Modelling and Analysis of Complex Networks

— Semester 3, Master of Data Science —

Jun Pang
University of Luxembourg

Motifs and Graphlets

Network Metrics

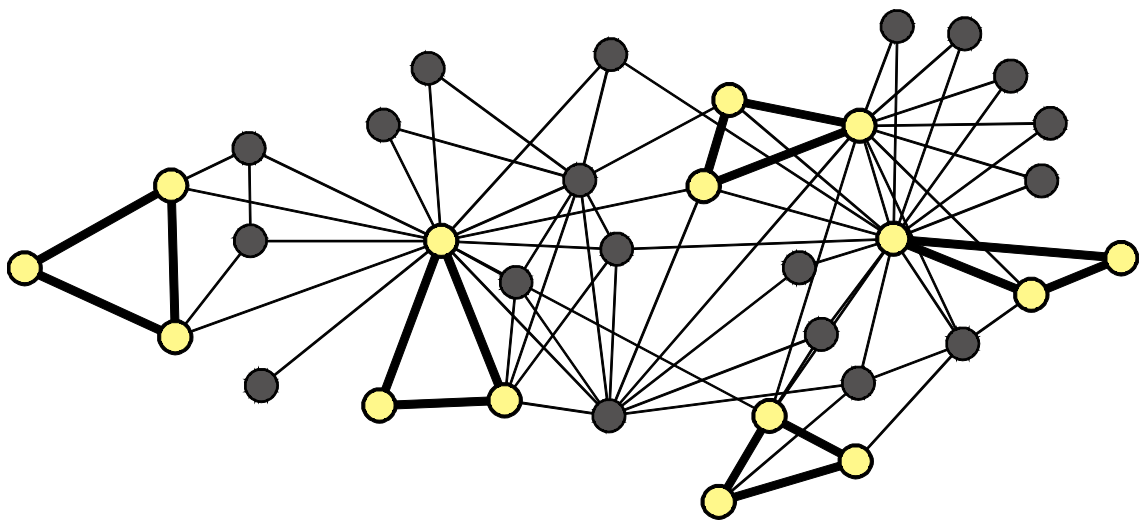
- Macro (network) level: diameter, clustering, size of giant component
- Micro (node) level: node degree, betweenness, PageRank score
- What about something in-between? A mesoscale characterisation of networks

Network Metrics

- Macro (network) level: diameter, clustering, size of giant component
- Micro (node) level: node degree, betweenness, PageRank score
- What about something in-between? A mesoscale characterisation of networks

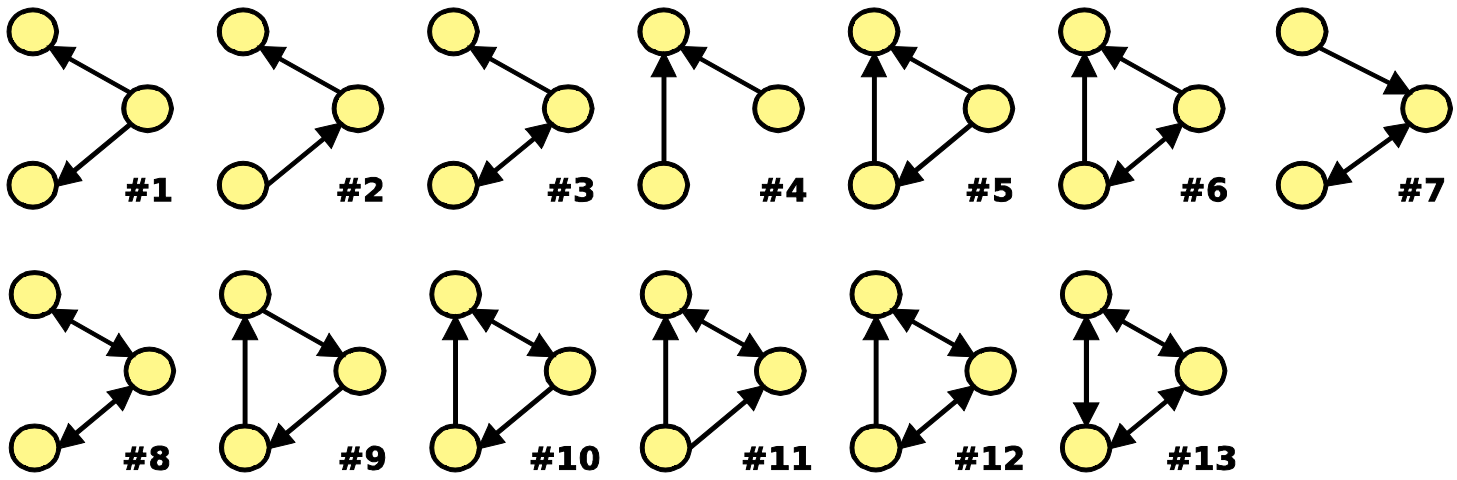
Subnetworks

- Subnetworks (subgraphs) are the building blocks of networks.
- They can characterise and discriminate networks.
- They can describe the network structure around a given node.



Subnetworks

Consider all possible (non-isomorphic) directed subgraphs of size 3.



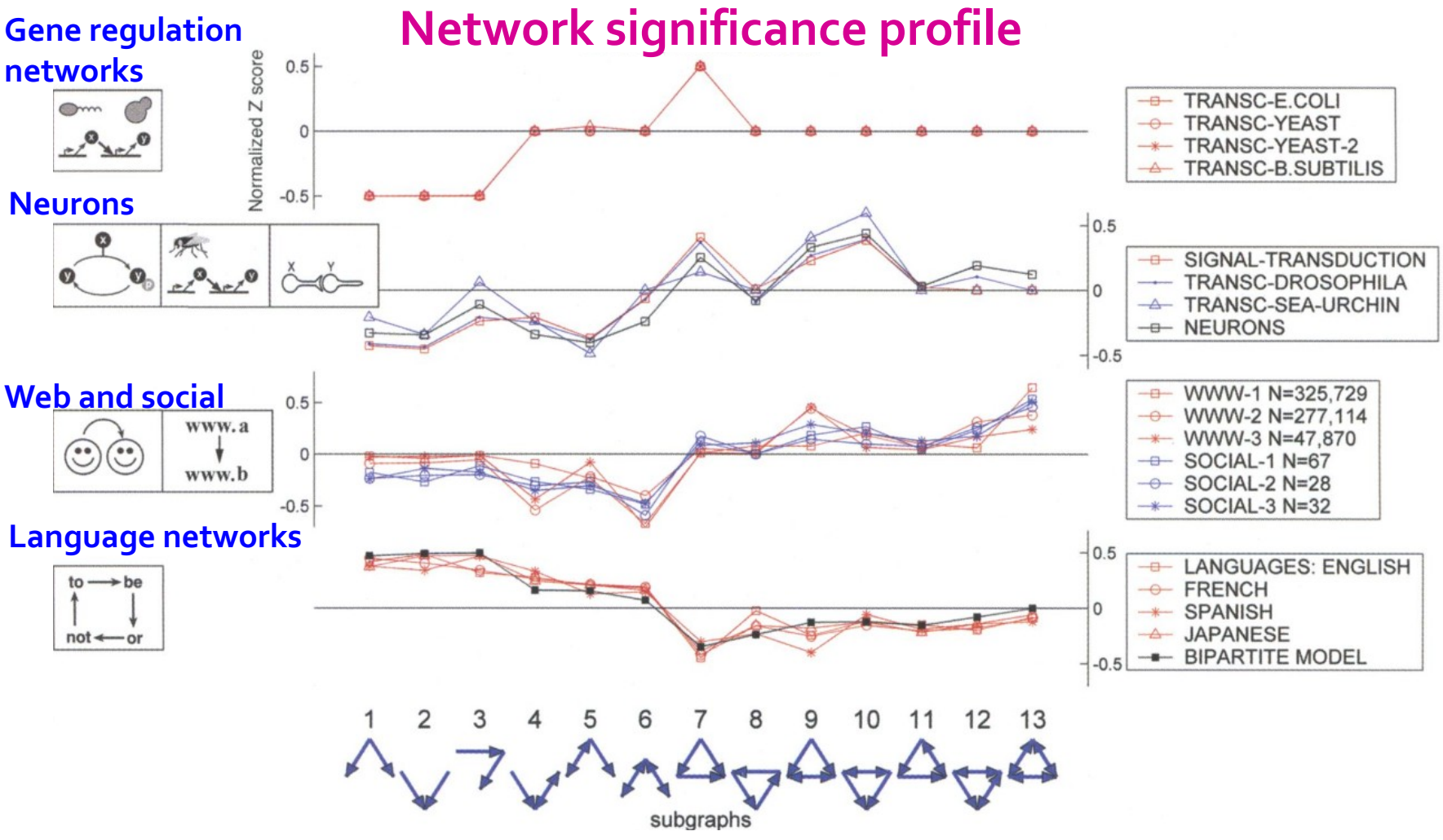
Decompose a network into subnetworks

Subnetworks

- For each subgraph, we can use a metric capable of classifying the subgraph's "significance"
  - Negative values indicate under-representation
  - Positive values indicate over-representation
- We create a network significance profile: a feature vector with values for all subgraph types
- We can compare profiles of networks from different domains.

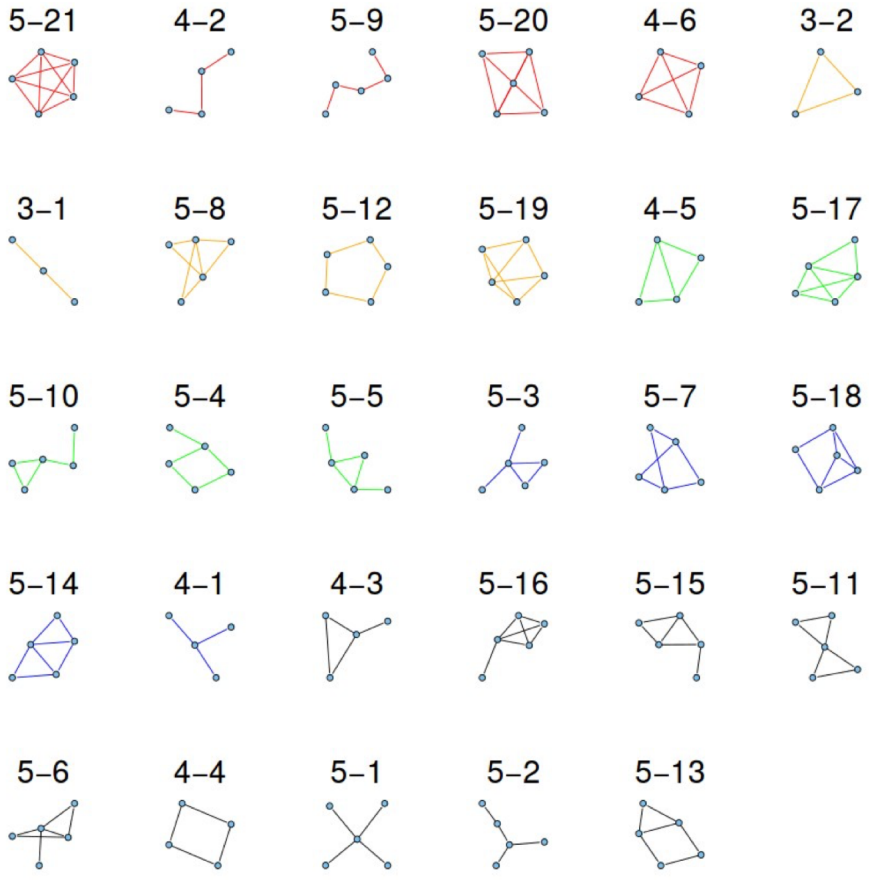
Subnetworks

Example: different type networks, networks from the same domain have similar profiles [Milo et al, 2004]

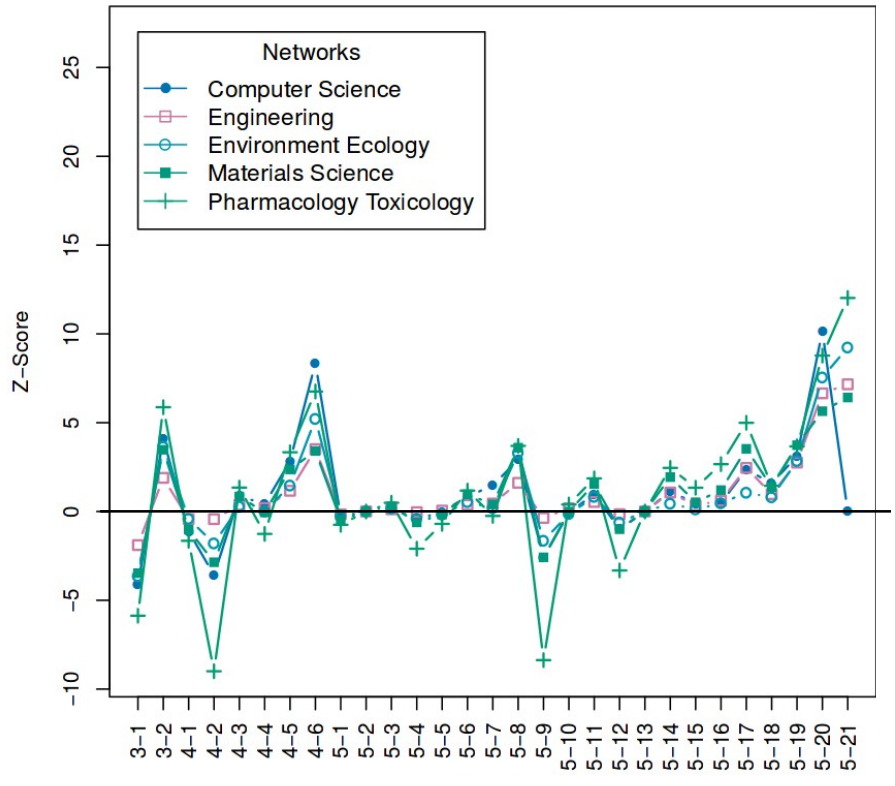


Subnetworks

Example: co-authorship network [Choobdar et al, 2021]



Subgraph types (corresponding to the X-axis of the plot)



Network significance profile

Motifs

Network Motifs

Network motifs: recurring, significant patterns of interconnections

- Pattern: small induced/non-induced subgraph
- Recurring: found many times, i.e., with high frequency
- Significant: more frequent than expected, i.e., in randomly generated networks

(Erdös-Rényi random graphs, scale-free networks, etc.)

