Advanced Network Analysis

Community Detection

Olga Chyzh [www.olgachyzh.com]

Network Community Detection

Reading Materials

- Barabasi, Albert-Laszlo. (2016). Network Science. Cambridge University Press. Chapter 9.
- Zachary, Wayne W. (1977). An information flow model for conflict and fission in small groups. Journal of Anthropological Research 33(4) : 452-473.
- Renshon, Jonathan. (2016). Status deficits and war. International Organization 70(3): 513-550.
- Gould, Roger V. (1991). Multiple Networks and Mobilization in the Paris Commune, 1971. Americal Sociological Review 56(6): 716-729.

Introduction: Belgium

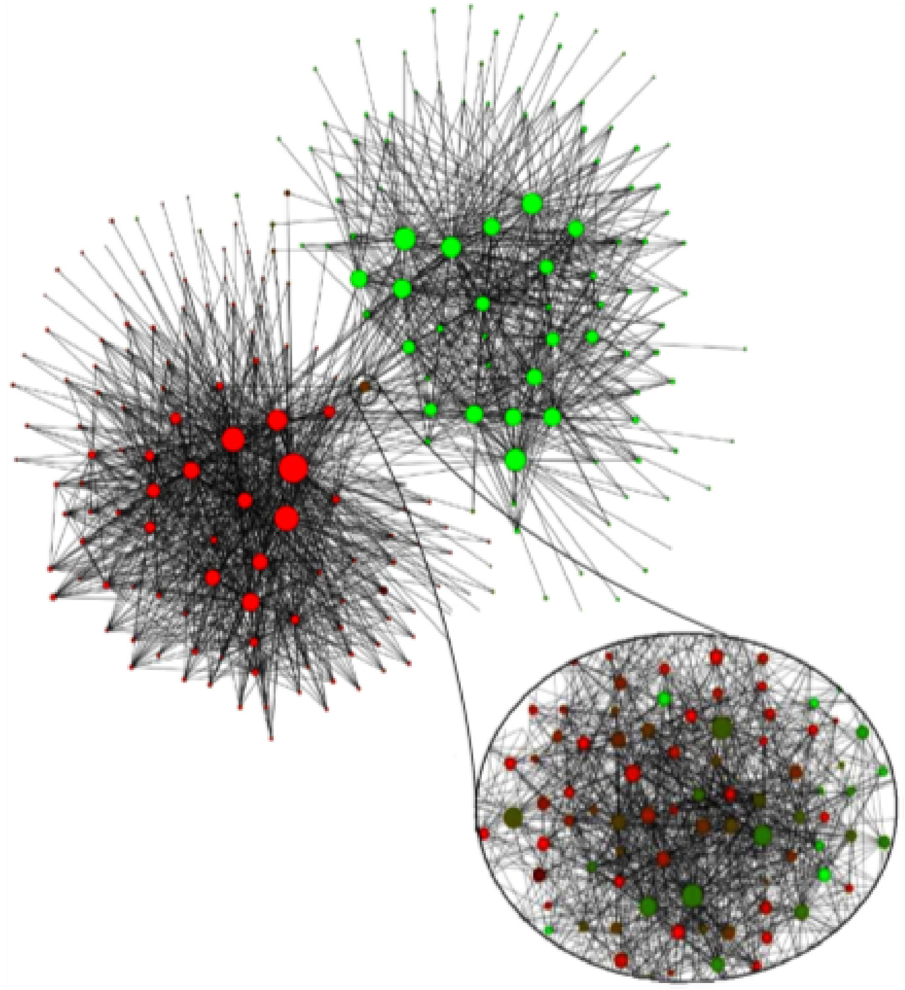
Introduction: Belgium

- Population: 11.5 mil
- Bilingual: 59% Flemish (speak Dutch), 40% Waloons (French)
- Is the society so densely knitted together that nobody notices who is of what ethnic group?
- Or do the two groups minimize the interactions?

