Web-graphs. Lecture 10.

Community detection. General description

- Community detection is the search for the **naturally occurring groups** in a network regardless of their number or size, which is used primarily as a tool for discovering and understanding the large-scale structure of networks
- The basic goal of community detection is similar to that of graph partitioning: we want to separate the network into groups of vertices that have few connections between them.
- The important difference from the partition problem is that the number or size of the groups is not fixed.
- Starting example: we fix the number of groups (=2), but we allow the groups to have any size. The sum of the sizes should equal the size n of the whole network.
- Thus, in this simple version of the problem, the number of groups is still specified but their sizes are not, and we wish to find the "natural" division of the network into two groups, the fault line (if any) along which the network inherently divides.
- **The question we're asking is not well defined.**

Dividing using Betweenness Measures

- Real complex networks can be described in terms of their tightly-knit regions and the weaker ties that link them together.
- Our focus: describe a method that can take a network and break it down into a set of tightly-knit regions, with sparser interconnections between the regions.

Dividing using Betweenness Measures

From Easley & Kleinberg,
Networks, crowds and markets

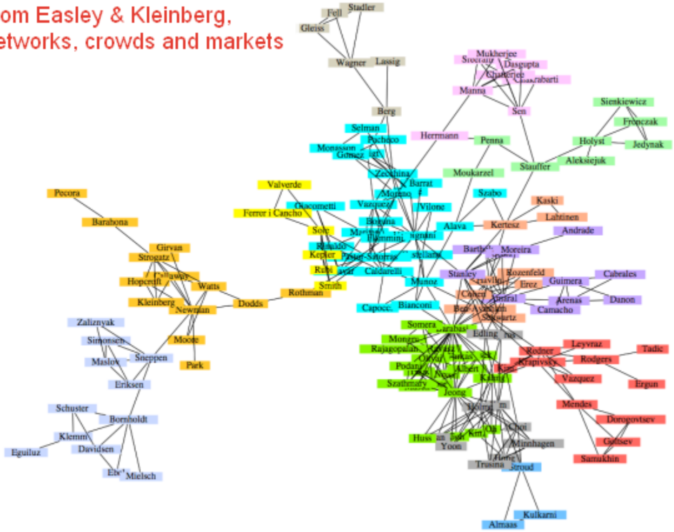
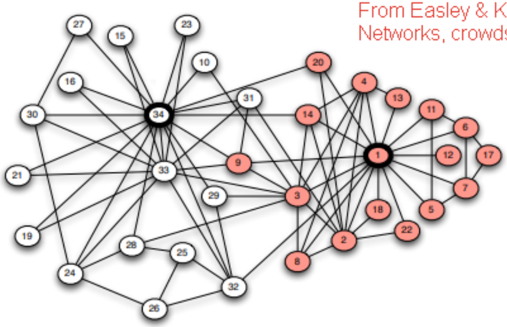


Figure 3.12: A co-authorship network of physicists and applied mathematicians working on networks [322]. Within this professional community, more tightly-knit subgroups are evident from the network structure.

Dividing using Betweenness Measures



From Easley & Kleinberg,
Networks, crowds and markets

Figure 3.13: A karate club studied by Wayne Zachary [421] — a dispute during the course of the study caused it to split into two clubs. Could the boundaries of the two clubs be predicted from the network structure?

Dividing using Betweenness Measures

- The **Girvan-Newman** method
- Bridges and local bridges often connect weakly interacting parts of the network, we should try removing these bridges and local bridges first.
- Local bridges are important because they form part of the shortest path between pairs of nodes in different parts of the network — without a particular local bridge, paths between many pairs of nodes may have to be "re-routed" a longer way.
- We can define an **edge betweenness** that counts the number of shortest paths that run along edges and edges that lie between communities can be expected to have high values of the edge betweenness.

Dividing using Betweenness Measures

From Newman. Networks. An Introduction

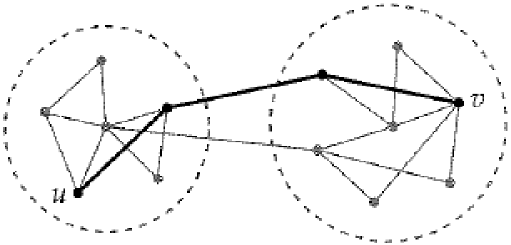
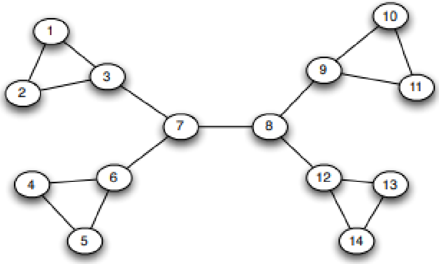