Network Motifs

Network motifs: recurring, significant patterns of interconnections

- Pattern: small induced/non-induced subgraph
- Recurring: found many times, i.e., with high frequency
- Significant: more frequent than expected, i.e., in randomly generated networks

  (Erdös-Rényi random graphs, scale-free networks, etc.)

Network Motifs

Network motifs: recurring, significant patterns of interconnections

- Pattern: small induced/non-induced subgraph
- Recurring: found many times, i.e., with high frequency
- Significant: more frequent than expected, i.e., in randomly generated networks

  (Erdös-Rényi random graphs, scale-free networks, etc.)

Network Motifs

Network motifs: recurring, significant patterns of interconnections

- Pattern: small induced/non-induced subgraph
- Recurring: found many times, i.e., with high frequency
- Significant: more frequent than expected, i.e., in randomly generated networks

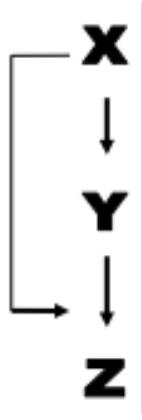
  (Erdös-Rényi random graphs, scale-free networks, etc.)

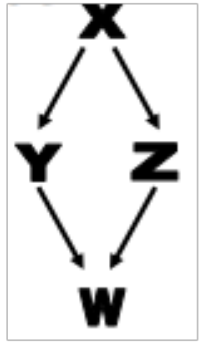
Network Motifs

- Help us understand how networks work
- Help us predict operation/reaction of the network in a given situation

Network Motifs

- Help us understand how networks work
- Help us predict operation/reaction of the network in a given situation

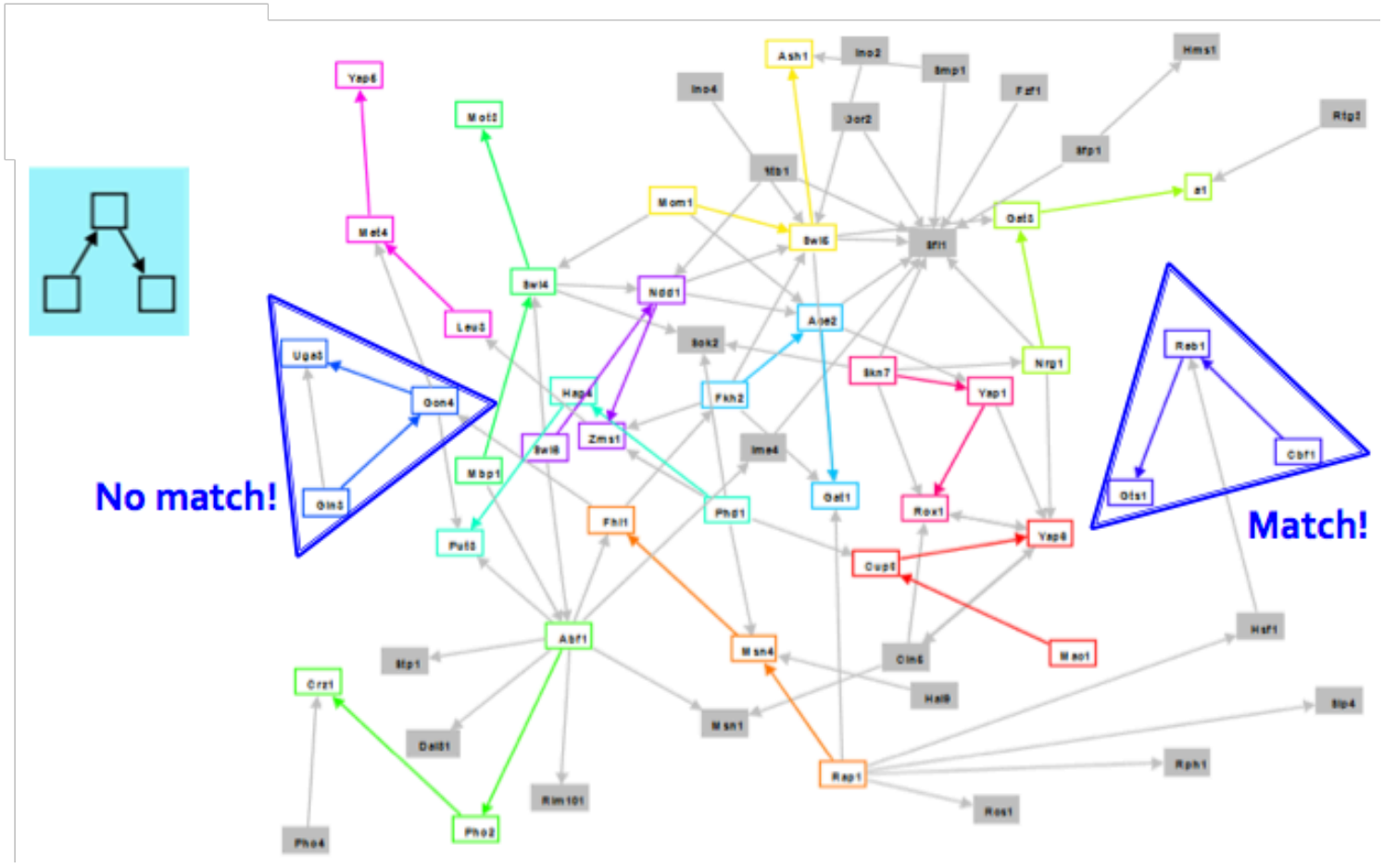




- Feed-forward loops: in the networks of neurons, where they neutralise “biological noise"
- Parallel loops: in food webs
- Single-input modules: in gene regulation networks

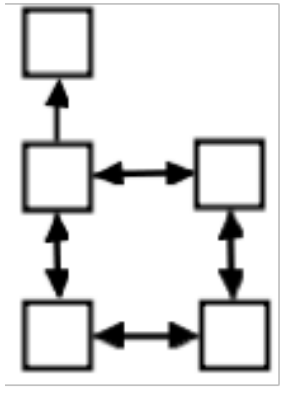
Network Motifs

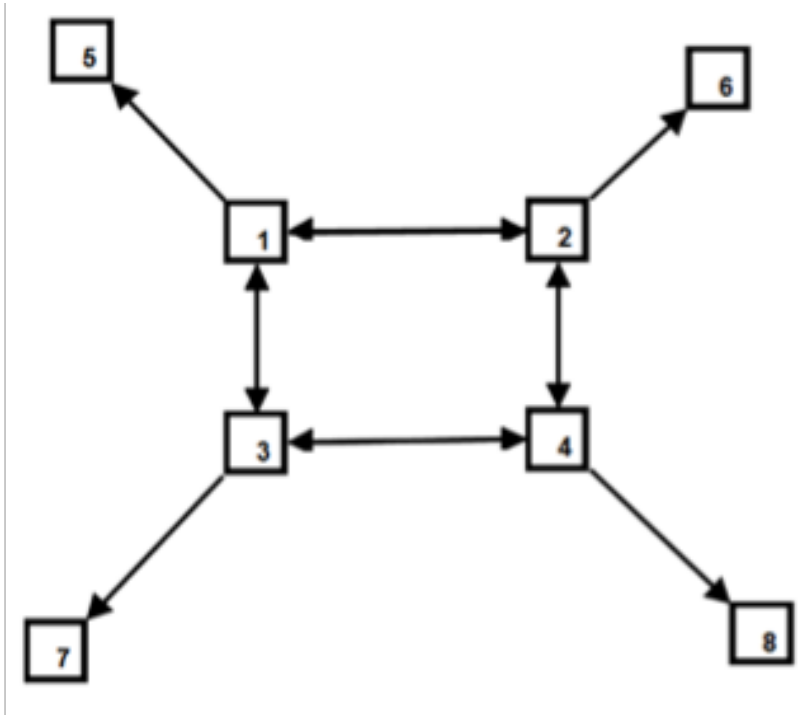
Induced subgraph (motif) of interest: formed from a subset of nodes of the graph and all of the edges connecting pairs of nodes in the subset



Network Motifs

Recurrence of subgraph (motif) of interest.

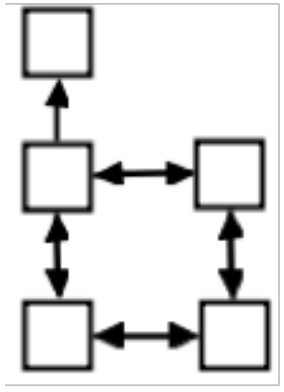


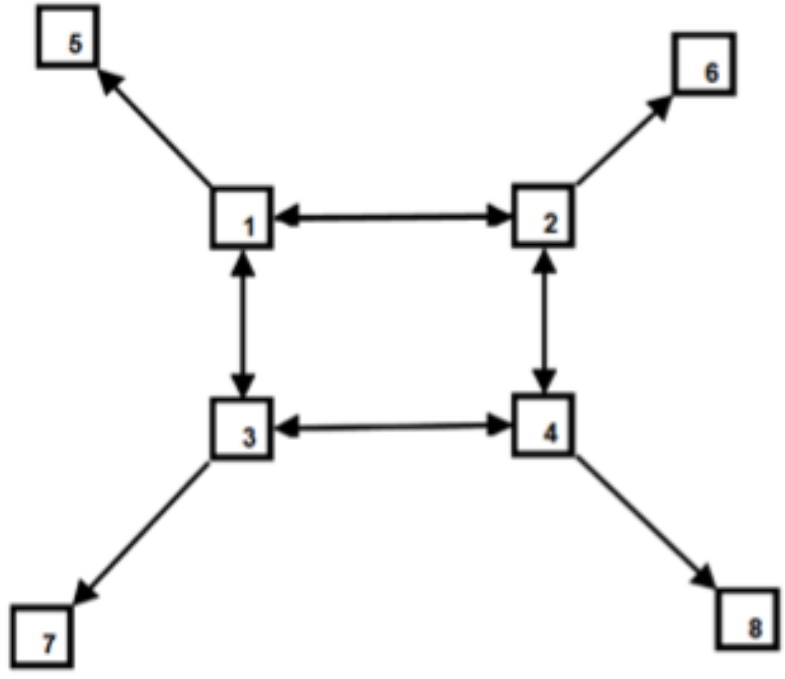


(Allow overlapping of motifs, there 4 occurrences of the motif:
{1,2,3,4,5}, {1,2,3,4,6}, {1,2,3,4,7}, {1,2,3,4,8})

Network Motifs

Recurrence of subgraph (motif) of interest.

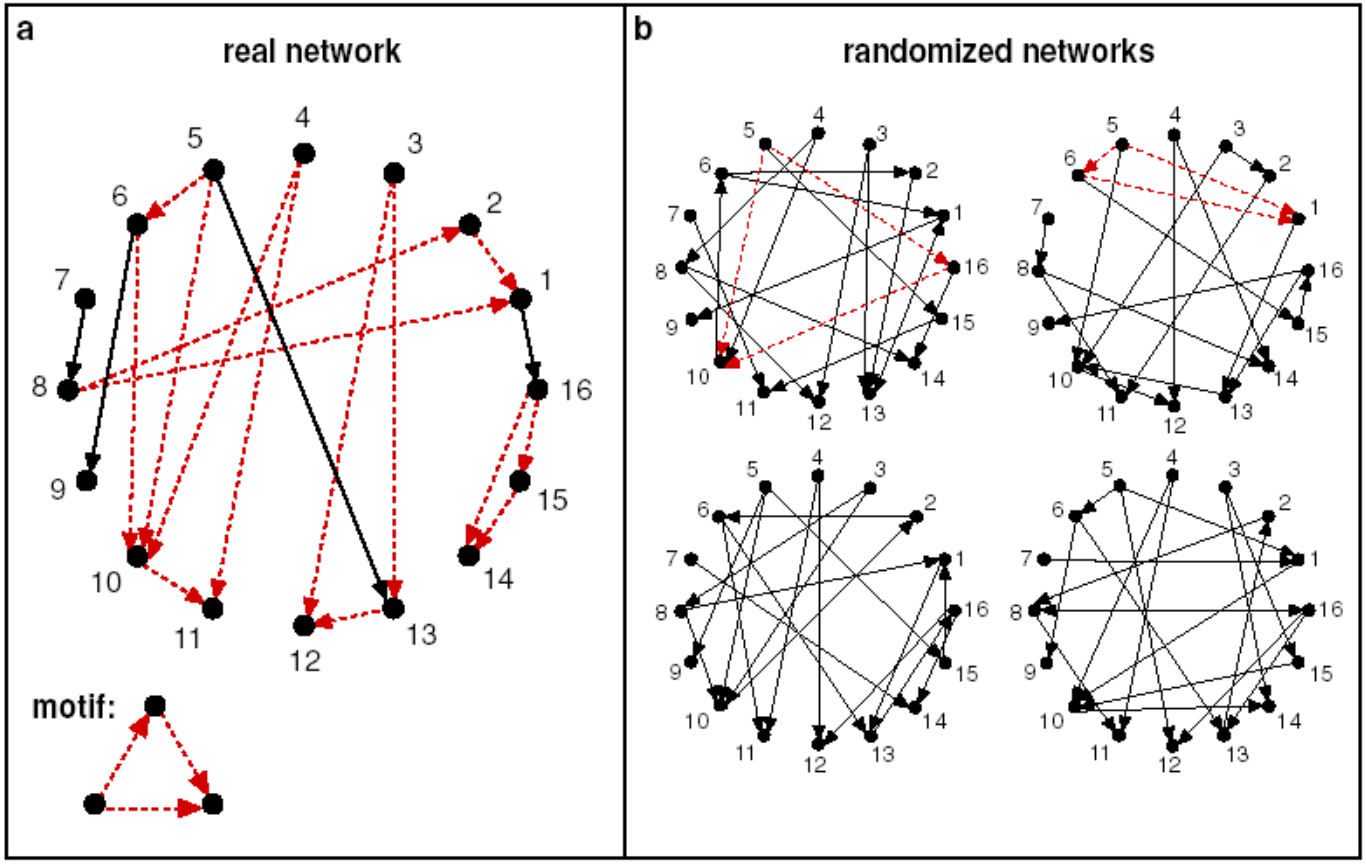




(Allow overlapping of motifs, there 4 occurrences of the motif:
{1,2,3,4,5}, {1,2,3,4,6}, {1,2,3,4,7}, {1,2,3,4,8})

Network Motifs

Significance of subgraph (motif) of interest.



