Big Data Processing in Social Networks

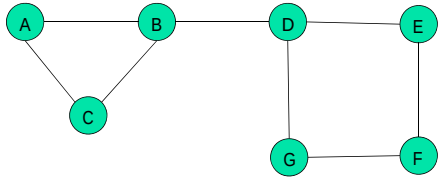
Kai Shen

Big Data in Social Networks

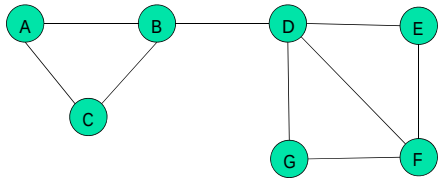
- Social networks (Facebook, Google+, …) are large
  - Containing big data on users and their interactions
- Uniqueness: Friendship and relationships
  - Assess locality of relationship ⇨ assess the evolution or maturity of the social network over time
  - Communities (strongly connected subgroups) ⇨ maybe common interests so we can cluster users and make recommendations

Social Network Graphs

- Modeled as a large graph
  - Nodes are users
  - Edges are friendship or relationships (typically undirected, may carry weights)

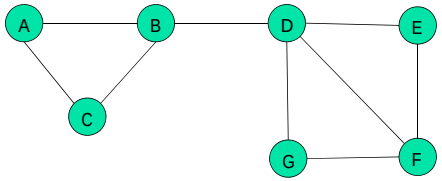


Friendship Locality



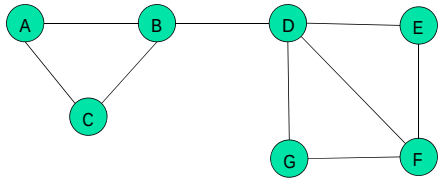
- Hypothesis: if X-Y connected and X-Z connected, then Y-Z connected
  - Probability: 9/16 = 56%
- A random (randomly formed edges) network with 7 nodes, 9 edges
  - Probability: 7/19 = 37%
  - ⇨ Yes, friendship locality
- Big data challenge? Parallel processing? Data partitioning?

Node Clustering (Traditional Approach)



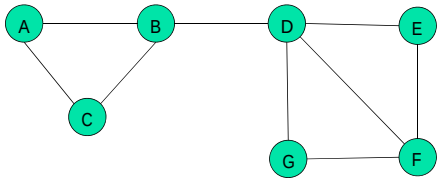
- Clustering to find strongly connected node communities
- K-means?
  - Define distance?
  - K=2, D & F are initial cluster centers?
  - K=2, B & F are cluster centers?
  - Big data processing – MapReduce()?

Node Clustering (Graph Approach)



- Use graph semantics: identify and cut edges that are least likely to be inside a cluster
- Betweenness of edge e
  - Number of pairs of nodes (X,Y) such that edge e lies on the shortest path between X and Y
  - How does it tell the likelihood of an edge to be inside a cluster?
- Remove edges on order of their betweenness until we settle

Node Clustering (Direct Approach)



- Finding cliques
  - Find the largest clique is NP-complete, approximations are even hard

Random Walk

- Random walk
  - From a starting point, a walker randomly follows links to surf over the social network graph. We see the probability for it to visit other nodes.
  - Does it remind you of something?
- Limit the walk step (or enforce a stopping probability) to add locality
  - May find related nodes; not quite able to find full communities
- Big data processing – MapReduce()?

Count Triangles

- Why?
  - Assess the friendly locality
  - Assess the maturity or evolution of communities
- How?
  - For each node, identify each neighbor pair and see if they are connected? Use a node rank to avoid triple counting.
    - Complexity? Assume N nodes and M edges.
  - For each edge (X,Y), identify each neighbor of X and see if it is connected to Y
    - Complexity?
  - Actually you can get to $O(M^{3/2})$

Count Triangles

- Big data challenges – parallel solution, easy data partitioning, good load balancing.
  - How to partition the data? Or more specifically, how to partition the edges?
- Triangle counting – find three points P_1, P_2, and P_3 such that P_1-P_2, P_1-P_3, and P_2-P_3 are all connected
  - Think of a N×N×N space, hash N nodes into B buckets, and we have B×B×B subspaces and as many tasks (each handles on subspace) for parallel data processing
  - Now we need to assign data to these tasks
    - each edge (X,Y), if used as P_1-P_2, is needed by B tasks
    - can also be used as P_2-P_3 and P_1-P_3, so must go to 3B tasks altogether

Text Mining in Social Networks

- Tweet / message texts contain information
- Along with location and other user profiles can enable useful analysis and processing
- Again, find matching advertisement for recommendation
- To track the flu infection
  - Easily parallelizable with good data locality ⇨ MapReduce()
  - Realtime analysis: get data to the right place quickly, less tolerance to slow tasks

Disclaimer

- Preparation of these slides used materials in the online book “Mining of Massive Datasets” by Rajaraman, Leskovec, and Ullman. The slides are intended for the sole purpose of instruction of computer networks at the University of Rochester. All copyrighted materials belong to their original owner(s).