FDP: NETWORK SCIENCE

Community Detection

5 December 2018

Rakhi Saxena
Deshbandhu College,
University of Delhi,
Delhi, India.

Outline

1. Introduction
2. Basics of Communities
3. Community Detection Methods
4. Graph Partitioning
5. Hierarchical Clustering
6. Modularity Optimization
7. Fast Algorithms for Community Detection
8. Overlapping Communities
9. Summary

Outline

Communities in the Real World

- Observation: A characerstic common to many real-world networks is community structure [RCC$^+$04]

Communities in the Real World

- Observation: A characerstic common to many real-world networks is community structure [RCC+04]

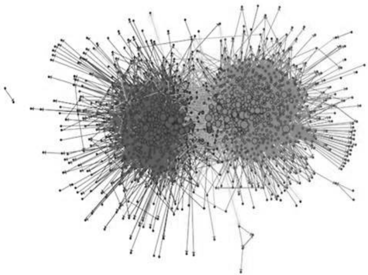


Figure: Directed network of hyperlinks between weblogs on US politics, recorded in 2005 by Adamic and Glance [AG05]

Communities in the Real World

- Observation: A characerstic common to many real-world networks is community structure [RCC+04]

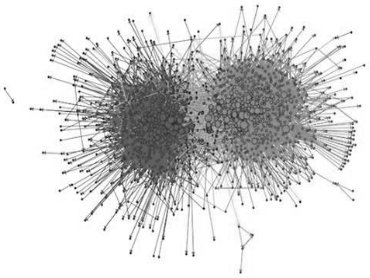


Figure: Directed network of hyperlinks between weblogs on US politics, recorded in 2005 by Adamic and Glance [AG05]

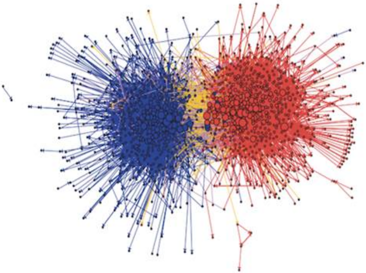
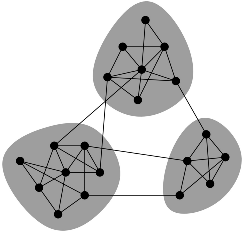


Figure: Community structure of the analysed political blogs. Red for conservative, and blue for liberal. Orange links connect communities.

Images Courtesy of: [AG05]

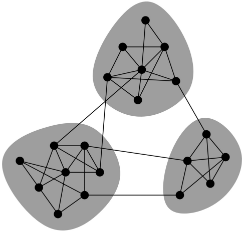
Community Structure in Networks

Communities: Groups of vertices that are densely connected among themselves while being sparsely connected to the rest of the network



Community Structure in Networks

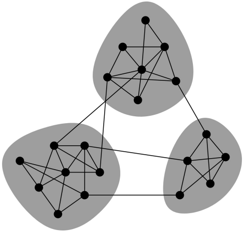
Communities: Groups of vertices that are densely connected among themselves while being sparsely connected to the rest of the network



Let $N_i(T)$ denote the neighbours of i in $T \subseteq V : N_i(T) = \{j \in T : \{i,j\} \in E\}$

Community Structure in Networks

Communities: Groups of vertices that are densely connected among themselves while being sparsely connected to the rest of the network

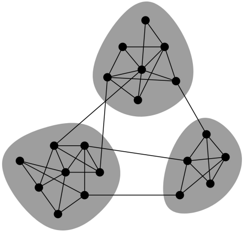


Let $N_i(T)$ denote the neighbours of i in $T \subseteq V$: $N_i(T) = \{j \in T : \{i,j\} \in E\}$

Community Detection Problem: Partition network to maximize (minimize) internal (external) connections

Community Structure in Networks

Communities: Groups of vertices that are densely connected among themselves while being sparsely connected to the rest of the network



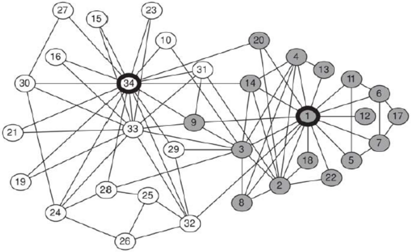
Let $N_i(T)$ denote the neighbours of i in $T \subseteq V \ : \ N_i(T) = \{j \in T : \{i, j\} \in E\}$

Community Detection Problem: Partition network to maximize (minimize) internal (external) connections

Given graph $G(V, E)$, the task of community detection is to find a partition Π of vertices s.t.

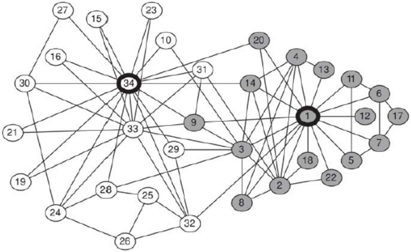
$$\forall i \in V, \forall C \in \Pi \ : \ |N_i(C)| \leq |N_i(\Pi_i)| \qquad (1)$$

Example 1: Zachary Karate Club



- Sociologist Wayne Zachary documented 78 pairwise interactions between 31 karate club members [Zac77]
- Conflict between club president and instructor split the club into two
- Some members followed the instructor and others the president
- Breakup unveiled the ground truth, representing club's underlying community structure

Example 1: Zachary Karate Club



- Sociologist Wayne Zachary documented 78 pairwise interactions between 31 karate club members [Zac77]
- Conflict between club president and instructor split the club into two
- Some members followed the instructor and others the president
- Breakup unveiled the ground truth, representing club's underlying community structure

Community finding algorithms often tested based on ability to infer these two communities from the structure of the network before the split. Zachary's paper has 3553 citations!!

Other Real World Communities

Social Networks: circles of friends / group of individuals who pursue the same hobby together / live in the same neighborhood / co-workers, alumini, relatives, nationalities...

Citation Networks: papers on related topic

Biological Networks: groups of molecules/genes/proteins that form functional modules

Blogosphere: indicate political affiliation/ similar ideology

Web: pages of related content

Explicit/ Implicit Communities in Networks

- Two types of groups in social media

Explicit/ Implicit Communities in Networks

- Two types of groups in social media
- Explicit Groups: formed by user subscriptions
    - Facebook, Yahoo! Groups, LinkedIn

Explicit/ Implicit Communities in Networks

- Two types of groups in social media
- Explicit Groups: formed by user subscriptions
  - Facebook, Yahoo! Groups, LinkedIn
- Implicit Groups: implicitly formed by social interactions
  - Individuals with the same taste for certain movies on a movie rental site.
  - Individuals who write blogs on the same or similar topics

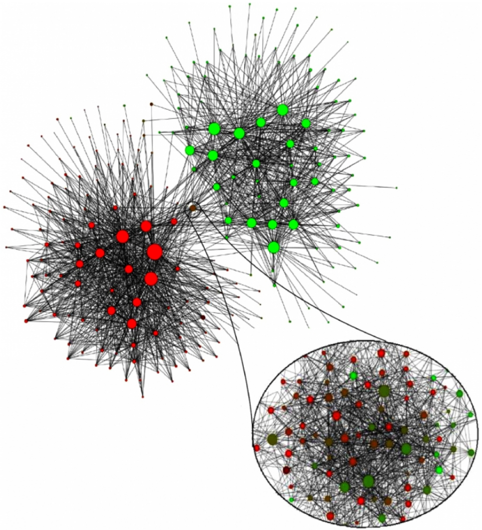
Explicit/ Implicit Communities in Networks

- Two types of groups in social media
- Explicit Groups: formed by user subscriptions
    - Facebook, Yahoo! Groups, LinkedIn
- Implicit Groups: implicitly formed by social interactions
    - Individuals with the same taste for certain movies on a movie rental site.
    - Individuals who write blogs on the same or similar topics

• Explicit communities are not interesting for community detection researchers

• Implicit communities are challenging to find as they are hidden in the network wiring

Implicit Community: Telecom Network



Communities extracted from the call pattern of two million consumers of the largest Belgian mobile phone company. The nodes correspond to communities, size of each node is proportional to the number of individuals in the corresponding community. Color of each community represents the language spoken in the particular community, red for French and green for Dutch. Community connecting the two main clusters captures the culturally mixed capital city Brussels. Source: [BGLM08]

Why Detect Communities?

- Some behaviours are only observable in a group setting and not on an individual level
- Communities provide global view of user interactions, when local-view of individual behaviour is often noisy and ad hoc

Why Detect Communities?