Key idea: subgraphs that occur in a real network much more often than in a random network have functional significance [Milo et al, 2002].

Network Motifs

Significance of subgraph (motif) of interest.

- Motifs are over-represented in a network when compared to randomised networks:

$$Z_i = \frac{N_i^{real} - \overline{N_i}^{rand}}{std(N_i^{rand})}$$

N_i^{real} is the number of subgraphs of type i in network G^{real};
N_i^{rand} is the number in randomised network G^{rand}

- Network significance profile:

$$SP_i = \frac{Z_i}{\sqrt{\sum Z_j^2}}$$

SP is a vector of normalised Z-scores, and it emphasises the relative significance of subgraphs.

Network Motifs

Significance of subgraph (motif) of interest.

- Motifs are over-represented in a network when compared to randomised networks:

$$Z_i = \frac{N_i^{real} - \overline{N_i}^{rand}}{std(N_i^{rand})}$$

N_i^{real} is the number of subgraphs of type i in network G^{real};
N_i^{rand} is the number in randomised network G^{rand}

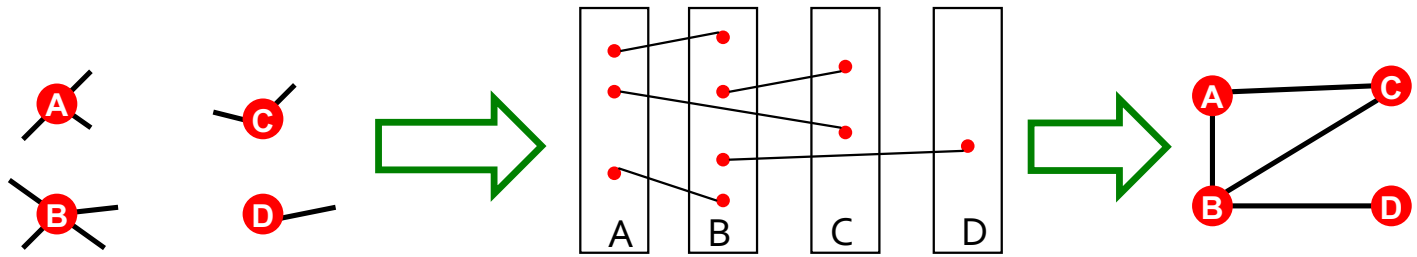
- Network significance profile:

$$SP_i = \frac{Z_i}{\sqrt{\sum Z_j^2}}$$

SP is a vector of normalised Z-scores, and it emphasises the relative significance of subgraphs.

Configuration Model

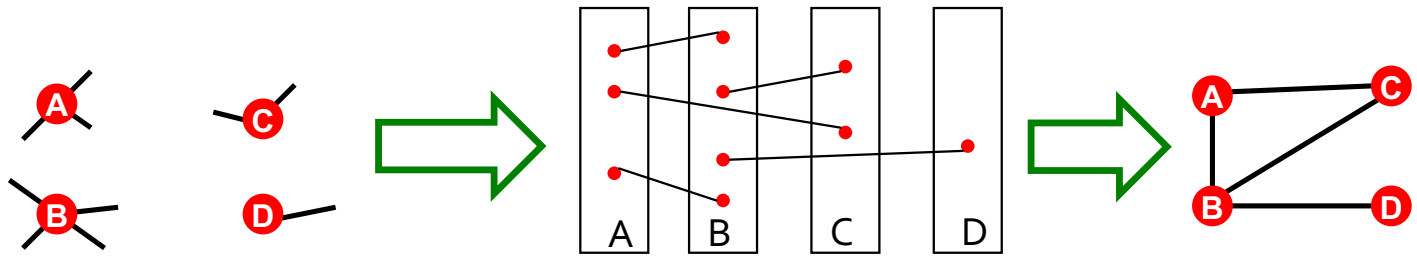
- Given a degree sequence $k_1, k_2, \ldots, k_N$
- Assign each node $i \in V$ with k_i number of stubs
- Select random pairs of unmatched stubs and connect them
- Repeat 3 while there are unmatched stubs



- A “null" model of networks: compare the real network G^{real} and a random network G^{rand} with the same degree sequence.

Configuration Model

- Given a degree sequence $k_1, k_2, \ldots, k_N$
- Assign each node $i \in V$ with k_i number of stubs
- Select random pairs of unmatched stubs and connect them
- Repeat 3 while there are unmatched stubs



- A “null" model of networks: compare the real network G^{real} and a random network G^{rand} with the same degree sequence.

Network Motifs

Detecting network motif:

- Count subgraphs i in G^{real}
- Count subgraphs i in random networks G^{rand} (each G^{rand} has the same numbers of nodes, edges, degree distribution as G^{real})
- Assign Z-score to subgraph type i:

$$Z_i = \frac{N_i^{real} - \overline{N_i}^{rand}}{std(N_i^{rand})}$$

- High Z-score: subgraph i is a network motif of G^{real}

Network Motifs

Detecting network motif:

- Count subgraphs i in G^{real}
- Count subgraphs i in random networks G^{rand} (each G^{rand} has the same numbers of nodes, edges, degree distribution as G^{real})
- Assign Z-score to subgraph type i:

$$Z_i = \frac{N_i^{real} - \overline{N_i}^{rand}}{std(N_i^{rand})}$$

- High Z-score: subgraph i is a network motif of G^{real}

Network Motifs

Detecting network motif:

- Count subgraphs i in G^{real}
- Count subgraphs i in random networks G^{rand} (each G^{rand} has the same numbers of nodes, edges, degree distribution as G^{real})
- Assign Z-score to subgraph type i:

$$Z_i = \frac{N_i^{real} - \overline{N_i}^{rand}}{std(N_i^{rand})}$$

- High Z-score: subgraph i is a network motif of G^{real}

Network Motifs

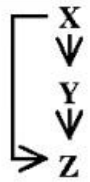
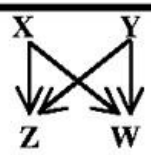
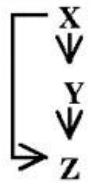
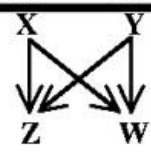
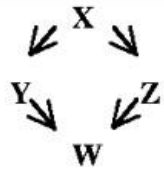
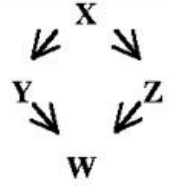
Detecting network motif:

- Count subgraphs i in G^{real}
- Count subgraphs i in random networks G^{rand} (each G^{rand} has the same numbers of nodes, edges, degree distribution as G^{real})
- Assign Z-score to subgraph type i:

$$Z_i = \frac{N_i^{real} - \overline{N_i}^{rand}}{std(N_i^{rand})}$$

- High Z-score: subgraph i is a network motif of G^{real}

Network Motifs

Network	Nodes	Edges	N_{real}	$N_{rand} \pm SD$	Z score	N_{real}	$N_{rand} \pm SD$	Z score	N_{real}	$N_{rand} \pm SD$	Z score
Gene regulation (transcription)					**Feed-forward loop**			**Bi-fan**			
E. coli	424	519	40	7 ± 3	10	203	47 ± 12	13			
*S. cerevisiae**	685	1,052	70	11 ± 4	14	1812	300 ± 40	41			
Neurons					**Feed-forward loop**			**Bi-fan**			**Bi-parallel**
C. elegans†	252	509	125	90 ± 10	3.7	127	55 ± 13	5.3	227	35 ± 10	20
Food webs					**Three chain**			**Bi-parallel**			
Little Rock	92	984	3219	3120 ± 50	2.1	7295	2220 ± 210	25			
Ythan	83	391	1182	1020 ± 20	7.2	1357	230 ± 50	23			
St. Martin	42	205	469	450 ± 10	NS	382	130 ± 20	12			
Chesapeake	31	67	80	82 ± 4	NS	26	5 ± 2	8			
Coachella	29	243	279	235 ± 12	3.6	181	80 ± 20	5			
Skipwith	25	189	184	150 ± 7	5.5	397	80 ± 25	13			
B. Brook	25	104	181	130 ± 7	7.4	267	30 ± 7	32			

Z-scores of individual motifs for different networks.

Network Motifs

- Network of neurons and a gene network contain similar motifs:
  - Feed-forward loops and bi-fan structures
- Food webs have parallel loops:
  - Prey of a particular predator share prey
- WWW network has bidirectional links:
  - Allowing the shortest path among sets of related pages

Network Motifs

- Network of neurons and a gene network contain similar motifs:
  - Feed-forward loops and bi-fan structures
- Food webs have parallel loops:
  - Prey of a particular predator share prey
- WWW network has bidirectional links:
  - Allowing the shortest path among sets of related pages

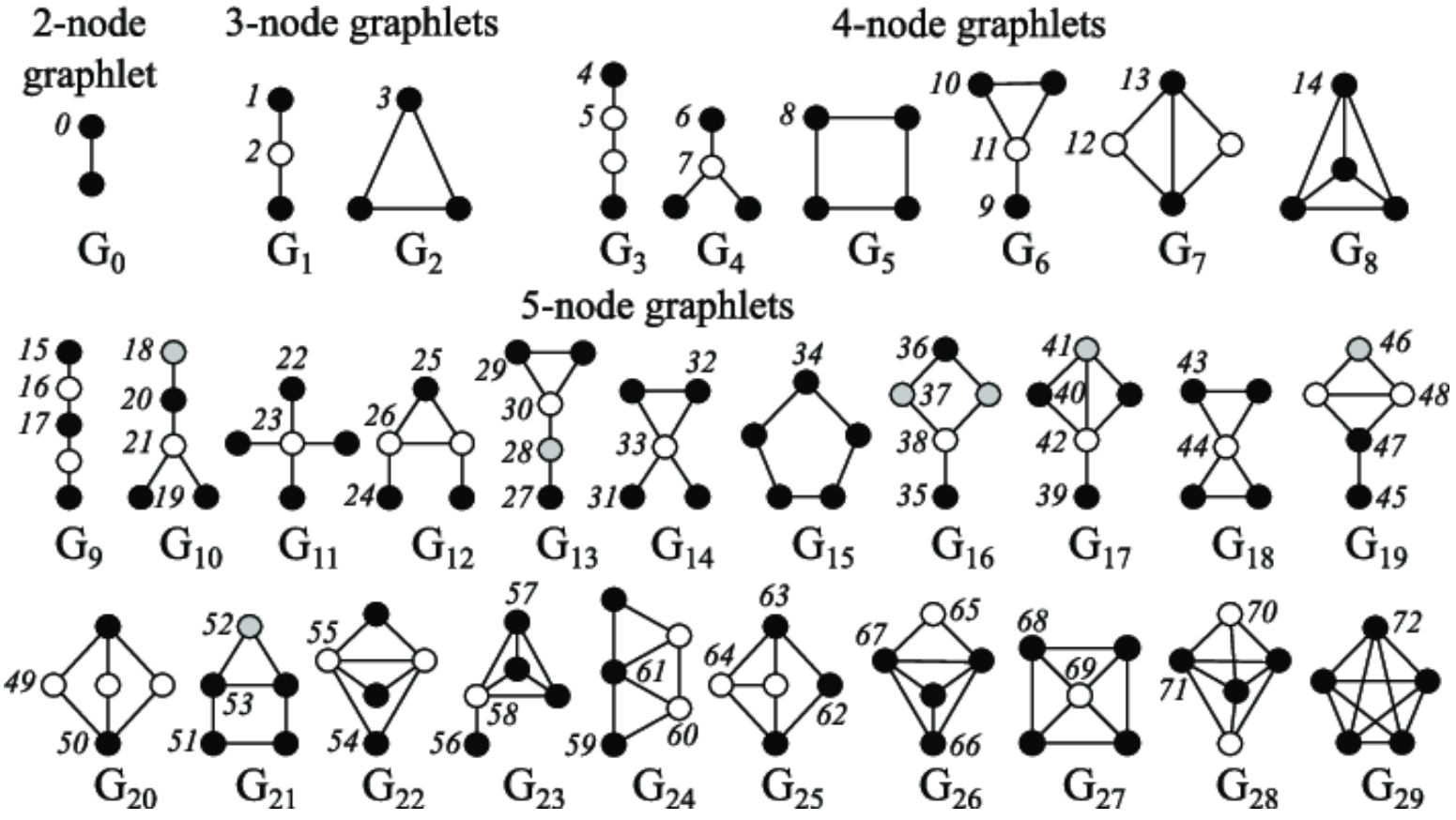
Network Motifs

- Network of neurons and a gene network contain similar motifs:
  - Feed-forward loops and bi-fan structures
- Food webs have parallel loops:
  - Prey of a particular predator share prey
- WWW network has bidirectional links:
  - Allowing the shortest path among sets of related pages

Graphlets

Graphlets

Graphlets: connected non-isomorphic subgraphs



For $n = 3, 4, 5, \ldots, 10$, there are $2, 6, 21, \ldots, 11716571$ graphlets.

