Efficient Aggregation for Graph Summarization

Yuanyuan Tian (University of Michigan)
Richard A. Hankins (Nokia Research Center)
Jignesh M. Patel (University of Michigan)
Motivation

- Graphs are everywhere
  - Social networks, biological networks
- Graph datasets growing rapidly in size.

- Impossible to understand by mere visual inspection.
- Need: Graph Summarization

Number of Facebook Users

DB Coauthorship
7,445 nodes, 19,971 edges
Existing Methods

- Statistical methods
  - Limited information, hard to interpret & manipulate.
- Frequent subgraph mining methods
  - Produce a large number of results.
- Graph partitioning methods
  - Largely ignore node attributes.
- Graph compression
  - Compact storage.
- Graph visualization
  - Ben’s keynote talk.
- MDL-based graph summarization (Nisheeth’s talk)
Solution: **Graph Aggregation**

- Two well-defined novel graph aggregation operations: SNAP & k-SNAP
  - Summarization based on user-selected node attributes and relationships.
  - Produce summaries with controllable resolutions.
  - Provide “drill-down” and “roll-up” abilities to navigate multi-resolution summaries.

- Efficient algorithms
  - Produce meaningful summaries for real applications.
  - Efficient and scalable for very large graphs.
SNAP Operation

- Group nodes by user-selected node attributes & relationships
- Nodes in each group are homogenous w.r.t. attributes and relationships
- The grouping with the minimum # groups

For example:
- All students in the blue group have the same gender and are in the same dept
- Every student in the blue group has:
  - at least one “friend” in the green group
  - at least one “classmate” in the purple group
  - at least one “friend” in the orange group
  - at least one “classmate” in the orange group
Evaluating SNAP Operation

Top-Down Approach

- **Step 1:** group nodes just based on user-selected attributes.
- **Iterative Step:**
  
  while a group breaks homogeneity requirement for relationships
  split the group based on its relationships with other groups
Limitations of SNAP Operation

- Problems with the SNAP operation
  - Homogeneity requirement for relationships
    - Noise and uncertainty

SNAP

Reality

- Users have no control over the resolutions of summaries
- SNAP operation can result in a large number of small groups

- **k-SNAP** operation:
  - Relax the homogeneity requirement for relationships
  - Let users control the resolutions of summaries
  - Provide “drill-down” and “roll-up” abilities to navigate summaries with different resolutions.
k-SNAP Operation

- Users control # groups in the resulting summary: k
  - Maintain homogeneity requirement for attributes.
  - Relax homogeneity requirement for relationships.
- Assess the quality of a summary

\[ \Delta = \sum_{g_i g_j} \{ \delta_{g_i g_j}(g_i) + \delta_{g_i g_j}(g_j) \} \]

\[ \delta_{g_i g_j}(g_i) = \begin{cases} |P_{g_i g_j}(g_i)| & \text{if } p_{ij} \leq 0.5 \\ |g_i| - |P_{g_i g_j}(g_i)| & \text{otherwise} \end{cases} \]

\[ \Delta = 0 \]

- 5% ≤ 50% (weak)
- \( \Delta + = 3 + 4 \) ← extra participants
- 95% > 50% (strong)
- \( \Delta + = (100-95)+(20-19) \) ← missing participants

......
Evaluating k-SNAP Operation

- **Goal:** Find the summary of size k with the minimum $\Delta$ value (best quality)
  - Proved to be **NP-Complete!**
    - Infeasible to produce exact k-SNAP summaries.
  - Alternative: **heuristics**
    - Top-Down Approach
    - Bottom-Up Approach
Top-Down Approach

- Similar to the SNAP evaluation algorithm (coarse → fine)
- **(Difference)** At each iteration, it needs to decide:
  - which group to split?
  - how to split the group?
- **Heuristics:**
  - Split a group into two subgroups at each iteration
  - Find $g_i$ with the maximum $\delta_{g_i,g_j}(g_i)$ (the most contribution to $\Delta$)
  - Split group $g_i$ based on whether the nodes in $g_i$ connect to $g_j$.

\[
\Delta = \sum_{g_i,g_j} \max \{ \delta_{g_i,g_j}(g_i) + \delta_{g_i,g_j}(g_j) \}
\]
Bottom-Up Approach

- Compute the SNAP summary first (fine → coarse)
- Iteratively merge two groups until the # groups is k
  - Which two groups to merge?
  - Heuristics:
    - Same attribute values
    - Similar neighbors
    - Similar participation ratio

\[ \text{MergeDist}(g_i, g_j) = \sum_{k \neq i,j} |p_{i,k} - p_{k,j}| \]

Merge two groups with the minimum \text{MergeDist}. 
Experimental Evaluation

- Implementation
  - C++ on top of PostgreSQL

- Evaluation Platform
  - 2.8GHz P4, 2GB RAM, 250GB SATA disk, FC2
  - PostgreSQL: version 8.1.3, 512 MB buffer pool

- Evaluation Measures:
  - Effectiveness & Efficiency
  - Verified by the SIGMOD repeatability committee.
Effectiveness: DB Coauthorship

DBLP Database Coauthorship Graph
(7,445 nodes, 19,971 edges)

Node Attributes:
name (string), numPub (int), prolific (LP, P, HP)
LP:[1, 5], P:[6, 20], HP:[21, -]

Relationship: coauthorship

SNAP
Attribute: prolific
Relationship: coauthorship

3,569 groups,
11,293 group relationships
Effectiveness: DB Coauthorship

**SNAP**
Attribute: prolific

**k-SNAP**
Attribute: prolific
Relationship: coauthorship

<table>
<thead>
<tr>
<th>K</th>
<th>Effectiveness Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>36</td>
</tr>
<tr>
<td>5</td>
<td>9.74, 1.66, 1.55</td>
</tr>
<tr>
<td>6</td>
<td>1.26, 1.23, 2.23</td>
</tr>
<tr>
<td>7</td>
<td>1.26, 1.23, 2.23</td>
</tr>
</tbody>
</table>
Effectiveness: DB Coauthorship

Impact of Double-Blind Reviewing on SIGMOD

- SIGMOD: 1994-2000: 0.125, 2001-2007: 0.217
k-SNAP: Top-Down vs. Bottom-Up

Dataset: DBLP DB Coauthorship Graph

Quality
- Measure: $\Delta /k$
- Top-down beats bottom-up for small $k$ values

Execution Time
- Top-down is much faster than bottom-up

Overall, top-down is the winner!
Efficiency: Synthetic Graphs

Dataset: Synthetic Power-Law Graphs (by GTgraph) (avg degree: 5)
Conclusion

- Database-style aggregation for graph summarization
  - Customized summaries
  - Controllable resolutions
  - "drill-down" and "roll-up" abilities
  - Meaningful summaries for real applications
  - Efficient and scalable for very large graphs

- Incorporated in Periscope/GQ graph querying system
  - Combined with other graph operations to perform complex analysis on graphs (VLDB’08 Demo)
Questions?
Suggestions?
Thanks! 😊