Algorithm Foundations of Data Science and Engineering

Lecture 12: Community Detection

YANHAO WANG

DaSE @ ECNU
(for course related communications)
yhwang@dase.ecnu.edu.cn

Nov 29, 2021

Outline

Image Compression via PCA

Image compression is used to minimize the amount of memory needed to represent an image.

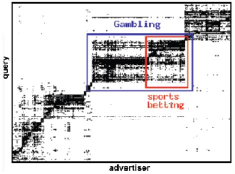
1. Downloading dataset (NWPU VHR-10 dataset is an aerial photography dataset, which can be downloaded from URL [a]);
2. Image compression
   - standardizing the sizes
   - PCA-based image compression
   - Evaluating the performance of image compression
3. Writing a report in Chinese or English (You will be given a report template)

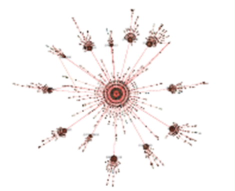
Your project report and source code are due by Jun. 28, 2020.

[a] `http://www.escience.cn/people/gongcheng/NWPU-VHR-10.html`

Network and communities

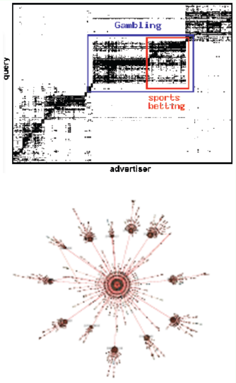
We often think of networks being organized into modules, clusters, communities.





Network and communities

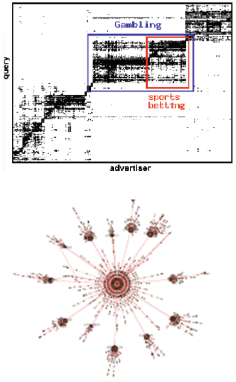
We often think of networks being organized into modules, clusters, communities.



- Network visualization.

Network and communities

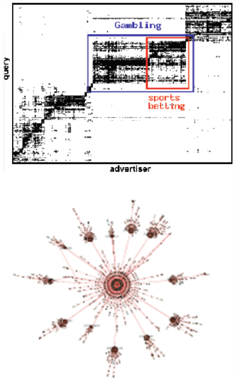
We often think of networks being organized into modules, clusters, communities.



- Network visualization.
- Find densely linked clusters.

Network and communities

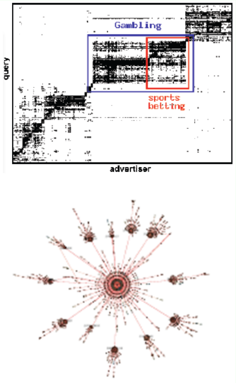
We often think of networks being organized into modules, clusters, communities.



- Network visualization.
- Find densely linked clusters.
- Find micro-markets by partitioning the query VS. advertiser graph.

Network and communities

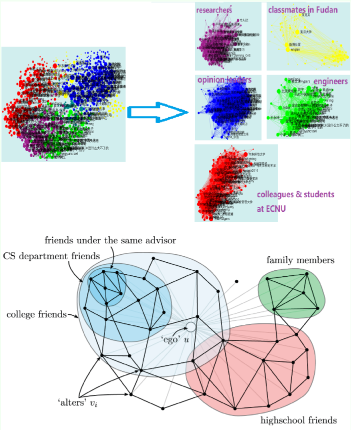
We often think of networks being organized into modules, clusters, communities.



- Network visualization.
- Find densely linked clusters.
- Find micro-markets by partitioning the query VS. advertiser graph.
- Spammer detection (water army).