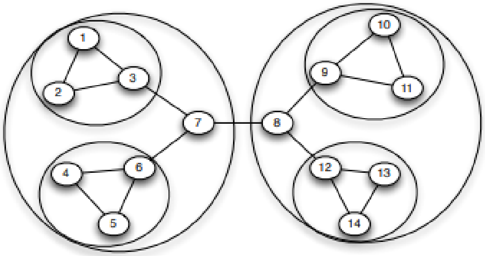
Figure 11.6: Identification of between-group edges. This simple example network is divided into two groups of vertices (denoted by the dotted lines), with only two edges connecting the groups. Any path joining vertices in different groups (such as vertices u and v) must necessarily pass along one of these two edges. Thus if we consider a set of paths between all pairs of vertices (such as geodesic paths, for instance), we expect the between-group edges to carry more paths than most. By counting the number of paths that pass along each edge we can in this way identify the between-group edges.

Dividing using Betweenness Measures

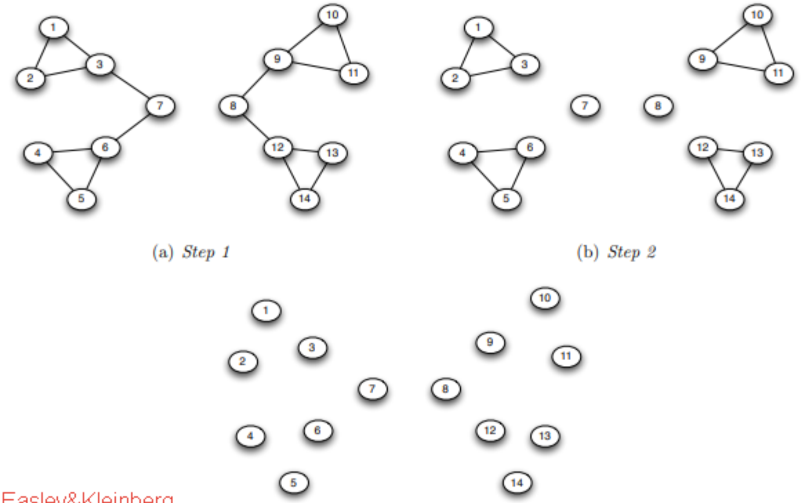
From Easley & Kleinberg. Networks, crowds and markets



(a) *A sample network*



Dividing using Betweenness Measures



(a) *Step 1* (b) *Step 2*

From Easley&Kleinberg.
Networks, crowds and markets

(c) *Step 3*

Figure 3.16: The steps of the Girvan-Newman method on the network from Figure 3.14(a).

Dividing using Betweenness Measures

From Easley & Kleinberg. Networks, crowds and markets

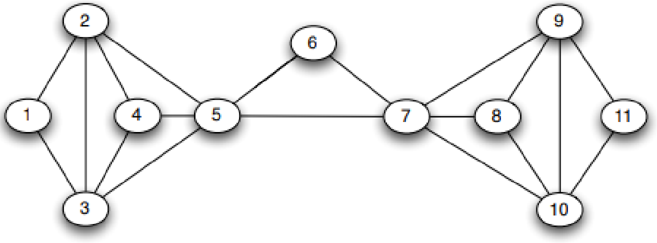
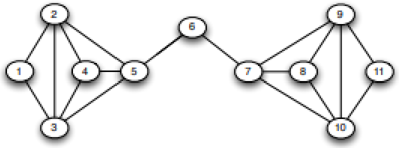


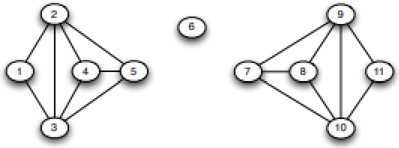
Figure 3.15: A network can display tightly-knit regions even when there are no bridges or local bridges along which to separate it.

Dividing using Betweenness Measures

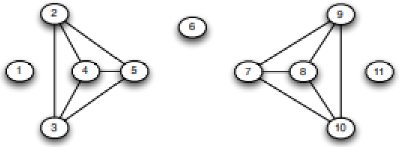
From Easley & Kleinberg. Networks, crowds and markets



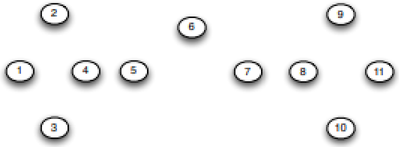
(a) *Step 1*



(b) *Step 2*



(c) *Step 3*



(d) *Step 4*

Figure 3.17: The steps of the Girvan-Newman method on the network from Figure 3.15.