Graphlet Degree Vector

Graphlets: connected non-isomorphic subgraphs

- Use graphlets to obtain node-level subgraph metric
- Degree counts #(edges) that a node touches
- Graphlet degree vector counts #(graphlets) that a node touches

Graphlet Degree Vector

Graphlets: connected non-isomorphic subgraphs

- Use graphlets to obtain node-level subgraph metric
- Degree counts #(edges) that a node touches
- Graphlet degree vector counts #(graphlets) that a node touches

Graphlet Degree Vector

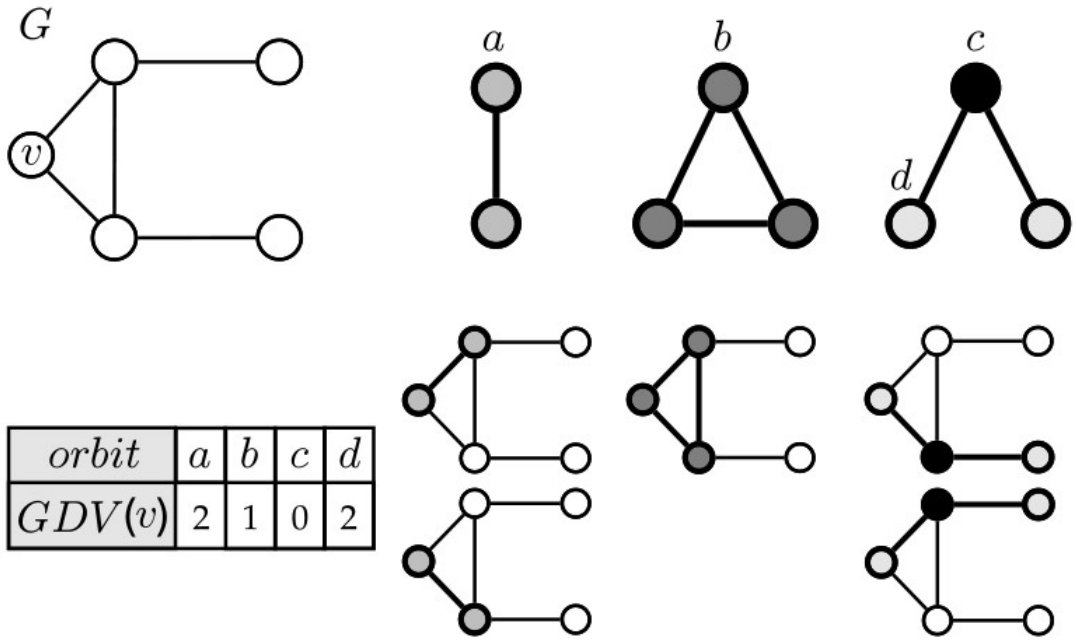
Graphlets: connected non-isomorphic subgraphs

- Use graphlets to obtain node-level subgraph metric
- Degree counts #(edges) that a node touches
- Graphlet degree vector counts #(graphlets) that a node touches

Graphlet Degree Vector

Graphlets: connected non-isomorphic subgraphs

- Use graphlets to obtain node-level subgraph metric
- Degree counts #(edges) that a node touches
- Graphlet degree vector counts #(graphlets) that a node touches



orbit	*a*	*b*	*c*	*d*
GDV(*v*)	2	1	0	2

Graphlet Degree Vector

- Graphlet degree vector provides a measure of a node's local network topology
- Finding size-k motifs/graphlets requires solving two challenges: 1) Enumerating all size-k connected subgraphs and 2) Counting #(occurrences of each subgraph type)
- Just knowing if a certain subgraph exists in a graph is a hard computational problem (NP-complete)!
- Computation time grows exponentially as the size of the motif/graphlet increases (feasible size is usually small 3-8)

Graphlet Degree Vector

- Graphlet degree vector provides a measure of a node's local network topology
- Finding size-k motifs/graphlets requires solving two challenges: 1) Enumerating all size-k connected subgraphs and 2) Counting #(occurrences of each subgraph type)
- Just knowing if a certain subgraph exists in a graph is a hard computational problem (NP-complete)!
- Computation time grows exponentially as the size of the motif/graphlet increases (feasible size is usually small 3-8)

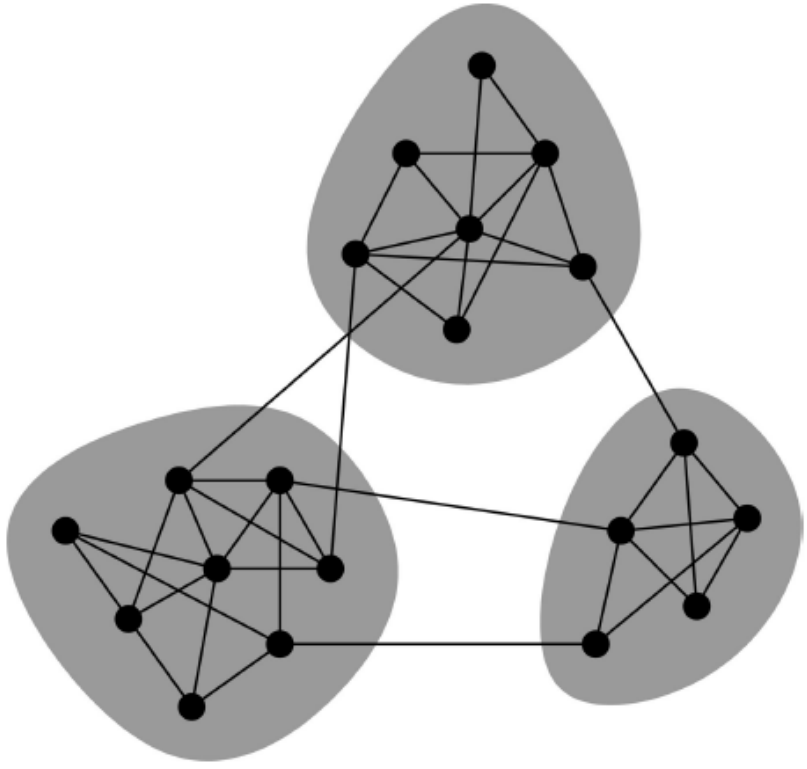
Graphlet Degree Vector

- Graphlet degree vector provides a measure of a node's local network topology
- Finding size-k motifs/graphlets requires solving two challenges: 1) Enumerating all size-k connected subgraphs and 2) Counting #(occurrences of each subgraph type)
- Just knowing if a certain subgraph exists in a graph is a hard computational problem (NP-complete)!
- Computation time grows exponentially as the size of the motif/graphlet increases (feasible size is usually small 3-8)

Community Structure in Networks

Networks & Communities

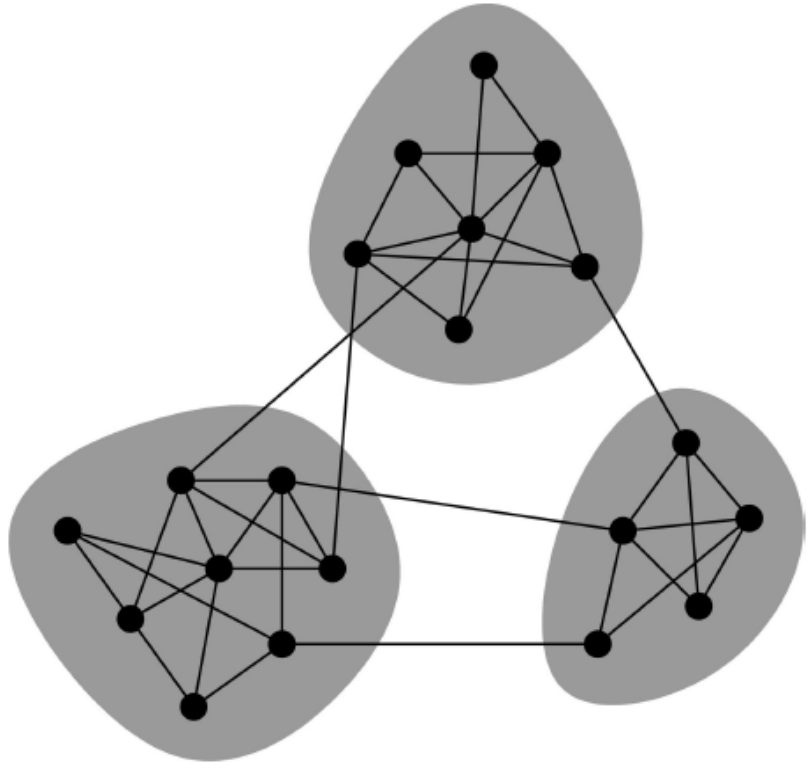
We often think of networks “look" like this:



But, what led to such a conceptual picture?

Networks & Communities

We often think of networks “look" like this:



But, what led to such a conceptual picture?

Conceptual Picture of Networks

- How does information flow through the network?
  - What structurally distinct roles do nodes play?
  - What roles do different links play?
- How do people find out about new jobs?
  - Mark Granovetter, part of his PhD in 1960s
  - People find the information through personal contacts.
- But, contacts were often acquaintances.
  - An acquaintance is someone you know casually but not as well as a friend.
  - This is surprising, one would expect your friends to help you out more than casual acquaintances.
- Why acquaintances are most helpful?

Conceptual Picture of Networks

- How does information flow through the network?
  - What structurally distinct roles do nodes play?
  - What roles do different links play?
- How do people find out about new jobs?
  - Mark Granovetter, part of his PhD in 1960s
  - People find the information through personal contacts.
- But, contacts were often acquaintances.
  - An acquaintance is someone you know casually but not as well as a friend.
  - This is surprising, one would expect your friends to help you out more than casual acquaintances.
- Why acquaintances are most helpful?

Conceptual Picture of Networks

- How does information flow through the network?
  - What structurally distinct roles do nodes play?
  - What roles do different links play?
- How do people find out about new jobs?
  - Mark Granovetter, part of his PhD in 1960s
  - People find the information through personal contacts.
- But, contacts were often acquaintances.
  - An acquaintance is someone you know casually but not as well as a friend.
  - This is surprising, one would expect your friends to help you out more than casual acquaintances.
- Why acquaintances are most helpful?

Conceptual Picture of Networks

- How does information flow through the network?
  - What structurally distinct roles do nodes play?
  - What roles do different links play?
- How do people find out about new jobs?
  - Mark Granovetter, part of his PhD in 1960s
  - People find the information through personal contacts.
- But, contacts were often acquaintances.
  - An acquaintance is someone you know casually but not as well as a friend.
  - This is surprising, one would expect your friends to help you out more than casual acquaintances.
- Why acquaintances are most helpful?

Conceptual Picture of Networks

- How does information flow through the network?
  - What structurally distinct roles do nodes play?
  - What roles do different links play?
- How do people find out about new jobs?
  - Mark Granovetter, part of his PhD in 1960s
  - People find the information through personal contacts.
- But, contacts were often acquaintances.
  - An acquaintance is someone you know casually but not as well as a friend.
  - This is surprising, one would expect your friends to help you out more than casual acquaintances.
- Why acquaintances are most helpful?

Conceptual Picture of Networks

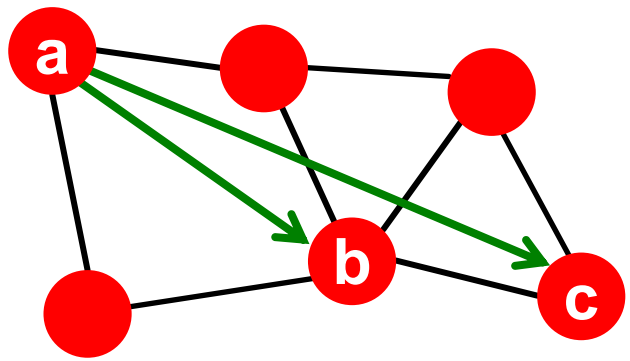
Granovetter's answer:

- Two perspectives on friendships:
  - Structural: Friendships span different parts of the network
  - Interpersonal: Friendship is either strong or weak

Conceptual Picture of Networks

Granovetter's answer:

- Two perspectives on friendships:
  - Structural: Friendships span different parts of the network
  - Interpersonal: Friendship is either strong or weak
- Structural role (triadic closure = high clustering coefficient):



Which edge is more likely?