- Some behaviours are only observable in a group setting and not on an individual level
- Communities provide global view of user interactions, when local-view of individual behaviour is often noisy and ad hoc
- Applications in
  - Recommendation systems [RKSR02]
  - Finding functional modules in biological networks [GN02, PDFV05]
  - Controlling spread of epidemics/ Determining influential spreaders [KSB17, ZZWZ13]
  - Graph visualization and summarization [LSDK18]
  - Prediction of missing links/ Identification of false links [Cla08, DJWL16]
  - Finding socially important nodes in networks [SKB18]

Outline

1. Introduction
2. **Basics of Communities**
3. Community Detection Methods
4. Graph Partitioning
5. Hierarchical Clustering
   - Ravasz Algorithm
   - Girvan-Newman Algorithm
6. Modularity Optimization
   - Newman Algorithm
7. Fast Algorithms for Community Detection
   - Louvain Algorithm
   - Label Propagation Algorithm
8. Overlapping Communities
   - Clique Percolation Algorithm
   - Link Clustering
9. Summary

Community Detection - Basics

Fundamental Hypothesis

A network's community structure is uniquely encoded in its wiring diagram

Community Detection - Basics

Fundamental Hypothesis

A network's community structure is uniquely encoded in its wiring diagram

There is a ground truth about a network's community organization, that can be uncovered by inspecting its connectivity patterns (adjacency matrix)

Community Detection - Basics

Connectedness Hypothesis

All members of a community must be reachable through other members of the same community

Community Detection - Basics

Connectedness Hypothesis

All members of a community must be reachable through other members of the same community – necessary, not sufficient

Community Detection - Basics

Connectedness Hypothesis

All members of a community must be reachable through other members of the same community – necessary, not sufficient

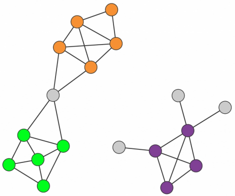


Figure: Connectedness and Density Hypothesis [Bar16]

- Connected Hypothesis: Each community corresponds to a connected subgraph

Density Hypothesis

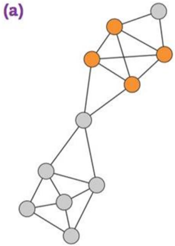
Nodes that belong to a community have higher probability to link to the other members of that community than to nodes in other community

- Density Hypothesis: Nodes in a community have more links between them than outside the community

Community definitions consistent with Connectedness and Density Hypothesis

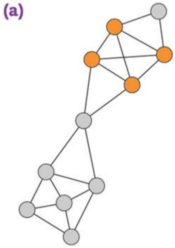
Community definitions consistent with Connectedness and Density Hypothesis

Clique: Group of individuals all of whose members know each other



Community definitions consistent with Connectedness and Density Hypothesis

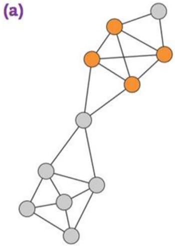
Clique: Group of individuals all of whose members know each other



- Not practical - too restrictive in case of even medium size communities
- In real networks, triangles frequent compared to large cliques

Community definitions consistent with Connectedness and Density Hypothesis

Clique: Group of individuals all of whose members know each other

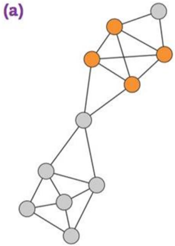


- Not practical - too restrictive in case of even medium size communities
- In real networks, triangles frequent compared to large cliques

- **Relaxation possible —**

Community definitions consistent with Connectedness and Density Hypothesis

Clique: Group of individuals all of whose members know each other

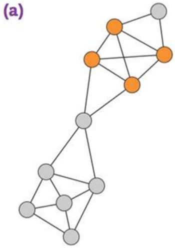


- Not practical - too restrictive in case of even medium size communities
- In real networks, triangles frequent compared to large cliques

• Relaxation possible — by counting connections of nodes wrt connected subgraph C

Community definitions consistent with Connectedness and Density Hypothesis

Clique: Group of individuals all of whose members know each other



- Not practical - too restrictive in case of even medium size communities
- In real networks, triangles frequent compared to large cliques

• Relaxation possible — by counting connections of nodes wrt connected subgraph C

• Internal degree (k_i^{int}) — number of links that connect i to other nodes in C

• External degree (k_i^{ext}) —number of links that connect i to the rest of network

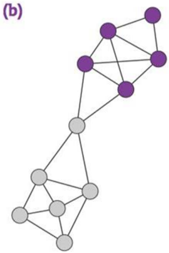
Strong and Weak Communities

Strong and Weak Communities

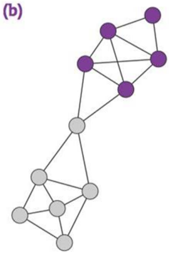
Strong Community: C is strong community if $\forall v_i \in C, k^{int}(v_i) \geq k^{ext}(v_i)$

Strong and Weak Communities

Strong Community: C is strong community if $\forall v_i \in C, k^{int}(v_i) \geq k^{ext}(v_i)$



Strong and Weak Communities

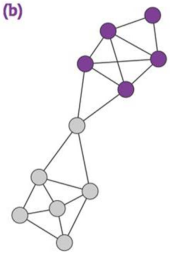


Strong Community: C is strong community if $\forall v_i \in C, k^{int}(v_i) \geq k^{ext}(v_i)$

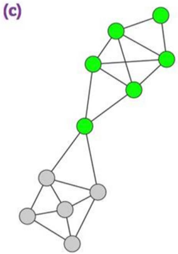
Weak Community: C is weak community if $\forall v_i \in C, \sum(k^{int}(v_i)) \geq \sum(k^{ext}(v_i))$

Strong and Weak Communities

Strong Community: C is strong community if $\forall v_i \in C, k^{int}(v_i) \geq k^{ext}(v_i)$



Weak Community: C is weak community if $\forall v_i \in C, \sum(k^{int}(v_i)) \geq \sum(k^{ext}(v_i))$



Images courtesy of [Bar16]

Outline

Roadmap for rest of the session....

Roadmap for rest of the session....

- Several algorithms to detect communities

Roadmap for rest of the session....

- Several algorithms to detect communities
  - Graph Partitioning Algorithms based on min-cut
  - Hierarchical Clustering Algorithms
  - Algorithms that optimize Modularity
  - Fast Algorithms for detecting communities in large networks
  - Algorithms to detect Overlapping Communities

Outline

Graph Partitioning

- Community: is a vertex subset $C \subset V$, such that $\forall v \in C, v$ has at least as many edges connecting to vertices in C as to vertices in $V \setminus C$
- Cut: A partition of the vertices into disjoint sets
- Min-Cut: Cut in which number of edges between the two sets is minimized

Graph Partitioning

- Community: is a vertex subset $C \subset V$, such that $\forall v \in C$, v has at least as many edges connecting to vertices in C as to vertices in $V \setminus C$
- Cut: A partition of the vertices into disjoint sets
- Min-Cut: Cut in which number of edges between the two sets is minimized

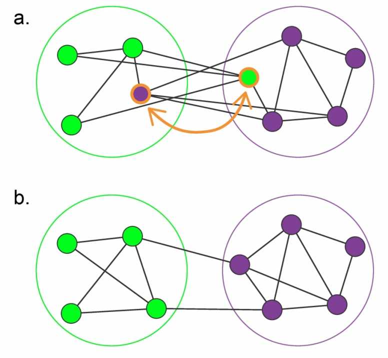
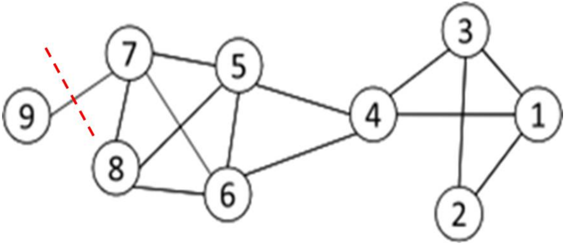


Image courtesy of [Bar16]

Community Detection $\Rightarrow$ Minimum-Cut problem

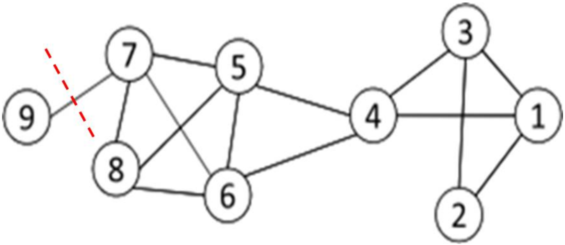
Min-Cut Problem

- Min-cut can return imbalanced partition (one set is a singelton, eg node 9)



Min-Cut Problem

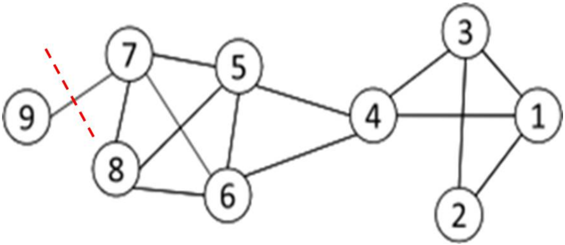
- Min-cut can return imbalanced partition (one set is a singelton, eg node 9)



- Change the objective function to consider community size
- Given partition of k communities $\{C_1, C_2, ...C_k\}$
- Two common minimization objectives

Min-Cut Problem

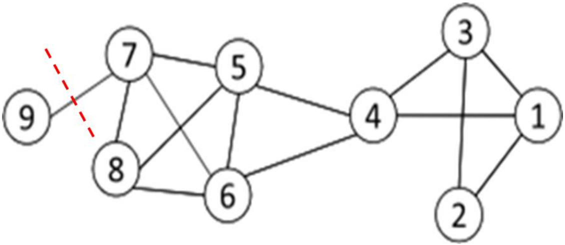
- Min-cut can return imbalanced partition (one set is a singelton, eg node 9)



- Change the objective function to consider community size
- Given partition of k communities $\{C_1, C_2, ...C_k\}$
- Two common minimization objectives
- Ratio Cut Objective Function $\pi = \frac{1}{k}\sum_{i=1}^{k}\frac{cut(C_i,\bar{C_i})}{|C_i|}$

Min-Cut Problem

- Min-cut can return imbalanced partition (one set is a singelton, eg node 9)

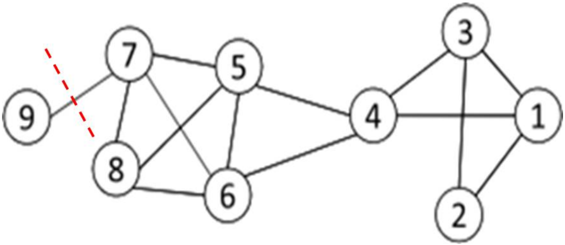


- Change the objective function to consider community size
- Given partition of k communities $\{C_1, C_2, ...C_k\}$
- Two common minimization objectives
- Ratio Cut Objective Function $\pi = \frac{1}{k}\sum_{i=1}^{k} \frac{cut(C_i, \bar{C_i})}{|C_i|}$
- Normalized Cut Objective Function $\pi = \frac{1}{k}\sum_{i=1}^{k} \frac{cut(C_i, \bar{C_i})}{vol(C_i)}$

  where $vol(v_i) = \sum_{v_j \in V_i} d(v_j)$

Min-Cut Problem

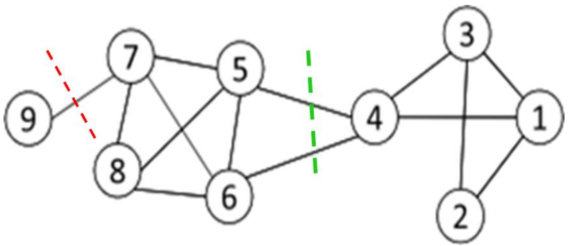
- Min-cut can return imbalanced partition (one set is a singelton, eg node 9)