The Girvan-Newman Method

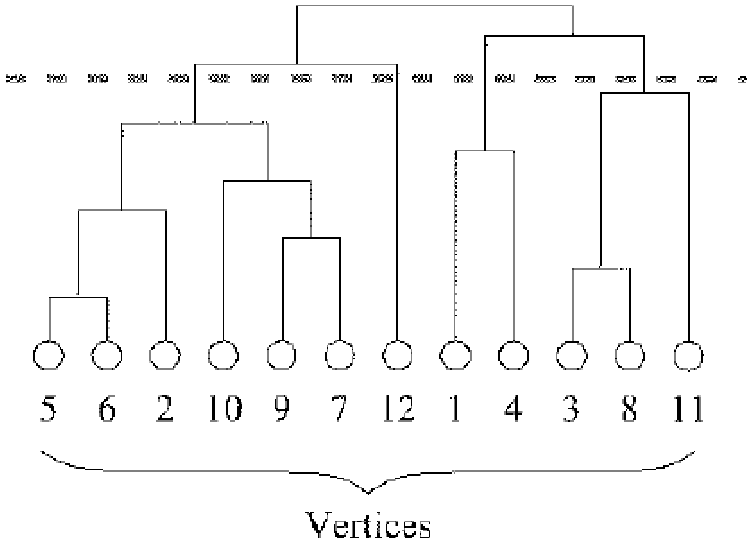
The Girvan-Newman method can be summarized as follows

1. Find the edge of highest betweenness — or multiple edges of highest betweenness, if there is a tie — and remove these edges from the graph. This may cause the graph to separate into multiple components. If so, this is the first level of regions in the partitioning of the graph.
2. Now recalculate all betweennesses, and again remove the edge or edges of highest betweenness. This may break some of the existing components into smaller components; if so, these are regions nested within the larger regions.
3. Proceed in this way as long as edges remain in graph, in each step recalculating all betweennesses and removing the edge or edges of highest betweenness.

The method drawback: high numeric complexity of the betweenness calculation. **Method is very slow**.

The Girvan-Newman Method: dendrogram

From Newman. Networks. An Introduction



Maximization based methods

- First guess: find the division with minimum cut size, as in the corresponding graph partitioning problem, but without any constraint on the sizes of our groups.
- If we divide a network into two groups with any number of vertices allowed in the groups then the optimum division is simply to put all the vertices in one of the groups and none of them in the other.
- As an answer to our community detection problem, however, it is clearly not useful.

Maximization based methods

- One way to do better would be to impose loose constraints of some kind on the sizes of the groups.
- An example of this type of approach is *ratio cut* partitioning in which, instead of minimizing the standard cut size R, we instead minimize the ratio $R/(n_1 \cdot n_2)$
- The denominator $(n_1 \cdot n_2)$ has its largest value when n_1 and n_2 are equal $n_l = n_2 = \frac{1}{2}n$.
- **BUT:** as a tool for discovering the natural divisions in a network the ratio cut is not ideal.
- Although it allows group sizes to vary it is still biased towards a particular choice of **equally sized groups**.
- More importantly, there is no principled rationale behind its definition.

Maximization based methods

- An alternative strategy is to focus on a different measure of the quality of a division other than the simple cut size or its variants.
- Cut size is not itself a good measure because a good division of a network into communities is not merely one in which there are few edges between communities, but the one where there are **fewer than expected such edges**.
- Our goal therefore will be to find a measure that quantifies how many edges lie within groups in our network relative to the number of such edges expected on the basis of chance.