Blondel et al. (2008)

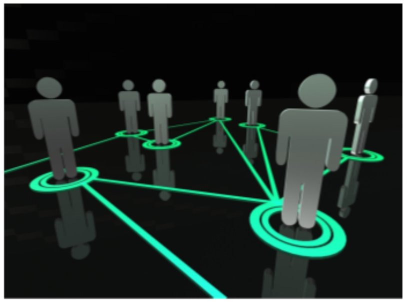
- Applied a community finding algorithm to the call patterns of a big mobile phone operator.
- Goal: identify individuals who regularly talk to one another.

Blondel et al. (2008)



Communities: the Basics

Network Levels of Analysis



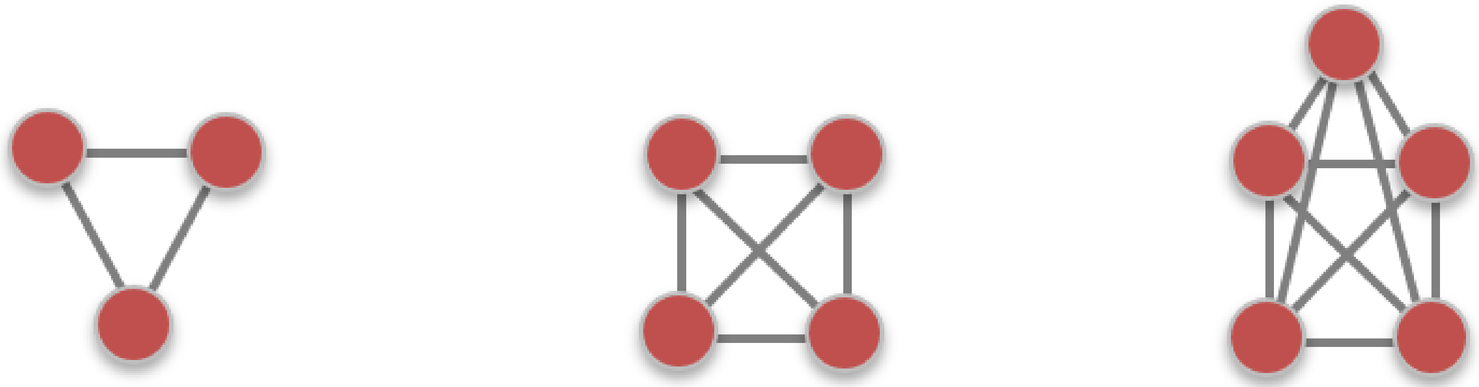
Microscopic Mesoscopic Macroscopic

What Is A Community?

- A community is a locally dense connected subgraph. (Barabasi, 2016, 325)
    - All members of a community must be reached through other members of the same community (connectedness).
    - Nodes that belong to the same community have a higher probability of linking than nodes outside of the community (density).
    - Examples: communities among the karate club members, communities of international states, communities of legislators, neighborhood communities.
- Note that these features do not uniquly define a community, just offer some guidelines.

Cliques as Communities

A clique is a complete subgraph of k-nodes.



- May be too restrictive.

Strong and Weak Communities

Consider a connected subgraph C of N_c nodes.

Internal degree, k_i^{int} is the set of links of node *i* that connectes to the other nodes in the same community.

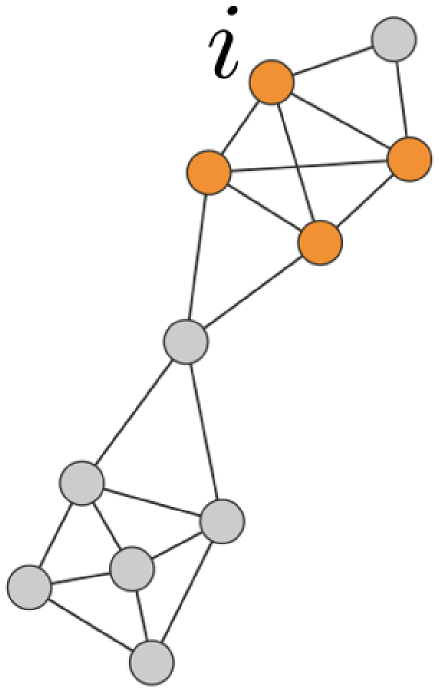
External degree, k_i^{ext} is the set of links of node *i* that connects to the rest of the network.

If $k_i^{ext} = 0$, all neighbors of i belong to C, and C is a good community for i.

