Near Linear-Time Community Detection in Networks with Hardly Detectable Community Structure

Aria Rezaei
arezaei@ce.sharif.edu

Mahdieh Soleymani
solyemani@sharif.edu

Saeed Mahloujifar
mahloujifar@ce.sharif.edu
Section I

Introduction

What are communities and why are they so important?
What are communities?

• Communities are informally defined as a group of nodes that
  • are **STRONGLY** connected to each other
  • but **WEAKLY** connected to the rest of the network
What are communities?

• There is no single definition of *community*.
• We can calculate how good a group of nodes *resembles* a community.

• Popular measures include:
  • Modularity
  • Conductance
  • Expansion
  • ...
How are communities useful?

- Communities are the building blocks of Social Networks
- Analyzing a structure starts with learning about the bricks
- They are used in a variety of applications in many fields, e.g.:
  - Social Network Analysis
  - Biology
  - Software Engineering
  - Transportation
How can we retrieve communities?

• Global Optimization
  • Objective Function = Quality Measures
    • Decrease Conductance
    • Increase Modularity
    • ...
Global Optimization

• Conventional methods use global optimization

• Drawbacks
  • People don’t join communities to increase a global function! They join communities to increase their own SATISFACTION.
What is satisfaction?

- A node \( v \) is **satisfied** if he is assigned to community \( C \) while there is no other community like \( C' \) so that:

\[
d_C(v) < d_{C'}(v)
\]

Given free will, the red man will choose \( C_2 \) regardless of any global function.
What is satisfaction?

• A node $v$ is **satisfied** if it is assigned to community $C$ while there is no other community like $C'$ so that:
  $$d_C(v) < d_{C'}(v)$$

• In social networks, a **dissatisfied** node is highly undesirable!

• Conventional methods show a small number of dissatisfied nodes in their results (less than 5%).
Global Optimization

• Conventional methods use *global optimization*

• **Drawbacks**
  
  • People don’t join communities to increase a global function! They join communities to increase their own *SATISFACTION*.
  
  • The *Resolution Limit* of modularity. The inability to find communities smaller than a threshold.

• The need for *node-centered* methods is real.
• Let nodes be agents in a game with a personal utility function.
  • A nash or local equilibrium yields a community structure [Chen et al.]
• Label Propagation Algorithm
Label Propagation Algorithm

• Starts with every node holding a \textit{unique} label
• Nodes change their labels to the \textit{most frequent} label in their neighborhood.
• When there is no change in labels, we are done.

Node $v$ changing its label
Section II

Background and Motivation

Why is Label Propagation Algorithm (LPA) worth studying?
Part A

When does LPA prevail?
Label Propagation Algorithm

- Attributes:
  - Time:
    - $K$ sweeps through nodes
    - Every sweep costs $O(M)$
    - Total time: $O(KM)$
  - Raghavan et al. state that 95% of nodes reach final state in 5 iterations.
  - An upper-bound for $K$ in real networks is yet to be found.
  - Leung et al. found a $\log N$ upper-bound for $K$ in a class of networks.
Label Propagation Algorithm

• Attributes:
  • Weighted/Negative graphs.
  • No need to know the number of communities beforehand.
  • Distributed/Parallel environments ➔ Almost constant time.
  • Security ensured;
    • Nodes only receive information from their neighbors
    • Topology of the graph is unknown to nodes.
When does LPA fail?
Label Propagation Algorithm

• Drawbacks
  • Flood-fills
**Flood-Fills**

- Flood fills happen when:
  - Propagation of a label grows out of hand.
  - The corresponding community grows too fast.
  - It sweeps through all other communities.
  - The result is **ONE GIANT COMMUNITY** reported as the answer.

- This is highly undesirable.
1. A community grows out of control during the first iterations.

Sizes are proportional to the size of the corresponding communities.
2. Communities try to propagate their labels.

Sizes are proportional to the size of the corresponding communities.
3. The cores of $C_2$ and $C_3$ are still weak; they join $C_1$.

Sizes are proportional to the size of the corresponding communities
4. \( C_4 \) starts to form its core, but it is too late.

Sizes are proportional to the size of the corresponding communities
5. The propagation power of $C_1$ is much higher than others.
6. In the end, All nodes have the label representing $C_1$. 

Sizes are proportional to the size of the corresponding communities
Preventing Flood-Fills

• What is done in the past?
  • Leung et al. introduced *Hop Attenuation*
    • Labels weaken when they traverse through the network

Node $v$'s label loses its strength ($\delta$) as it traverses through the network.
What is done in the past?
- Leung et al. introduced *Hop Attenuation*
- They also introduced *Node Preference*
  - Every node shall not be treated the same
  - Using some importance function
  - A node $v$ might be more impactful than another node $u$
  - This way, we can choose the best strategy based on our needs
What is done in the past?

