Efficient and principled Method for detecting Communities in Networks

Brian Ball, Brian Karrer, and M. E. J. Newman

*Physical Review*, 2011
Outline

1. Introduction
2. A generative model for link communities
3. Detecting overlapping communities
4. Implementation
5. Test on synthetic and real networks
6. Detecting non-overlapping communities
7. Results for running time
8. Conclusion
The concept of “community”

- There is no generally accepted definition of what a community is.

- Communities in a network are groups of vertices:
  - with relatively dense connections within groups
  - but sparser connections between them.

- **Note:** They may be disjoint or overlapping.
Detecting communities is a fundamental problem.
The authors derive an algorithm for community detection applied to overlapping and non-overlapping communities.

It gives good results on both real-world networks and synthetic benchmarks, competitive with previous methods.
Figure: An illustration for a result of the algorithm.
A Generative Network Model

Definition 1

- with a given number \( n \) of vertices
- with a given number \( K \) of communities
- has \( n \cdot K \) parameters \( \theta_{iz} \) representing the propensity of vertex \( i \) to have edges of color \( z \).

\[
E_{ijz} \sim \begin{cases} 
\text{Pois}(\theta_{iz} \cdot \theta_{jz}) & \text{if } i \neq j \\
\text{Pois}(\frac{1}{2} \cdot \theta_{iz} \cdot \theta_{jz}) & \text{if } i = j
\end{cases}
\] (1)
Why Poisson Distribution

Consider the following sampling process of links: Assume the sample size to be $m$ which is the degree of the network. The probability of forming a link $E_{ijz}$ follows Binomial distribution:

$$P(E_{ijz}) = \binom{m}{E_{ijz}} P(x_i, x_j, z)^{E_{ijz}} \times \left(1 - P(x_i, x_j, z)\right)^{m-E_{ijz}}.$$  

(a)

Under the condition $m \to \infty$, we have

$$\lim_{m \to \infty} P(E_{ijz}) = \frac{(m \cdot P(x_i, x_j, z))^{E_{ijz}}}{E_{ijz}!} e^{-m \cdot P(x_i, x_j, z)}.$$  

(b)
Poisson Distribution (Cont.)

\[
P(E_{ijz}) \approx \frac{(m \cdot P(x_i, x_j, z))^{E_{ijz}}}{E_{ijz}!} e^{-m \cdot P(x_i, x_j, z)}
\]

(c)

By factorizing \(m \cdot P(x_i, x_j, z)\) into two symmetric parts:

\[
\left( \sqrt{m \cdot P(z)P(x_i|z)} \right) \cdot \left( \sqrt{m \cdot P(z)P(x_j|z)} \right)
\]

We now have:

\[
\theta_{iz} = \sqrt{m \cdot P(z)P(x_i|z)}
\]

(d)

So for each \(E_{ijz}\)

\[
E_{ijz} \sim \text{Pois}(\theta_{iz} \cdot \theta_{jz})
\]

(e)
Likelihood

As \( E_{ij} = \sum_z E_{ijz} \),

\[
\lim_{m \to \infty} \frac{P(E_{ij}|\theta)}{E_{ij}!} = \frac{(\sum_z \theta_{iz} \cdot \theta_{jz})^{E_{ij}}}{E_{ij}!} e^{-\sum_z \theta_{iz} \cdot \theta_{jz}}
\]

Likelihood over the whole network:

\[
P(G|\theta) = \prod_{i,j} P(E_{ij}|\theta_i, \theta_j) \approx \prod_{i,j} \frac{(\sum_z \theta_{iz} \cdot \theta_{jz})^{E_{ij}}}{E_{ij}!} e^{-\sum_z \theta_{iz} \cdot \theta_{jz}}
\]
Detecting Overlapping Communities

- Given a undirected graph \( G \) with \( n \times n \) adjacency matrix \( A, (A_{ij} = 0, 1 \text{ or } 2) \)
- We fit the model to \( G \) by maximizing \( \log P(G | \theta) \)

\[
\log P(G | \theta) = \sum_{ij} A_{ij} \cdot \log(\sum_{z} \theta_{iz} \cdot \theta_{jz}) - \sum_{ij} \theta_{iz} \cdot \theta_{jz}
\]

\[
\geq \sum_{ij} [A_{ij} \cdot q_{ij}(z) \cdot \log \frac{\theta_{iz} \cdot \theta_{jz}}{q_{ij}(z)} - \theta_{iz} \cdot \theta_{jz}] \quad (2)
\]

for all \( q_{ij}(z) > 0 \) satisfying \( \sum_{z} q_{ij}(z) = 1. \)
Detecting Overlapping Communities

Maximizing the log likelihood is to solve the following equations:

\[ q_{ij}(z) = \frac{\theta_{iz} \cdot \theta_{jz}}{\sum_z \theta_{iz} \cdot \theta_{jz}} \tag{3} \]

\[ \theta_{iz} = \sqrt[\frac{1}{2}]{\sum_j A_{ij} \cdot q_{ij}(z)} \]

choose a random set of initial values.

iterate until \( \theta_{iz}^{(l)} \) converge.