- Change the objective function to consider community size
- Given partition of k communities $\{C_1, C_2, ...C_k\}$
- Two common minimization objectives
- Ratio Cut Objective Function $\pi = \frac{1}{k}\sum_{i=1}^{k}\frac{cut(C_i,\bar{C_i})}{|C_i|}$
- Normalized Cut Objective Function $\pi = \frac{1}{k}\sum_{i=1}^{k}\frac{cut(C_i,\bar{C_i})}{vol(C_i)}$

  where $vol(v_i) = \sum_{v_j \in V_i} d(v_j)$
- Both Ratio Cut and Normalized Cut prefer a balanced partition

Ratio-Cut and Normalized-Cut Example

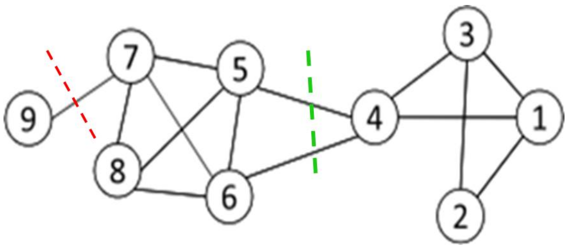


Consider the partition shown by Red cut:

Ratio-Cut $\pi_1 = \frac{1}{2}(\frac{1}{1} + \frac{1}{8}) = 0.56$

Normalized-cut $\pi_1 = \frac{1}{2}(\frac{1}{1} + \frac{1}{27}) = \frac{14}{27} = 0.52$

Ratio-Cut and Normalized-Cut Example



Consider the partition shown by Red cut:

Ratio-Cut $\pi_1 = \frac{1}{2}(\frac{1}{1} + \frac{1}{8}) = 0.56$

Normalized-cut $\pi_1 = \frac{1}{2}(\frac{1}{1} + \frac{1}{27}) = \frac{14}{27} = 0.52$

Consider the partition shown by Green cut:

Ratio-Cut $\pi_2 = \frac{1}{2}(\frac{2}{4} + \frac{2}{5}) = \frac{9}{20} = 0.45 < RC(\pi_1)$

Normalized-cut $\pi_2 = \frac{1}{2}(\frac{2}{12} + \frac{2}{16}) = \frac{7}{48} = 0.15 < NC(\pi_1)$

Graph Partitioning is challenging

- The number of possible ways network can be partitioned into communities grows exponentially with the network size N

The number of possible partitions: Bell number $B_N = \frac{1}{e}\sum_{j=1}^{\infty}\frac{j^N}{j!}$

Graph Partitioning is challenging

- The number of possible ways network can be partitioned into communities grows exponentially with the network size N

The number of possible partitions: Bell number $B_N = \frac{1}{e}\sum_{j=1}^{\infty}\frac{j^N}{j!}$

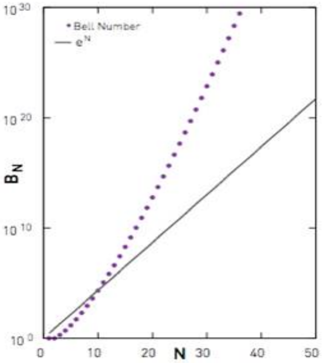


Image courtesy of [Bar16]

Graph Partitioning is challenging

- The number of possible ways network can be partitioned into communities grows exponentially with the network size N

The number of possible partitions: Bell number $B_N = \frac{1}{e}\sum_{j=1}^{\infty}\frac{j^N}{j!}$

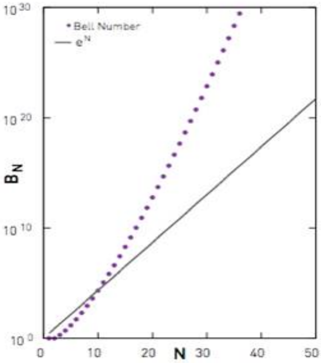


Image courtesy of [Bar16]

- Brute-force approaches computationally infeasible $O(e^N)$
- We need algorithms that can identify communities without inspecting all partitions

Limitation of Graph Partitioning

- In graph partitioning the number and the size of communities is predefined
- In community detection, number and size of groups is determined from the intrinsic network structure
- Need for alternative community detection methods

Outline

1. Introduction
2. Basics of Communities
3. Community Detection Methods
4. Graph Partitioning
5. **Hierarchical Clustering**
   - Ravasz Algorithm
   - Girvan-Newman Algorithm
6. Modularity Optimization
   - Newman Algorithm
7. Fast Algorithms for Community Detection
   - Louvain Algorithm
   - Label Propagation Algorithm
8. Overlapping Communities
   - Clique Percolation Algorithm
   - Link Clustering
9. Summary

Community Detection Method: Hierarchical Clustering

Community Detection Method: Hierarchical Clustering

- Define a similarity measure quantifying some (usually topological) type of similarity between node pairs

Community Detection Method: Hierarchical Clustering

- Define a similarity measure quantifying some (usually topological) type of similarity between node pairs
  - Commonly used measures - Jaccard index, Cosine similarity, Hamming distance between rows of the adjacency matrix

Community Detection Method: Hierarchical Clustering

- Define a similarity measure quantifying some (usually topological) type of similarity between node pairs
  - Commonly used measures - Jaccard index, Cosine similarity, Hamming distance between rows of the adjacency matrix
- Group similar nodes into communities according to chosen measure

Community Detection Method: Hierarchical Clustering

- Define a similarity measure quantifying some (usually topological) type of similarity between node pairs
  - Commonly used measures - Jaccard index, Cosine similarity, Hamming distance between rows of the adjacency matrix
- Group similar nodes into communities according to chosen measure
  - Commonly used schemes - single linkage, complete linkage, average linkage

Community Detection Method: Hierarchical Clustering

- Define a similarity measure quantifying some (usually topological) type of similarity between node pairs
  - Commonly used measures - Jaccard index, Cosine similarity, Hamming distance between rows of the adjacency matrix
- Group similar nodes into communities according to chosen measure
  - Commonly used schemes - single linkage, complete linkage, average linkage
- Agglomerative or Divisive Procedure?

Community Detection Method: Hierarchical Clustering

- Define a similarity measure quantifying some (usually topological) type of similarity between node pairs
  - Commonly used measures - Jaccard index, Cosine similarity, Hamming distance between rows of the adjacency matrix
- Group similar nodes into communities according to chosen measure
  - Commonly used schemes - single linkage, complete linkage, average linkage
- Agglomerative or Divisive Procedure?
  - Agglomerative: merge nodes with high similarity into the same community
  - Divisive: isolate communities by removing low similarity links that tend to connect communities

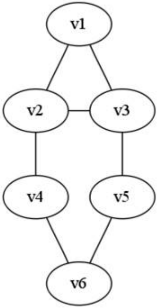
Community Detection Method: Hierarchical Clustering

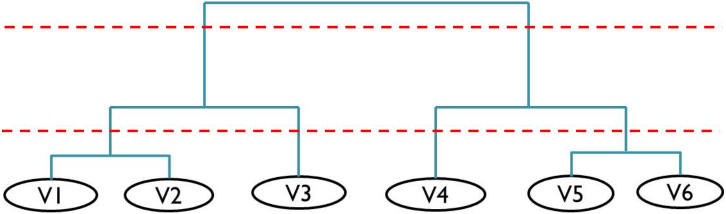
- Define a similarity measure quantifying some (usually topological) type of similarity between node pairs
  - Commonly used measures - Jaccard index, Cosine similarity, Hamming distance between rows of the adjacency matrix
- Group similar nodes into communities according to chosen measure
  - Commonly used schemes - single linkage, complete linkage, average linkage
- Agglomerative or Divisive Procedure?
  - Agglomerative: merge nodes with high similarity into the same community
  - Divisive: isolate communities by removing low similarity links that tend to connect communities
- Both procedures generate a dendrogram in order to extract the underlying community organization

Community Detection Method: Hierarchical Clustering

- Define a similarity measure quantifying some (usually topological) type of similarity between node pairs
  - Commonly used measures - Jaccard index, Cosine similarity, Hamming distance between rows of the adjacency matrix
- Group similar nodes into communities according to chosen measure
  - Commonly used schemes - single linkage, complete linkage, average linkage
- Agglomerative or Divisive Procedure?
  - Agglomerative: merge nodes with high similarity into the same community
  - Divisive: isolate communities by removing low similarity links that tend to connect communities
- Both procedures generate a dendrogram in order to extract the underlying community organization
  - Dendrogram - visualizes the order in which the nodes are assigned to specific communities
  - User chooses the cut in the dendrogram to identify the communities

Example: Hierarchical Clustering

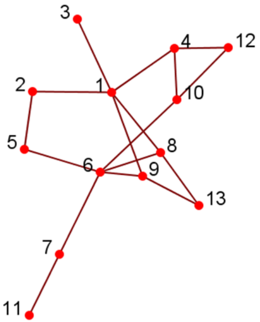




Finding similarity between node pairs

- **Assumption: Nodes that connect to each other and share neighbors are likely to belong to the same community**

$$Jaccard(v_i, v_j) = \frac{|N_i \cap N_j|}{|N_i \cup N_j|} \qquad Cosine(v_i, v_j) = \frac{|N_i \cap N_j|}{\sqrt{|N_i \cdot N_j|}}$$



$$Jaccard(5, 8) = \frac{|\{6\}|}{|\{1,2,6,13\}|} = \frac{1}{4}$$
$$Cosine(5, 8) = \frac{1}{\sqrt{2 \cdot 3}} = \frac{1}{\sqrt{6}}$$

Cosine Similarity uses the cosine of the angle between the i-th and j-th rows/column vectors of the adjacency matrix. Thus cosine similarity of v_i and v_j is the number of common neighbours divided by the geometric mean of their degrees.