If $k_i^{int} = 0$, all neighbors of i belong to other communities, then i should be assigned to a different community.

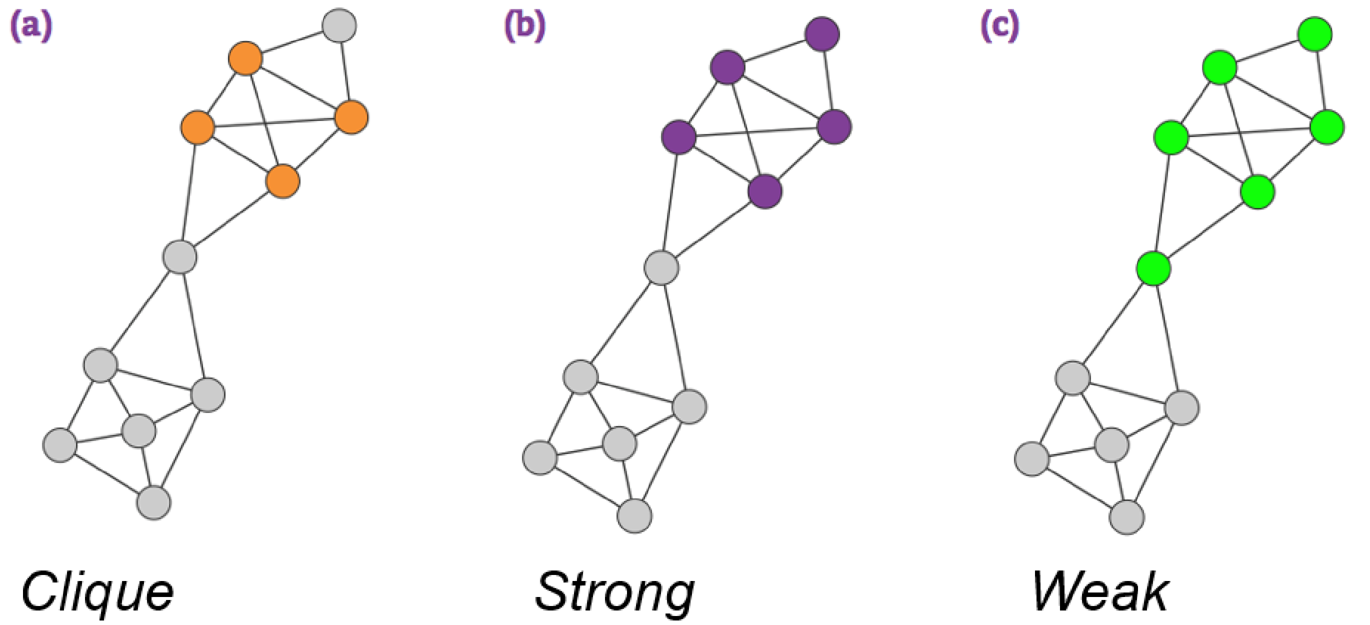
Example

$k_i^{ext} = 1, k_i^{int} = 3$



Strong and Weak Communities

In a *strong community* each node of C has more links within the community than with the rest of the graph, $k_i^{int}(C) > k_i^{ext}(C)$. In a *weak community*, the total internal degree of C exceeds its total external degree, $k_i^{in}(C) > k_i^{out}(C)$.



Number of Partitions

How many ways can we partition a network into 2 communities?

Divide a network into two equal non-overlapping subgraphs, such that the number of links between the nodes in the two groups is minimized.

Two subgroups of size n1 and n2. Total number of combinations: $\frac{N!}{n_1!n_2!}$

- $N = 10 \implies 256$ partitions
- $N = 100 \implies 10^{26}$ partitions

If the number and size of the communities are unknown at the beginning, the number of possible partitions is a Bell Number.

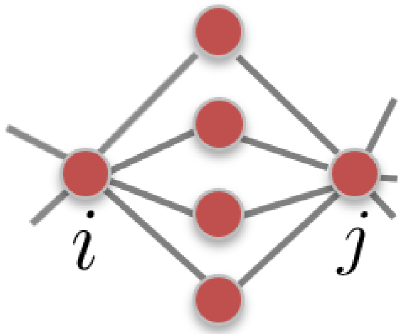
Brute force approach is unfeasible, need an algorithm.