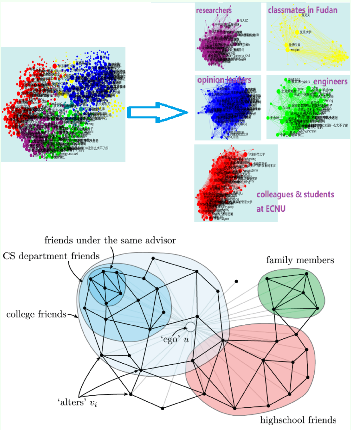
Network and communities Cont'd

Discovering social circles, circles of trust:



Network and communities Cont'd

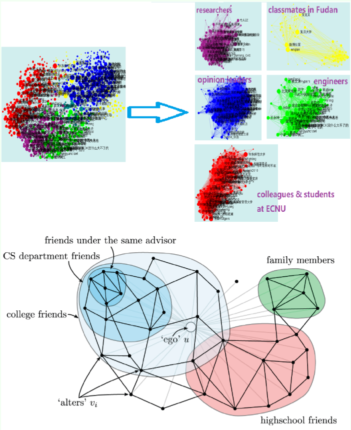
Discovering social circles, circles of trust:



- Trust network.

Network and communities Cont'd

Discovering social circles, circles of trust:



- Trust network.
- Social circles.

Problem setting

Community structure indicates that the network divides naturally into groups of nodes with dense connections internally and sparser connections between groups.

Problem setting

Community structure indicates that the network divides naturally into groups of nodes with dense connections internally and sparser connections between groups.
Graph is large

Problem setting

Community structure indicates that the network divides naturally into groups of nodes with dense connections internally and sparser connections between groups.

Graph is large

- Assume the graph fits in main memory.

Problem setting

Community structure indicates that the network divides naturally into groups of nodes with dense connections internally and sparser connections between groups.

Graph is large

- Assume the graph fits in main memory.
  - For example, to work with a 200M node and 2B edge graph one needs approx. 16GB RAM.

Problem setting

Community structure indicates that the network divides naturally into groups of nodes with dense connections internally and sparser connections between groups.

Graph is large

- Assume the graph fits in main memory.
  - For example, to work with a 200M node and 2B edge graph one needs approx. 16GB RAM.
  - But the graph is too big for running anything more than linear time algorithms

Problem setting

Community structure indicates that the network divides naturally into groups of nodes with dense connections internally and sparser connections between groups.

Graph is large

- Assume the graph fits in main memory.
  - For example, to work with a 200M node and 2B edge graph one needs approx. 16GB RAM.
  - But the graph is too big for running anything more than linear time algorithms
- Community structures:

Problem setting

Community structure indicates that the network divides naturally into groups of nodes with dense connections internally and sparser connections between groups.

Graph is large

- Assume the graph fits in main memory.
  - For example, to work with a 200M node and 2B edge graph one needs approx. 16GB RAM.
  - But the graph is too big for running anything more than linear time algorithms
- Community structures:
  - Global structure.

Problem setting

Community structure indicates that the network divides naturally into groups of nodes with dense connections internally and sparser connections between groups.

Graph is large

- Assume the graph fits in main memory.
  - For example, to work with a 200M node and 2B edge graph one needs approx. 16GB RAM.
  - But the graph is too big for running anything more than linear time algorithms
- Community structures:
  - Global structure.
  - Local structure.

Global community structures

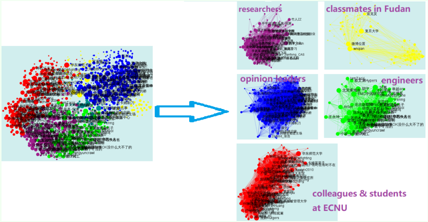
Goal

Partition nodes of a network into disjoint sets.

Global community structures

Goal

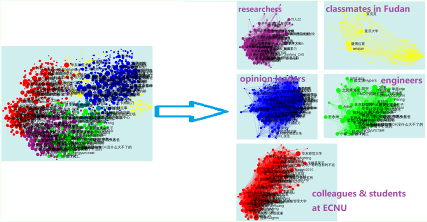
Partition nodes of a network into disjoint sets.