Deciding Similarity between Community Pairs

As nodes are merged into small communities.... need to measure how similar two communities are

- Single Linkage : Smallest similarity between all pairs of nodes i, j which belongs to two different groups
- Complete Linkage: Maximum similarity between all pairs of nodes i, j which belongs to two different groups
- Average Linkage: Average similarity between all pairs of nodesi, j which belongs to two different groups

Example: Schemes for grouping community pairs

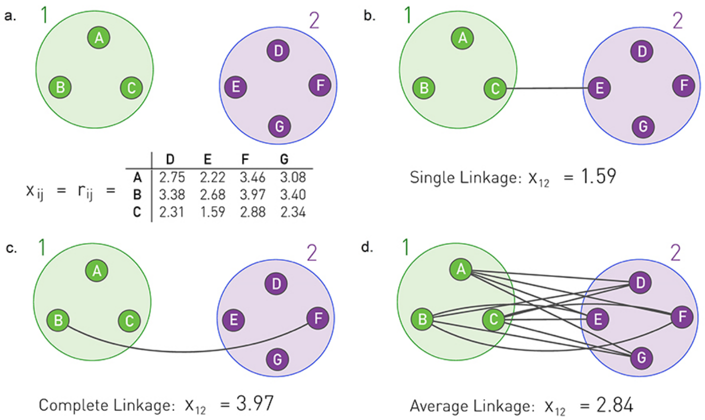


Image courtesy of [Bar16]

Algorithms for Hierarchical Clustering

- Two commonly used hierarchical community detection algorithms:
  - Agglomerative Method: Ravasz Algorithm [RSM$^+$02]
  - Divisive Method: Girvan-Newman Algorithm [GN02]

Agglomerative Method: Ravasz Algorithm [RSM+02]

- Define topological overlap matrix X^o, where

$$x^o_{ij} = \frac{J(i,j)}{min(k_i, k_j) + 1 - \Theta(A_{ij})} \tag{2}$$

  $J(i,j)$ is the number of common neighbors (add 1 if there is direct link between i, j; k_i, k_j are respective degrees of two nodes ; $\Theta(A_{ij}) = 1$, iff $A_{ij} > 0$, else 0
- For grouping communities, Ravasz algorithm uses the average cluster similarity method

Agglomerative Method: Ravasz Algorithm [RSM+02]

- Define topological overlap matrix X^o, where

$$x^o_{ij} = \frac{J(i,j)}{min(k_i, k_j) + 1 - \Theta(A_{ij})} \tag{2}$$

  $J(i,j)$ is the number of common neighbors (add 1 if there is direct link between i, j; k_i, k_j are respective degrees of two nodes ; $\Theta(A_{ij}) = 1$, iff $A_{ij} > 0$, else 0

- For grouping communities, Ravasz algorithm uses the average cluster similarity method

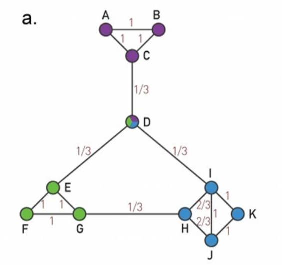
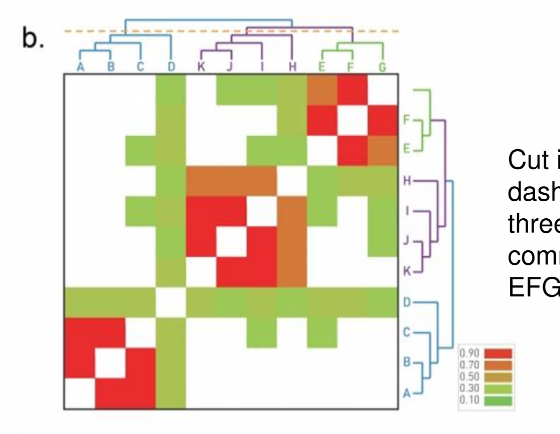


Image courtesy of [Bar16]

● $x^0_{AB} = 1$: A and B have a link to each other and have the same neighbors
● $x^0_{AE} = 0$: A and E do not have common neighbors, nor link to each other
● High topological overlap for members of the same dense local network neighborhood like nodes H, I, J, K

Ravasz Algorithm identifies nested community structure



Cut indicated by dashed line recovers three obvious communities (ABCD, EFG, and HIJK)

Image courtesy of [Bar16]

Ravaz Algorithm: Computational Complexity

- Construction of similarity matrix has computational complexity $0(N^2)$
- Finding group similarity has computational complexity $0(N^2)$
- The construction of the dendrogram can be performed in $0(N \log N)$
- Overall complexity is $0(N^2)$, much faster than brute-force algorithm for finding communities

Divisive Method: Girvan-Newman Algorithm [GN02]

1. Compute the centrality of each link
   - centrality is high if nodes *i* and *j* belongs to different communities else small
2. Remove the link with the largest centrality. In case of a tie, choose one link randomly.
3. Recalculate the centrality of each link for the altered network.
4. Repeat steps 2 and 3 until all links are removed.
5. Cut the dendrogram to extract the underlying community organization

Divisive Method: Girvan-Newman Algorithm contd.

Two centrality measures used by the algorithm:

1. Link Betweenness -
   - x_{ij} is the number of shortest paths that go through the link $i \rightarrow j$
   - computational complexity $0(N^2)$

Divisive Method: Girvan-Newman Algorithm contd.

Two centrality measures used by the algorithm:

1. Link Betweenness -
   - x_{ij} is the number of shortest paths that go through the link $i \to j$
   - computational complexity $0(N^2)$
2. Random Walk Betweenness -
   - A pair of nodes m and n are chosen at random
   - A walker starts at m, following each adjacent link with equal probability until it reaches n
   - x_{ij} is the probability that the link $i \to j$ was crossed by the walker after averaging over all possible choices for the starting nodes m and n
   - computational complexity $0(N^3)$

Divisive Method: Girvan-Newman Algorithm contd.

Two centrality measures used by the algorithm:

1. Link Betweenness -
   - x_{ij} is the number of shortest paths that go through the link $i \rightarrow j$
   - computational complexity $0(N^2)$
2. Random Walk Betweenness -
   - A pair of nodes m and n are chosen at random
   - A walker starts at m, following each adjacent link with equal probability until it reaches n
   - x_{ij} is the probability that the link $i \rightarrow j$ was crossed by the walker after averaging over all possible choices for the starting nodes m and n
   - computational complexity $0(N^3)$

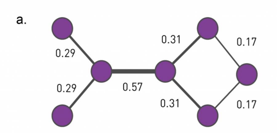


Figure: Link Betweenness

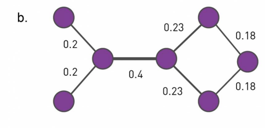
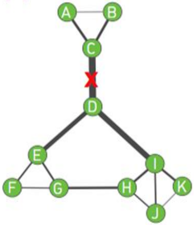


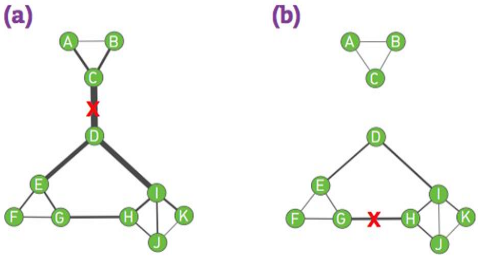
Figure: Random Walk Betweenness

Example: Girvan Newman Algorithm

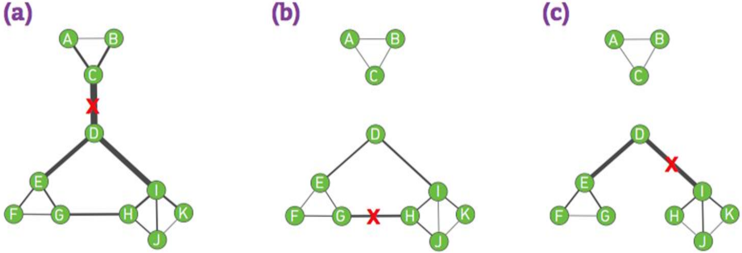
(a)



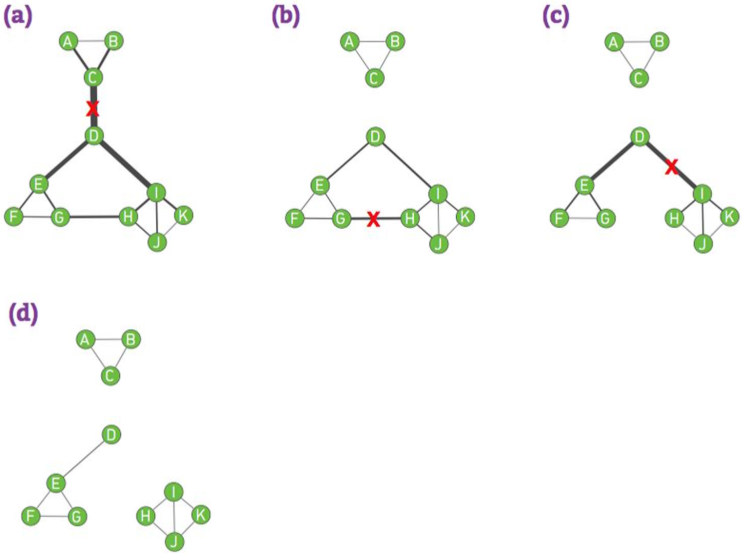
Example: Girvan Newman Algorithm



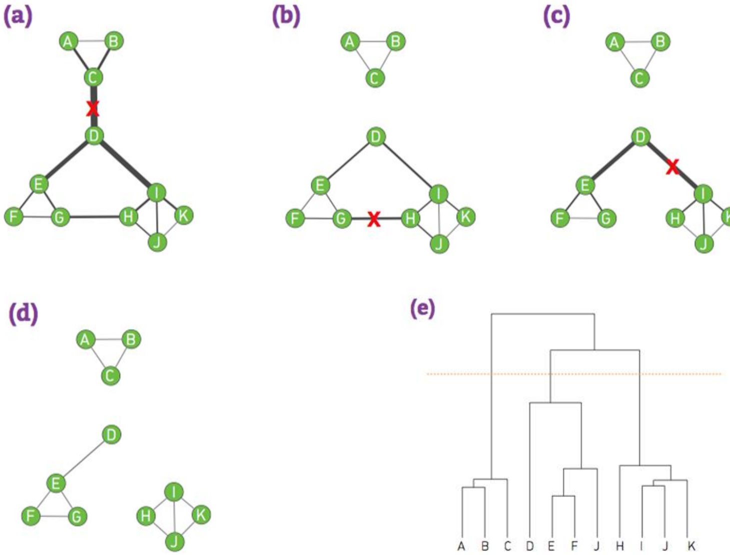
Example: Girvan Newman Algorithm



Example: Girvan Newman Algorithm



Example: Girvan Newman Algorithm



Images courtesy of [Bar16]