Ravasz Algorithm

Step 1: Define the Similarity Matrix

High for node pairs that likely belong to the same community, low for those that likely belong to different communities. Nodes that connect directly to each other and/or share multiple neighbors are more likely to belong to the same dense local neighborhood, hence their similarity should be large.

Topological overlap matrix: $x_{ij} = \frac{J_N(i,j)}{min(k_i,k_j)}$, where $J_N(i,j)$ is the number of common neighbors of nodes i and j, $+1$ if there is a direct link between i and j, and $min(k_i, k_j)$ is the smaller of the degrees k_i and k_j.



Ravasz Algorithm

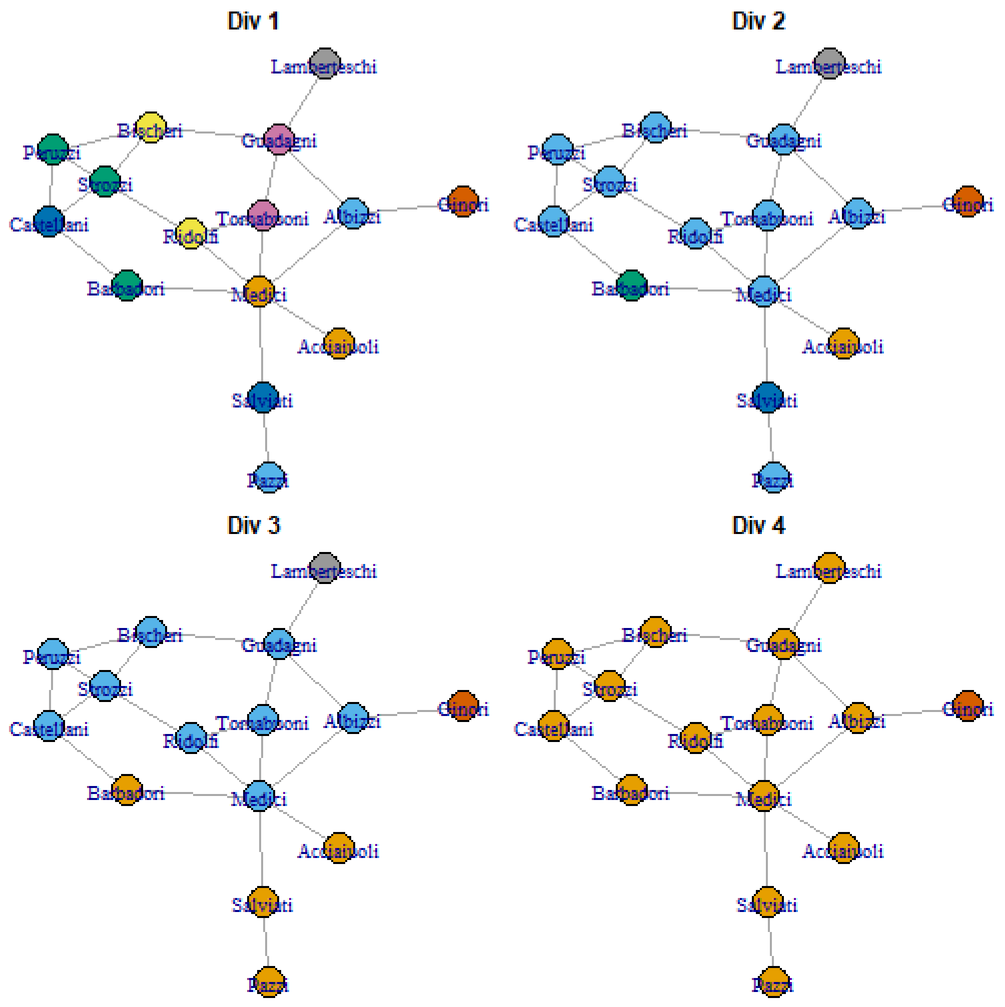
Step 2: Define Group Similarity

Nodes are merged into groups based on their mutual similarity, e.g. average cluster linkage.