Global community structures

Goal

Partition nodes of a network into disjoint sets.

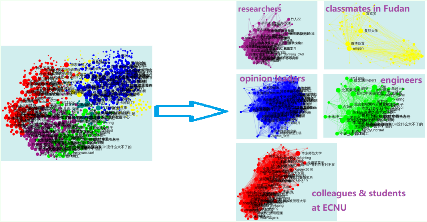


- Clustering based on vertex similarity

Global community structures

Goal

Partition nodes of a network into disjoint sets.

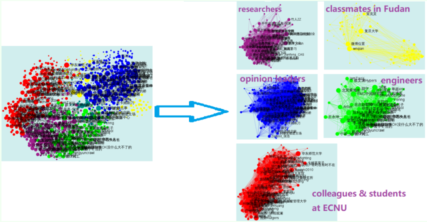


- Clustering based on vertex similarity
- Latent space models

Global community structures

Goal

Partition nodes of a network into disjoint sets.

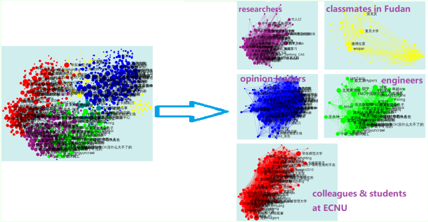


- Clustering based on vertex similarity
- Latent space models
- Spectral clustering

Global community structures

Goal

Partition nodes of a network into disjoint sets.

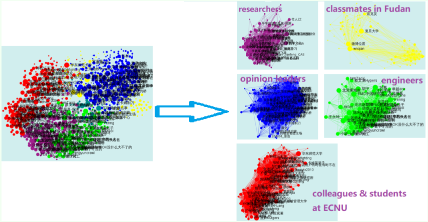


- Clustering based on vertex similarity
- Latent space models
- Spectral clustering
- Modularity maximization

Global community structures

Goal

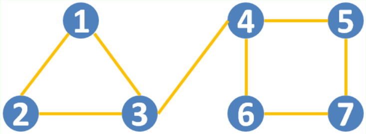
Partition nodes of a network into disjoint sets.



- Clustering based on vertex similarity
- Latent space models
- Spectral clustering
- Modularity maximization

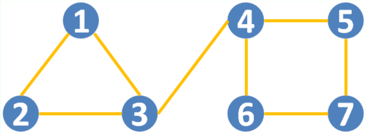
In the study of complex networks, a network is said to have community structure if the nodes of the network can be easily grouped into (potentially overlapping) sets of nodes such that each set of nodes is densely connected internally.

What makes a good community?



Input an undirected graph $G = (V, E)$:

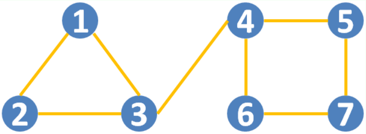
What makes a good community?



Input an undirected graph $G = (V, E)$:

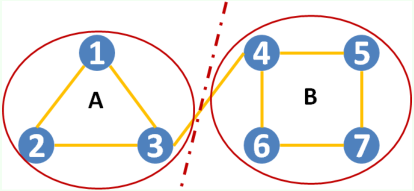
- Partitioning task: Divide vertices into 2 disjoint groups A and $B = V \backslash A$.

What makes a good community?



Input an undirected graph $G = (V, E)$:

- Partitioning task: Divide vertices into 2 disjoint groups A and $B = V \backslash A$.
- How can we define a "good" community in G?



What makes a good community? Cont'd

What makes a good community?

What makes a good community? Cont'd

What makes a good community?

- Maximize the number of intra-community connections.

What makes a good community? Cont'd

What makes a good community?

- Maximize the number of intra-community connections.
- Minimize the number of inter-community connections

What makes a good community? Cont'd

What makes a good community?