Assortative mixing

- Suppose we have a network in which the vertices are classified according to some characteristic that has a finite set of possible values.
- The values are merely enumerative-they don't fall in any particular order.
- For instance, the vertices could represent people and be classified according to nationality, race, or gender. Or they could be web pages classified by what language they are written in, or biological species classified by habitat, or any of many other possibilities.
- The network is **assortative** if a significant fraction of the edges in the network **run between vertices of the same type**
- A simple way to quantify assortativity would be to measure the fraction of withing class edges. However, this is not a very good measure (see arguments in the previous slides)

Assortative mixing

- A good measure turns out to be the following.
  - We find the fraction of edges that run between vertices of the same type
  - Then we subtract from that figure the fraction of such edges **we would expect** to find if edges were positioned at **random without regard for vertex type**
- c_i – the class (type) of vertex $i = 1, \ldots, N$. $c_i \in \{1, \ldots, n_c\}$
- Then the total number of edges that run between vertices of the same type is

$$\sum_{\text{edges}(i,j)} \delta(c_i, c_j) = \frac{1}{2} \sum_{i,j} A_{ij} \delta(c_i, c_j)$$

- Let's count expected number of edges between nodes i and j
  - Consider a particular edge attached to vertex i with degree d_i
  - There are by definition $2m$ ends of edges in the entire network.
  - The chances that the other end of our particular edge is one of the d_j are equal to $d_j/2m$ if connections are made purely at random
  - The total expected number of edges between vertices i and j is then $d_i d_j/2m$

  the expected number of edges between all pairs of vertices of the same type is

$$\frac{1}{2} \sum_{ij} \frac{d_i d_j}{2m} \delta(c_i, c_j)$$

Assortative mixing

- An expression for the difference between the actual and expected number of edges in the network that join vertices of like types

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

- Conventionally, one calculates not the number of such edges but the fraction, which is given by this same expression divided by the number m of edges

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

- Q is called **the modularity function**
- It is strictly less than 1, takes positive values if there are more edges between vertices of the same type than we would expect by chance, and negative ones if there are less.

Modularity and community detection

- The modularity function is precisely the type of measure we need to solve our current community detection problem.
- We return to the problem of the **two** communities identification.
- If we consider the vertices in our two groups to be vertices of two types then good divisions of the network into communities are precisely those that have high values of the corresponding modularity
- Thus one way to detect communities in networks is to look for the divisions that have the **highest modularity scores**
- This is the most commonly used method for community detection/evaluation of the results quality.
- Like graph partitioning, **modularity maximization is a hard problem**
- We turn again to **heuristic algorithms**, algorithms that attempt to maximize the modularity in an intelligent way that gives reasonably good results most of the time

Simple modularity maximization

- The analog of the Kernighan-Lin algorithm
- This algorithm divides networks into two communities starting from some initial division, such as a random division into equally sized groups.
- The algorithm then considers **each vertex** in the network in turn and calculates **how much the modularity would change** if that vertex were moved to the other group.
- It then chooses among the vertices the one whose movement would most increase, or least decrease, the modularity and moves it.
- Then it repeats the process, but with the important constraint that a vertex once moved cannot be moved again on this round of the algorithm.
- When all vertices have been moved exactly once, we go back over the states through which the network has passed and select the one with the highest modularity.
- We then use that state as the starting condition for another round of the same algorithm, and we keep repeating the whole process until **the modularity no longer improves**.

Simple modularity maximization

From Newman. Networks. An Introduction

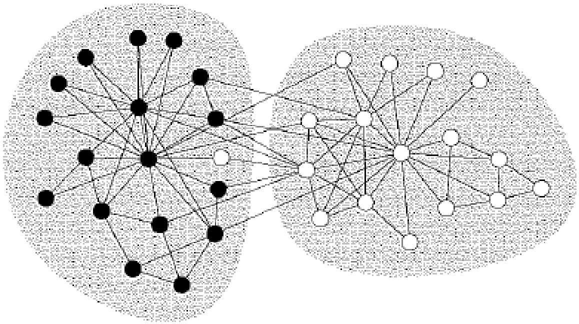


Figure 11.4: Modularity maximization applied to the karate club network. When we apply our vertex-moving modularity maximization algorithm to the karate club network, the best division found is the one indicated here by the two shaded regions, which split the network into two groups of 17 vertices each. This division is very nearly the same as the actual split of the network in real life (open and solid circles), following the dispute among the club's members. Just one vertex is classified incorrectly.

Algorithm computation complexity

- The vertex moving algorithm is quite efficient
- At each step of the algorithm we have to evaluate the modularity change due to the movement of each of $O(n)$ vertices
- each such evaluation, like the corresponding ones for the Kernighan-Lin algorithm, can be achieved in time $O(m/n)$
- Thus each step takes time $O(m)$
- There are n steps in one complete round of the algorithm for a total time of $O(mn)$.
- This is considerably better than the $O(mn^2)$ of the Kernighan-Lin algorithm
- The algorithm is in fact one of the best of the many proposed algorithms for modularity maximization