SC435
Introduction to Complex Networks

Community Detection

Graph Partitioning & Community Detection

- Task: Divide the nodes of a network into a set of groups so as to minimize the number of edges running between them. division is based on the connection pattern of the edges.
- Useful in revealing patterns and organization in a network beyond the scale of a single vertex.
- Depending on whether we have specified the number of groups
  - Graph Partitioning
  - Community Detection
- Different algorithms that deal with each one of the problems.
  - In graph Partitioning the number of groups k is predefined.$\Rightarrow$ divide the network vertices into k non-overlapping groups of given sizes such that the number of edges across groups are minimized
  - In community detection neither the number of groups nor their sizes are pre-defined.$\Rightarrow$ *Few* edges between groups and *many* edges within groups

Graph Partitioning

Bisection

- **Graph Bisection:** Divide a network into two non-overlapping groups such that the number of links between the nodes in the two groups (cut-size) is minimized.
- Inspect all the possible divisions into two groups and choose the one with the smallest cut-size $\binom{n}{n_1}$.
- Number of bisections increases exponentially with the size of the network. For $n_1 = n_2$, number of bisections $\sim \frac{2^{n+1}}{\sqrt{n}} = e^{(n+1)ln2} - e^{\frac{1}{2}ln(n)}$

Graph Partitioning

Kernighan-Lin Algorithm (greedy-heuristic)

1. Randomly split network (nodes) into groups.
2. Find the pair $i \in$ Group 1, $j \in$ Group 2 that if swapped reduces the cut-set size the most. All pairs of i, j must be tested. If no pair reduces the cut-set size then pick the pair that increases it the most.
3. Total change in cut-size

$$\boxed{\Delta = k_i^{ext} - k_i^{int} + k_j^{ext} - k_j^{int} - 2A_{ij}}$$

4. Continue (without picking the same nodes again) until there are no nodes remaining that have not been picked.
5. Choose the Partitioning that has the smallest cut.
6. Start with this partition and go to 2.
7. Repeat many times.

Graph Partitioning

Spectral Partitioning

- Total number of edges running between two groups

$$\boxed{R = \frac{1}{2} \sum_{i,j \text{ in diff. groups}} A_{ij}}$$

- Define s_i, one for each node i

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

$$\frac{1}{2}(1 - s_i s_j) = \begin{cases} 1, & \text{if } i \text{ and } j \text{ are in different groups} \\ 0, & \text{if } i \text{ and } j \text{ are in the same group} \end{cases}$$

- cut-size

$$R = \frac{1}{4} \sum_{i,j} A_{ij}(1 - s_i s_j) = \frac{1}{4} \mathbf{s}^{\mathsf{T}} \mathbf{L} \mathbf{s} = \sum_{i,j} (s_i - s_j)^2$$

Graph Partitioning is equivalent to finding the ground state of a certain type of Ising model in which the spins live on the nodes of the network.

Graph Partitioning

Spectral Partitioning

- Goal: Find $\mathbf{s}$ that minimizes the cut-size (Discrete Optimization problem)
- For bisection of equal size

$$\sum_i s_i = 0$$

- Approximate solution:
  - Let $\mathbf{s} = \sum_i a_i \mathbf{u}_i$.
  - Since $\mathbf{s}$ is orthogonal to $\mathbf{u}_i = \mathbf{1}/\sqrt{n}$. Therefore, $a_1 = 0$.

$$R = \frac{1}{4} \sum_{i=2}^{n} a_i^2 \lambda_i$$

  - Since

$$R = \frac{1}{4} \sum_{i=2}^{n} a_i^2 \lambda_i \geq \frac{n\lambda_2}{4}$$

  - Finding minimum of R is equivalent to setting $s_i = +(-1)$ if the *ith* component of u_2 is positive(negative).

Community Detection

Modularity Maximization

- Community detection problem is not well posed.
- It has been argued that cut size is not a good metric for evaluating community detection algorithms.
- Modularity Maximization $\Rightarrow$ Assortative mixing (division is one where edges across groups are less than the expected ones if edges were places in random)
- Quantitative Measure $\Rightarrow$ Modularity (high when connections are between nodes of the same type, low otherwise)
- Still a hard problem: Calculating modularity over all partitions is hard.