Girvan-Newman Algorithm: Computational Complexity

- Calculation of centrality is the time limiting step
- Fastest known link betweenness algorithm is $0(LN)$
- Recalculating centrality of each link introduces a factor of L
- Hence the algorithm scales as $0(L^2N)$ or $0(N^3)$ for sparse networks

Ambiguity in Hierarchical Clustering

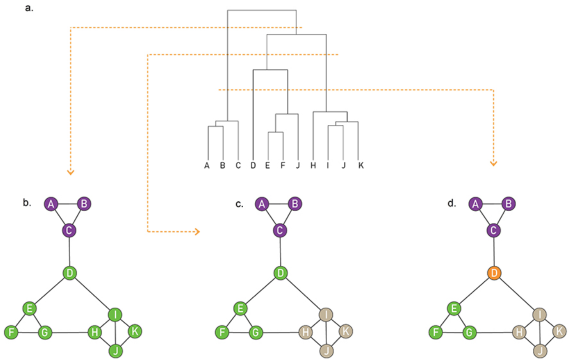


Image courtesy of [Bar16]

- Hierarchical clustering does not tell where to cut the dendrogram
  - Can obtain (b) two, (c) three or (d) four communities depending on where the cut is made

Ambiguity in Hierarchical Clustering

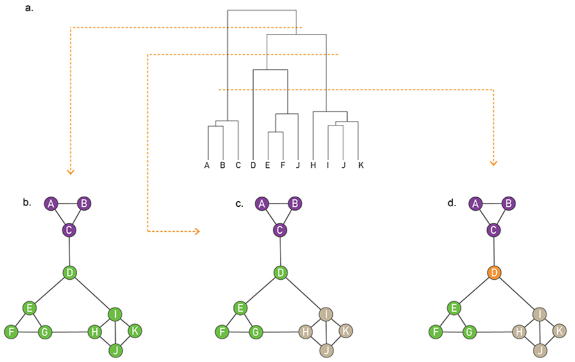


Image courtesy of [Bar16]

- Hierarchical clustering does not tell where to cut the dendrogram
  - Can obtain (b) two, (c) three or (d) four communities depending on where the cut is made
- Can visually decide which cut captures best the underlying community structure for small network; Impossible to visually inspect large networks

Ambiguity in Hierarchical Clustering

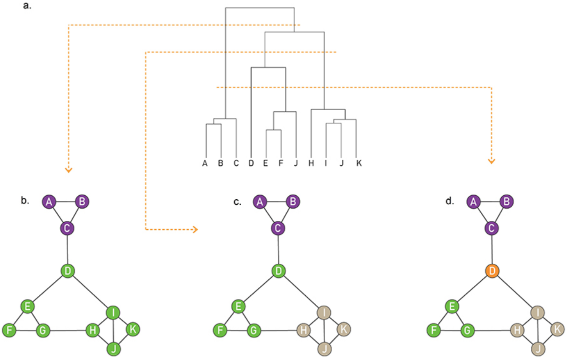


Image courtesy of [Bar16]

- Hierarchical clustering does not tell where to cut the dendrogram
  - Can obtain (b) two, (c) three or (d) four communities depending on where the cut is made
- Can visually decide which cut captures best the underlying community structure for small network; Impossible to visually inspect large networks
- Solution: modularity: helps to select the optimal cut

Outline

Modularity Optimization

- Modularity measures the quality of each partition

Random Hypothesis: Randomly wired networks lack an inherent community structure

- In random networks the connection pattern is expected to be uniform, lacking local density fluctuations that could be interpreted as communities
- Systematic deviations from a random configuration allow us to define a quantity called modularity
- How? .....

Modularity Optimization

- Modularity measures the quality of each partition

Random Hypothesis: Randomly wired networks lack an inherent community structure

- In random networks the connection pattern is expected to be uniform, lacking local density fluctuations that could be interpreted as communities
- Systematic deviations from a random configuration allow us to define a quantity called modularity
- How? .....
  - Compare link density of a community with

Modularity Optimization

- Modularity measures the quality of each partition

Random Hypothesis: Randomly wired networks lack an inherent community structure

- In random networks the connection pattern is expected to be uniform, lacking local density fluctuations that could be interpreted as communities
- Systematic deviations from a random configuration allow us to define a quantity called modularity
- How? .....
  - Compare link density of a community with
  - link density obtained for the same group of nodes for a randomly rewired network

Modularity Optimization

- Modularity measures the quality of each partition

Random Hypothesis: Randomly wired networks lack an inherent community structure

- In random networks the connection pattern is expected to be uniform, lacking local density fluctuations that could be interpreted as communities
- Systematic deviations from a random configuration allow us to define a quantity called modularity
- How? .....
  - Compare link density of a community with
  - link density obtained for the same group of nodes for a randomly rewired network
- Modularity(M) — measure associated to a partition
- Modularity optimization offers a novel approach to community detection
- Find $\{C_c, c = 1, n_c\}$ that maximises M

Modularity

• Consider a network with N nodes and L links and a partition into n_c communities, each community having N_c nodes connected to each other by L_c links
• Measure difference between network's real wiring diagram (A_{ij}) and expected number of links between i and j if the network is randomly wired (p_{ij})

$$M_c = \frac{1}{2L} \sum_{(i,j)\ \in C_c} (A_{ij} - p_{ij}) \tag{3}$$

• Using the degree preserving null model to keep expected degree of each node unchanged

$$p_{ij} = \frac{k_i \cdot k_j}{2L} \tag{4}$$

$$\Rightarrow M_c = \frac{L_c}{L} - \left(\frac{k_c}{2L}\right)^2 \tag{5}$$

k_C is total degree of community C_C

Modularity - continued

- Positive M_c: Subgraph C_c is a potential community (has more links than expected by chance)
- Zero M_c: Connectivity between the N_c nodes is random
- Negative M_c: Nodes of C_c do not form a community

Modularity - continued

- Positive M_c: Subgraph C_c is a potential community (has more links than expected by chance)
- Zero M_c: Connectivity between the N_c nodes is random
- Negative M_c: Nodes of C_c do not form a community
- Partition's modularity computed by summing M_c over all n_c communities

$$M = \sum_{c=1}^{n_c} \left[\frac{L_c}{L} - \left(\frac{k_c}{2L} \right)^2 \right] \tag{6}$$

- Higher Modularity Implies Better Partition
- Taking the whole network as a single community $\Rightarrow M = 0$
- Each node belongs to a separate community $\Rightarrow M < 0$

Modularity - continued

- Positive M_c: Subgraph C_c is a potential community (has more links than expected by chance)
- Zero M_c: Connectivity between the N_c nodes is random
- Negative M_c: Nodes of C_c do not form a community
- Partition's modularity computed by summing M_c over all n_c communities

$$M = \sum_{c=1}^{n_c} \left[\frac{L_c}{L} - \left(\frac{k_c}{2L} \right)^2 \right] \tag{6}$$

- Higher Modularity Implies Better Partition
- Taking the whole network as a single community $\Rightarrow M = 0$
- Each node belongs to a separate community $\Rightarrow M < 0$

Maximal Modularity Hypothesis: For a given network the partition with maximum modularity corresponds to the optimal community structure

Example: Modularity Optimization

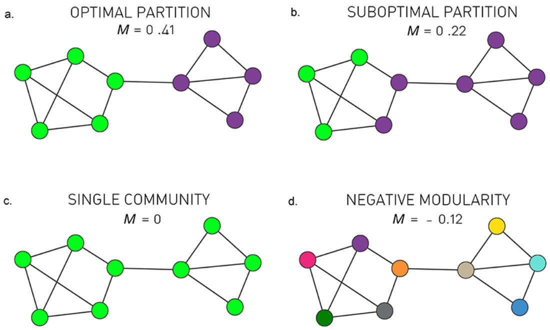


Figure: M for each cut of the dendrogram, finding a clear maximum when the network breaks into three communities.