Conceptual Picture of Networks

Granovetter's explanation: the strength of weak ties

- Structure:
  - Structurally embedded edges are also socially strong
  - Long-range edges spanning different parts of the network are socially weak
- Information
  - Long-range edges allow you to gather information from different parts of the network and get a job
  - Structurally embedded edges are heavily redundant in terms of information access

Conceptual Picture of Networks

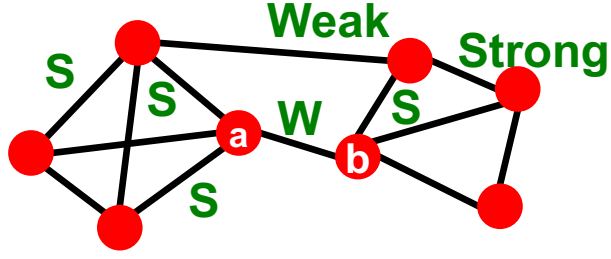
Granovetter's explanation: the strength of weak ties

- Structure:
  - Structurally embedded edges are also socially strong
  - Long-range edges spanning different parts of the network are socially weak
- Information
  - Long-range edges allow you to gather information from different parts of the network and get a job
  - Structurally embedded edges are heavily redundant in terms of information access

Conceptual Picture of Networks

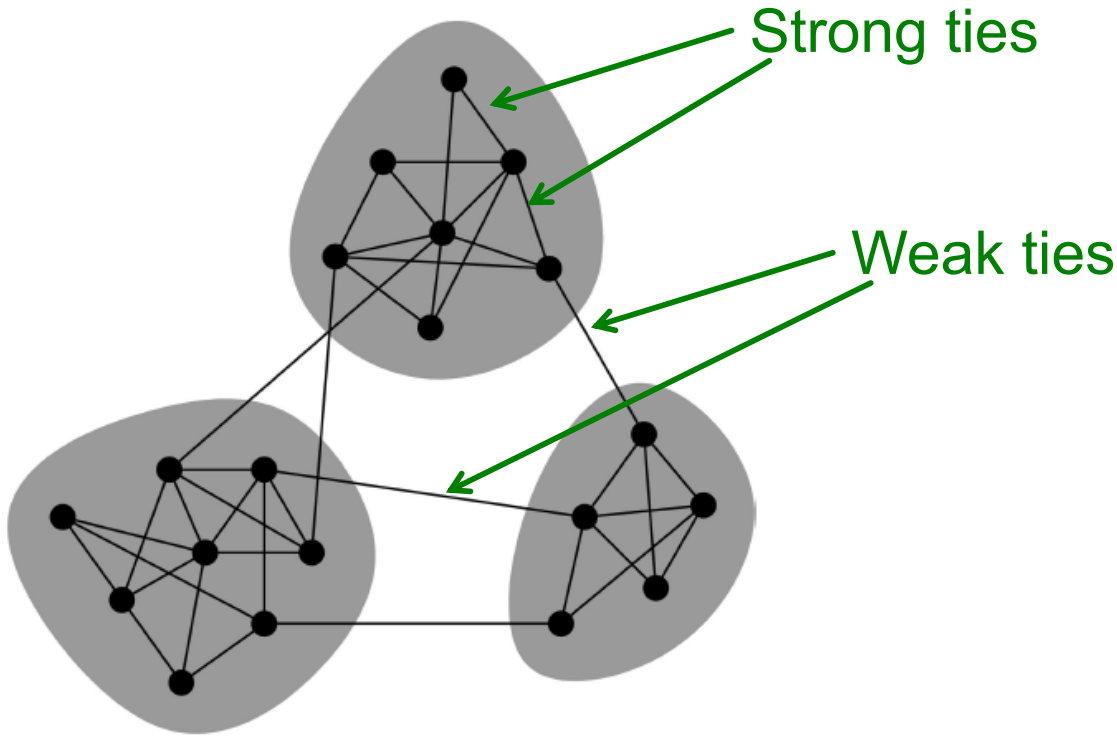
Granovetter's explanation: the strength of weak ties

- Structure:
  - Structurally embedded edges are also socially strong
  - Long-range edges spanning different parts of the network are socially weak
- Information
  - Long-range edges allow you to gather information from different parts of the network and get a job
  - Structurally embedded edges are heavily redundant in terms of information access



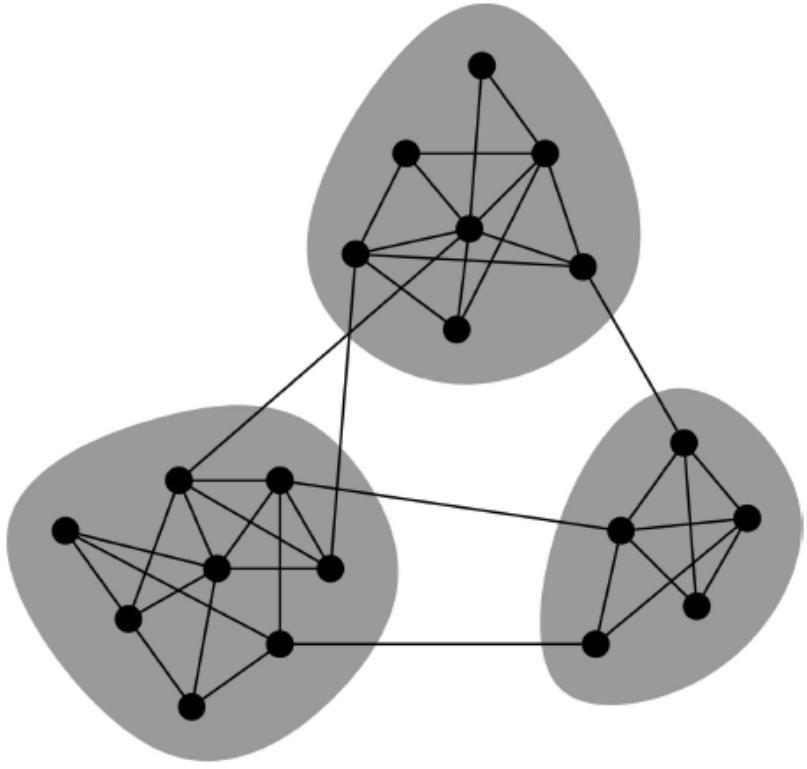
Conceptual Picture of Networks

Granovetter's theory leads to the conceptual picture of networks.



Network Communities

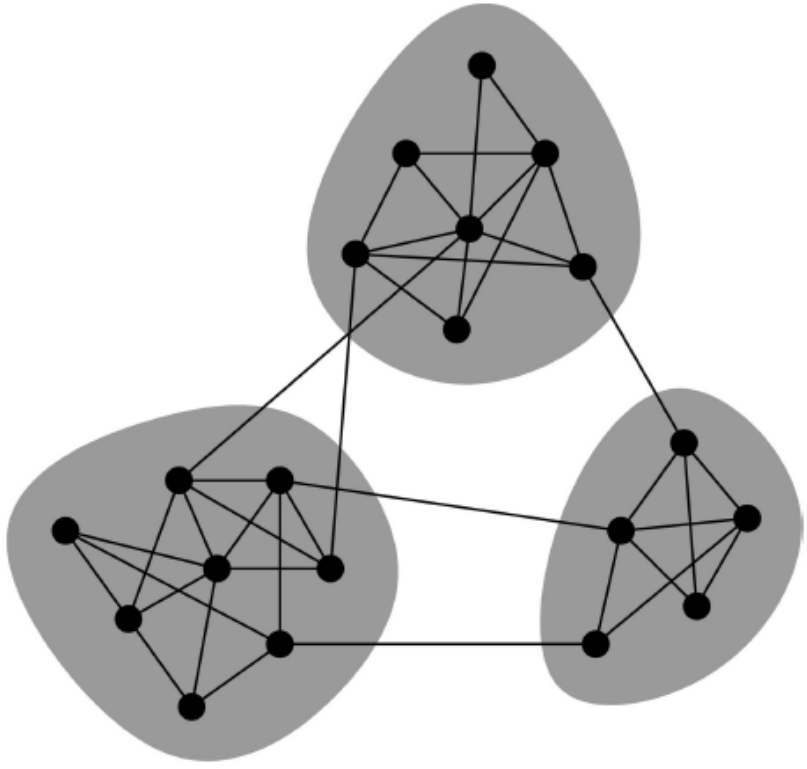
Granovetter's theory suggests that networks are composed of tightly connected sets of nodes



Network communities (clusters, modules): sets of nodes with lots of internal connections and few external ones.

Network Communities

Granovetter's theory suggests that networks are composed of tightly connected sets of nodes



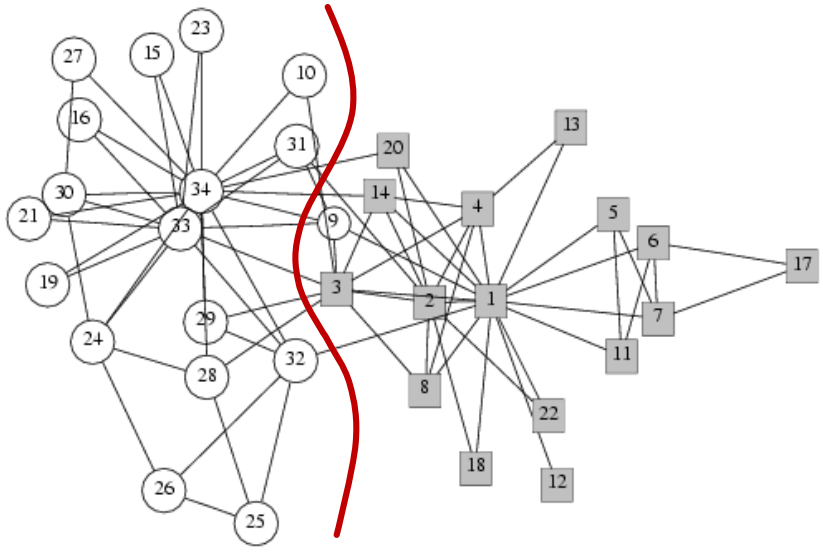
Network communities (clusters, modules): sets of nodes with lots of internal connections and few external ones.

Community Detection

- How to automatically find densely connected groups of nodes?
- Ideally such automatically detected clusters would then correspond to real groups.

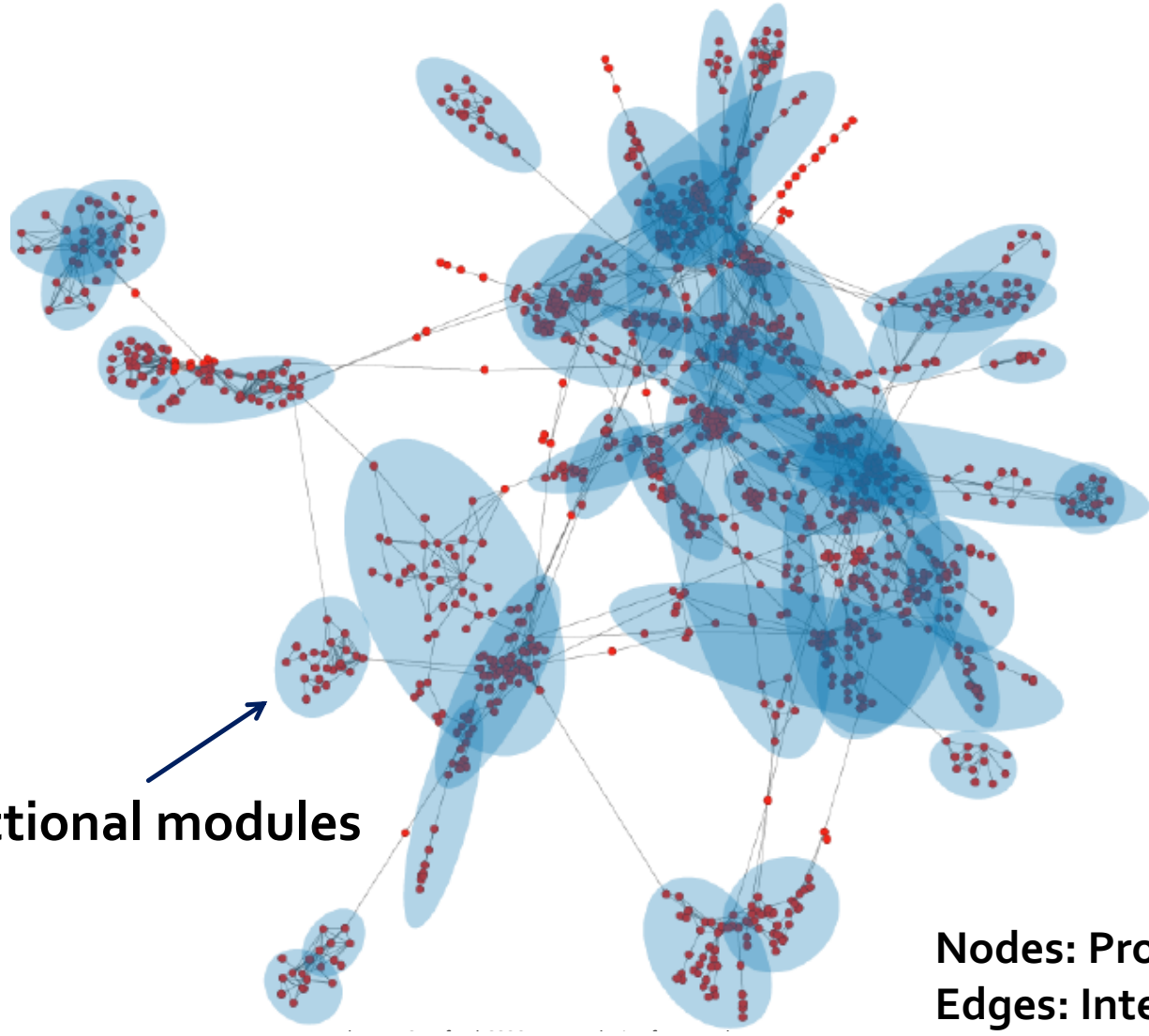
Community Detection

- How to automatically find densely connected groups of nodes?
- Ideally such automatically detected clusters would then correspond to real groups.
- Zachary's Karate club network:

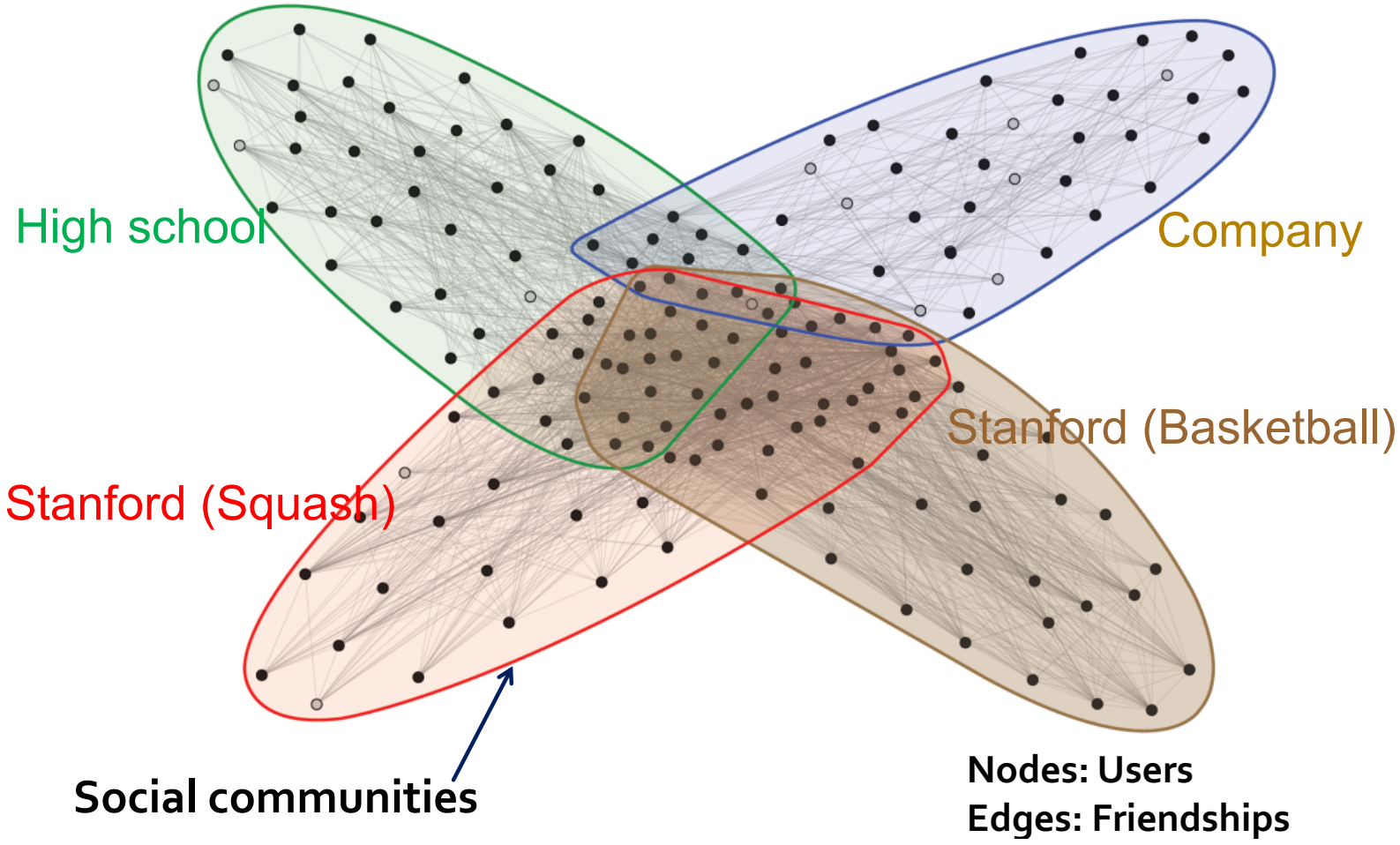


  - Observe social ties and rivalries in a university karate club. During his observation, conflicts led the group to split.
  - Split could be explained by a minimum cut in the network.

Community Detection



Community Detection



Community Detection

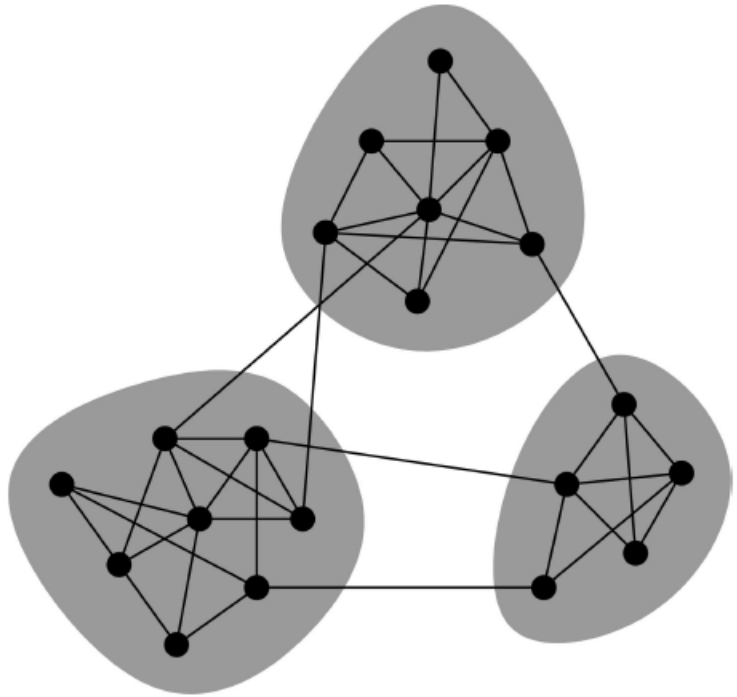
- Community detection is equivalent to “clustering" in unstructured data
- Clustering: unsupervised machine learning
  - Find groups of elements that are similar to each other
  - Hundreds of methods published since 1950 (k-means)
- Problem: what does “similar to each other" means?

Community Detection

- Community detection is equivalent to “clustering" in unstructured data
- Clustering: unsupervised machine learning
  - Find groups of elements that are similar to each other
  - Hundreds of methods published since 1950 (k-means)
- Problem: what does “similar to each other" means?

Community Detection

- Find groups of nodes that are strongly connected to each other, but weakly connected to the rest of the network.
- No formal definition
- Hundreds of methods published since 2003



Community Detection Algorithms

Clustering (community detection) algorithms are either:

- Hierarchical:
  - Agglomerative: begin with singleton groups and join successively by similarity (e.g., Newman, Louvain)
  - Divisive: begin with one group containing all points and divide successively (e.g., Girvan-Newman)
- Partitional: separate points in arbitrary number of groups, exchange according to similarity (e.g., graph partition).

Community Detection Algorithms

Clustering (community detection) algorithms are either:

- Hierarchical:
  - Agglomerative: begin with singleton groups and join successively by similarity (e.g., Newman, Louvain)
  - Divisive: begin with one group containing all points and divide successively (e.g., Girvan-Newman)
- Partitional: separate points in arbitrary number of groups, exchange according to similarity (e.g., graph partition).

Community Detection Algorithms

Agglomerative hierarchical clustering [Newman, 2010]: based on the similarity measure between nodes, sets of nodes

1. Assign each node to its own cluster
2. Find the cluster pair with the highest similarity and join them together into a cluster
3. Compute new similarities between new joined cluster and others
4. Go to step 2 until all nodes form a single cluster
5. Select clustering (cut the tree at desired level)

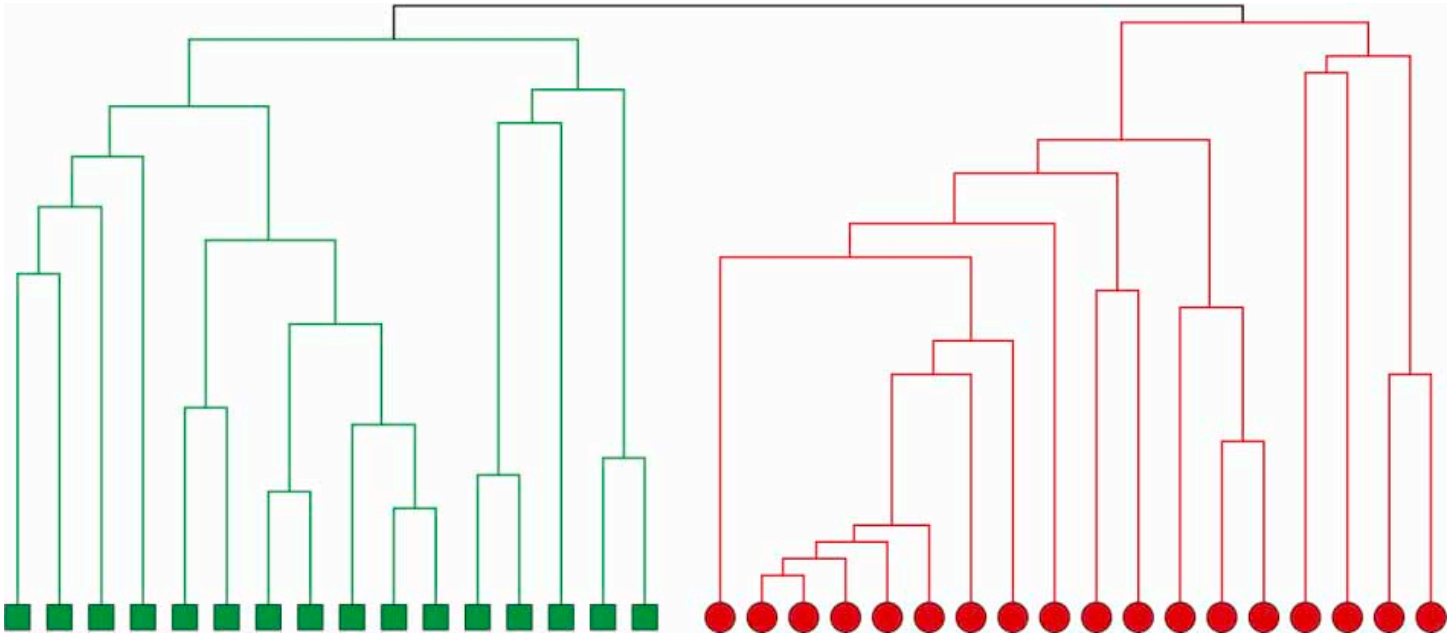
Community Detection Algorithms

Agglomerative hierarchical clustering [Newman, 2010]: based on the similarity measure between nodes, sets of nodes

1. Assign each node to its own cluster
2. Find the cluster pair with the highest similarity and join them together into a cluster
3. Compute new similarities between new joined cluster and others
4. Go to step 2 until all nodes form a single cluster
5. Select clustering (cut the tree at desired level)

Community Detection Algorithms

Agglomerative hierarchical clustering on Zachary's network, using average linkage as similarity measure

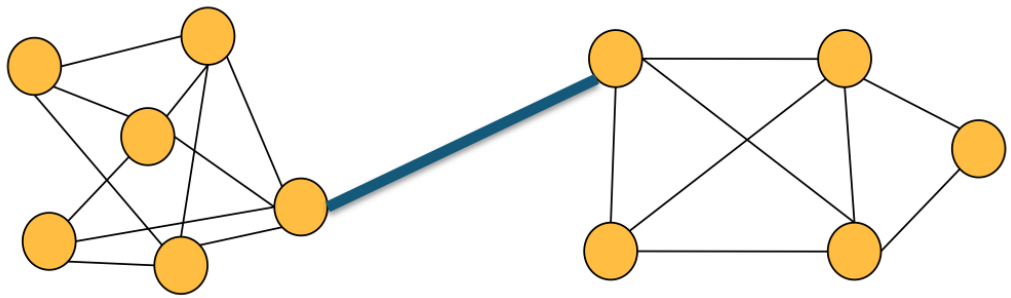


The result is a dendogram.

Community Detection Algorithms

Divisive hierarchical clustering [Girvan & Newman, 2002]:

- Edge betweenness: The betweenness of an edge is the number of shortest-paths in the network that pass through that edge.
- “Bridges" between communities have high edge betweenness.



Community Detection Algorithms

Divisive hierarchical clustering [Girvan & Newman, 2002]:

1. Compute betweenness for all edges in the network
2. Remove the edge with highest betweenness
3. Go to step 1 until no edges left

Community Detection Algorithms

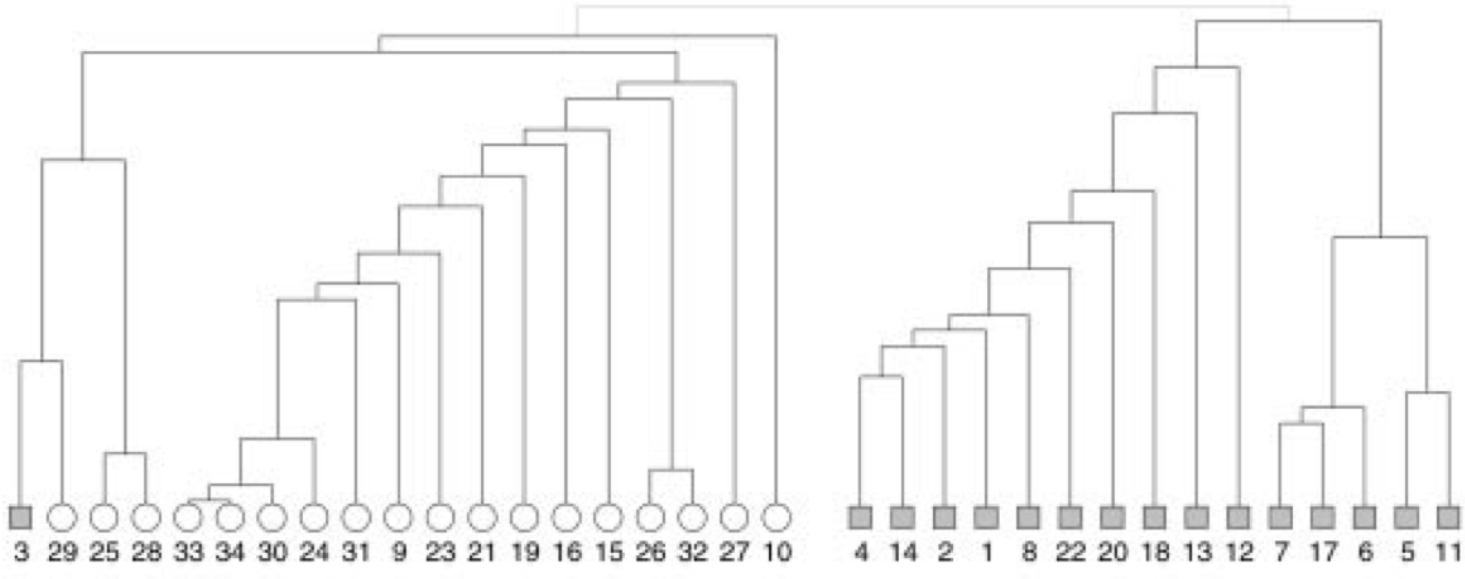
Divisive hierarchical clustering [Girvan & Newman, 2002]:

1. Compute betweenness for all edges in the network
2. Remove the edge with highest betweenness
3. Go to step 1 until no edges left

Community Detection Algorithms

Divisive hierarchical clustering [Girvan & Newman, 2002]:

1. Compute betweenness for all edges in the network
2. Remove the edge with highest betweenness
3. Go to step 1 until no edges left



Result is a dendogram

Community Detection Algorithms

Modularity:

- Introduced by Girvan & Newman, 2002
- The modularity is computed for a partition of a graph (each node belongs to one and only one community)
- It compares the observed fraction of edges inside communities vs. the expected fraction of edges inside communities in a random network

Community Detection Algorithms

Modularity: random graphs are not expected to have community structures, so we will use them as null models.

$$Q = \sum_{c \in C} [(\#\text{edges within group } c) - (\text{expected } \#\text{edges within group } c)]$$

In particular,

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

where P_{ij} is the expected number of edges between nodes i and j under the null model, c_i (c_j) is the community of node i (j), and $\delta(c_i, c_i) = 1$ if $c_i = c_j$ and 0 otherwise.

