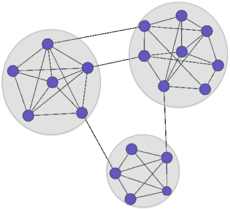
Community Detection Approaches in Networks

UNIVERSITY OF SAN FRANCISCO

James D. Wilson

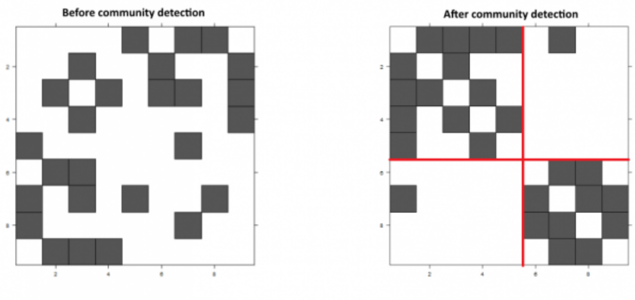
Community Structure



Informally: Communities in a network are subgraphs $C_1, \ldots, C_k \subseteq [n]$ such that

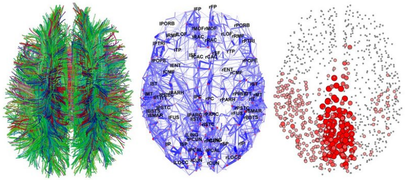
- Edge density within sets C_i is large
- Edge density between sets C_i is small

Community Structure



- In the adjacency matrix, re-ordering the rows and columns according to community labels / modules will result in densely connected "blocks" along the diagonal
- Networks with community structure are said to be assortative

Aims of Community Detection



Aim: Capture relevant structure of a complex system

- Example 1: Facebook friendship networks
  - User friendships → Geographic location of user
- Example 2: Human Connectome
  - Clustering of regions signify “functional regions”

Community Detection Approaches

- In general, community detection (when well-defined mathematically) is NP-hard. Thus, identifying communities requires approximate algorithms
- That being the case, there is no shortage of computational algorithms to identify communities
- We will describe several key approaches - i.e., ways to *define* community structure. For each of these approaches there are many algorithms available (which we won't detail here)
- A 100 + review on algorithms is available on github

Key Community Detection Approaches

Min-cut

- Identify cut of vertices that "cuts" the fewest edges

Modularity

- Partition that deviates most from organization in random graph

Spectral

- Focus on spectral properties of graph Laplacian

Stochastic Block Model

- Model-based approach. Relies on maximum likelihood estimation

Extraction

- Local, significance based algorithms

Features of Community Detection Methods

Common: Communities $\{C_j\}$ partition vertices $[n]$.

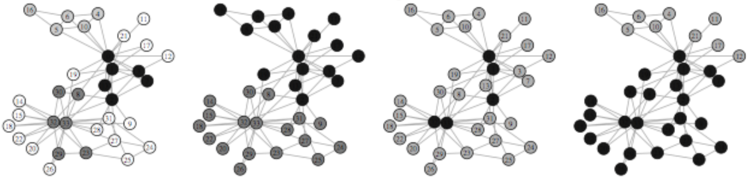
In real networks:

- Overlap: Vertices may belong to two or more communities
- Background: Vertices may belong to no community

Potential Issue: Many methods, many results...

3 comms + background

2 comms; all different partitions



- *Zhao, et al. 2011*

Recall Paths and Path Lengths

- A **path** exists between u and v if there is a collection of edges

$$P(u,v) = \{(u,u_1),(u_1,u_2),\ldots,(u_{k-1},u_k),(u_k,v)\} \subseteq E$$

  that connects u with v

- The **path length** of a path $P(u,v)$:

$$\mathcal{L}(P(u,v)) = A_{u,u_1} + A_{u_k,v} + \sum_{i=1}^{k} A_{u_i,u_{i+1}}$$

The Min-k-Cut Approach

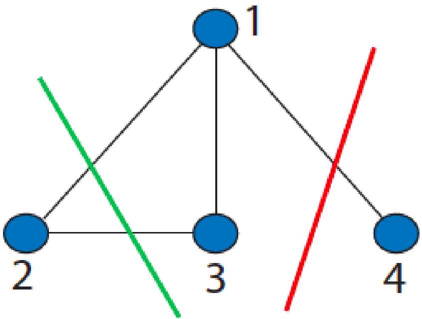
Goal (Min-cut Max flow problem): Find the partition of vertices $\Pi = C_1 \cup \ldots \cup C_k$ whose communities have the *minimum number of edges between them* (Goldberg and Tarjan, 1988)

The cut of two communities $C_1, C_2 \subset [n]$ is:

$$cut(C_1, C_2) = \frac{1}{2}\sum_{i,j} A_{i,j}\mathbb{I}(i \in C_1, j \in C_2)$$

We seek the partition -

$$\text{Min-k-Cut}(G) = \text{argmin}_{\Pi}\left(\sum_{\ell=1}^{k-1}\sum_{m=\ell+1}^{k} cut(C_\ell, C_m)\right)$$



Question: What happens if we search for the Min-2-cut when there are nodes that are only connected to one other node?