This is the Expectation-Maximization (EM) algorithm.
If $q_{ij}(z_0) = \max_z q_{ij}(z)$ then the edge connecting $i$ and $j$ belongs to community $z_0$.

One vertex can belong to more than one community.
To save memory use, we calculate number of edges of color $z$ connected to $i$:

$$k_{iz} = \sum_j A_{ij} \cdot q_{ij}(z).$$  \hfill (5)

Firstly, initialize $\{k_{iz}\}$ then compute new values $\{k_{iz}\}$ until $\{k_{iz}\}$ converge.

Since most of them converge to 0, hence the authors give 2 strategies for pruning set of variables by a certain threshold $\delta$. 

Efficient and principled Method for detecting Communities in Networks
Two Strategies

\[ k_{iz} = \sum_j A_{ij} \cdot q_{ij}(z) \]

\[ q_{ij}(z) = \frac{k_{iz} \cdot k_{jz}}{\sum_z k_{iz} \cdot k_{jz}} \]

Noticing

1. Many of \( k_{iz} \)'s are close to zero under the EM iteration, we no longer need to update them and we can exclude them from the calculations. A threshold \( \delta \) controls this.

2. \( q_{ij}(z) \)'s on all their adjacent links are also zero and therefore need not be calculated.
Test on synthetic networks with known communities and having similar properties.

Calculate the fractions of vertices assigned to the correct community.

Calculate the Jaccard index.
Test on real-world networks

Figure: Karate club network.
Test on real-world networks

Figure: The network of characters from Les Misérables.
Figure: Overlapping communities in the network of US passenger air transportation. The three communities produced by the calculation correspond roughly to the east and west coasts of the country and Alaska.
Detecting non overlapping communities

Supplement an extra step: assigning vertices to the community for which the value $\frac{k_{iz}}{k_z}$ is largest.
It can be verified that

\[
\frac{k_{iz}}{k_i} = \frac{\sum_j A_{ij} \cdot q_{ij}(z)}{\sum_{ij} A_{ij} \cdot q_{ij}(z)} = P(x_i | z) \quad (i)
\]

So it is essentially a ranking instead of a clustering.

A better way to detect non-overlapping communities is to directly find

\[
\arg\max_z k_{iz}
\]
It can be verified that

$$\frac{k_{iz}}{k_i} = \frac{\sum_j A_{ij} \cdot q_{ij}(z)}{\sum_{ij} A_{ij} \cdot q_{ij}(z)} = P(x_i|z)$$

So it is essentially a ranking instead of a clustering.

A better way to detect non-overlapping communities is to directly find

$$\arg\max_z k_{iz}$$
We have 3 different algorithms:

- Naive, $\delta = 0$: using naive EM algorithm.
- Fast, $\delta = 0$: using pruned algorithm with $\delta = 0$.
- Fast, $\delta = 0.001$: using pruned algorithm with $\delta = 0.001$
### Results for running time

