Web-graphs. Lecture 11.

Spectral modularity maximization

- We described the algorithm for modularity maximization analogous to the Kernighan-Lin algorithm
- It is natural to ask whether there also exists an analog for community detection of the spectral graph partitioning algorithm
- The modularity function reads as follows

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

  where $B_{ij} = A_{ij} - \frac{d_i d_j}{2m}$
- Note that

$$\sum_j B_{ij} = \sum_j A_{ij} - d_i \sum_j \frac{d_j}{2m} = d_i - d_i \frac{2m}{2m} = 0$$

$$\sum_i B_{ij} = \sum_i A_{ij} - d_j \sum_i \frac{d_i}{2m} = d_j - d_j \frac{2m}{2m} = 0$$

Spectral modularity maximization

- Let us again consider the division of a network into just two parts
- we will represent such a division by the quantities

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

- $\frac{1}{2}(1 + s_i s_j) = \begin{cases} 0, & \text{if } i \text{ and } j \text{ are in different groups} \\ 1, & \text{otherwise} \end{cases}$
- $\delta(c_i, c_j) = \frac{1}{2}(1 + s_i s_j)$
- $Q = \frac{1}{4m} \sum_{ij} B_{ij}(1 + s_i s_j) = \frac{1}{4m} \sum_{ij} B_{ij} s_i s_j$
- In matrix form

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

Spectral modularity maximization

- We wish to find the division of a given network that maximizes the modularity Q
- That is, we wish to find the value of $\mathbf{s}$ that maximizes Q for a given modularity matrix $\mathbf{B}$.
- The elements of $\mathbf{s}$ are constrained to take values ± 1, so that the vector always points to one of the corners of an n-dimensional hypercube
- There are no other constraints on the problem. The number of elements with value + 1 or -1 is not fixed.
- As before, this optimization problem is a hard one, but it can be tackled approximately and effectively by a **relaxation method**.

Spectral modularity maximization

- We relax the constraint that $\mathbf{s}$ must point to a corner of the hypercube and allow it to point in any direction, though keeping its length the same, meaning that it can take any real value subject only to the constraint that

$$\mathbf{s}^T\mathbf{s} = \sum_j s_j^2 = n$$

- The Lagrangian of the problem

$$\mathcal{L} = 2\sum_{ij} B_{ij}s_i s_j + \beta(n - \sum_j s_j^2),$$

  where we put coefficient 2 instead of $1/(4m)$ for simplicity of the further notations (there is no difference between maximization of Q and $8Q$).
- Differentiating by s_i we obtain

$$\frac{\partial}{\partial s_i}\mathcal{L} = 2\sum_j B_{ij}s_j - 2\beta s_i = 0$$

$$\sum_j B_{ij}s_j = \beta s_i$$

$$\mathbf{Bs} = \beta\mathbf{s}$$

Spectral modularity maximization

- $\mathbf{s}$ is one of the eigenvectors of the modularity matrix, β is the corresponding eigenvalue
- $Q = \frac{1}{4m}\beta \mathbf{s}^T \mathbf{s} = \frac{1}{4m}\beta$
- we should choose $\mathbf{s}$ to be the eigenvector $\mathbf{u}_1$, corresponding to the largest eigenvalue of the modularity matrix $\mathbf{B}$.
- As in the case of partitioning problem, we typically cannot choose $\mathbf{s} = \mathbf{u}_1$ since the elements of $\mathbf{s}$ are subject to the constraint $s_i = \pm 1$. But we do the best we can and choose it as close to $\mathbf{u}_1$ as possible, which means maximizing the product

$$\mathbf{s}^T \mathbf{u}_1 = \sum_i s_i u_{1i}$$

- the maximum is achieved when each term in the sum is non-negative, i.e., when

$$s_i = \begin{cases} +1, & \text{if } u_{1i} > 0 \\ -1, & \text{if } u_{1i} < 0 \end{cases}$$

In the unlikely event that a vector element is exactly zero, either value of s_i is equally good and we can choose whichever we prefer.

Louvain method

- Blondel, V. D., Guillaume, J. L., Lambiotte, R., & Lefebvre, E. (2008). Fast unfolding of communities in large networks. Journal of statistical mechanics: theory and experiment, 2008(10), P10008.
- The algorithm is divided into two phases that are repeated iteratively.
- Assume that we start with a weighted network of N nodes

Louvain method

- First, we assign a different community to each node of the network. So, in this initial partition there are as many communities as there are nodes
- Then, for each node i we consider the neighbours j of i and we evaluate the gain of modularity that would take place by removing i from its community and by placing it in the community of j.
- The node i is then placed in the community for which this gain is maximum, but only if this gain is positive
- If no positive gain is possible, i stays in its original community
- This process is applied repeatedly and sequentially for all nodes until no further improvement can be achieved and the first phase is then complete
- This first phase stops when a local maxima of the modularity is attained, i.e. when no individual move can improve the modularity
- The output of the algorithm depends on the order in which the nodes are considered