Modularity Maximization (Bisection)

- Modularity and form of modularity function

$$Q = \frac{1}{2m} \sum_{ij} \left(A_{ij} - \frac{k_i k_j}{2m} \right) \delta(c_i, c_j)$$

$$B_{ij} = A_{ij} - \frac{k_i k_j}{2m}$$

- B_{ij} has the property that its sum with either of the indices is zero

$$\sum_i B_{ij} = \sum_i A_{ij} - \frac{k_j}{2m} \sum_i k_i = k_i - k_j = 0$$

$$\sum_j B_{ij} = \sum_j A_{ij} - \frac{k_i}{2m} \sum_j k_j = k_j - k_j = 0$$

Modularity Maximization Problem (Bisection)

- Consider two groups i and j and define

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

$$\frac{1}{2}(1 + s_i s_j) = \begin{cases} 0, & \text{if } i \text{ and } j \text{ are in different groups} \\ 1, & \text{if } i \text{ and } j \text{ are in different groups} \end{cases}$$

- Notice,

$$\delta(c_i, c_j) = \frac{1}{2}(1 + s_i s_j)$$

- Then,

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

- Modularity Maximization Problem $\Rightarrow$ Given a particular set of values of B_{ij} find the quantities $s_i = \pm 1$ that maximize Q (discrete optimization problem)
- For small sized networks methods such as simulated annealing, genetic algorithms, extremal optimization work well.

Community Detection

Modularity Maximization (Random division of network into two equally sized groups)

1. Start with a random division of network into two equally sized groups.
2. At each step go through all the vertices and compute the change in modularity if the vertex was placed in the other group.

$$\Delta Q = -\frac{s_v}{m}\sum_{i:i\neq v} B_{iv}s_i = -\frac{s_v}{m}\left(\sum_{i:i\neq v} A_{iv}s_i - \sum_i \frac{k_i k_v}{2m}s_i + \frac{k_v^2 s_v}{2m}\right)$$

3. Among all the vertices examined swap groups to the one that has the best effect on the modularity (largest increase or smallest decrease).
4. Repeat the process by considering only the vertices that have not been already swapped.
5. As with Kernighan-Lin Algorithm after we go through all the vertices, we go through all the states of the network and keep the one with the higher modularity.
6. Repeat many times by starting with 2.

Spectral Modularity Maximization

- Notice that like L in Spectral Partitioning B also has a quadratic form

$$\boxed{Q = \frac{1}{4m}\mathbf{s}^{\mathbf{T}}\mathbf{B}\mathbf{s}}$$

- Maximizing Q with $s_i \in \{-1, 1\}$ is an integer optimization problem (challenging)
- Trick: Relaxation method

$$\boxed{\max_{s_i \in \{-1,1\}} \frac{1}{4m}\mathbf{s}^{\mathbf{T}}\mathbf{B}\mathbf{s} \sim \max_{s_i \in \mathbb{R}} \frac{1}{4m}\mathbf{s}^{\mathbf{T}}\mathbf{B}\mathbf{s}}$$

- Constraint of the main problem: $\mathbf{s}^{\mathbf{T}}\mathbf{s} = \mathbf{n}$
- Calculate the maxima allowing for constraints.
- Lagrangian multiplier method

$$\frac{\partial}{\partial \mathbf{s}}\left[\mathbf{s}^{\mathbf{T}}\mathbf{B}\mathbf{s} + \lambda(n - \mathbf{s}^{\mathbf{T}}\mathbf{s})\right] = 0 \Rightarrow \boxed{\mathbf{B}\mathbf{s} = \lambda s}$$

Spectral Modularity Maximization

- Spectral Modularity Maximization Problem.

$$\textbf{Maximize}\left[\frac{1}{4m}\mathbf{s}^{\mathbf{T}}\mathbf{B}\mathbf{s}=\frac{1}{4m}\mathbf{s}^{\mathbf{T}}\lambda\mathbf{s}=\frac{\lambda}{4m}\right]$$

- Pick the eigenvector $\mathbf{v_1}$ corresponding to the largest eigenvalue λ_1.
- Spectral partition

$$s_i=\begin{cases}+1, & \text{if sign}[v_1]_i>0\\ 1, & \text{if sign}[v_1]_i<0\end{cases}$$

