Spectral Graph Multisection Through Orthogonality

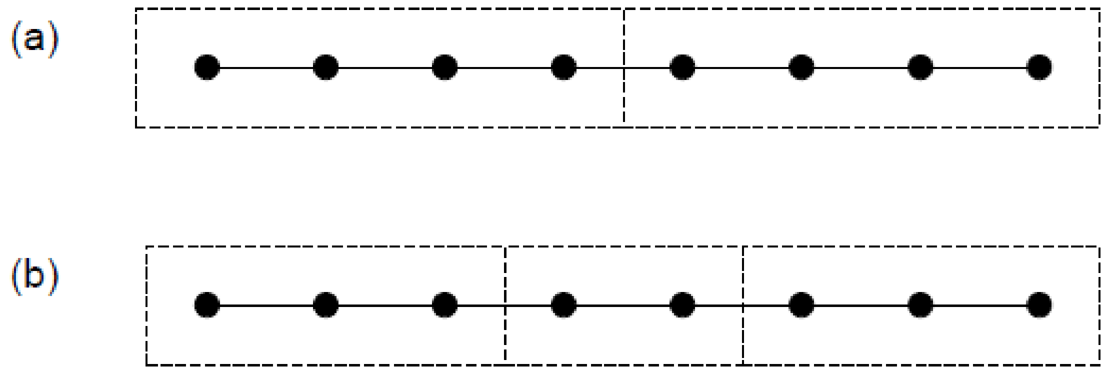
Huanyang Zheng and Jie Wu
CIS Department, Temple University

Outline

- Motivation
- Preliminary
- Algorithm
- Evaluation
- Future work

Motivation

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



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

Motivation

- Can we cut the graph into multisections directly?

- Yes, but former approaches on graph multusection 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 *si* 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 *si* and *sj* to indicate whether

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

not:

Preliminary

- Modularity is the partition metric, if node *i* and *j* belong to the same community, then the modularity gain is *Bij*.

- So the total modularity (or quality) is

$$Q = \frac{1}{4m}\sum_{ij} B_{ij}(s_i s_j + 1) = \frac{1}{4m}\sum_{ij} B_{ij} s_i s_j$$

- Since $\sum_{ij} B_{ij} = 0$,

Preliminary

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

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

$$\frac{\partial}{\partial s_i}\left[\sum_{ij} B_{ij} s_i s_j + \beta\left(n - \sum_i s_i^2\right)\right] = 0$$

Preliminary

- Then (use a vector *s* to denote [*s1 s2 …*

  $\sum_j B_{ij} s_j = \beta s_i$, 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 ± 1 d of (***we use more than one bit to present group allocation***).

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

- 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*], $H_2 = \begin{bmatrix} +1 & +1 \\ +1 & -1 \end{bmatrix}$, $H_{2K} = \begin{bmatrix} +H_K & +H_K \\ +H_K & -H_K \end{bmatrix}$
- Our approach $\overline{s_\iota}\,\overline{s_J} = \begin{cases} 1 \\ 0 \end{cases}$ vs classi $\frac{1}{2}(s_i s_j + 1) = \begin{cases} 1 \\ 0 \end{cases}$
- For example, $\overline{s_\iota} = [+1 \;\; +1]$ and $\overline{s_J} = [+1 \;\; -1]$

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*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(K4n2)*, where *K* is the estimated number of communities.

  - Recursive bisection algorithm: *O(n2logn)*.

Evaluation

- Our evaluations are based on the LFR benchmark, where the node degree and the community size follow power-law, with exponents β γ and , respectively.

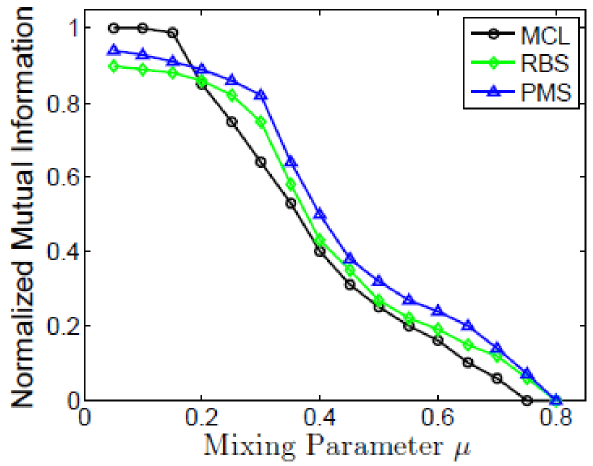
- Links between nodes in the same (different) community are called internal (external) links. A ***mixing parameter***, u, is the ratio of the external node degree to the total degree.