Louvain method

- The second phase of the algorithm consists in building a **new network whose nodes are now the communities** found during the first phase
- To do so, the weights of the links between the new nodes are given by the sum of the weight of the links between nodes in the corresponding two communities
- Links between nodes of the same community lead to self-loops for this community in the new network
- Once this second phase is completed, it is then possible to reapply the first phase of the algorithm to the resulting weighted network and to iterate

Louvain method

- **“pass”** – a combination of the two phases
- By construction, the number of meta-communities decreases at each pass, and as a consequence most of the computing time is used in the first pass.
- Th until there are no more changes and a maximum of modularity is attained.
- The algorithm is reminiscent of the self-similar nature of complex networks and naturally incorporates a notion of hierarchy, as communities of communities are built during the process

Louvain method

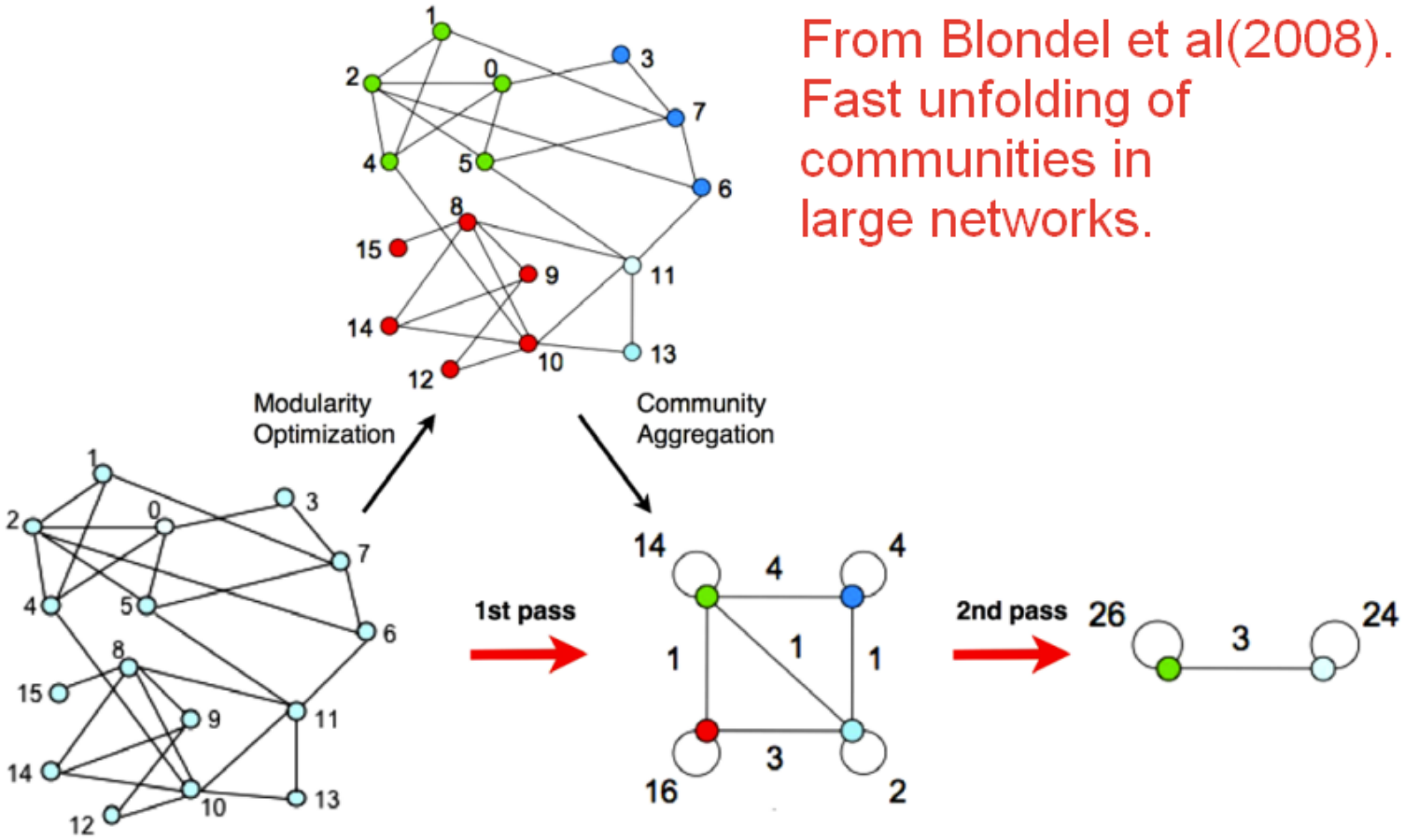

Figure 1. Visualization of the steps of our algorithm. Each pass is made of two phases: one where modularity is optimized by allowing only local changes of communities; one where the communities found are aggregated in order to build a new network of communities. The passes are repeated iteratively until no increase of modularity is possible.