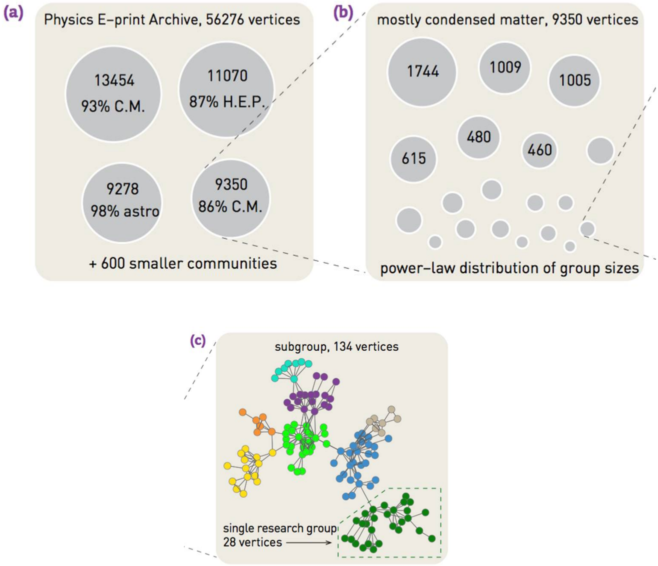
Image courtesy of [Bar16]

Greedy Procedure: Newman Algorithm [New03]

Newman Algorithm

1. Assign each node to a community of its own, starting with N communities of single nodes.
2. Inspect each community pair connected by at least one link and compute modularity difference ΔM if they are merged.
3. Identify the community pair for which ΔM is the largest and merge them.
    - (Note: modularity is always calculated for the full network)
4. Repeat Step 2 until all nodes merge into a single community, recording M for each step.
5. Select the partition for which M is maximal.

Example: Community structure of the collaboration network of physicists obtained using Newman Algorithm



Newman Algorithm: Computational Complexity

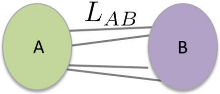
- Calculation of each ΔM - constant time $O(1)$
- Identification of largest ΔM - $O(L)$
- After merging communities, update of matrix - $O(N)$
- Algorithm requires $N - 1$ community mergers
- Complexity is $O[(L + N)N]$ or $O(N^2)$
- Optimized implementations reduce complexity to $O(N \log^2 N)$
- Complexity can be prohibitive for very large networks
- Optimized implementations reduce the algorithm's complexity to $O(N \log^2 N)$

Limits of Modularity - Resolution Limit [FB07]

- Modularity maximization forces together small weakly connected communities

Limits of Modularity - Resolution Limit [FB07]

- Modularity maximization forces together small weakly connected communities



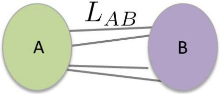
degree in communities A and B

k_A and k_B are total

$$\Delta M = \frac{L_{AB}}{L} - \frac{k_A \cdot k_B}{2L^2} \qquad (7)$$

Limits of Modularity - Resolution Limit [FB07]

- Modularity maximization forces together small weakly connected communities



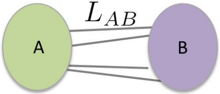
k_A and k_B are total degree in communities A and B

$$\Delta M = \frac{L_{AB}}{L} - \frac{k_A \cdot k_B}{2L^2} \qquad (7)$$

- If $\frac{k_A \cdot k_b}{2L} < 1$, and $L_{AB} \geq 1 \Rightarrow \Delta M > 0$
  - Merge A and B to maximize modularity

Limits of Modularity - Resolution Limit [FB07]

- Modularity maximization forces together small weakly connected communities



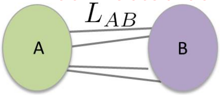
k_A and k_B are total degree in communities A and B

$$\Delta M = \frac{L_{AB}}{L} - \frac{k_A \cdot k_B}{2L^2} \qquad (7)$$

- If $\frac{k_A \cdot k_b}{2L} < 1$, and $L_{AB} \geq 1 \Rightarrow \Delta M > 0$
  - Merge A and B to maximize modularity
- Assuming for simplicity, $k_A \approx k_B = k \Rightarrow k \leq \sqrt{2L}$

Limits of Modularity - Resolution Limit [FB07]

- Modularity maximization forces together small weakly connected communities



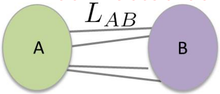
k_A and k_B are total degree in communities A and B

$$\Delta M = \frac{L_{AB}}{L} - \frac{k_A \cdot k_B}{2L^2} \quad (7)$$

- If $\frac{k_A \cdot k_b}{2L} < 1$, and $L_{AB} \geq 1 \Rightarrow \Delta M > 0$
  - Merge A and B to maximize modularity
- Assuming for simplicity, $k_A \approx k_B = k \Rightarrow k \leq \sqrt{2L}$
- If k_A and k_B are under the threshold ($\sqrt{2L}$), even a single link between them will force the two communities together when M is maximized

Limits of Modularity - Resolution Limit [FB07]

- Modularity maximization forces together small weakly connected communities



k_A and k_B are total degree in communities A and B

$$\Delta M = \frac{L_{AB}}{L} - \frac{k_A \cdot k_B}{2L^2} \qquad (7)$$

- If $\frac{k_A \cdot k_b}{2L} < 1$, and $L_{AB} \geq 1 \Rightarrow \Delta M > 0$
  - Merge A and B to maximize modularity
- Assuming for simplicity, $k_A \approx k_B = k \Rightarrow k \leq \sqrt{2L}$
- If k_A and k_B are under the threshold ($\sqrt{2L}$), even a single link between them will force the two communities together when M is maximized
- Example - For WWW graph sample with $L = 1,497,134$ modularity maximization will have difficulties resolving communities with total degree $k_C \leq 1,730$

Limits of Modularity - Modularity Maxima

- Networks lack a clear modularity maxima, developing instead a modularity plateau containing many partitions with hard to distinguish modularity
- Consider a network composed of n_c subgraphs with comparable link densities $k_c \approx \frac{2L}{n_c}$
- If we merge a pair of clusters, we change modularity with

$$\Delta M = \frac{L_{AB}}{L} - \frac{2}{n_c^2} \tag{8}$$

- Drop in modularity is less than $\Delta M = -\frac{2}{n_c^2}$

Limits of Modularity - Modularity Maxima

- Networks lack a clear modularity maxima, developing instead a modularity plateau containing many partitions with hard to distinguish modularity
- Consider a network composed of n_c subgraphs with comparable link densities $k_c \approx \frac{2L}{n_c}$
- If we merge a pair of clusters, we change modularity with

$$\Delta M = \frac{L_{AB}}{L} - \frac{2}{n_c^2} \tag{8}$$

- Drop in modularity is less than $\Delta M = -\frac{2}{n_c^2}$
- Example: For a network with $n_c = 20$ communities, at most $\Delta M = -0.005$, tiny compared to the maximal modularity $M = 0.87$

Example: Limits of Modularity

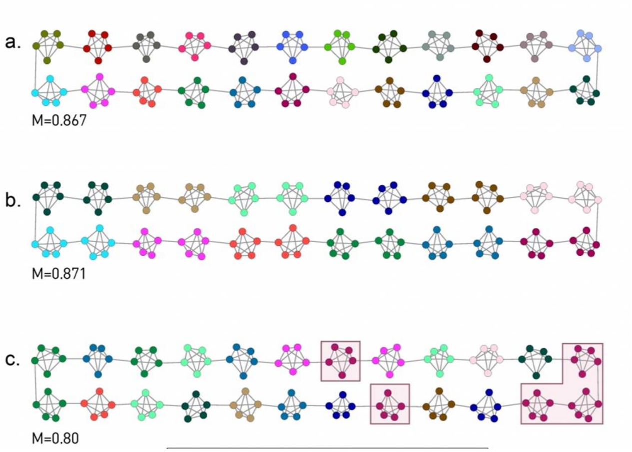


Figure: A ring network consisting of 24 cliques, each made of 5 nodes. Source: [Bar16]

(a) The best partition should correspond to the configuration where each cluster is a separate community. M=0.867
(b) When clusters are combined into pairs, M=0.871, higher than M of intuitive partition
(c) When clusters assigned randomly to communities - clusters that have no links to each other may end up in the same community. M=0.80 $\approx$ 0.87

Outline

1. Introduction
2. Basics of Communities
3. Community Detection Methods
4. Graph Partitioning
5. Hierarchical Clustering
   - Ravasz Algorithm
   - Girvan-Newman Algorithm
6. Modularity Optimization
   - Newman Algorithm
7. **Fast Algorithms for Community Detection**
   - Louvain Algorithm
   - Label Propagation Algorithm
8. Overlapping Communities
   - Clique Percolation Algorithm
   - Link Clustering
9. Summary

Fast Algorithms for Community Detection

- Used to identify communities in very large networks
- Recently two algorithms have gained popularity
  - Louvain algorithm [BGLM08] - aims to optimize modularity
  - Label Propagation [RAK07] algorithm - consensus based

Louvain Algorithm [BGLM08]

- **Step I**
  - Given: weighted network of N nodes
  - Assign each node to its own community
  - For each node *i*: evaluate gain in modularity if *i* is placed in community of one of its neighbors *j*
  - Move *i* in the community for which the modularity gain is the largest – but only if gain is +ve

Louvain Algorithm [BGLM08]

- **Step I**
  - Given: weighted network of N nodes
  - Assign each node to its own community
  - For each node *i*: evaluate gain in modularity if *i* is placed in community of one of its neighbors *j*
  - Move *i* in the community for which the modularity gain is the largest – but only if gain is +ve
  - If no positive gain is found, *i* stays in its original community
  - Process is applied to all nodes until no further improvement can be achieved

Louvain Algorithm [BGLM08]

- **Step I**
  - Given: weighted network of N nodes
  - Assign each node to its own community
  - For each node i: evaluate gain in modularity if i is placed in community of one of its neighbors j
  - Move i in the community for which the modularity gain is the largest – but only if gain is +ve
  - If no positive gain is found, i stays in its original community
  - Process is applied to all nodes until no further improvement can be achieved
  - Modularity change ΔM calculated using

$$\Delta M = \left[\frac{\sum_{in} + 2k_{i,in}}{2W} - \left(\frac{\sum_{tot} + k_i}{2W}\right)^2\right] - \left[\frac{\sum_{in}}{2W} - \left(\frac{\sum_{tot}}{2W}\right)^2 - \left(\frac{k}{2W}\right)^2\right]$$