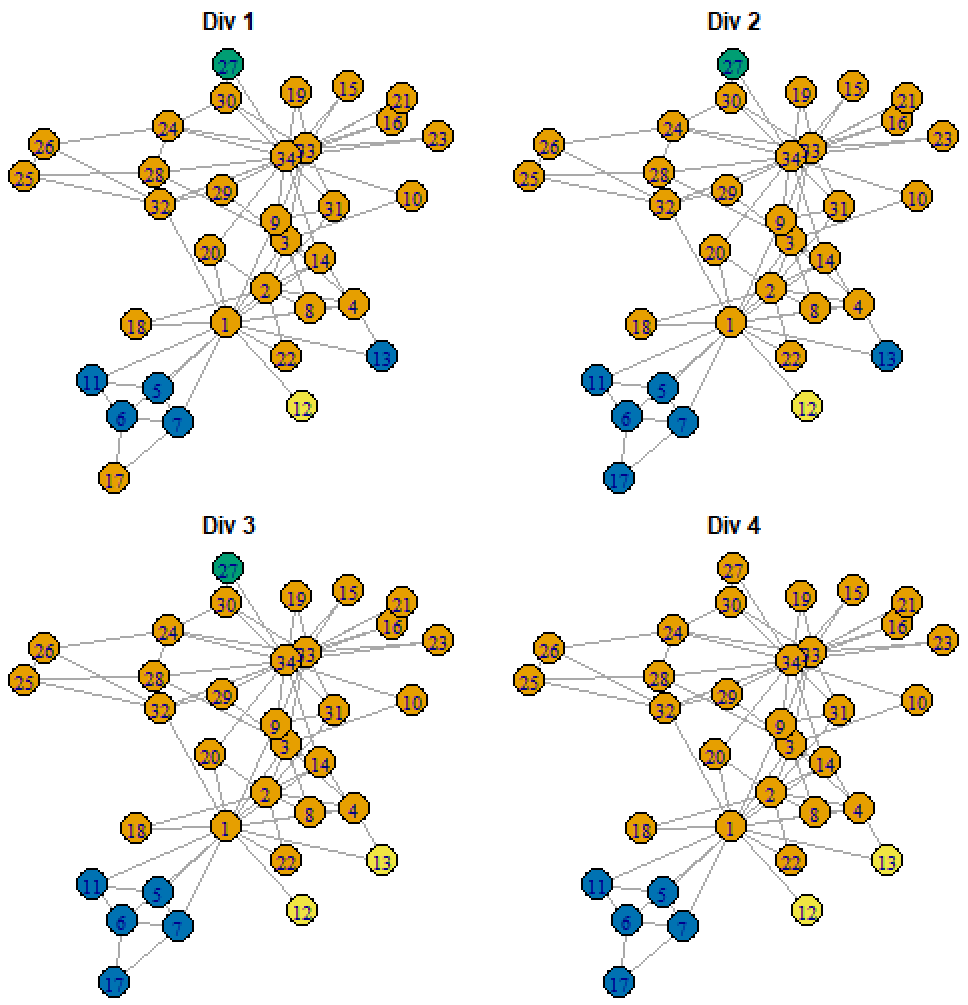
Step 3: Apply Hierarchical Clustering

- Assign each node to a community of its own and evaluate the similarity for all node pairs. The initial similarities between these “communities” are simply the node similarities.
- Find the community pair with the highest similarity and merge them to form a single community.
- Calculate the similarity between the new community and all other communities.
- Repeat from Step 2 until all nodes are merged into a single community.

