Summarization for a Heterogeneous Social Network


Prof. Shou-de Lin
CSIE/GINM, NTU
Existing Methods for Network Summarization (1/2)

• Use simple **statistics** to describe graph **structural characteristics**
  – E.g., degree distribution, average path length, clustering coefficient
  – Provide only limited information which is difficult to interpret and manipulate

• **Frequent pattern mining** methods
  – Understand repetitive behaviors in the network
  – However, it produces a large number of results that can easily overwhelm the user
Existing Methods for Network Summarization (2/2)

• **Community detection** methods
  – Purely based on connectivity, the attributes of nodes are largely ignored
  – There are more things to know than communities in a social network

• **Graph drawing/visualization** methods
  – Easily understand and interpret for smaller graphs
  – High space and time complexity for very large graph data
The SNAP and K-SNAP Approaches

• Summarization based on user-selected Node Attributes and Pairwise relationships
• K-SNAP produces summaries with controllable resolutions
• Efficient and scalable for very large heterogeneous graphs
• Demonstrate meaningful summarized results for real-world applications
SNAP Setup

- Nodes have attributes.
- Different types of edges exist.
- Summarize the network by grouping nodes into groups and identifying relations between groups.
  - Nodes in the same group should have identical attributes
  - Nodes in the same group should have identical kinds of connections to the other groups (homogeneous requirement).

Graph: G

Summary
An Example

• All students in the **blue group** have the same gender and are in the same dept

• Every student in **blue group** has
  – at least one “friend” in **green group**
  – at least one “classmate” in **purple group**
  – at least one “friend” in **orange group**
  – at least one “classmate” in **orange group**
SNAP Algorithm

• It is a **top-down** approach

• **Step 1**: group nodes based on user-selected attributes
  – I.e., the maximum A-compatible grouping

• **Iterative Steps**:
  
  While a group **violates homogeneous requirement for relationships**, **split** the group based on its relationships with other groups
An Illustration for SNAP

Group

Split
Extending SNAP to K-SNAP

- **SNAP**: nodes in each group are homogeneous w.r.t. user-selected attributes and relationships.
- **Drawbacks for SNAP**: 
  - homogeneity is often too restrictive in practice, as most real life graph data is subject to noise and uncertainty  
    - E.g., some edges are missing, or spurious  
  - Apply SNAP could result in a large number of small groups, in the worst case, each node may end up in an individual group
- **K-SNAP**: relax a bit the homogeneity of relationships
Extending to K-SNAP (cont.)

• Observations in SNAP
  – The existence of a group relationship(s) between two groups implies every node in both groups at least participates once in this group relationship(s).
  – The missing of a group relationship implies no single link connects any nodes across the two groups using this relationship.

• K-SNAP: Relax the above constraints
  – If most nodes in two groups participate in the group relationship => strong group relationship
  – If only a tiny fraction of nodes are connected between two groups => strong negative group relationship
K-SNAP Operation

- **K-SNAP** further allows users to control the resolutions of summaries.
- Here using the slider, users can drill-down to a larger summary with more details or roll-up to a smaller summary with less details.
K-SNAP Operation

- Use user-specified \( k (=\#\text{group}) \) for summary
  - **Maintain** homogeneity requirement for **attributes**
  - **Relax** homogeneity requirement for **relationships**

**\( \Delta(\Phi_A) \):** assess the quality of an A-compatible grouping by examining how different it is to a hypothetical \((A,R)\)-compatible grouping

\[
\Delta(\Phi_A) = \sum_{\mathcal{g}_i, \mathcal{g}_j \in \Phi_A} \sum_{E_t \in R} (\delta_{E_t, \mathcal{g}_j}(\mathcal{g}_i) + \delta_{E_t, \mathcal{g}_i}(\mathcal{g}_j))
\]

A kind of node grouping w.r.t. to an attribute \( \Phi_A = \{g_1, g_2, \ldots\} \)

\[
\delta_{E_t, \mathcal{g}_j}(\mathcal{g}_i) = \begin{cases} \left| P_{E_t, \mathcal{g}_j}(\mathcal{g}_i) \right| & \text{if } p_{i,j}^t \leq 0.5 \\ \left| \mathcal{g}_i \right| - \left| P_{E_t, \mathcal{g}_j}(\mathcal{g}_i) \right| & \text{otherwise} \end{cases}
\]

\( \Delta = 0 \)
5% <= 50% (weak)
\( \Delta += 3 + 4 \) **Extra participants**
95% > 50% (strong)
\( \Delta += (100-95) + (20-19) \) **Missing participants**
K-SNAP is a **NP-Complete** problem

- For SNAP, it is computationally feasible to **exactly** perform SNAP operation.
- However, it is infeasible to find the exact **optimal** answers for the K-SNAP operation.
- In the following, we propose two **heuristic** algorithms to **approximate** K-SNAP **efficiently**.
K-SNAP Algorithm

Two approximated heuristics

• **Top-down** approach (coarse $\rightarrow$ fine)
  – Start from the maximum grouping only based on attributes
  – Iteratively split groups until group number reach k

• **Bottom-up** approach (fine $\rightarrow$ coarse)
  – First compute the maximum grouping compatible with both attributes and relationships
  – Then iteratively merge groups until the results satisfies the user defined k value
Top-Down K-SNAP Algorithm

• At each iteration, it needs to decide
  – Which group to split?
  – How to split the group?