- Leung et al. introduced *Hop Attenuation*
- They also introduced *Node Preference*
- Barber and Clark changed *Objective Function* of LPA!
  - They realized that LPA is simply trying to put every “edge” inside a community rather than outside.
  - Global maxima is “every edge inside one giant community”.
  - They set a penalty function to prevent such actions.
Section III

Controlled Label Propagation

*Preventing flood-fills through gradual expansion of communities*
• Drawbacks
  • Flood-fills

• Why?
  • *Rapid* and *unfair* growth of a small number of communities

• Idea
  • Stop those that are growing *FAST*
  • Give enough time for those that are growing *SLOWLY*
Controlled Label Propagation

• Idea
  • Set a limit on the number of nodes holding a label.
  • *Downside*:
    • What if a community is *actually* big?
    • Nodes are *forced* to stay out of their favorite community
  • Gradually increase this limit as time goes on.
  • Nodes *might* go to the wrong community at first.
  • They will *come back* as communities expand over time.
**START**: every node holds a unique label $L_0(v)$ and $t \leftarrow 1$.

While $t < T$

$c(t)$ is the capacity for the $t$ iteration.

For all $v \in V$

$L^*$ is the most frequent label with free space in $v$’s neighborhood.

Set $L^*$ is $v$’s label.

*A label has free space if the number of nodes holding it is smaller than $c(t)$*
• The $C(t)$ function:
  • Starts at 1 and ends in $N$.
  • Any node can join any community in the last iteration.
  • Continuous increase in the capacity is not desired.
  • Expand every $T$ iterations.
  • We call the $T$ iterations between two increases in capacity, a **CYCLE**

$$C(t) = \left(\left\lfloor \frac{kt}{T} \right\rfloor + 1 \right) \times \frac{N}{k}$$
Label Propagation Algorithm

• Drawbacks
  • Flood-fills
  • Ties are broken universally randomly!
    • Too much randomness
    • Small mistakes in early steps have huge effects later on.
    • No mechanism to escape from a bad decision

• Idea
  • In early steps, change good labels for an equally good one.
  • In later steps, never change a good label for an equally good one.
**START**: every node holds a unique label \( L_0(v) \) and \( t \leftarrow 1 \).

While \( t < T \)

- \( C(t) \) is the capacity for the \( t \) iteration.
- For all \( v \in V \)
  - \( L^* \) is the most frequent label with free space in \( v \)’s neighborhood.
  - If \( P(t) \) holds, then we set \( v \)’s label \( L^* \).
  - Else, we only set \( v \)’s label \( L^* \) if \( F(L^*) > F(L_t(v)) \).

\( F(L) \) is the number of nodes holding the label \( L \).
• Attributes:
  • Compatible with previous heuristics to overcome flood-fills

Our heuristic can be used on top of recently found methods to prevent flood-fill more effectively.
Controlled Label Propagation

• Attributes:
  • Time:
    • There are $k$ cycles.
    • Every cycle is a LPA itself.
    • So, $O(MI)$ for every cycle.
    • $O(MIk)$
Experiments

Testing our algorithm using real networks and networks with planted partitions.
Controlled Label Propagation

• Attributes:
  • Accuracy:
    • How does CLPA fare on real world networks?
    • How does it prevent flood-fills?
    • What is the cost of preventing flood-fills?
      • More time?
      • More memory?
      • Less Accuracy?
Part A

Real Networks
Controlled Label Propagation

• Attributes:
  • Accuracy:
    • First experiment: Real world networks
      • Does accuracy fall when trying to prevent flood-fills?
      • Does it increase or decrease the overall quality of communities?
Real World Networks - Setup

• Networks:
  • 6 co-authorship network: GRQ, HepTh, HepPh, Astro, CondMat, DBLP
  • Two location-based social network: Gowalla, Brightkite
  • An email client named Enron
  • A piece of Youtube’s network
  • A piece of Amazon’s network

• Measure: Modularity
Real World Networks - Results

Modularity of CLPA and LPA results on 11 real world networks. The difference is also shown in percentage.