Different Splits of the Florentine Network



The Ravasz Algorithm Splits of Zachary's Data



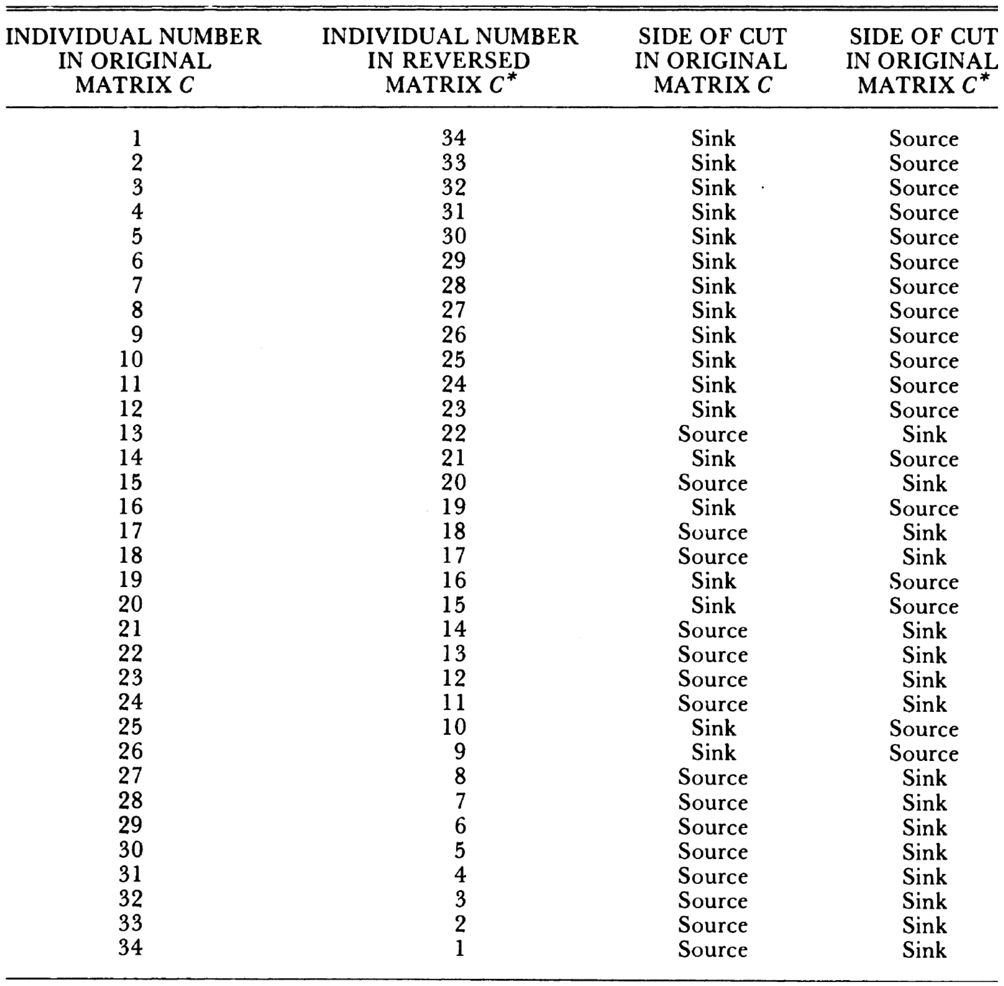
Zachary (1977)

- Karate club of 34 members;
- 78 pairwise links between members who regularly interacted outside the club;
- A conflict between the club's president and the instructor split the club into two.
- Today community finding algorithms are often tested based on their ability to infer these two communities from the structure of the network before the split.

Zachary (1977)

- Maximum Flow Minimum Cut Procedure: divides network nodes into two subsets, one containing the source and the other, the sink. The edges connecting the two subsets, called `the cut.` The cut with the minimum sum of the capacities of all of its edges represents a bottleneck in the network. (464)

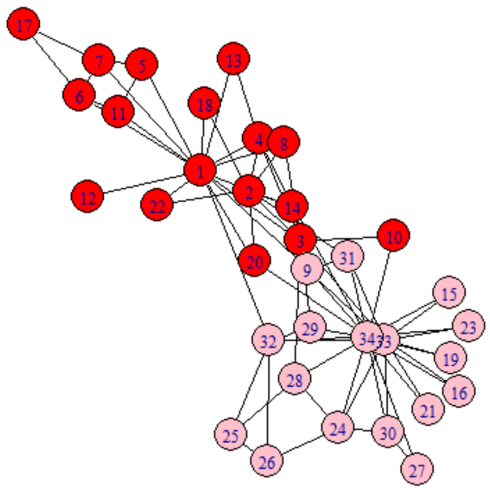
Zachary (1977)