• Heuristics

\[ CT(g_i) = \max_{g_j} \{ \delta_{g_i,g_j}(g_i) \} \]

– **Split** a group into two subgroups at each iteration
– Find \( g_i \) with the **maximum** \( \delta_{g_i,g_j}(g_i) \) by \( CT(g_i) \)
  • i.e., the **most contribution to** \( \Delta \)-value
– **Split** group \( g_i \) based on whether the nodes in \( g_i \) connect to \( g_t \)

\[ g_t = \arg \max_{g_j} \{ \delta_{g_i,g_j}(g_i) \} \]
\[ \delta_{E,t,g_j}(g_i) = \begin{cases} \left| P_{E,t,g_j}(g_i) \right| & \text{if } p_{i,j}^t \leq 0.5 \\ \left| g_i \right| - \left| P_{E,t,g_j}(g_i) \right| & \text{otherwise} \end{cases} \]

\[ CT(g_i) = \max_{g_j} \left\{ \delta_{E,g_j}(g_i) \right\} \]

\[ \delta_{\text{green}}(\text{orange}) = 4: \] #nodes in green link to nodes in orange
\[ \delta_{\text{orange}}(\text{green}) = 3: \] #nodes in orange link to nodes in green

\[ p_{i,j}^t = \frac{(3+4)}{(100+40)} = 5\% \]

\[ (3,4): 5\% \]

\[ (4,2): 10\% \]

\[ (95,19): 95\% \]

\[ (80,82): 90\% \]
\[ (75,15): 90\% \]

\[ g_t = \text{argmax}_{g_j} \{ \delta_{E,g_j}(\text{green}) \} \]
\[ g_t = \text{argmax}_{g_j} \{ \text{orange:3, blue:100-80, green:100-95} \} \]
\[ g_t = \text{blue} \]

CT(\text{purple}) = \max\{2, 20-15, 20-19\} = 5

CT(\text{orange}) = \max\{4, 40-32, 4\} = 8

CT(\text{green}) = \max\{3, 100-80, 100-95\} = 20

CT(\text{blue}) = \max\{80-70, 80-82, 80-75\} = 10

\[ g_t = \text{argmax}_{g_j} \{ \delta_{E,g_j}(g_i) \} \]
Bottom-Up K-SNAP Algorithm

• First compute the maximum (A,R)-compatible grouping
• Iteratively **merge two** groups until #group=k
• **Which two groups to merge?**
• Heuristics
  1. Same attributes values
  2. Similar participation ratio
     i.e., merge two groups with minimum $\text{MergeDist}$

\[
\text{MergeDist}(G_i, G_j) = \sum_{k \neq i,j} |p_{i,k} - p_{j,k}|
\]
Experiments on SNAP

• DBLP Database Graph
  – 7445 nodes, 19971 edges

• Node Attributes
  – PubNum: Integer
  – Prolific: (LP: [1, 5], P: [6, 20], HP: [21, -])

• Relationship: coauthorship

Resulted Summary Graph using SNAP on both Prolific and Coauthorship
3569 groups, 11293 group relationships

Yellow: HP value Prolific
Blue: P value Prolific
White: LP value Prolific
k-SNAP on DB

Attribute only

K=7

K=4

K=5

K=6

2009/11/24 SNA09F, Prof. Shou-de Lin
k-SNAP on AI

SNAP on AI

K=4

K=5

K=6

K=7

2009/11/24 SNA09F, Prof. Shou-de Lin
## SNAP Efficiency on DBLP

<table>
<thead>
<tr>
<th>Description</th>
<th>#Nodes</th>
<th>#Edges</th>
<th>Avg. Degree</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1 $DB$</td>
<td>7,445</td>
<td>19,971</td>
<td>5.4</td>
</tr>
<tr>
<td>D2 $D1+AL$</td>
<td>14,533</td>
<td>37,386</td>
<td>5.1</td>
</tr>
<tr>
<td>D3 $D2+OS+CC$</td>
<td>22,435</td>
<td>55,007</td>
<td>4.9</td>
</tr>
<tr>
<td>D4 $D3+AI$</td>
<td>30,664</td>
<td>70,669</td>
<td>4.6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th># Groups</th>
<th># Group Relationships</th>
<th>Time(sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>3569</td>
<td>11293</td>
</tr>
<tr>
<td>D2</td>
<td>7892</td>
<td>26031</td>
</tr>
<tr>
<td>D3</td>
<td>11379</td>
<td>35682</td>
</tr>
<tr>
<td>D4</td>
<td>15052</td>
<td>44318</td>
</tr>
</tbody>
</table>

- 44 sec for largest dataset (fast)
- However, the SNAP results are very large (i.e., summary size are comparable to input graphs)
- In practice, k-SNAP is more desired than SNAP
K-SNAP Efficiency

- On DBLP dataset

- On Synthetic Power-law Graphs (AvgDegree=5)
Potential Drawbacks of These Methods

• High-order relationships (combination of edge types) are ignored
• Grouping are depends on node attributes (not really community detection)
• It is hard to determine a meaningful k value for k-SNAP