<table>
<thead>
<tr>
<th>Running conditions</th>
<th>Time (s)</th>
<th>Iterations</th>
<th>Log-likelihood</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>US air transportation, n = 700, m = 3327, K = 3</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>naive, $\delta = 0$</td>
<td>15.71</td>
<td>55719</td>
<td>-8924.58</td>
</tr>
<tr>
<td>fast, $\delta = 0$</td>
<td>14.67</td>
<td>55719</td>
<td>-8924.58</td>
</tr>
<tr>
<td>fast, $\delta = 0.001$</td>
<td>2.17</td>
<td>26063</td>
<td>-9074.21</td>
</tr>
<tr>
<td><strong>Network science collaborations [43], n = 379, m = 914, K = 3</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>naive, $\delta = 0$</td>
<td>0.93</td>
<td>13165</td>
<td>-3564.74</td>
</tr>
<tr>
<td>fast, $\delta = 0$</td>
<td>0.82</td>
<td>13165</td>
<td>-3564.74</td>
</tr>
<tr>
<td>fast, $\delta = 0.001$</td>
<td>0.13</td>
<td>10747</td>
<td>-3577.85</td>
</tr>
<tr>
<td><strong>Network science collaborations, n = 379, m = 914, K = 10</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>naive, $\delta = 0$</td>
<td>3.19</td>
<td>18246</td>
<td>-2602.15</td>
</tr>
<tr>
<td>fast, $\delta = 0$</td>
<td>3.15</td>
<td>18246</td>
<td>-2602.15</td>
</tr>
<tr>
<td>fast, $\delta = 0.001$</td>
<td>0.49</td>
<td>12933</td>
<td>-2611.96</td>
</tr>
<tr>
<td><strong>Network science collaborations, n = 379, m = 914, K = 20</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>naive, $\delta = 0$</td>
<td>6.16</td>
<td>19821</td>
<td>-2046.95</td>
</tr>
<tr>
<td>fast, $\delta = 0$</td>
<td>6.09</td>
<td>19821</td>
<td>-2046.95</td>
</tr>
<tr>
<td>fast, $\delta = 0.001$</td>
<td>0.94</td>
<td>14010</td>
<td>-2094.85</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Running conditions</th>
<th>Time (s)</th>
<th>Iterations</th>
<th>Log-likelihood</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Political blogs [44], n = 1490, m = 16778, K = 2</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>naive, $\delta = 0$</td>
<td>11.42</td>
<td>13773</td>
<td>-48761.1</td>
</tr>
<tr>
<td>fast, $\delta = 0$</td>
<td>11.46</td>
<td>13773</td>
<td>-48761.1</td>
</tr>
<tr>
<td>fast, $\delta = 0.001$</td>
<td>4.14</td>
<td>13861</td>
<td>-48765.6</td>
</tr>
<tr>
<td><strong>Physics collaborations [45], n = 40421, m = 175693, K = 2</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>naive, $\delta = 0$</td>
<td>4339.57</td>
<td>424077</td>
<td>-1.367 x 10^6</td>
</tr>
<tr>
<td>fast, $\delta = 0$</td>
<td>2557.91</td>
<td>424077</td>
<td>-1.367 x 10^6</td>
</tr>
<tr>
<td>fast, $\delta = 0.001$</td>
<td>253.41</td>
<td>61665</td>
<td>-1.378 x 10^6</td>
</tr>
<tr>
<td><strong>Amazon copurchasing [46], n = 403394, m = 2443408, K = 2</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>naive, $\delta = 0$</td>
<td>170646.9</td>
<td>1222937</td>
<td>-2.521 x 10^7</td>
</tr>
<tr>
<td>fast, $\delta = 0$</td>
<td>105042.3</td>
<td>1222937</td>
<td>-2.521 x 10^7</td>
</tr>
<tr>
<td>fast, $\delta = 0.001$</td>
<td>11635.0</td>
<td>120612</td>
<td>-2.538 x 10^7</td>
</tr>
<tr>
<td><strong>LiveJournal [47, 48], n = 4847571, m = 42851237, K = 2</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fast, $\delta = 0$</td>
<td>39834.3</td>
<td>26163</td>
<td>-4.611 x 10^8</td>
</tr>
<tr>
<td>fast, $\delta = 0.001$</td>
<td>3093.7</td>
<td>1389</td>
<td>-4.660 x 10^8</td>
</tr>
</tbody>
</table>
Conclusion

Advantages
- detects on both overlapping and non-overlapping communities.
- has a rigorous mathematical foundation.
- can apply to networks of millions of vertices.
- gives results competitive with previous algorithms.

Disadvantages
- offers no criterion for determining $K$. 

1. This generative model can be used to well understand the formation of the links based on the clustering or community detection.

2. Too many parameters to estimate \((nK + mK)\), easily to get trapped in local optimum and over-fitting.

3. The update functions are exactly the same with PLSA proposed for bipartite graph.

4. Through experiments, this model is not robust to unbalanced networks.
In this paper, as the network is homogeneous, $P(z)$ can be absorbed into $\theta_{iz}$. 

$$
P(w, d) = \sum_z P(w|z)P(d|z)P(z)$$

$$
\implies \prod_{\langle w, d \rangle} P(w|z)P(d|z)P(z)
$$

$$
\implies P(z|w, d) = \frac{P(w|z)P(d|z)P(z)}{\sum_z P(w|z)P(d|z)P(z)}
$$

$$
P(d|z) = \frac{\sum_w c(w, d)P(z|w, d)}{\sum_{w,d} c(w, d)P(z|w, d)}
$$

$$
P(w|z) = \frac{\sum_d c(w, d)P(z|w, d)}{\sum_{w,d} c(w, d)P(z|w, d)}
$$