INDIVIDUAL NUMBER IN ORIGINAL MATRIX C	INDIVIDUAL NUMBER IN REVERSED MATRIX C^*	SIDE OF CUT IN ORIGINAL MATRIX C	SIDE OF CUT IN ORIGINAL MATRIX C^*
1	34	Sink	Source
2	33	Sink	Source
3	32	Sink	Source
4	31	Sink	Source
5	30	Sink	Source
6	29	Sink	Source
7	28	Sink	Source
8	27	Sink	Source
9	26	Sink	Source
10	25	Sink	Source
11	24	Sink	Source
12	23	Sink	Source
13	22	Source	Sink
14	21	Sink	Source
15	20	Source	Sink
16	19	Sink	Source
17	18	Source	Sink
18	17	Source	Sink
19	16	Sink	Source
20	15	Sink	Source
21	14	Source	Sink
22	13	Source	Sink
23	12	Source	Sink
24	11	Source	Sink
25	10	Sink	Source
26	9	Sink	Source
27	8	Source	Sink
28	7	Source	Sink
29	6	Source	Sink
30	5	Source	Sink
31	4	Source	Sink
32	3	Source	Sink
33	2	Source	Sink
34	1	Source	Sink

Can We Replicate the Result?

```r
library(igraph) #Zachary's karate club dataset is built into the igra
karate <- make_graph("Zachary")
mf<-max_flow(karate, source=V(karate)[1], target=V(karate)[34])
V(karate)$color<- ifelse(V(karate) %in%  mf$partition1, "red","pink")
plot(karate, edge.color="black", vertex.frame.color="black")
```



Can We Find the Same Result?

```
mf$partition1
```

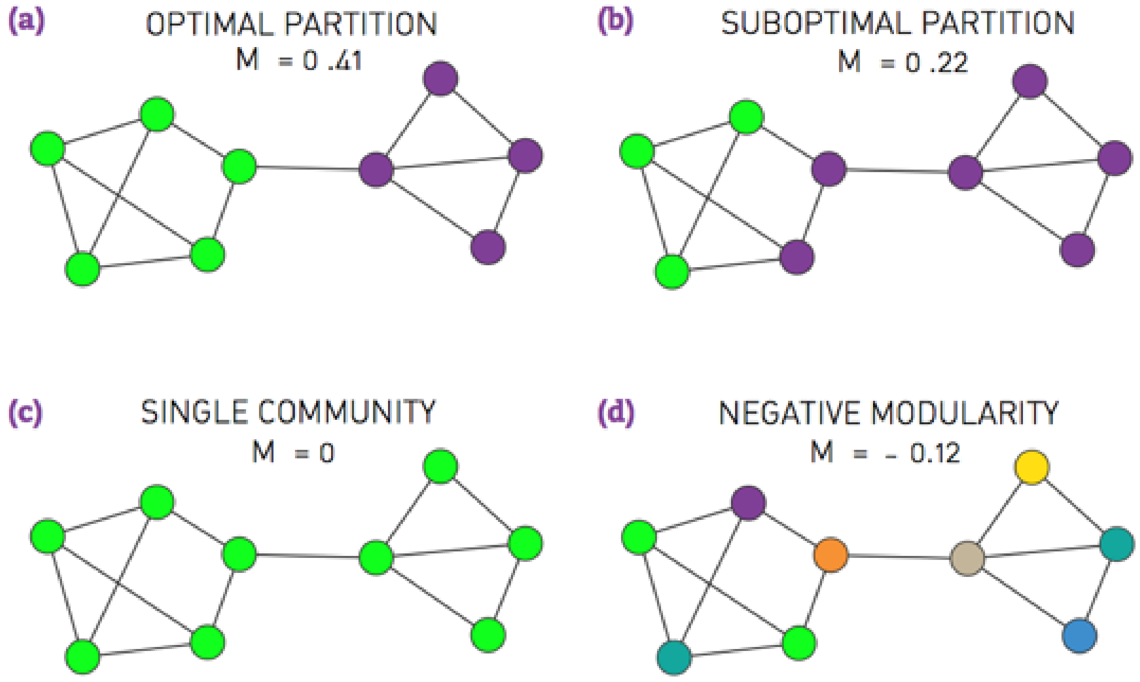
```
## + 17/34 vertices, from d3a0304:
##  [1]  1  2  3  4  5  6  7  8 10 11 12 13 14 17 18 20 22
```

```
mf$partition2
```

```
## + 17/34 vertices, from d3a0304:
##  [1]  9 15 16 19 21 23 24 25 26 27 28 29 30 31 32 33 34
```

