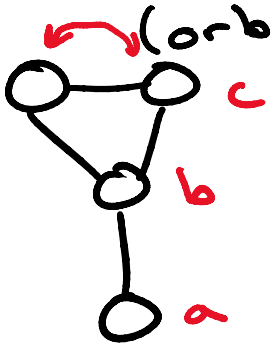
→ easier to count

Vertex similarity measures

Really: just extending our graph measures to single vertices

How: we "root" each subgraph at the vertex and count the rooted embedding

Consider:



For each root at some v, we need to re-count

Subgraph degree signatures

- consider vertices u and v
- consider all possible orbits

- consider all possible orbits across some set of subgraphs

- For orbit i with counts u_i, v_i

$$D_i(u,v) = w_i \frac{|\log(u_i+1) - \log(v_i+1)|}{\log(\max(u_i, v_i)+2)}$$

w_i: weighting term to mitigate automorphisms

overall:

$$D(u,v) = \frac{\sum_i D_i(u,v)}{\sum_i w_i}$$

For similarity:

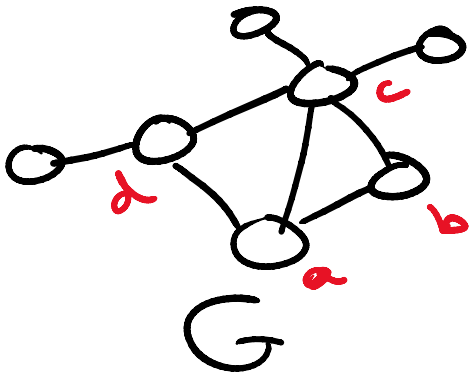
$$S(u,v) = 1 - D(u,v)$$

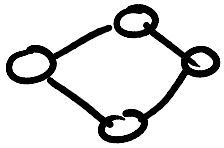
Uses: vertex classification

study similar/differing regions of same interaction network

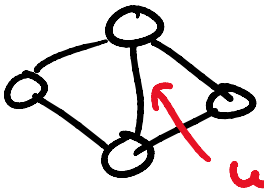
graph alignment

Induced vs. non-induced





non-induced



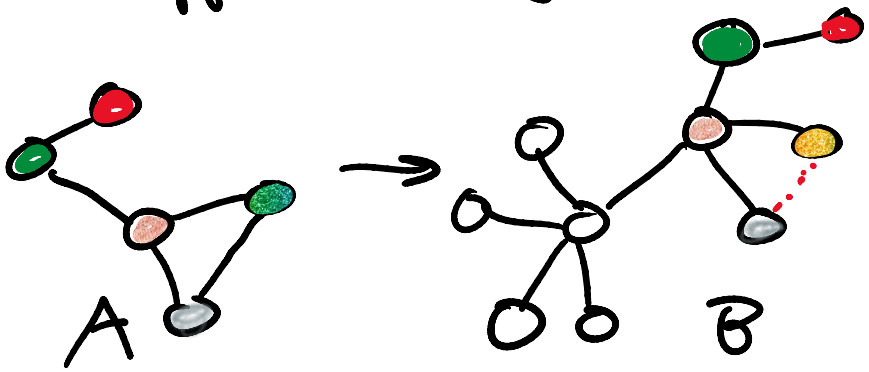
induced

we have all edges from G

Graph Alignment

Consider graph alignment as an "approximate" isomorphism problem

- Aligning two similar-sized networks is approximate graph isomorphism
- Aligning a small to large network is approx. subgraph isomorphism



For this problem, we need to consider some cost function

same cost func...

- There are many such metrics, across differents fields/application

- Functional metrics - explicit measures of functional similarity within biological networks

- Structural metrics

  Edge correctness - how many edges match

  Edit distance - how many edge or vertex delettions or subtractions to make match

Note: this is also a computationally challenging problem - NP-hard

→ If we consider using subgraph counts, this gets expensive quite quickly

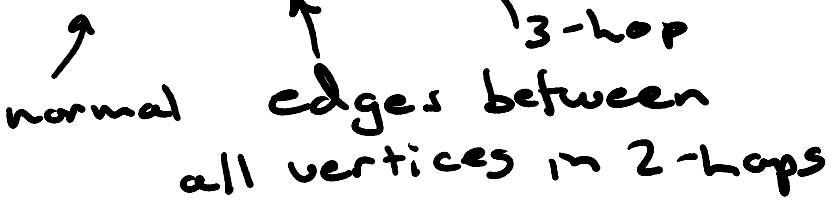
How can we use subgraph counts to perform an alignment?

- we want to align $G \to H$

- we want to align $G \to H$
  - we count all orbits for all vertices
  - we can compute pairwise similarities for $v \in G$, $u \in H$
  - we can start aligning greedily

Note: if we have to consider possible non-existing edges/vertices

→ we can use power graphs G^1, G^2, G^3



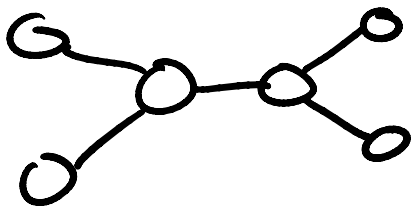
Repeat until done

Approximation Algorithms

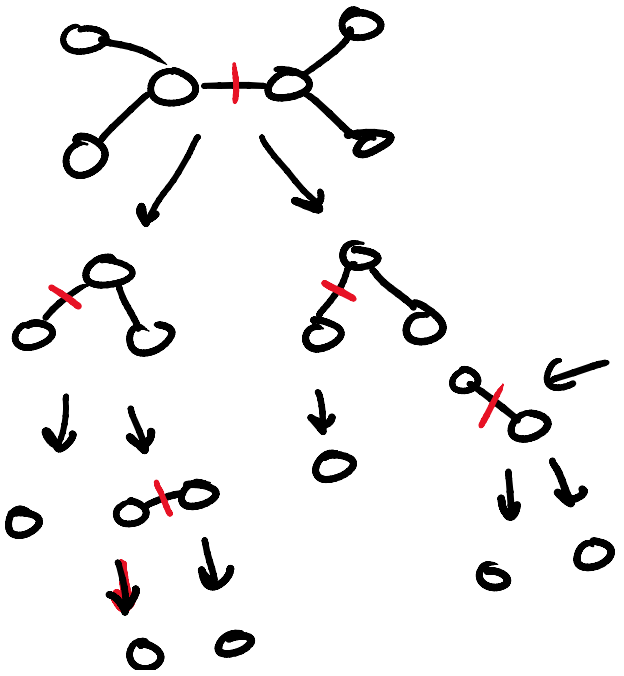
Sampling-based: with skewed degree distributions, want to sample edges

Color-coding:

- we color our graph with k colors
- we count "colorful" embeddings of same subgraph ← every vertex has a unique
- Why: low error, relatively can be quite fast via dynamic programming



Subgraph → we partition using single edge cuts down to a single vertex



← counts of this subgraph are product of its counts of "children"

Complexity: $O(m 2^k)$ per iteration