



## Social Data Management Communities

---

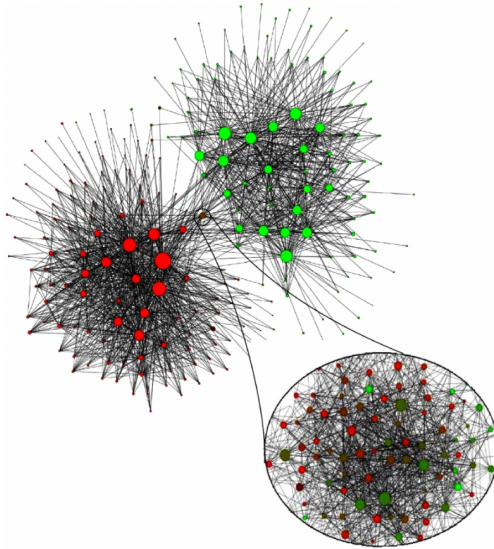
**Silviu Maniu<sup>1</sup>**

November 12th, 2021

<sup>1</sup>Université Paris-Saclay

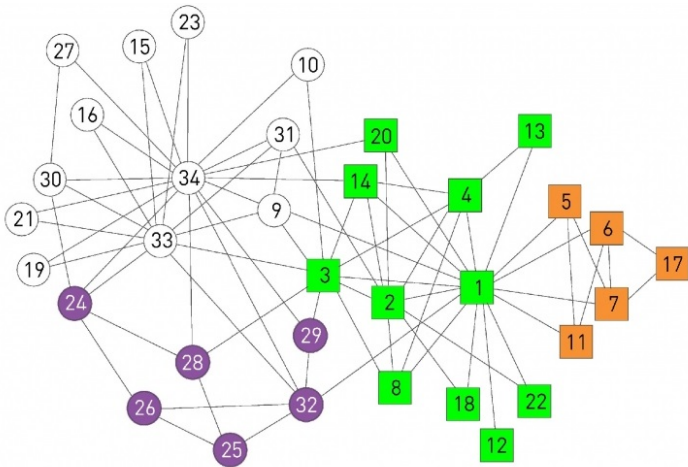
Communities in Graphs

# Graph Communities



Communities in the Belgium call graph

# Graph Communities



Communities in Zachary's Karate club

# Detecting Communities – Hypothesis 1

## **Hypothesis**

*A graph's community structure is uniquely encoded in its topology.*

# Detecting Communities – Hypothesis 2

## **Hypothesis**

*A community is a locally dense connected subgraph in a network.*

# Detecting Communities

A few approaches:

1. **Maximum Cliques**: a community is a subgraph whose nodes are all connected to each other.
2. **Strong Communities**: relaxation of cliques, depending on the *internal degree* (number of neighbors in the community) vs. *external degree* (neighbours outside of the community) – a strong community has a greater internal degree than external degree.

# Detecting Communities

A naïve algorithm for detecting 2 communities:

1. divide the graph in two (find a **cut**) and decide if they are strong communities, and
2. choose the best cut over all possible cuts.

This can generalize to more communities, but it needs to generate an **exponential** number of cuts.



# Polynomial Algorithms

We need polynomial algorithms to be able to detect efficiently the communities in a graph.

One approach is **hierarchical clustering**:

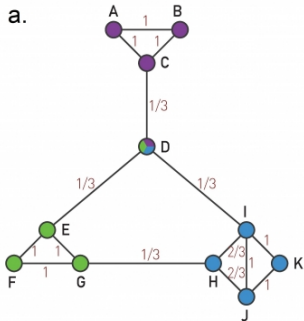
- uses a similarity matrix  $X$ , where  $x_{ij}$  encodes the similarity between nodes  $i$  and  $j$ , and
- based on this, **agglomerative algorithms** merge nodes into the same community, while
- **divisive algorithms** isolate communities by removing low similarity links.

# Agglomerative Algorithm – Ravasz

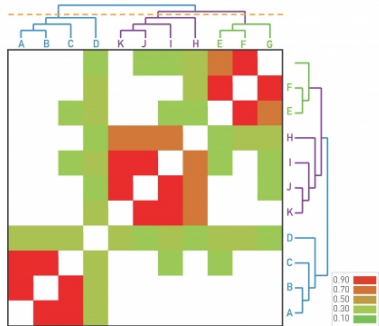
1. **Define the similarity matrix:** various ways, but the algorithm uses the *topological overlap matrix*, encoding the number of common neighbors over the maximum possible.
2. **Define group similarity:** computed as the *average cluster similarity* – the average of  $x_{ij}$  over all node pairs
3. **Apply the Hierarchical Clustering:**
  - 3.1 assign each node to a community of their own,
  - 3.2 find the community pairs with highest similarity and merge them,
  - 3.3 compute the similarity between all communities
  - 3.4 repeat until only one community exists.
4. The community structure will be encoded in the *Dendogram*, showing the order in which communities were merged (see next slide).

# Agglomerative Algorithm – Ravasz

a.



b.