<table>
<thead>
<tr>
<th>Name</th>
<th>N</th>
<th>M</th>
<th>CLPA</th>
<th>LPA</th>
<th>Δ(CLPA, LPA)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GRQ</td>
<td>5.24K</td>
<td>14.48K</td>
<td>0.797</td>
<td>0.735</td>
<td>8.435374 %</td>
</tr>
<tr>
<td>HepTh</td>
<td>9.88K</td>
<td>25.97K</td>
<td>0.671</td>
<td>0.627</td>
<td>7.017544 %</td>
</tr>
<tr>
<td>HepPh</td>
<td>12.01K</td>
<td>0.12M</td>
<td>0.497</td>
<td>0.488</td>
<td>1.844262 %</td>
</tr>
<tr>
<td>CondMat</td>
<td>12.01K</td>
<td>0.12M</td>
<td>0.633</td>
<td>0.578</td>
<td>9.515571 %</td>
</tr>
<tr>
<td>Astro</td>
<td>18.77K</td>
<td>0.20M</td>
<td>0.45</td>
<td>0.323</td>
<td>39.31889 %</td>
</tr>
<tr>
<td>Enron</td>
<td>36.69K</td>
<td>0.18M</td>
<td>0.473</td>
<td>0.338</td>
<td>39.94083 %</td>
</tr>
<tr>
<td>Brightkite</td>
<td>58.23K</td>
<td>0.21M</td>
<td>0.623</td>
<td>0.557</td>
<td>11.84919 %</td>
</tr>
<tr>
<td>Gowalla</td>
<td>0.20M</td>
<td>0.95M</td>
<td>0.618</td>
<td>0.503</td>
<td>22.86282 %</td>
</tr>
<tr>
<td>DBLP</td>
<td>0.32M</td>
<td>1.05M</td>
<td>0.697</td>
<td>0.622</td>
<td>12.05788 %</td>
</tr>
<tr>
<td>Amazon</td>
<td>0.33M</td>
<td>0.93M</td>
<td>0.786</td>
<td>0.709</td>
<td>10.86037 %</td>
</tr>
<tr>
<td>Youtube</td>
<td>1.13M</td>
<td>2.99M</td>
<td>0.682</td>
<td>0.555</td>
<td>22.88288 %</td>
</tr>
</tbody>
</table>
Part B

Networks with Planted Partitions
Controlled Label Propagation

• Attributes:
  • Accuracy:
    • First experiment: Real world networks
      • Does accuracy fall when trying to prevent flood-fills?
      • Does it increase or decrease the overall quality of communities?
    • Second experiment: Networks with planted partition
      • How does CLPA exactly prevent flood-fills?
      • How does different characteristics of a network contribute to CLPA’s success or failure?
• Networks:
  • Lancichinetti et al. benchmark.
  • There are three features for a network:
    1. Maximum degree
    2. Mean degree
    3. Mixing parameter ($\mu$): for every $v$, $\mu = \frac{N(\text{neighbors of links outside of its community})}{N(\text{neighbors of links inside its community})}$
• As $\mu$ grows, detecting communities becomes harder.
A network with $d_{\text{max}} = 100$ and $\bar{d} = 20$

Networks with Planted Partition - Results
Networks with Planted Partition - Results

A network with $d_{\text{max}} = 150$ and $\bar{d} = 30$
A network with $d_{\text{max}} = 200$ and $\bar{d} = 40$
Dissatisfaction rate increases rapidly when \( \mu \) passes a threshold.
Landing points of different algorithms on three networks.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>20 - 100</th>
<th>30 - 150</th>
<th>40 - 200</th>
</tr>
</thead>
<tbody>
<tr>
<td>LPA</td>
<td>0.6</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>DPA</td>
<td>0.7</td>
<td>0.65</td>
<td>0.65</td>
</tr>
<tr>
<td>BPA</td>
<td>0.55</td>
<td>0.6</td>
<td>0.5</td>
</tr>
<tr>
<td>CLPA</td>
<td>0.7</td>
<td>0.75</td>
<td>0.75</td>
</tr>
<tr>
<td>Infomap</td>
<td>0.6</td>
<td>0.55</td>
<td>0.55</td>
</tr>
<tr>
<td>CNM</td>
<td>0.55</td>
<td>0.65</td>
<td>0.65</td>
</tr>
<tr>
<td>Louvain</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
</tr>
</tbody>
</table>

Landing point of a curve. We have set the threshold here to be 0.5.
Further Works

*What can be the next step?*
Pros:
• Fast.
• Flexible.
• Ensures security.
• Scalable.

Cons:
• Flood-fills
• Low accuracy
• Too much randomness
Flood-fills

• There are two ways to overcome flood-fills
  • Change objective function of LPA (Barber & Clarks)
  • Monitor propagation of labels; Stop “over”-propagations

• Unanswered questions on flood-fills
  • In what networks do they appear?
  • Can we anticipate flood-fills before running the algorithm?
  • Can we detect communities that are more likely to flood the graph?
Future Work

• Our algorithm looks at the problem in a different angle
• Controlling the propagation manually is **BAD**.
• Bias is induced to solution.
• Can we find some adaptive function for $C(t)$?
• Can we find other methods preventing flood-fills from the **VERY FIRST ITERATION**?
Thank you everyone.

That is it for today ...