Evaluation

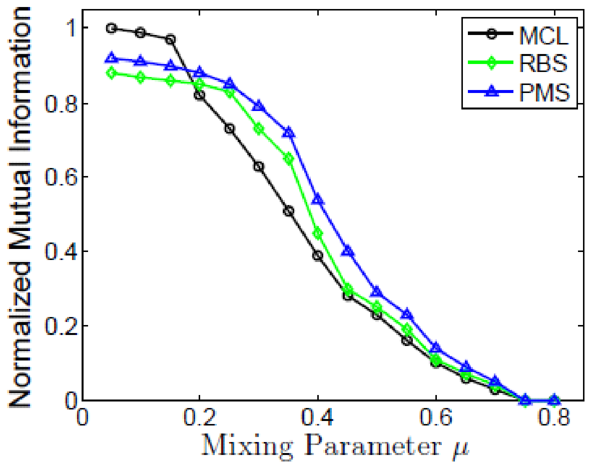
- Algorithms in comparison:
  - Ø The recursive bisection algorithm (denoted as RBS) by Dr. Newman.
  - Ø The Markov Cluster algorithm (MCL).
  - Ø Proposed algorithm is denoted as PMS.

Evaluation

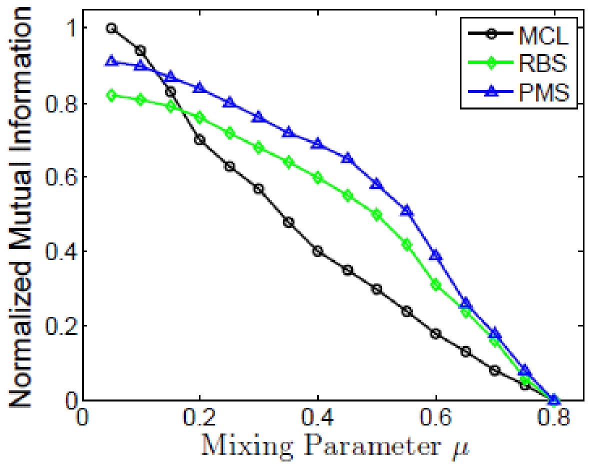
- LFR benchmark with N=128 nodes



(a) $\gamma = 3$, $\beta = 2$



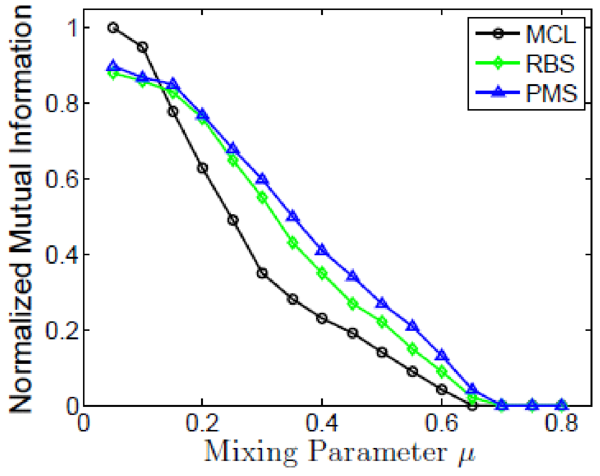
(b) $\gamma = 2$, $\beta = 2$



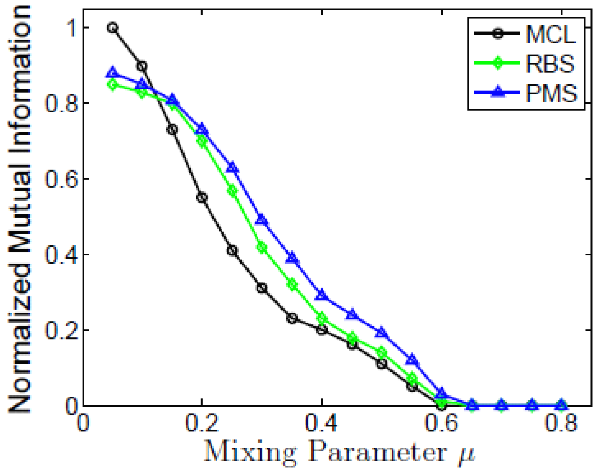
(c) $\gamma = 2$, $\beta = 3$

Evaluation

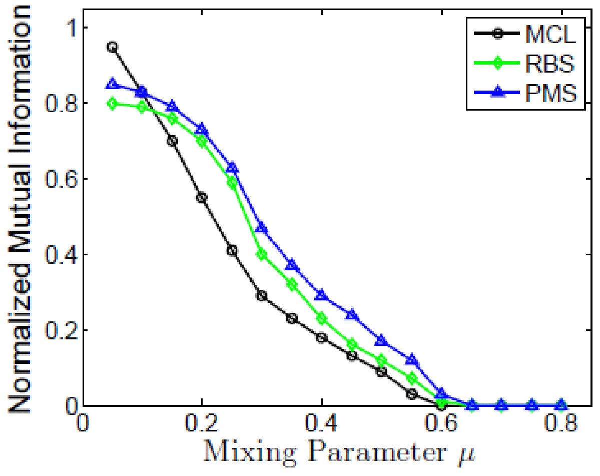
- LFR benchmark with N=256 nodes



(a) $\gamma = 3,\ \beta = 2$



(b) $\gamma = 2,\ \beta = 2$



(c) $\gamma = 2,\ \beta = 3$

Evaluation

- Simulation summery:

- 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