    - $\sum_{in}$ is the sum of the weights of the links inside C
    - $\sum_{tot}$ is the sum of the link weights of all nodes in C
    - k_i is the sum of the weights of the links incident to node i
    - $k_{i,in}$ is the sum of the weights of the links from i to nodes in C
    - W is the sum of the weights of all links in the network

Louvain Algorithm ... contd

- **Step II**
  - Construct a new network whose nodes are the communities identified during Step I
  - Weight of link between two nodes is the sum of the weight of the links between the nodes in the corresponding communities
  - Links between nodes of the same community lead to weighted self-loops
- Repeat Steps I - II - each iteration called a *pass*
- Number of communities decreases with each pass
- Passes repeated until there are no more changes and maximum modularity is attained

Example: Louvain Algorithm

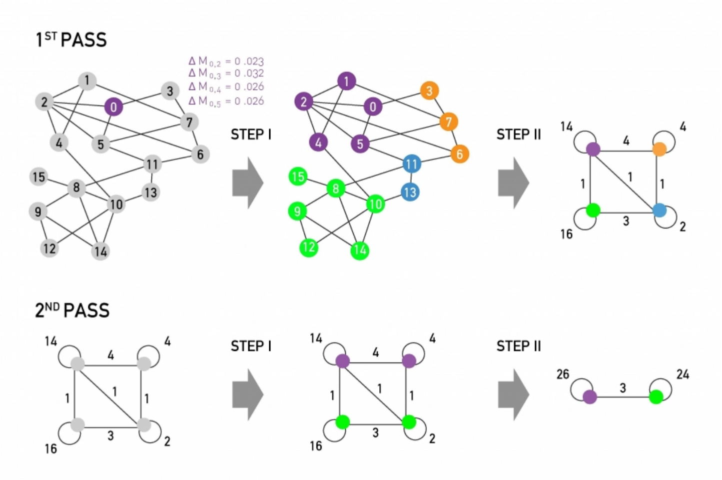


Image courtesy of [BGLM08]

Louvain Algorithm - Computational Complexity

- Algorithm more limited by storage demands than by computational time
- Number of computations scale linearly with L for first pass
- For subsequent passes, complexity of the algorithm is at most $O(L)$
- Therefore algorithm allows identification of communities in networks with millions of nodes

Label Propagation Algorithm [RAK07]

- Consensus Based Algorithm
- Every node is assigned a unique label
- In subsequent iterations, each node takes the most frequent label of its neighbors in a synchronous manner
- Method stops when the label of each node is one of the most frequent labels in its neighborhood

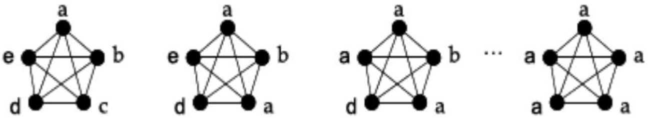


Figure: Nodes are updated one by one as we move from left to right. Due to a high density of edges (highest possible in this case), all nodes acquire the same label

Image courtesy of [RAK07]

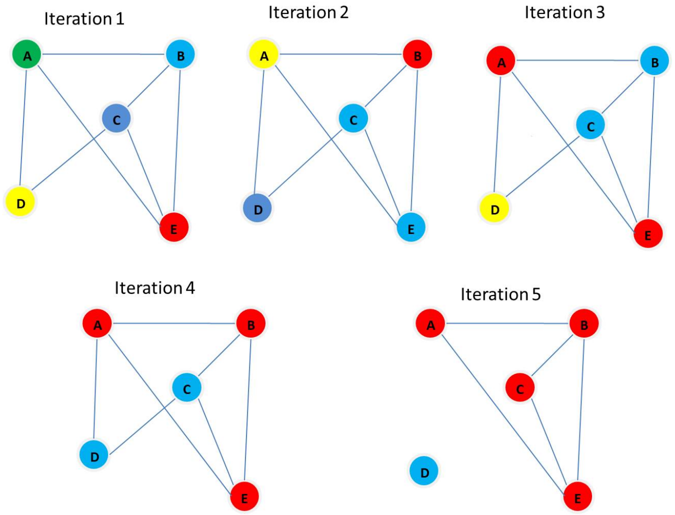
Label Propagation Algorithm...contd.

1. Initialize the labels of all nodes in the network. For node x, $C_x(0) = x$
2. Initialize $t = 1$
3. Arrange the nodes in random order and set it to X
4. For each $x \in X$ chosen in that specific order

$$C_x(t) = f(C_{x1}(t), ..., C_{xm}(t), C_{x(m+1)}(t-1), ..., C_{xk}(t-1)) \quad (9)$$

5. f here returns the label occurring with the highest frequency among the neighbors and ties are broken uniformly randomly
6. If every node has a label that the maximum number of their neighbors have, stop the algorithm. Else increment t and go to Step 3

Example: Label Propagation Algorithm

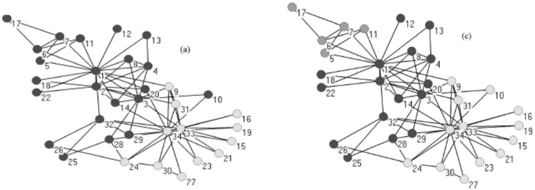


Ambiguity in Label propagation Algorithm

- Problem: Algorithm yields different results based on the initial configuration (which is decided randomly)

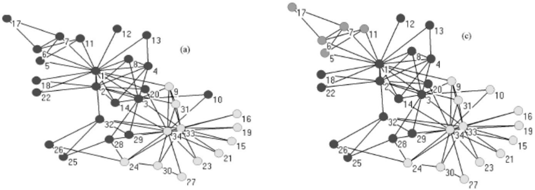
Ambiguity in Label propagation Algorithm

- Problem: Algorithm yields different results based on the initial configuration (which is decided randomly)



Ambiguity in Label propagation Algorithm

- Problem: Algorithm yields different results based on the initial configuration (which is decided randomly)



- Solution: Run the method a large number of times (say, 1000 times) and then build a consensus labelling

LPA: Computational Complexity

- Takes near-linear time
- Initialization of nodes with unique labels takes $O(N)$ time
- Each iteration of LPA takes linear time in number of edges $O(L)$
- At each node x
  - Grouping neighbors according to their labels $(Od(x))$
  - Picking the group of maximum size and assigning its label to $x \Rightarrow O(d(x))$
- Overall time is $O(L)$ since process repeated at each node

Outline

Overlapping Communities

- In the real world, a node can belong to multiple communities. Example:
  - of colleagues that share professional interests
  - of family members and relatives
  - of individuals sharing hobby

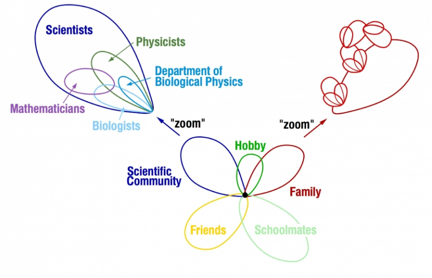


Figure: Communities surrounding Tamas Vicsek, who introduced the concept of overlapping communities [PDFV05]

- Overlapping communities not limited to social systems
  - same genes often implicated in multiple diseases → indicating disease modules of different disorders overlap

Overlapping Communities - Example

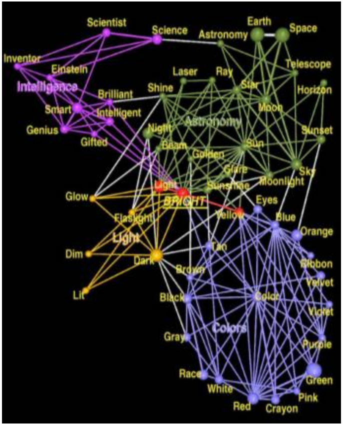


Image Courtesy of [PBV07]

Bright :-

- community containing light-related words (glow or dark);
- community capturing different colors (yellow, brown)
- community consisting of astronomical terms (sun, ray).
- community linked to intelligence (gifted, brilliant).

Algorithms to Detect Overlapping Communities

- Clique Percolation Algorithm [PDFV05]
- Link Clustering Algorithm [ABL10]

Clique Percolation Algorithm [PDFV05]

- Algorithm views a community as the union of overlapping cliques
- Two k-cliques are considered adjacent if they share k – 1 nodes
- A k-clique community is the largest connected subgraph obtained by the union of all adjacent k-cliques
- k-cliques that can not be reached from a particular k-clique belong to other k-clique communities

Example: Clique Percolation Algorithm

- To identify k=3 clique-communities roll a triangle across the n/w such that each subsequent triangle shares one link(2 nodes) with the previous triangle

Example: Clique Percolation Algorithm

- To identify k=3 clique-communities roll a triangle across the n/w such that each subsequent triangle shares one link(2 nodes) with the previous triangle

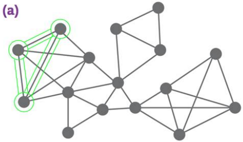


Figure: Start with a k-clique (eg-3-clique)

Example: Clique Percolation Algorithm

- To identify k=3 clique-communities roll a triangle across the n/w such that each subsequent triangle shares one link(2 nodes) with the previous triangle

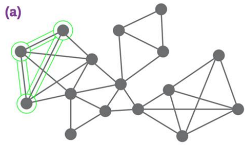


Figure: Start with a k-clique (eg-3-clique)

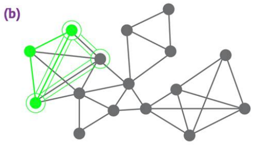


Figure: "Roll" the clique over adjacent cliques

Example: Clique Percolation Algorithm

- To identify k=3 clique-communities roll a triangle across the n/w such that each subsequent triangle shares one link(2 nodes) with the previous triangle

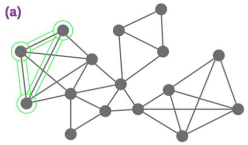


Figure: Start with a k-clique (eg-3-clique)

