Spectral Graph Multisection Through Orthogonality

Huanyang Zheng and Jie Wu
CIS Department, Temple University
Outline

- Motivation
- Preliminary
- Algorithm
- Evaluation
- Future work
Motivation

- Traditional graph spectral clustering algorithm is based on recursive bisection.

  (a) 

  (b)

- Recursive bisection: 4/2/2 nodes
- Optimal partition: 3/3/2 nodes
Motivation

- Can we cut the graph into multissections directly?

- Yes, but former approaches on graph multisection have a high time complexity and a low partition accuracy.

- We propose a multisection algorithm with a low time complexity and a competitive partition accuracy.
Preliminary

• Spectral bisection algorithm by Dr. Newman

• Let $s_i$ node $i$.
  
  $s_i = \begin{cases} 
  +1 & \text{if node } i \text{ belongs to group 1.} \\
  -1 & \text{if node } i \text{ belongs to group 2.} 
  \end{cases}$

• We can use $s_i$ and $s_j$ to indicate whether

  $\frac{1}{2} (s_is_j + 1) = \begin{cases} 
  1 & \text{if node } i \text{ and } j \text{ belong to the same community} \\
  0 & \text{otherwise} 
  \end{cases}$

not:
Preliminary

• Modularity is the partition metric, if node \( i \) and \( j \) belong to the same community, then the modularity gain is \( B_{ij} \).

• So the total modularity (or quality) is

\[
Q = \frac{1}{4m} \sum_{i,j} B_{ij} (s_is_j + 1) = \frac{1}{4m} \sum_{i,j} B_{ij} s_is_j
\]

• Since \( \sum_{i,j} B_{ij} = 0 \).
Preliminary

- **Relax** the constraint $s_i \in \{-1, +1\}$ and $\sum_i s_i^2 = n$, if there are $n$ nodes in total.

- This is a classic optimization (maximize the modularity under the above constraint), which can be solved by Lagrange multipliers:

  $$\frac{\partial}{\partial s_i} \left[ \sum_{ij} B_{ij} s_is_j + \beta(n - \sum_i s_i^2) \right] = 0$$
• Then (use a vector $s$ to denote $[s_1 \ s_2 \ldots]$

\[ \sum_j B_{ij} s_j = \beta s_i, \text{ or in matrix notation, } Bs = \beta s. \]

• This implies that the vector $s$ should be an eigenvector of $B$.

• Recall

\[ Q = \frac{1}{4m} s^T Bs = \frac{1}{4m} s^T \beta s = \frac{n}{4m} \beta \]
Preliminary

• To maximize $Q$, the vector $s$ should be the eigenvector corresponding to the largest eigenvalue of the modularity matrix.

• Each element in this eigenvector stands for the group allocation of the corresponding node.

• Round each element in the eigenvector to $\{-1,+1\}$, then we obtain the partition result.
Algorithm

• Basic idea: use a vector to present the group allocation, ins $\pm 1$ d of  
  (we use more than one bit to present group allocation).

$$\bar{s}_i = \begin{cases} 
  h_1 & \text{if node } i \text{ belongs to group 1.} \\
  h_2 & \text{if node } i \text{ belongs to group 2.} \\
  \vdots & \text{ } \\
  h_K & \text{if node } i \text{ belongs to group } K. 
\end{cases}$$

• Our algorithm:

$$s_i = \begin{cases} 
  +1 & \text{if node } i \text{ belongs to group 1.} \\
  -1 & \text{if node } i \text{ belongs to group 2.} 
\end{cases}$$

• Classic approach:
Algorithm

• These vectors are mutually orthogonal to each other, which are produce by Hadamard matrix (hi is the i-th row of it):

  \[ H1 = [1], \quad H2 = \begin{bmatrix} +1 & +1 \\ +1 & -1 \end{bmatrix}, \quad H2K = \begin{bmatrix} +H_K & +H_K \\ +H_K & -H_K \end{bmatrix} \]

• Our approach \( \overline{s_i s_j} = \begin{cases} 1 \\ 0 \end{cases} \) vs classi \( \frac{1}{2} (s_i s_j + 1) = \begin{cases} 1 \\ 0 \end{cases} \)

• For example, \( \overline{s_i} = [+1 \quad +1] \quad \text{and} \quad \overline{s_j} = [+1 \quad -1] \)
Algorithm

- Basic idea: use a self-defined operation, *matrix inflation*, to present the modularity.

- Our approach:
  \[ Q = \frac{1}{2Km} \bar{s}^T \overline{B_K} \bar{s} \]

- Classic approach:
  \[ Q = \frac{1}{4m} s^T B s \]

- Then the following process is the same as the classic approach, but now we can do multisection directly.
Algorithm

- Self-defined operation: matrix inflation
- Definition: the Kronecker product of the matrix $M$ and a $K \times K$ identity matrix.
- Example

$$M = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \quad \text{and} \quad \overline{M_2} = \begin{bmatrix} 1 & 0 & 2 & 0 \\ 0 & 1 & 0 & 2 \\ 3 & 0 & 4 & 0 \\ 0 & 3 & 0 & 4 \end{bmatrix}$$
Algorithm

- Additional issue: we use a randomized matrix inflation to keep relaxation effective.

- Time complexity (the graph has $n$ nodes):
  - Our method: $O(K4n^2)$, where $K$ is the estimated number of communities.
  - Recursive bisection algorithm: $O(n^2 \log n)$. 
Evaluation

• Our evaluations are based on the LFR benchmark, where the node degree and the community size follow power-law, with exponents $\gamma$ and $\beta$, respectively.

• Links between nodes in the same (different) community are called internal (external) links. A *mixing parameter*, $u$, is the ratio of the external node degree to the total degree.
Evaluation

- Algorithms in comparison:
  - The recursive bisection algorithm (denoted as RBS) by Dr. Newman.
  - The Markov Cluster algorithm (MCL).
  - Proposed algorithm is denoted as PMS.
Evaluation

- LFR benchmark with N=128 nodes
• LFR benchmark with N=256 nodes
Evaluation

- Simulation summary:
  - Our algorithm outperforms recursive bisection algorithm. This is because our algorithm has “global” view of the partition, and recursive bisection is based on the “local” view.
  - Our algorithm has a competitive performance with a low time complexity.
Future work

• Real data evaluation.
• Tests in large-scale networks.
Thank you!

Q & A