Normalized Cut

- **Problem with Min-Cut:** Tends to find many singleton communities!
- To address this, one can normalize the cut between two communities by their size (Ratio-Cut) or by their volume (Norm-cut)

Normalized-Cut (Shi and Malik, 2000):

Define the volume of a collection $B \subset [n]$ as: $vol(B) = \sum_{i \in B} d_i$. Then,

$$\text{Min-Norm-k-Cut}(G) = \text{argmin}_{\Pi} \left(\sum_{\ell=1}^{k-1} \sum_{m=\ell+1}^{k} \frac{cut(B_\ell, B_m)}{vol(B_\ell)} + \frac{cut(B_\ell, B_m)}{vol(B_m)} \right)$$

Normalized Cut

- Addresses the issue of singleton communities but ...
- **Issue**: When $k > 2$, finding the solution to the Norm-Cut is NP-hard.
- Fortunately, an approximate solution can be found!

Recall Connected Components

- A **connected component** of an undirected graph is a collection of vertices $C \subseteq V$ such that
  - $P(u, v) \neq \varnothing$ for all $u, v \in C$
  - $P(u, v) = \varnothing$ for all $u \in C$ and $v \in V \backslash C$

Note: Partitioning a network into its connected components is an "extreme" example of community detection. So we want to identify communities that are "like" disjoint connected components

Spectral Clustering and The Graph Laplacian

- Define $D = \text{diag}(d_1, \ldots, d_n) \in \mathbb{R}^{n \times n}$ where
- Graph Laplacian L:

$$L = D - A$$

- Normalized graph laplacian L_{norm}:

$$L_{norm} = D^{-1}L = I - D^{-1}A$$

Properties of the Graph Laplacian

- λ is an eigenvalue of L_{norm} with eigenvector v iff λ and v solve the eigenproblem $Lv = \lambda Dv$
- 0 is an eigenvalue of L and L_{norm} with eigenvector $\mathbf{1}$
- L and L_{norm} are nonnegative definite and have n real-valued eigenvalues $0 = \lambda_1 \leq \ldots \leq \lambda_n$
- L is symmetric

Key Property of the Graph Laplacian

Theorem 1.

Let G be an undirected graph with non-negative weights and let L_{norm} be its normalized graph laplacian.

Let k = the multiplicity of the eigenvalue 0 of L_{norm}. Then,

(1) k is the number of connected components $C_1, \ldots, C_k$ in G

(2) The eigenspace of 0 is spanned by the indicator vectors $\mathbf{1}_{C_i}$

Key Point:

- If G clustered into k disjoint connected components, then we can perfectly identify the k clusters using the k smallest eigenvectors

Spectral Clustering

Algorithm

Input: Adjacency matrix $A \in \mathbb{R}_+^{n \times n}$, number of communities k

1 Calculate normalized graph laplacian L_{norm}

2 Compute

$$X = \text{the } n \times k \text{ matrix of the } k \text{ smallest eigenvectors of } L_{norm}$$

3 Cluster the rows of X using k-means

Output: Clusters $C_1, \ldots, C_k$

Properties of Spectral Clustering

- Requires a prespecified number of clusters k
- Works perfectly in an ideal scenario
- Requires the use of another clustering method (k-means)
- The solution to a relaxed version of the normalized-cut problem

Reference (*seriously, read this*): Ulrike Von Luxburg "A tutorial on spectral clustering" (2006)

Stochastic Block Model (SBM)

- Model-based approach to community detection
- $G = (V = [n], E)$ with binary adjacency matrix A
- Assumes that G has k blocks generated as follows:

1. Community labels $\mathbf{c} = (c_1, \ldots, c_n)$ generated at random:

$$c_1, \ldots, c_n \overset{iid}{\sim} \text{multinomial}(1, \pi = \{\pi_1, \ldots, \pi_k\})$$

