Public-Private Model in Graphs

- Brian Brubach
- Soheil Ehsani
- Karthik Sankararaman
Overview

• Introduction of the model
• Simple Example to illustrate the model
• Comparison to other well-studied models
• Algorithm to illustrate the all-pairs shortest path
• Community Detection aka Densest sub-graph problem
• Extension to other sub-additive functions e.g. MaxCut
• Algorithm for Vertex Cover
• Experimental Results*
• Future Directions
The Public-Private Model

- Introduced by Chierichetti, Epasto, Kumar, Lattanzi, Mirrokni
  - KDD 2015 Best Paper Award
- The public graph $G = (V, E)$ is known
- For each node $u$, there is an unknown private graph $G_u = (V, E_u)$
  - For all $(v, w)$ in $E_u$ both $v$ and $w$ are at most distance 2 from $u$. Why?
  - WLOG $E \cap E_u = \emptyset$
- Together they form the public-private graph $G \cup G_u$
Motivation: Social Networks

- Facebook, Google+, Twitter
- Nodes represent people/users
- Edges represent connections (e.g., friendship, group membership)
  - Private graph edges represent private friend lists, private groups, etc
  - Among 1.4 million New York Facebook users, 52.6% hid their friends (Dey, Jelveh, Ross 2012)
Motivation: Social Networks

- Very large graphs (Big data!)
  - YouTube: 1,000,000+ nodes
- Problem: processing the public-private graph for each node/person is too slow
- Goal: preprocess the public graph to answer queries fast when the private graph is revealed
  - How fast?
The Public-Private Model

- Known public graph $G = (V, E)$
- Unknown private graph $G_u = (V, E_u)$
  - For all $(v, w)$ in $E_u$ both $v$ and $w$ are at most distance 2 from $u$
  - WLOG $E \cap E_u = \emptyset$
- Goal:
  - Preprocess the public graph using $\text{poly}(|E|)$ time and $\tilde{O}(|V|)$ space
  - When $G_u$ is revealed, answer queries using time/space $\tilde{O}(|E_u|)$ and $\text{poly}(\lg |V|)$
Warm-up: Number of Connected Components

- Algorithm
  - Label the components of the public graph and store total number of components
    - $O(m)$ time, $O(n \lg n)$ space
  - Count the number of different components that $G_u$ connects
    - $O(|E_u|)$ time
All Pairs Shortest Path (APSP)

- Important problem in Social Networks

- In learning algorithms, distance between two people can be used as a feature
  - E.g. Gives information of likelihood of a person following a celebrity

- Can be solved exactly in $O(n^3)$ time offline
  - Too slow for large graphs

- Will later describe a $O(poly \ log \ n)$ approximation in near-linear time
APSP in public-private model

- Will use the poly-log approximation to get an algorithm in the public-private model
  - Here, we look at the restricted model where distance from u is at most 2 in private graph

- Compute a poly log (n) approximation on the public graph

- For a private graph query with u, we need to find \( \text{dist}(u, \ast) \)
  - We can have the following cases (described in the next few slides) for \( \text{dist}(u,v) \)

- Take the one with the minimum of all of them as \( \text{dist}(u,v) \) in the union graph
Case 1

- dist(u,v) in union graph is same dist(u,v) in public graph

- In this case, no new computation needs to be done
Case 2

- $\text{dist}(u,v)$ in union graph is $1 + \text{dist}(w,v)$ where $w$ is a neighbor of $u$ in private graph and $\text{dist}(w,v)$ is the distance in public graph.
Case 3

- $\text{dist}(u,v)$ in union graph is $2 + \text{dist}(z,v)$ where $z$ is at distance 2 of $u$ in private graph and $\text{dist}(z,v)$ is the distance in public graph.
O(poly log n) approximation to APSP

- Due to Das Sharma, Gollapudi, Najork, Panigrahy [WSDM 2010]

- A sampling based approach

- Choose a random subset of vertices and find distance to this random subset

- Use this distance to estimate distance between any two pairs
Estimating dist(u, v)

\[ n = 11 \]
\[ r = \lceil \log n \rceil = 3 \]

\begin{align*}
\text{SKETCH}(u) &= \{ q, u_1, u_2, u_3 \} \\
\text{SKETCH}(v) &= \{ q, v_1, v_2, v_3 \} \\
\text{CommonSketch} &= \text{SKETCH}(u) \cap \text{SKETCH}(v) \\
dist(u, v) &= \min\{ \text{dist}(u, w) + \text{dist}(w, v) : w \in \text{CommonSketch} \} 
\end{align*}
Analysis

- Single run of the algorithm gives a $O(\text{polylog } n)$ approximation in expectation
  - Proof omitted here

- Success probability can be amplified by running the algorithm $O(\log n)$ times and taking the sketches to be the union of the sketches in each iteration

- Finally computing the distances using the common sketch as before on this union of sketches gives a $O(\text{polylog } n)$ with high probability
  - Chernoff Bound type arguments on the generated subsets
Putting it together

- Preprocessing takes $O(m \ polylog n)$ time
  - The closest vertex computation can be performed by BFS from each set $S_i$ to all vertices

- For each vertex a $O(polylog n)$ sketch stored; Hence total space $O(n \ polylog n)$

- Query takes $O(|E_u| \ polylog n)$ time
Community Detection

- Central question in Social Network: Do node A and node B in a graph share a core similarity?
  - E.g.: Same geographical location in Yelp, Papers in similar topics in DBLP

- Many notions and various algorithms in the Social Networks literature

- Important problem outside CS community
  - E.g.: Communities in protein interaction graphs studied by Biologists
Example of Community Detection

Nodes: A topic-dedicated stack exchange

Edges: If a user is part of both the sites

Colors: Different communities
Densest Subgraph

- Concept of Community Detection often formalized as the densest subgraph problem
  - Formal definition in the following slide

- Often well-captures the intuitive definition of “well-connected” nodes
The Densest Subgraph

- Find a set $S$ of vertices maximizing

$$\frac{\{(u, v) \in E \mid u, v \in S\}}{|S|}$$

Density = 1.25
Future Works

- Can we give a similar approach for other functions, such as
  - sub-modular
  - matroid
- Can we formulate this method as a general tool which includes all cases such as
  - union
  - intersection
  - maximum
  - minimum
- Can we modify the model to capture other real world problems?
  - What if we allow the private graph to delete edges (eg. “unfollowing” on Facebook)?
  - What if two private graphs $G_u$ and $G_v$ are revealed together (eg. friend request)?
Thank You!