Community Detection

Louvain Algorithm

Step-1

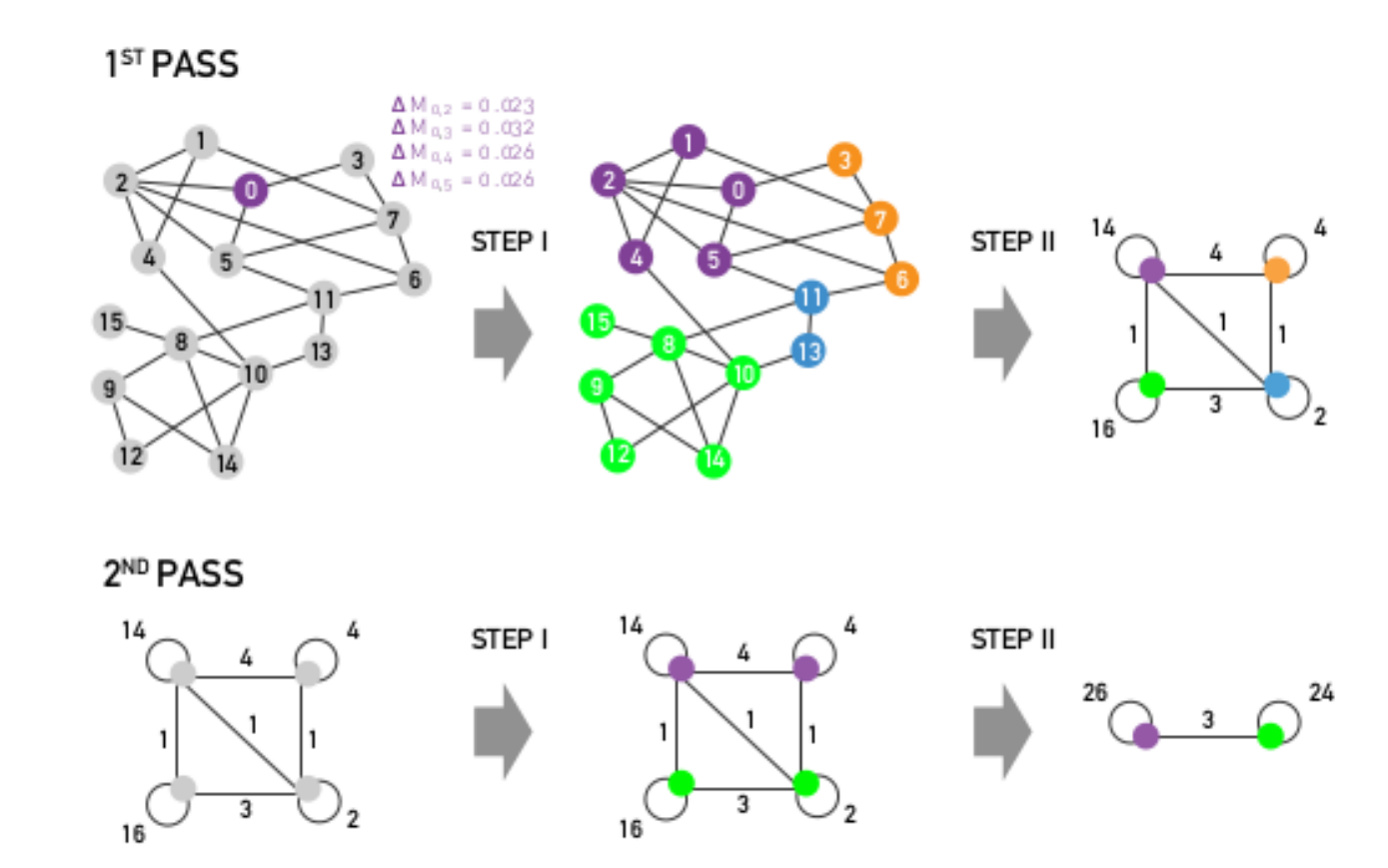
- Initially assigning each node to a different community.
- For each node i evaluate the gain in modularity if node i is placed in the community of one of its neighbors j.
- Move node i in the community for which the modularity gain is the largest, but only if this gain is positive. If no positive gain is found, i stays in its original community.
- This process is applied to all nodes until no further improvement can be achieved.

Step-2

- Construct a new network whose nodes are the communities identified during Step-1. The weight of the link between two nodes is the sum of the weight of the links between the nodes in the corresponding communities. Links between nodes of the same community lead to weighted self-loops.

Repeat step-1 and step-2 until there are no more changes and maximum modularity is attained.

Community Detection



$$\Delta M = \left[\frac{\sum_{in}+2K_{i,in}}{2W} - \left(\frac{\sum_{tot}+2K_i}{2W}\right)^2\right] - \left[\frac{\sum_{in}}{2W} - \left(\frac{\sum_{tot}}{2W}\right)^2 - \left(\frac{k_i}{2W}\right)^2\right]$$