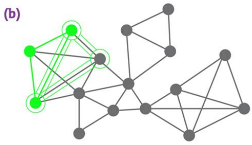


Figure: "Roll" the clique over adjacent cliques

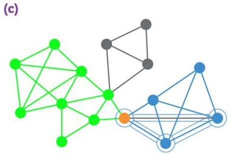


Figure: Clique Communities for k=3: Orange nodes belong to multiple communities

Example: Clique Percolation Algorithm

- To identify k=3 clique-communities roll a triangle across the n/w such that each subsequent triangle shares one link(2 nodes) with the previous triangle

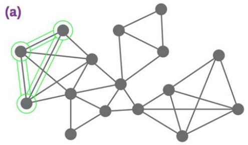


Figure: Start with a k-clique (eg-3-clique)

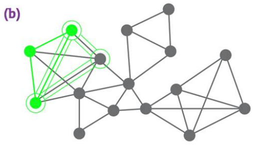


Figure: "Roll" the clique over adjacent cliques

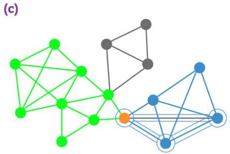


Figure: Clique Communities for k=3: Orange nodes belong to multiple communities

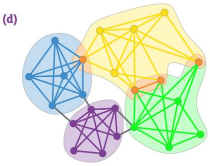


Figure: Clique Communities for k=4; complete 4 node subgraphs sharing at least three nodes

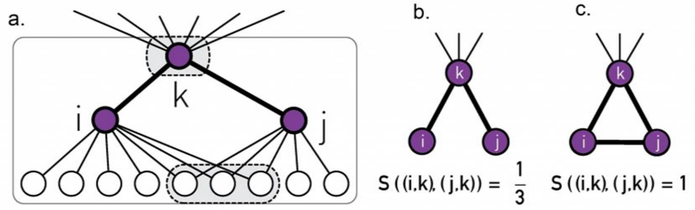
Images courtesy of [PBV07]

Clique Percolation: Computational Complexity

- Algorithm's community definition is based on cliques instead of maximal cliques, which can be identified in polynomial time
- However, if there are large cliques in the network, it is more efficient to identify all cliques using an algorithm with $O(e^N)$ complexity
- Despite this high computational complexity, the algorithm is relatively fast, processing the mobile call network of 4 million mobile phone users in less then one day

Finding Overlapping Communities: Link Clustering [ABL10]

- Intuition:
  - Nodes can belong to multiple communities
  - But links tend to be community specific
- Step 1: Define Link Similarity S
  - S measures the relative number of common neighbors endnodes have
  - $S((i,k),(j,k)) = \frac{|n_+(i) \cap n_+(j)|}{|n_+(i) \cup n_+(j)|}$
  - where $n_+(i)$ is the list of the neighbors of node i, including itself
  - Small $S \rightarrow$ less overlap between neighborhood of the two links



Link Clustering ... contd

- Step 2: Apply Hierarchical Clustering
    - Apply hierarchical clustering to identify link communities using the similarity matrix *S*
- Figure shows single-linkage procedure, iteratively merging communities with the largest similarity link pairs
- Cuts result in the link communities and the overlapping node communities

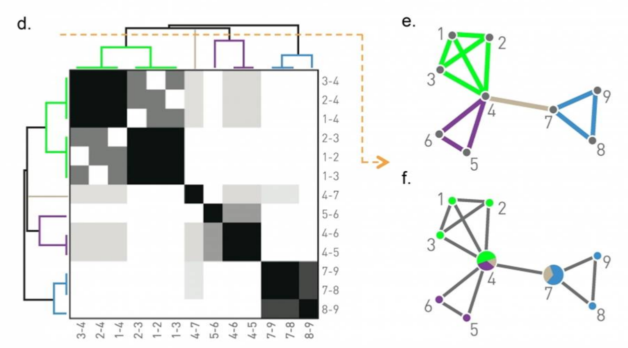


Image Courtesy of [Bar16]

Link Clustering: Computational Complexity

- Similarity calculation:
  - requires $max(k_i, k_j)$ steps for link pair with degrees k_i and k_j
  - For a scale-free network with degree exponent γ, calculation of similarity has complexity $O(N^2/(\gamma - 1))$, determined by the size of the largest node, *kmax*
- Hierarchical clustering: requires $O(L^2)$ time steps
- Total computational complexity is $O(N^2/(\gamma - 1)) + O(L^2)$
- For sparse graphs the latter term dominates, leading to $O(N^2)$

Outline

Summary

- Community identification relies on following hypotheses
  - **Fundamental Hypothesis:** A network's community structure is uniquely encoded in its wiring diagram
  - **Connectedness and Density Hypothesis:** A community corresponds to a locally dense connected subgraph
  - **Random Hypothesis:** Randomly wired networks do not have communities
  - **Maximal Modularity Hypothesis:** The partition with the maximum modularity offers the best community structure
- Several community detection algorithms available in literature
- Many of the developed algorithms are available as software packages

Summary of Community Detection Algorithms

Algorithm	Nature	Complexity	Ref.
Ravaz	Hierarchical Agglomerative	$O(N^2)$	[RSM$^+$02]
Girvan-Newman	Hierarchical Divisive	$O(N^2)$	[GN02]
Greedy Modularity	Modularity Optimization	$O(N^2)$	[New03]
Greedy Modularity	(Optimized) Modularity Optimization	$O(N \log^2 N)$	[CNM04]
Louvain	Modularity Optimization	$O(L)$	[BGLM08]
Label Propagation	Consensus Based	$O(L)$	[RAK07]
Clique percolation	Overlapping Community Detection	$O(e^N)$	[PDFV05]
Link Clustering	Hierarchical Agglomerative Overlapping CD	$O(\frac{N^2}{(\gamma-1))} + O(L^2)$	[ABL10]

References I

- Yong-Yeol Ahn, James P. Bagrow, and Sune Lehmann, *Link Communities reveal Multiscale Complexity in Networks*, Nature **466** (2010), no. 7307, 761–764.
- Lada A. Adamic and Natalie Glance, *The Political Blogosphere and the 2004 U.S. Election: Divided They Blog*, Proceedings of the 3rd International Workshop on Link Discovery (New York, NY, USA), ACM, 2005, pp. 36–43.
- Albert-László Barabási, *Network Science*, Cambridge University Press, New York, 2016.
- V.D. Blondel, J.L. Guillaume, R. Lambiotte, and E.L.J.S. Mech, *Fast Unfolding of Communities in Large Networks*, Journal of Statistical Mechanics: Theory and Experiment (2008), no. 10.
- *Hierarchical Structure and the Prediction of Missing Links in Networks*, Nature **453** (2008), 98–101.

References II

- Aaron Clauset, M. E. J. Newman, , and Cristopher Moore, *Finding Community Structure in Very Large Networks*, Physical Review E (2004), 1– 6.
- Jingyi Ding, Licheng Jiao, Jianshe Wu, and Fang Liu, *Prediction of missing links based on community relevance and ruler inference*, Knowledge-Based Systems **98** (2016), 200 – 215.
- Santo Fortunato and M Barthelemy, *Resolution Limit in Community Detection*, Proceedings of the National Academy of Sciences (2007).
- M. Girvan and M. E. J. Newman, *Community Structure in Social and Biological Networks*, Proceedings of the National Academy of Sciences **99** (2002), no. 12, 7821–7826.

References III

- Sharanjit Kaur, Rakhi Saxena, and Vasudha Bhatnagar, *Leveraging Hierarchy and Community Structure for Determining Influencers in Networks*, Big Data Analytics and Knowledge Discovery (DaWaK), vol. 10440, Springer, 2017, pp. 383–390.
- Yike Liu, Tara Safavi, Abhilash Dighe, and Danai Koutra, *Graph Summarization Methods and Applications: A Survey*, ACM Computing Surveys **51** (2018), no. 3, 62:1–62:34.
- M.E.J. Newman, *Fast algorithm for detecting community structure in networks*, Physical Review E **69** (2003).
- Gergely Palla, Albert-Laszlo Barabasi, and Tamas Vicsek, *Quantifying Social Group Evolution*, Nature **446** (2007), no. 7136, 664–667.

References IV

- Gergely Palla, Imre Derényi, Illés Farkas, and Tamás Vicsek, *Uncovering the Overlapping Community Structure of Complex Networks in Nature and Society*, Nature **435** (2005), no. 7043, 814–818.
- Usha Nandini Raghavan, Réka Albert, and Soundar Kumara, *Near Linear Time Algorithm to Detect Community Structures in Large-Scale Networks*, Physical Review E **76** (2007).
- Filippo Radicchi, Claudio Castellano, Federico Cecconi, Vittorio Loreto, and Domenico Parisi, *Defining and Identifying Communities in Networks*, Proceedings of the National Academy of Sciences **101** (2004), no. 9, 2658–2663.

References V

- P. Krishna Reddy, Masaru Kitsuregawa, P. Sreekanth, and S. Srinivasa Rao, *A Graph Based Approach to Extract a Neighborhood Customer Community for Collaborative Filtering*, International Workshop on Databases in Networked Information Systems, vol. 2544, Springer, 2002, pp. 188–200.
- E. Ravasz, A. L. Somera, D. A. Mongru, Z. N. Oltvai, and A.-L. Barabási, *Hierarchical Organization of Modularity in Metabolic Networks*, Science **297** (2002), no. 5586, 1551–1555.
- Rakhi Saxena, Sharanjit Kaur, and Vasudha Bhatnagar, *Social Centrality using Network Hierarchy and Community Structure*, Data Mining and Knowledge Discovery **32** (2018), no. 5, 1421–1443.
- W. W. Zachary, *An Information Flow Model for Conflict and Fission in Small Groups*, Journal of Anthropological Research **33** (1977), 452–473.

References VI

Xiaohang Zhang, Ji Zhu, Qi Wang, and Han Zhao, *Identifying Influential Nodes in Complex Networks with Community Structure*, Knowledge-Based Systems **42** (2013), 74 – 84.