2. Conditional on $\mathbf{c}$, $A(u, v)$ are independent Bernoulli rvs with

$$\mathbb{E}[A(u, v)|\mathbf{c}] = P_{c_u, c_v}$$

Reference: Holland, et al. "Stochastic block models: first steps" (1983)

Stochastic Block Model (SBM)

- Observe $G = G_o$, calculate likelihood $\mathcal{L}(\Theta | G_o, k)$ with $\Theta = \{\mathbf{P}, c\}$
- Finding c becomes an estimation problem:

$$\widehat{\Theta} = \arg\max_{\Theta} \mathcal{L}(\Theta | G_o, k)$$

- Requires approximate algorithms like MCMC or variational EM
- **Issue:** Approximate algorithms can be slow!

Properties of SBM

- Requires pre-specified k
- “Best" performance on identifying disjoint communities
- Block labels are consistent (as $n \to \infty$) if for all $i \neq \ell$:

$$nP_{i,i} - nP_{i,\ell} \geq \sqrt{k(nP_{i,i} + (k-1)nP_{i,\ell})}$$

- Algorithms like MCMC and variational EM can be slow!

Modularity

Aim: find the partition of G whose communities contain the highest density of edges relative to the expected density of edges

Remarks:

- Requires a notion of what a *random* network looks like
- The choice of a null network model affects resulting communities
- This is the most widely adopted approach to community detection!

Reference: Mark E Newman "Modularity and community structure in networks" (2004)

Modularity

- Graph $G = (V = [n], E)$, adjacency matrix $A = [A_{u,v}]$
- Modularity ($\mathcal{Q}$): Measures the “significance" of partition **c**:

$$\mathcal{Q}(\mathbf{c}) = \frac{1}{2|E|} \sum_{u,v} \left[\left(A(u,v) - \frac{d_u d(v)}{2|E|} \right) \mathbb{I}\{c_u = c_v\} \right]$$

- Measures the average departure of observed edge density from expected edge density

Modularity Maximization

Aim: Find the labels $c^* \in \{1, \dots, k\}^n$ that maximizes modularity:

$$c^* = \arg\max_c \{\mathcal{Q}\}$$

- NP hard optimization problem
- *Many* approximate algorithms developed

Reference: Santo Fortunato, "Community detection in graphs" (2009). [100+ page review paper]

Community Extraction

Basic Idea:

- Identify communities $C_i \subseteq V$ one at a time via iterative search
- Remove/avoid $C_1, \ldots, C_i$ when searching for C_{i+1}

Virtues:

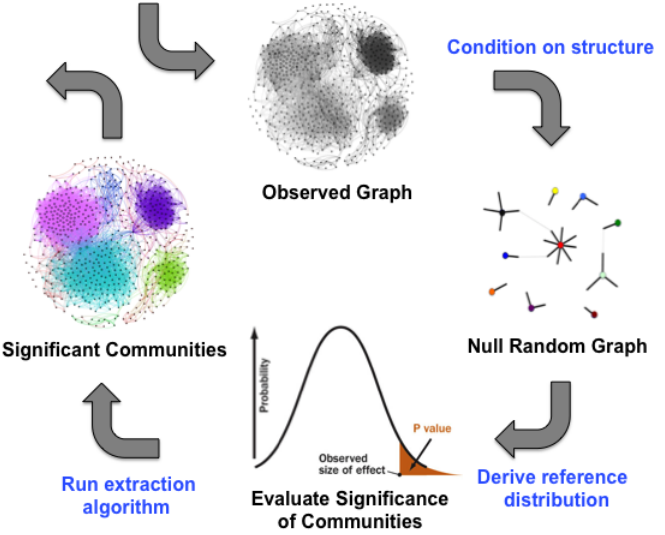
- Possible to accommodate overlap
- Automatic selection of number of communities
- Parallelizable! Can easily scale to large networks.

