What is Graph Mining?

Monday, January 8, 2024 8:57 AM

Plan for today:

- About me and the course ✓
- Go over syllabus ✓
- Go over website –
- Talk about project
- Talk about paper presentation
- Lecture
  - Define graphs
  - Real-world graph properties
  - Graph Mining and applications
  - Graph computation and processing
  - NetworkX and connected components

Project details

- Groups of 1-4
- Take a look at data repos
- Prior project ideas:
  - * Predicting chess match outcomes using a competition network
  - * Vertex centrality to identify population centers on a road network
  - * Stock market / coin prediction
  - * Recommender systems using

* Recommender systems using SNAP data

- Grad students: use your ongoing project

Paper Presentations

- One paper selected per student (from website)
- 20-30 minute presentation

Graph Definitions

Graph → tuple of

$V(G)$ = vertex set of graph G

$E(G)$ = edge set

$w_V(G)$ = vertex weights

$w_E(G)$ = edge weights

$D_V(G)$ = vertex metadata

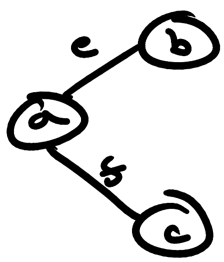
$D_E(G)$ = edge metadata

$D_V(G)$ = vertex metadata

$D_E(G)$ = edge metadata

$G = \{V, E, W_V, W_E, D_V, D_E\}$

$G = \{V, E\}$ → at a minimum we need vertices and edges to define graph



G

$V(G) = \{a, b, c\}$

$E(G) = \{e = (a, b),\ f = (a, c)\}$

For our GM context

vertices => represent discrete objects, entities, etc.

edges => represent connections or interactions between those objects, entities, etc.

weights => strength of connections for edges or importance or some other measure for vertices

or some other measure
for vertices

metadata => generic catch-all
for other vertex/edge
labels or properties

Temporal graphs => graphs that
evolve over time

$G_{t=0}$ $G_{t=1}$ $G_{t=2}$ $G_{t=3}$

Real World Graphs

$\hookrightarrow$ social networks, info networks
biological networks, etc.

These graphs all have similar properties

Sparsity: $|E(G)| \lll |V(G)|^2$

Degree skew: # of low degree
vertices >>> # of
high degree vertices

$d(v)$ = # of edges
attached to v

$d(v) = $ # of edges attached to v

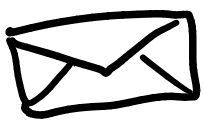
vertices ... # of high degree vertices

Hubs: high degree or otherwise "important" vertices

Irregular: information, social, etc. networks are often not physically constrained

Small-world: average shortest path lengths are small relative to $|V(G)|$

"6 degrees of Kevin Bacon"

Note  : Properties are typical but not every real-world graph has them

Graph Processing

"vertex-centric" processing

- Each $v \in V(G)$ has some state $S(v)$
- We iteratively update $S(v)$
  → often using the states of v's neighbors
- Many/most graph algorithms can be implented in a vertex-centric way

Algorithmically:

Input: $G = \{V, E\}$

For all $v \in V(G)$:
  $S(v) = $ initialize()

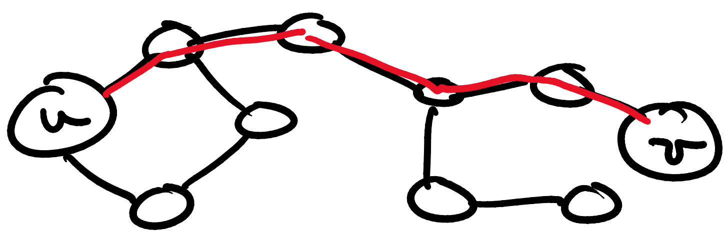
For some # of iterations:
  updateAlgoState()
  For all $v \in V(G)$:
    For all $u \in N(v)$: ← neighborhood of v
      $S(v) = $ updateState$(S(v), S(u))$

For all $v \in V(G)$:

$S(v) = \text{finalize}()$

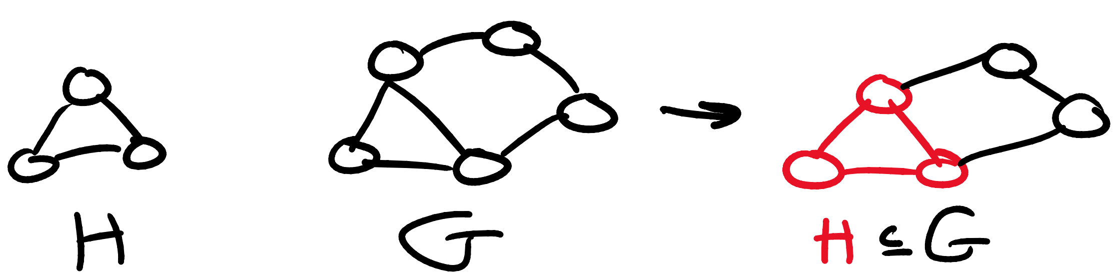
Implement connectivity decomposition

u,v are connected if there exists a path along edges from v to u



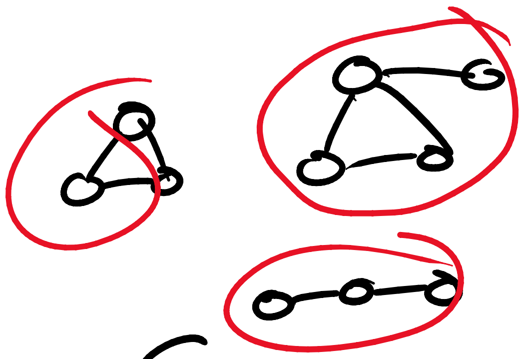
G is connected if there exists such a path for all u,v pairs

Connectivity decomposition: find all maximal connected subgraphs within same G



H G $H \subseteq G$

H is a subgraph of G



3 connected maximal subgraphs

G

Easy way to solve using our framework:

Each vertex gets a unique label

do
all vertices assumes maximum label from its 1-hop neighborhood
while some vertex updates a label

at the end → each unique label corresponds to a unique connected component