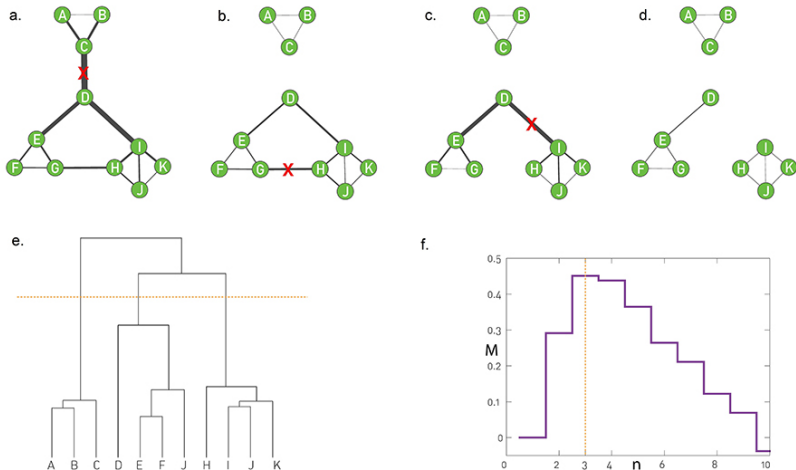


# Divisive Algorithm – Girvan-Newman

Divisive algorithms remove edges:

1. **Define centrality:**  $x_{ij}$  needs to select nodes in different communities, e.g., betweenness.
2. **Apply the Hierarchical Clustering:**
  - 2.1 remove the link with the largest centrality
  - 2.2 recompute the centrality of all other links
  - 2.3 repeat until no links exist
3. The community structure will be encoded in the **Dendogram**, showing the order in which edges were removed (see next slide).

# Divisive Algorithm – Girvan-Newman



# Detecting Communities – Hypothesis 3

## **Hypothesis**

*Random networks lack a community structure.*

# Modularity

Consider a graph having some partition into communities  $C$  having  $L_C$  links. If  $L_C$  is greater than the number of links expected by a random wiring *having the same degree distribution*, then it is a **potential community**.

This is measured by the **modularity**:

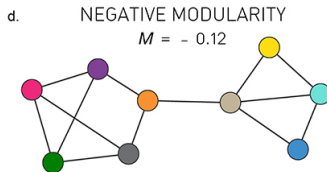
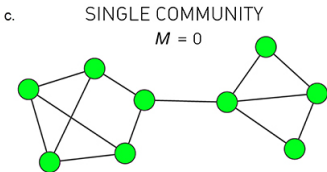
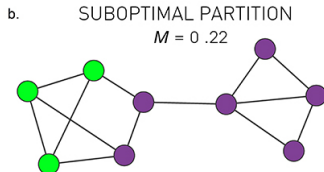
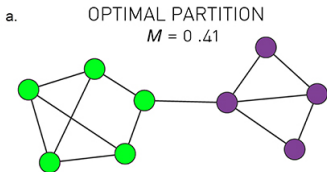
$$M_C = \frac{1}{2L} \sum_{(i,j) \in C} (A_{ij} - p_{ij}),$$

where  $p_{ij}$  can be computed by randomizing the original network, e.g.,:

$$p_{ij} = \frac{k_i k_j}{2L}.$$

**For the entire graph**: we sum the modularities over all communities.

# Modularity – Examples





## **Hypothesis**

*The partition with maximum modularity corresponds to the optimal community structure.*

# Modularity – Algorithm

For computation efficiency concerns, all algorithms use a **greedy approach**:

1. Assign each node to its own community.
2. Inspect each community pair connected by at least one link and merge the ones having the highest increase in modularity  $\Delta M$  *for the whole network*.
3. Repeat until all nodes are in a single community.
4. Choose the partition with the highest modularity.

**Louvain algorithm**: optimizes the modularity **for each node** – moves one node in the nearest community w.r.t. modularity

# Community Detection Algorithms – Complexity

algorithm	type	complexity
Ravasz	agglomerative	$\mathcal{O}(N^2)$
Girvan–Newman	divisive	$\mathcal{O}(N^3)$
greedy optimized	modularity	$\mathcal{O}(N \log^2 N)$
Louvain	modularity	$\mathcal{O}(L)$
Infomap	flow	$\mathcal{O}(N \log N)$

# Open Issues in Community Detection

- **Do communities really exist?:** given a network, do we know it is always organized in communities?
- **Are the hypotheses valid?:** is a community only identified by its wiring diagram?
- **Does everybody belong to a community?**
- **How do we know which measure is the valid one?:** centrality, similarity, modularity, flow, etc.

# Acknowledgments

Figures in slides 3, 4, 11, 13, and 16 taken from the book “Network Science” by A.-L. Barabási. The contents is partly inspired by the flow of Chapter 9 of the same book. <http://barabasi.com/networksciencebook/>

## References i



Blondel, V. D., Guillaume, J.-L., Lambiotte, R., and Lefebvre, E.

**Fast unfolding of communities in large networks.**

*Journal of Statistical Mechanics: Theory and Experiment*, 2008(10).



Girvan, M. and Newman, M. E. J. (2002).

**Community structure in social and biological networks.**

*Proceedings of the National Academy of Sciences*, 99(12):7821–7826.



Newman, M. E. J. (2004).

**Fast algorithm for detecting community structure in networks.**

*Phys. Rev. E*, 69.



Ravasz, E., Somera, A. L., Mongru, D. A., Oltvai, Z. N., and Barabási, A.-L. (2002).

**Hierarchical organization of modularity in metabolic networks.**

*Science*, 297(5586):1551–1555.