Community Detection Algorithms

Modularity: random graphs are not expected to have community structures, so we will use them as null models.

$$Q = \sum_{c \in C} [(\#\text{edges within group } c) - (\text{expected } \#\text{edges within group } c)]$$

In particular,

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

where P_{ij} is the expected number of edges between nodes i and j under the null model, c_i (c_j) is the community of node i (j), and $\delta(c_i, c_i) = 1$ if $c_i = c_j$ and 0 otherwise.

Community Detection Algorithms

How to compute P_{ij}?

The "configuration" random graph model chooses a graph with the same degree distribution as the original graph with n nodes and m edges uniformly at random.

- There are $2m$ stubs available in the configuration mode
- Let p_i be the probability of picking at random a stub incident with node i (with degree k_i)

$$p_i = \frac{k_i}{2m}$$

- The probability of connecting node i to node j is then

$$p_i p_j = \frac{k_i}{2m}\frac{k_j}{2m} = \frac{k_i k_j}{4m^2}$$

- Thus, the probability of an edge existing between i and j

$$P_{ij} = 2m \times p_i p_j = \frac{k_i k_j}{2m}$$

Community Detection Algorithms

How to compute P_{ij}? The “configuration" random graph model chooses a graph with the same degree distribution as the original graph with n nodes and m edges uniformly at random.

- There are $2m$ stubs available in the configuration mode
- Let p_i be the probability of picking at random a stub incident with node i (with degree k_i)

$$p_i = \frac{k_i}{2m}$$

- The probability of connecting node i to node j is then

$$p_i p_j = \frac{k_i}{2m}\frac{k_j}{2m} = \frac{k_i k_j}{4m^2}$$

- Thus, the probability of an edge existing between i and j

$$P_{ij} = 2m \times p_i p_j = \frac{k_i k_j}{2m}$$

Community Detection Algorithms

How to compute P_{ij}? The "configuration" random graph model chooses a graph with the same degree distribution as the original graph with n nodes and m edges uniformly at random.

- There are $2m$ stubs available in the configuration mode
- Let p_i be the probability of picking at random a stub incident with node i (with degree k_i)

$$p_i = \frac{k_i}{2m}$$

- The probability of connecting node i to node j is then

$$p_i p_j = \frac{k_i}{2m}\frac{k_j}{2m} = \frac{k_i k_j}{4m^2}$$

- Thus, the probability of an edge existing between i and j

$$P_{ij} = 2m \times p_i p_j = \frac{k_i k_j}{2m}$$

Community Detection Algorithms

How to compute P_{ij}? The "configuration" random graph model chooses a graph with the same degree distribution as the original graph with n nodes and m edges uniformly at random.

- There are $2m$ stubs available in the configuration mode
- Let p_i be the probability of picking at random a stub incident with node i (with degree k_i)

$$p_i = \frac{k_i}{2m}$$

- The probability of connecting node i to node j is then

$$p_i p_j = \frac{k_i}{2m}\frac{k_j}{2m} = \frac{k_i k_j}{4m^2}$$

- Thus, the probability of an edge existing between i and j

$$P_{ij} = 2m \times p_i p_j = \frac{k_i k_j}{2m}$$

Community Detection Algorithms

How to compute P_{ij}? The "configuration" random graph model chooses a graph with the same degree distribution as the original graph with n nodes and m edges uniformly at random.

- There are $2m$ stubs available in the configuration mode
- Let p_i be the probability of picking at random a stub incident with node i (with degree k_i)

$$p_i = \frac{k_i}{2m}$$

- The probability of connecting node i to node j is then

$$p_i p_j = \frac{k_i}{2m}\frac{k_j}{2m} = \frac{k_i k_j}{4m^2}$$

- Thus, the probability of an edge existing between i and j

$$P_{ij} = 2m \times p_i p_j = \frac{k_i k_j}{2m}$$

Community Detection Algorithms

How to compute P_{ij}? The "configuration" random graph model chooses a graph with the same degree distribution as the original graph with n nodes and m edges uniformly at random.

- There are $2m$ stubs available in the configuration mode
- Let p_i be the probability of picking at random a stub incident with node i (with degree k_i)

$$p_i = \frac{k_i}{2m}$$

- The probability of connecting node i to node j is then

$$p_i p_j = \frac{k_i}{2m}\frac{k_j}{2m} = \frac{k_i k_j}{4m^2}$$

- Thus, the probability of an edge existing between i and j

$$P_{ij} = 2m \times p_i p_j = \frac{k_i k_j}{2m}$$

Community Detection Algorithms

Properties of modularity:

$$Q = \frac{1}{2m} \sum_{ij} (A_{ij} - \frac{k_i k_j}{2m}) \delta(c_i, c_j)$$

- Q depends on nodes in the same clusters only
- Large modularity means better communities (better than random intra-cluster density)
- Modularity values take range $[-1, 1]$
  - It is positive if the number of edges within groups exceeds the expected number
  - $Q \geq 0.3 - 0.7$ means significant community structure

Community Detection Algorithms

Properties of modularity:

$$Q = \frac{1}{2m} \sum_{ij} (A_{ij} - \frac{k_i k_j}{2m}) \delta(c_i, c_j)$$

- Q depends on nodes in the same clusters only
- Large modularity means better communities (better than random intra-cluster density)
- Modularity values take range $[-1, 1]$
  - It is positive if the number of edges within groups exceeds the expected number
  - $Q \geq 0.3 - 0.7$ means significant community structure

Community Detection Algorithms

Properties of modularity:

$$Q = \frac{1}{2m} \sum_{ij} (A_{ij} - \frac{k_i k_j}{2m}) \delta(c_i, c_j)$$

- Q depends on nodes in the same clusters only
- Large modularity means better communities (better than random intra-cluster density)
- Modularity values take range $[-1, 1]$
  - It is positive if the number of edges within groups exceeds the expected number
  - $Q \geq 0.3 - 0.7$ means significant community structure

Community Detection Algorithms

Properties of modularity:

$$Q = \frac{1}{2m} \sum_{ij} (A_{ij} - \frac{k_i k_j}{2m}) \delta(c_i, c_j)$$

- Q depends on nodes in the same clusters only
- Large modularity means better communities (better than random intra-cluster density)
- Modularity values take range $[-1, 1]$
  - It is positive if the number of edges within groups exceeds the expected number
  - $Q \geq 0.3 - 0.7$ means significant community structure

Community Detection Algorithms

Modularity can be naturally extended to weighted/multi-edge networks.

$$Q = \frac{1}{2m} \sum_{ij} (A_{ij} - \frac{k_i k_j}{2m}) \delta(c_i, c_j)$$

- A_{ij} represents the edge weight between nodes i and j
- k_i and k_j are the sum of the weights of the edges attached to nodes i and j, respectively
- $2m$ is the sum of all of the edge weights in the network

Community Detection Algorithms

Modularity can be naturally extended to weighted/multi-edge networks.

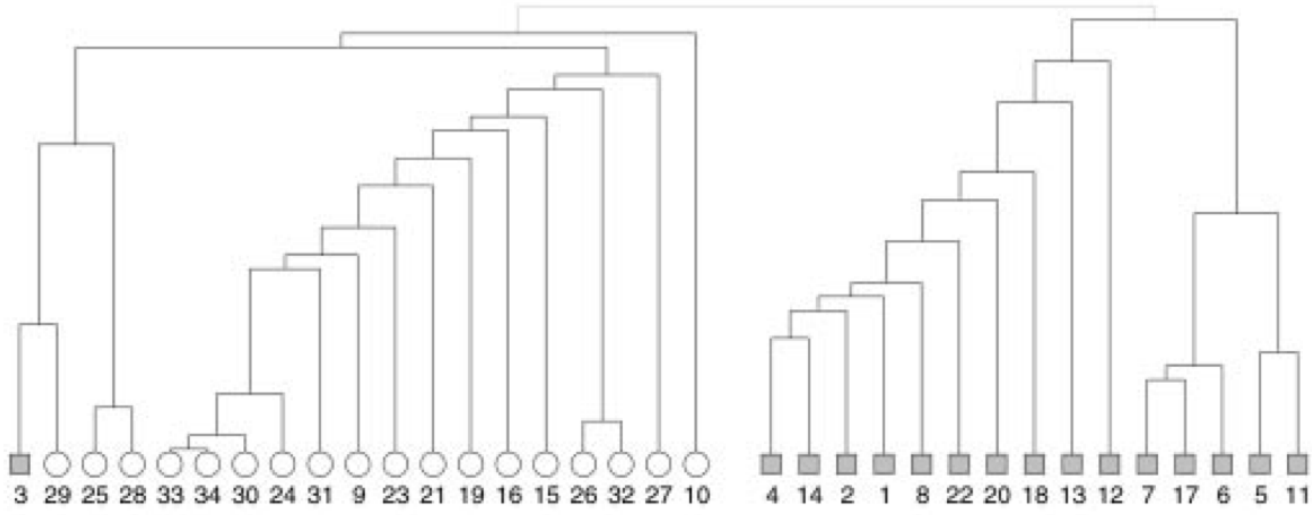
$$Q = \frac{1}{2m} \sum_{ij} (A_{ij} - \frac{k_i k_j}{2m}) \delta(c_i, c_j)$$

- A_{ij} represents the edge weight between nodes i and j
- k_i and k_j are the sum of the weights of the edges attached to nodes i and j, respectively
- $2m$ is the sum of all of the edge weights in the network

Community Detection Algorithms

Back to Girvan & Newman algorithm:

- Create a dendrogram by removing edges
- Cut the dendrogram at the best level using modularity
- It seems that the objective is to optimise modularity.

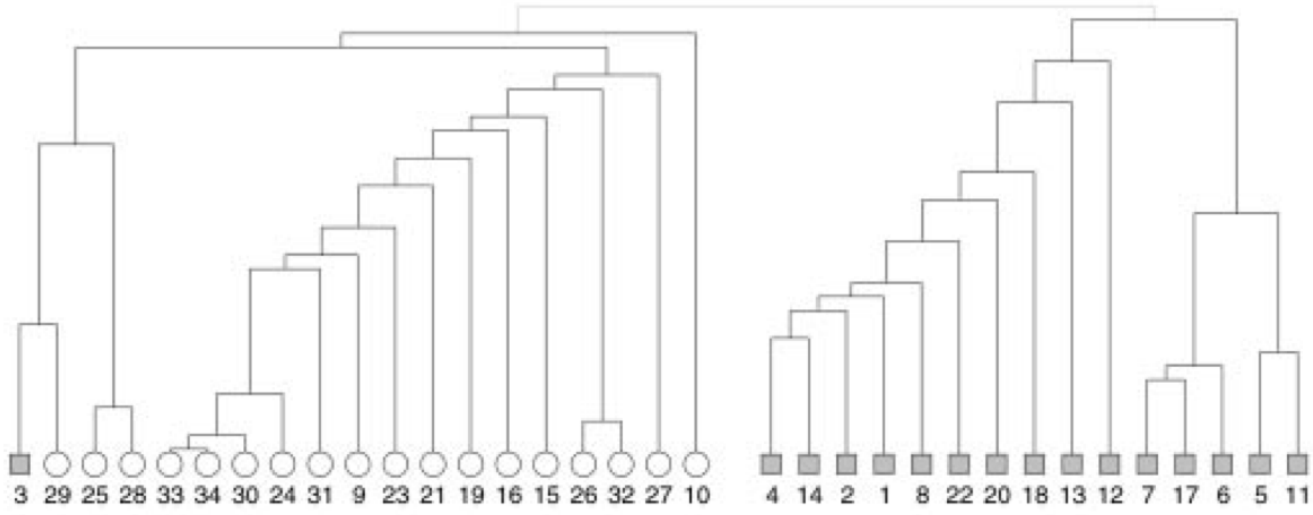


Why not identify communities by maximising modularity directly!

Community Detection Algorithms

Back to Girvan & Newman algorithm:

- Create a dendrogram by removing edges
- Cut the dendrogram at the best level using modularity
- It seems that the objective is to optimise modularity.



Why not identify communities by maximising modularity directly!

Community Detection Algorithms

Louvain algorithm [Blondel et al., 2008]

- Simple, greedy algorithm for community detection
  ($\mathcal{O}(n \log n)$, easy to implement)
- Supports weighted graphs
  (see 'modularity' adapted for weighted networks)
- Yield a hierarchical community structure
- Considered state-of-the-art
  - Rapid convergence properties
  - High modularity output (i.e., "better communities")

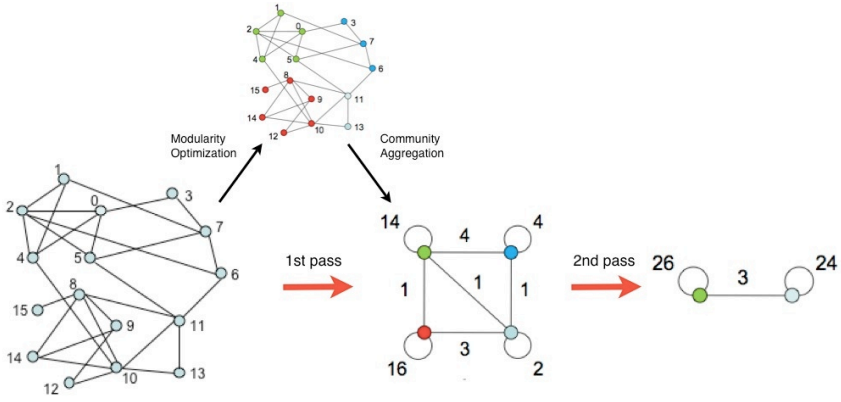
Community Detection Algorithms

Louvain algorithm [Blondel et al., 2008]

- Simple, greedy algorithm for community detection
  ($\mathcal{O}(n \log n)$, easy to implement)
- Supports weighted graphs
  (see 'modularity' adapted for weighted networks)
- Yield a hierarchical community structure
- Considered state-of-the-art
  - Rapid convergence properties
  - High modularity output (i.e., "better communities")

Community Detection Algorithms

Louvain algorithm greedily maximises modularity



1. Phase 1: modularity is optimised by allowing only local changes of communities
2. Phase 2: the identified communities are aggregated in order to build a new network of communities
3. Goto Phase 1 (repeated until no increase of modularity is possible)

Community Detection Algorithms

Louvain algorithm: Phase 1

- Assign a different community to each node
- For each node i, the algorithm performs two calculations:
  1. Compute the modularity gain ($\triangle Q$) when putting node i into the community of some neighbour j
  2. Move i to a community of node j that yields the largest modularity gain $\triangle Q$
  3. If no increase is possible, i remains in its original community.
- Repeat until no movement yields a modularity gain
- The output depends on the order in which the nodes are considered.
- The ordering of the nodes does not have a significant influence on the overall modularity that is obtained.