- Maximize the number of intra-community connections.
- Minimize the number of inter-community connections

Express community quality as a function of the "edge cut" of the community, where cut is the set of edges (edge weights) with only one node in the community, and can be defined as

$$cut(A) = \sum_{i \in A, j \notin A} w_{ij}.$$

What makes a good community? Cont'd

What makes a good community?

- Maximize the number of intra-community connections.
- Minimize the number of inter-community connections

Express community quality as a function of the "edge cut" of the community, where cut is the set of edges (edge weights) with only one node in the community, and can be defined as

$$cut(A) = \sum_{i \in A, j \notin A} w_{ij}.$$

A good community makes a minimal cut.

What makes a good community? Cont'd

What makes a good community?

- Maximize the number of intra-community connections.
- Minimize the number of inter-community connections

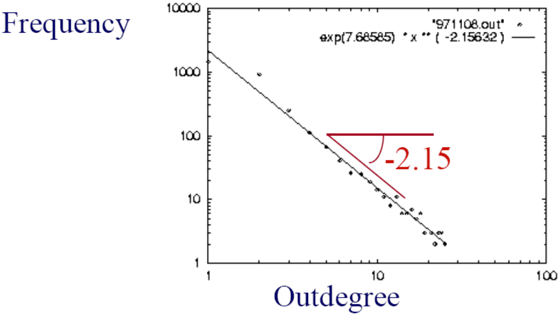
Express community quality as a function of the "edge cut" of the community, where cut is the set of edges (edge weights) with only one node in the community, and can be defined as

$$cut(A) = \sum_{i \in A, j \notin A} w_{ij}.$$

A good community makes a minimal cut. A cut is minimum if the size or weight of the cut is not larger than the size of any other cut. There are polynomial-time methods to solve the min-cut problem, notably the EdmondsCKarp algorithm, which complexity is $O(|V||E|^2)$.

Outline

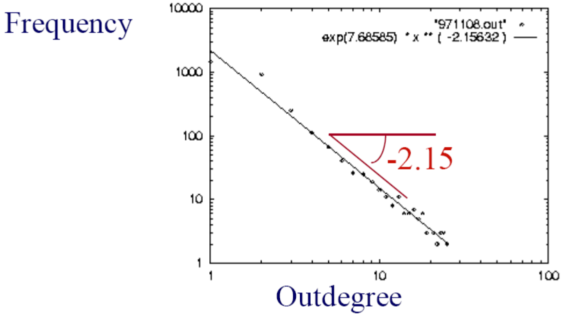
Power-law I



Internet topology [SIGCOMM 99]

- Out-degree distribution is plotted in log-log scale.

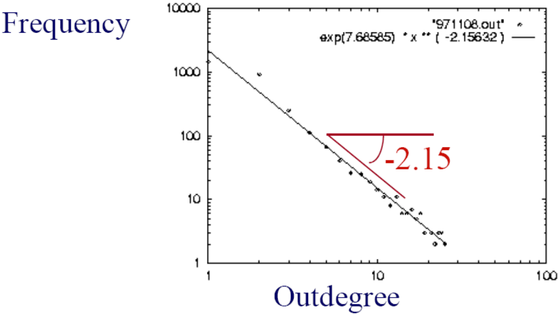
Power-law I



Internet topology [SIGCOMM 99]

- Out-degree distribution is plotted in log-log scale.
- It forms a line with a slope ~ -2.15

Power-law I



Internet topology [SIGCOMM 99]

- Out-degree distribution is plotted in log-log scale.
- It forms a line with a slope ~ -2.15
- $freq. = deg.^{-2.15}$

What is the power-law?

Due to Matthew effect, Pareto's law, "rich-get-richer", or the 80/20 principle, there are many settings with power law.

What is the power-law?

Due to Matthew effect, Pareto's law, "rich-get-richer", or the 80/20 principle, there are many settings with power law.

- 80% of Italy's land owned by 20% of the population.