Other Ways to Partition Networks

- Modularity: a measure of the extent to which like is connected to like in a network.
  - A *greedy algorithm* iteratively joins nodes if the move increases the new partition's modularity.



```r
library(sna)
data(coleman)
#make into an igraph object
friends<-graph_from_adjacency_matrix(coleman[2,,], mode="undirected",
friends <- igraph::delete.vertices(friends , which(igraph::degree(fr
LO = layout_with_fr(friends) #Layout
cfg<-cluster_fast_greedy(friends)
modularity(cfg)
```

```
## [1] 0.5904201
```

```r
cfg$membership
```

```
##  [1] 5 5 1 1 5 5 5 5 5 1 5 5 5 5 1 5 5 5 5 5 5 3 5 1 4 5 1 4 4 1 1 1 4 1 4
## [62] 3 3 3 3 3 3 3 3
```

Communities in the Friendship Network

```r
plot(cfg, friends, layout=LO, main="Greedy")
```

Other Examples:

```
wc <- cluster_walktrap(friends) #community structure via short rando
modularity(wc)
```

```
## [1] 0.5935815
```

```
membership(wc)
```

```
##  1  2  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 26 27
##  2  2  1  1  2  2  2  2  1  1  2  2  2  2  1  2  1  1  2  2  2  3  2  2  5
## 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69
##  3  3  3  3  3  3  3  3  3  3  3  4  3  4  3  4  4  4  4  4  4  4  4  4  4
```

Fast Greedy Algorithm

```r
fg <- cluster_fast_greedy(friends) #Used in Renshon (2016)
modularity(fg)
```

```
## [1] 0.5904201
```

```r
fg$membership
```

```
##  [1] 5 5 1 1 5 5 5 5 5 1 5 5 5 5 1 5 5 5 5 5 5 3 5 1 4 5 1 4 4 1 1 1 4 1 4
## [62] 3 3 3 3 3 3 3 3
```

Egde Betweenness

```
ceb<-cluster_edge_betweenness(friends)
modularity(ceb)
```

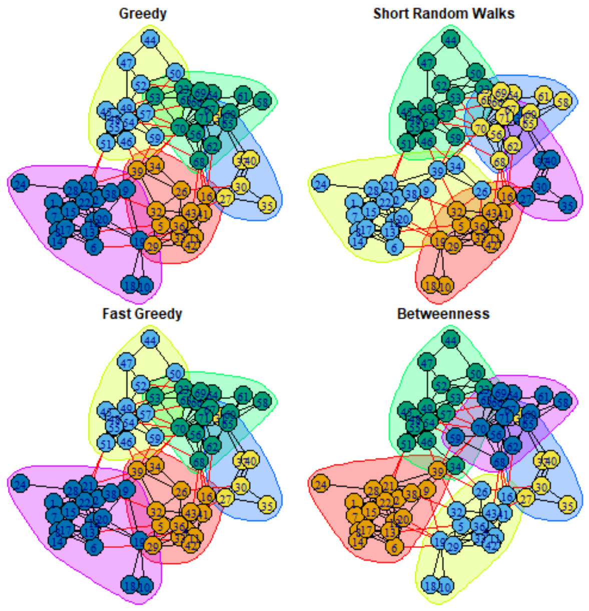
```
## [1] 0.5993285
```

```
ceb$membership
```

```
##  [1] 1 1 2 2 1 1 1 1 2 2 1 1 1 1 2 1 2 2 1 1 1 3 1 2 4 1 2 4 4 2 2 3 4 2 4
## [62] 5 5 5 5 5 5 5 5
```

Comparing Clustering Algorithms

```
par(mfrow=c(2, 2), mar=c(0,0,1,0))
plot(cfg, friends, layout=LO, main="Greedy")
plot(wc, friends, layout=LO, main="Short Random Walks")
plot(fg, friends, layout=LO,main= "Fast Greedy")
plot(ceb, friends, layout=LO, main="Betweenness")
```


Greedy
Short Random Walks
Fast Greedy
Betweenness