Community Detection Algorithms

Louvain algorithm: Phase 1

- Assign a different community to each node
- For each node i, the algorithm performs two calculations:
  1. Compute the modularity gain ($\triangle Q$) when putting node i into the community of some neighbour j
  2. Move i to a community of node j that yields the largest modularity gain $\triangle Q$
  3. If no increase is possible, i remains in its original community.
- Repeat until no movement yields a modularity gain
- The output depends on the order in which the nodes are considered.
- The ordering of the nodes does not have a significant influence on the overall modularity that is obtained.

Community Detection Algorithms

Louvain algorithm: Phase 1

- Assign a different community to each node
- For each node i, the algorithm performs two calculations:
  1. Compute the modularity gain ($\triangle Q$) when putting node i into the community of some neighbour j
  2. Move i to a community of node j that yields the largest modularity gain $\triangle Q$
  3. If no increase is possible, i remains in its original community.
- Repeat until no movement yields a modularity gain
- The output depends on the order in which the nodes are considered.
- The ordering of the nodes does not have a significant influence on the overall modularity that is obtained.

Community Detection Algorithms

Louvain algorithm: Phase 1

- Assign a different community to each node
- For each node i, the algorithm performs two calculations:
  1. Compute the modularity gain ($\triangle Q$) when putting node i into the community of some neighbour j
  2. Move i to a community of node j that yields the largest modularity gain $\triangle Q$
  3. If no increase is possible, i remains in its original community.
- Repeat until no movement yields a modularity gain
- The output depends on the order in which the nodes are considered.
- The ordering of the nodes does not have a significant influence on the overall modularity that is obtained.

Community Detection Algorithms

Louvain algorithm: Phase 1

- Assign a different community to each node
- For each node i, the algorithm performs two calculations:
  1. Compute the modularity gain ($\triangle Q$) when putting node i into the community of some neighbour j
  2. Move i to a community of node j that yields the largest modularity gain $\triangle Q$
  3. If no increase is possible, i remains in its original community.
- Repeat until no movement yields a modularity gain
- The output depends on the order in which the nodes are considered.
- The ordering of the nodes does not have a significant influence on the overall modularity that is obtained.

Community Detection Algorithms

Louvain algorithm: Phase 1

- Assign a different community to each node
- For each node i, the algorithm performs two calculations:
  1. Compute the modularity gain ($\triangle Q$) when putting node i into the community of some neighbour j
  2. Move i to a community of node j that yields the largest modularity gain $\triangle Q$
  3. If no increase is possible, i remains in its original community.
- Repeat until no movement yields a modularity gain
- The output depends on the order in which the nodes are considered.
- The ordering of the nodes does not have a significant influence on the overall modularity that is obtained.

Community Detection Algorithms

Louvain algorithm: Phase 2

- Each partition c_i obtained in Phase 1 forms a new super-node i (a self-loop with weight corresponds to the intra-community weight)
- Super-nodes i and j are connected if there is at least one edge between nodes of their corresponding partitions (c_i and c_j)
- The weight of the edge between new super-nodes i and j is the sum of weights from all edges between their corresponding partitions (c_i and c_j)

Community Detection Algorithms

Louvain algorithm: Phase 2

- Each partition c_i obtained in Phase 1 forms a new super-node i (a self-loop with weight corresponds to the intra-community weight)
- Super-nodes i and j are connected if there is at least one edge between nodes of their corresponding partitions (c_i and c_j)
- The weight of the edge between new super-nodes i and j is the sum of weights from all edges between their corresponding partitions (c_i and c_j)

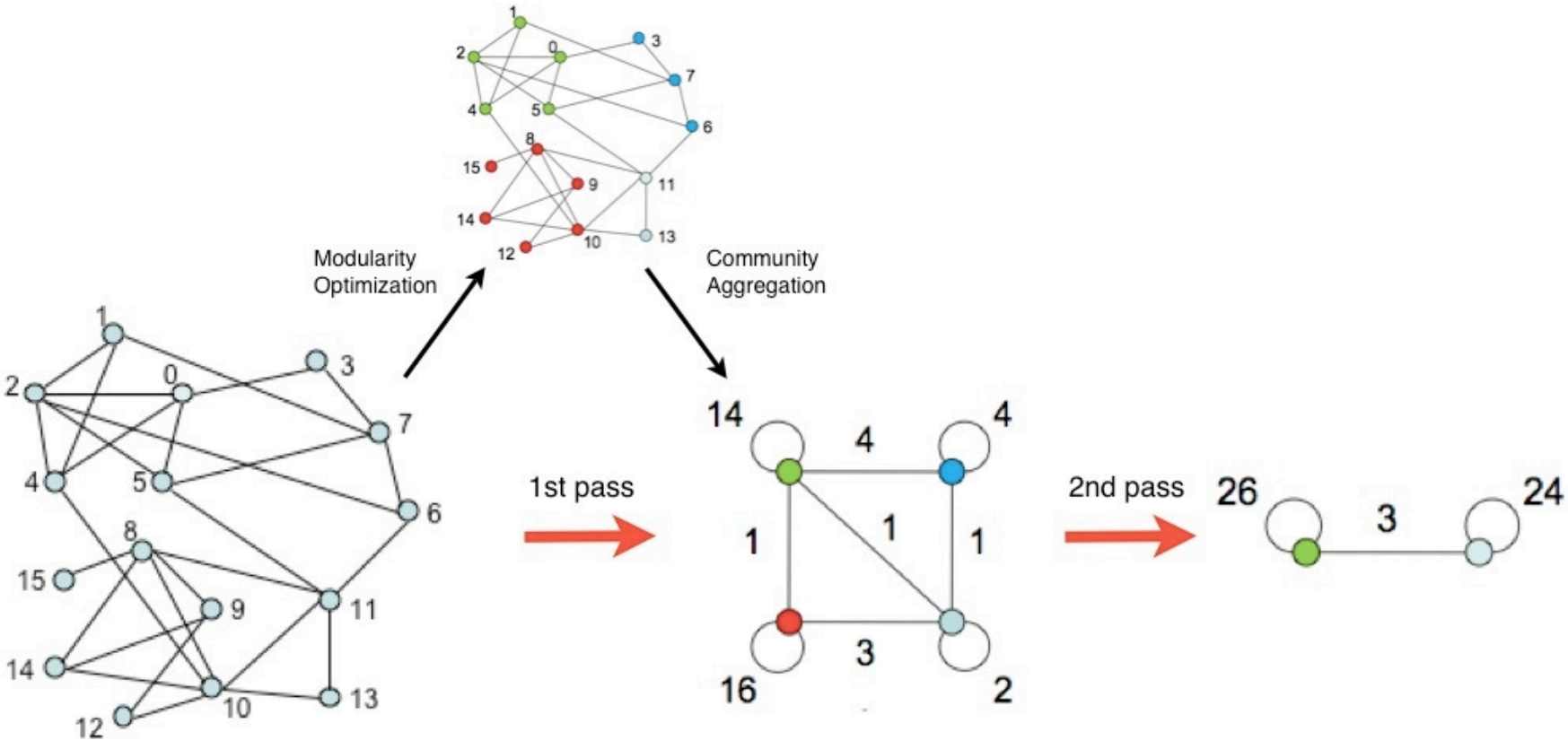
Community Detection Algorithms

Louvain algorithm: Phase 2

- Each partition c_i obtained in Phase 1 forms a new super-node i (a self-loop with weight corresponds to the intra-community weight)
- Super-nodes i and j are connected if there is at least one edge between nodes of their corresponding partitions (c_i and c_j)
- The weight of the edge between new super-nodes i and j is the sum of weights from all edges between their corresponding partitions (c_i and c_j)

Community Detection Algorithms

Louvain algorithm



Community Detection Algorithms

Louvain algorithm

- Louvain produces only non-overlapping communities.
- It may yield arbitrarily badly connected communities.
- In the worst case, communities may even be disconnected, especially when running the algorithm iteratively.
- Leiden algorithm [Traag, Waltman, van Eck, 2019]

Community Detection Algorithms

Louvain algorithm

- Louvain produces only non-overlapping communities.
- It may yield arbitrarily badly connected communities.
- In the worst case, communities may even be disconnected, especially when running the algorithm iteratively.
- Leiden algorithm [Traag, Waltman, van Eck, 2019]

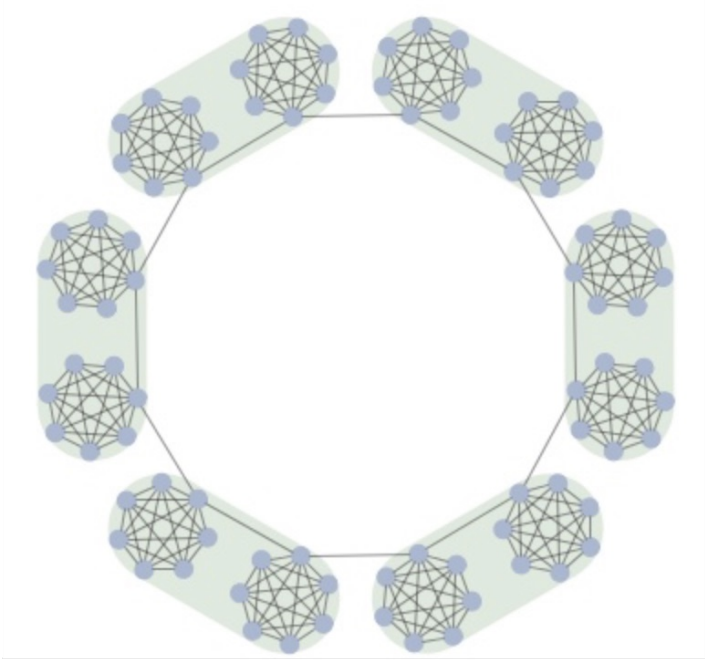
Community Detection Algorithms

Louvain algorithm

- Louvain produces only non-overlapping communities.
- It may yield arbitrarily badly connected communities.
- In the worst case, communities may even be disconnected, especially when running the algorithm iteratively.
- Leiden algorithm [Traag, Waltman, van Eck, 2019]

Community Detection Algorithms

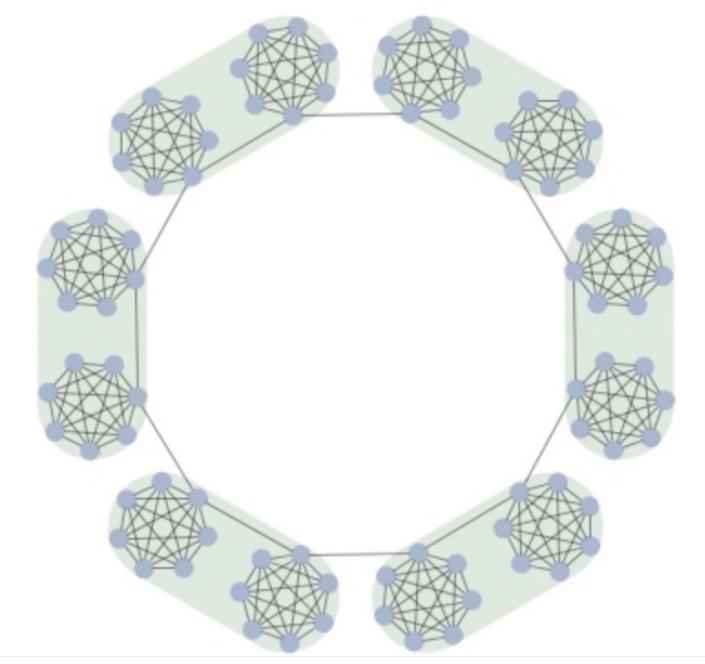
Modularity$\stackrel{?}{=}$ Definition of good communities



- A ring of cliques which are as dense as possible
- Single edges between them (as separate as possible)
- Community detection algorithm: each clique is a community

Community Detection Algorithms

Modularity $\stackrel{?}{=}$ Definition of good communities



With modularity: small graphs (OK); large graphs (the max of modularity obtained by merging cliques)

Community Detection Algorithms

Modularity$\stackrel{?}{=}$ Definition of good communities

- Discovery that modularity has a “favourite scale"
- For a graph of given density and size, communities cannot be smaller/larger than a fraction of nodes
- Modularity optimisation will never discover small (large) communities in large (small) networks
- Unable to find no community (networks without community structure)

Community Detection Algorithms

Modularity $\stackrel{?}{=}$ Definition of good communities

- Discovery that modularity has a “favourite scale"
- For a graph of given density and size, communities cannot be smaller/larger than a fraction of nodes
- Modularity optimisation will never discover small (large) communities in large (small) networks
- Unable to find no community (networks without community structure)