Community Extraction Methods

Methods:

- OSLOM: Lancichinetti, et al. "Finding statistically significant communities in networks" (2011) – resampling based method
- Extraction: Zhao, et al. "Community extraction for social networks" (2011) – score-based residualizing
- ESSC: Wilson, et al. "A testing based extraction algorithm for identifying significant communities in networks" (2014) – hypothesis testing based extraction

Significance based Community Extraction



Measure Significance Locally

$G_{obs} = ([n], E)$: obs. graph with degree sequence $\mathbf{d} = \{d(1), \ldots, d(n)\}$

Statistic

Let $u \in [n]$; $B \subseteq [n]$

$$D_{obs}(u : B) = \sum_{v \in B} \mathbb{I}(\{u, v\} \in E) \;\; = \;\; \# \text{ edges between } u \text{ and } B$$

Null Distribution (G_{null})

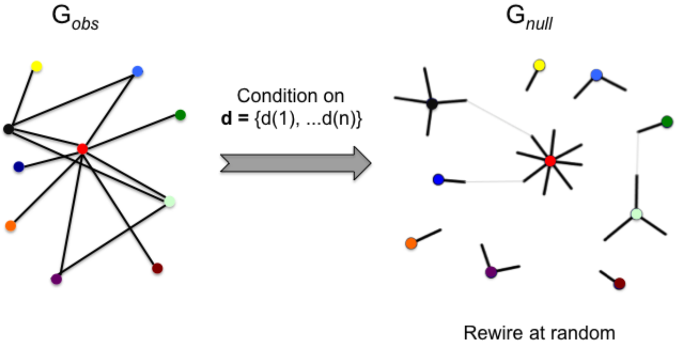
Configuration model $P_{\mathbf{d}}$ based on the degree sequence $\mathbf{d}$

P-Values

Use $P_{\mathbf{d}}$ to evaluate connection between u and B via $D_{obs}(u : B)$

Configuration Null Model (G_{null})

Generative model for G_{obs}



Result: Distribution $P_{\mathbf{d}}$ on family of graphs with degree sequence **d**

The Configuration Null

Features

- Preserves the observed degree sequence **d**.
- Uniform distribution on simple graphs with degree sequence **d**
- Pr(edge between u,v) $\propto d_u d(v)$
- No preferential interconnection between vertices

Idea: Use $P_{\mathbf{d}}$ to test connectivity of a set of nodes in G_{obs}

Local Connectivity p-values

- Quantify statistical signficance of a collection of nodes via the p-value:

$$p_n(u : B) = \Pr\left(D_{null}(u : B) \geq D_{obs}(u : B)\right)$$

- **Fact**: As $n \to \infty$, we have that

$$p_n(u : B) \to \mathbb{P}\{\text{Bin}(d_u, r(B)) \geq D_{obs}(u : B)\}$$

- Above, $r(B)$ is the volume, or density of the collection B.
- Thus, we can test the strength of a collection B using a Binomial tail probability!

The ESSC Algorithm

Single Extraction

Given: Graph $G = ([n], E)$. Significance level $\alpha \in (0, 1)$

Input: Initial set $B_0 \subseteq [n]$

Loop: Until $B_{t+1} = B_t$

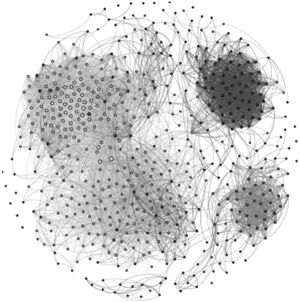
- For each $u \in [n]$, compute p-value $p(u : B_t)$
- Order the vertices of G so that $p(u_1 : B_t) \leq \cdots \leq p(u_n : B_t)$
- Let $k \geq 0$ be the largest integer such that $p(u_k : B_t) \leq (k/n)\alpha$
- Let $B_{t+1} = \{u_1, \ldots, u_k\}$ and increment $t := t + 1$

Return: Community $C = B_t$

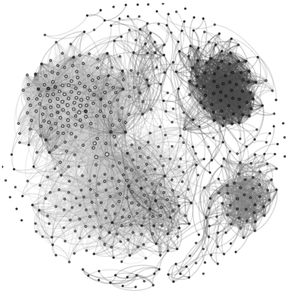
The ESSC Algorithm

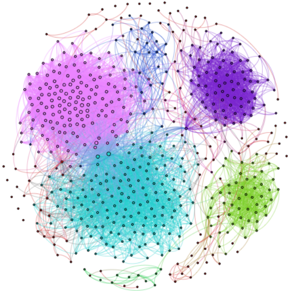
- Repeat *Single Extraction* using vertex neighborhoods as an initial set
- Final collection: set of unique fixed points of search global search
- Code and Readme available at
  https://github.com/jdwilson4/ESSC

Application: Personal Facebook Network



- Network with 561 vertices and 8375 edges
  - **Vertices** (561): friend of mine on Facebook
  - **Edges** (8375): friendships
  - Each individual categorized into one of 8 different locations

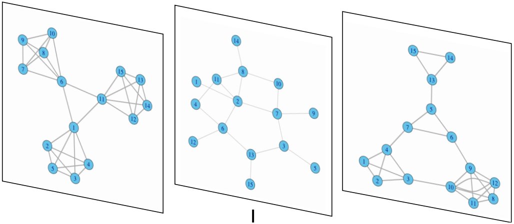




Results of ESSC with $\alpha = 0.05$

- 7 communities detected (size $\approx 68 \pm 51$)
- 18% background vertices (0.90 match with Acquaintances)
- Community match score = 0.87 / 1

Brief Excursion into Multilayer Networks



Network model for multidimensional system

- Unordered sequence of m networks $\mathbf{G}(m, n) = \{([n], E_\alpha)\}_{\ell=1}^{m}$
- $G_\ell = ([n], E_\ell)$ describes relational structure of layer ℓ
  - Each layer represents a relational type or sample
- Can incorporate layer dependencies (correlation, temporal dependence)

Multilayer Community Detection

Three general approaches: Aggregate, Separate, and Super Adjacency

Big Idea: Identify densely connected collections of vertices that persist across layers

Important Considerations:

- What is a multilayer community?
- How to measure significance of a community?
- How to identify communities?

Aggregate Analysis

Ex: m layers; adjacency matrices $\{A_\ell\}_{\ell=1,\ldots,m}$

Approach: Use weighted average:

$$\overline{A} = \sum_{\ell=1}^{m} w_\ell A_\ell$$

Pros/Cons

- ☺ Homogeneous community structure
- ☹ Heterogeneous community structure

Separate Analysis

Ex: m layers; adjacency matrices $\{A_\ell\}_{\ell=1,\ldots,m}$

Approach: Use A_ℓ separately and combine results

Pros/Cons

- ☺ Heterogeneous community structure
- ☹ No use of inter-layer relationships
- ☹ How to combine results?

Super Adjacency Matrices

Ex: $m = 3$ layers; adjacency matrices $\{A_\ell\}_{\ell=1,\ldots,3}$

Approach: Use super adjacency matrix (or, e.g. Laplacian):

$$A_{super} = \begin{pmatrix} A_1 & R_{1,2} & R_{1,3} \\ R_{2,1} & A_2 & R_{2,3} \\ R_{3,1} & R_{3,2} & A_3 \end{pmatrix}, \qquad R_{i,j} = \text{inter-layer connections } i \text{ and } j$$

Pros/Cons

- ☺ Heterogeneous community structure
- ☹ Choice of $R_{i,j}$?

Multilayer Extraction

Setting: Observe multilayer network $\mathbf{G}(m, n) = \{([n], E_\ell)\}_{\ell=1}^m$

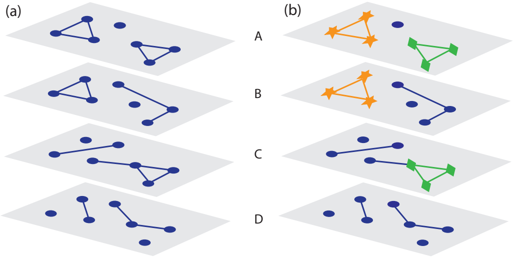
Objective: Find vertex-layer communities $\{(B_\kappa, \mathcal{L}_\kappa)\}_\kappa$ s.t.

- $B_\kappa \subseteq$ vertices $([n])$, $\mathcal{L}_\kappa \subseteq$ layers $([m])$
- Vertices B_κ densely connected *within* layers $\mathcal{L}_\kappa$

Upshot:

- Can capture heterogeneous community structure
- Identified communities (layers and/or vertices) may overlap
- Not all vertices nor layers need belong to a community

Multilayer Extraction: Example



- Two communities: (i) $(\{\star, \star, \star\}, \{A, B\})$ (ii) $(\{\blacksquare, \blacksquare, \blacksquare\}, \{A, C\})$
- Layer D, and vertex $\bullet$ belong to no communities

Score-based Approach to Multilayer Extraction

Given $\mathbf{G}(m, n) = \{([n], E_\ell)\}_{\ell=1}^{m}$ and candidate community $(B, \mathcal{L})$

Outline:

- Formulate null model for $\mathbf{G}(m, n)$
- Compare edges in $(B, \mathcal{L})$ w/ expected number in null model
- Score $(B, \mathcal{L})$ according to significance

Reference: Wilson et al. "Community Extraction in Multilayer Networks with Heterogeneous Community Structure" (2016)