What is the power-law?

Due to Matthew effect, Pareto's law, "rich-get-richer", or the 80/20 principle, there are many settings with power law.

- 80% of Italy's land owned by 20% of the population.
- Richest 20% obtain 82.70% income [world GDP, 1989].

What is the power-law?

Due to Matthew effect, Pareto's law, "rich-get-richer", or the 80/20 principle, there are many settings with power law.

- 80% of Italy's land owned by 20% of the population.
- Richest 20% obtain 82.70% income [world GDP, 1989].
- Bible: rank VS. frequency (log-log)

What is the power-law?

Due to Matthew effect, Pareto's law, "rich-get-richer", or the 80/20 principle, there are many settings with power law.

- 80% of Italy's land owned by 20% of the population.
- Richest 20% obtain 82.70% income [world GDP, 1989].
- Bible: rank VS. frequency (log-log)
- Web: hit count VS. volume

What is the power-law?

Due to Matthew effect, Pareto's law, "rich-get-richer", or the 80/20 principle, there are many settings with power law.

- 80% of Italy's land owned by 20% of the population.
- Richest 20% obtain 82.70% income [world GDP, 1989].
- Bible: rank VS. frequency (log-log)
- Web: hit count VS. volume
- File: count VS. size

What is the power-law?

Due to Matthew effect, Pareto's law, "rich-get-richer", or the 80/20 principle, there are many settings with power law.

- 80% of Italy's land owned by 20% of the population.
- Richest 20% obtain 82.70% income [world GDP, 1989].
- Bible: rank VS. frequency (log-log)
- Web: hit count VS. volume
- File: count VS. size
- Business

What is the power-law?

Due to Matthew effect, Pareto's law, "rich-get-richer", or the 80/20 principle, there are many settings with power law.

- 80% of Italy's land owned by 20% of the population.
- Richest 20% obtain 82.70% income [world GDP, 1989].
- Bible: rank VS. frequency (log-log)
- Web: hit count VS. volume
- File: count VS. size
- Business
  - 80% of a company's profits come from 20% of its customers.

What is the power-law?

Due to Matthew effect, Pareto's law, "rich-get-richer", or the 80/20 principle, there are many settings with power law.

- 80% of Italy's land owned by 20% of the population.
- Richest 20% obtain 82.70% income [world GDP, 1989].
- Bible: rank VS. frequency (log-log)
- Web: hit count VS. volume
- File: count VS. size
- Business
  - 80% of a company's profits come from 20% of its customers.
  - 80% of a company's complaints come from 20% of its customers.

What is the power-law?

Due to Matthew effect, Pareto's law, "rich-get-richer", or the 80/20 principle, there are many settings with power law.

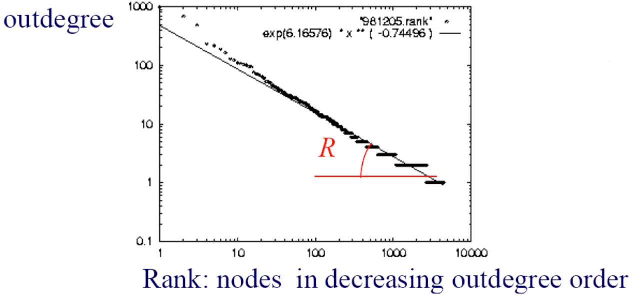
- 80% of Italy's land owned by 20% of the population.
- Richest 20% obtain 82.70% income [world GDP, 1989].
- Bible: rank VS. frequency (log-log)
- Web: hit count VS. volume
- File: count VS. size
- Business
  - 80% of a company's profits come from 20% of its customers.
  - 80% of a company's complaints come from 20% of its customers.
  - 80% of a company's profits come from 20% of the time its staff spent

What is the power-law?

Due to Matthew effect, Pareto's law, "rich-get-richer", or the 80/20 principle, there are many settings with power law.

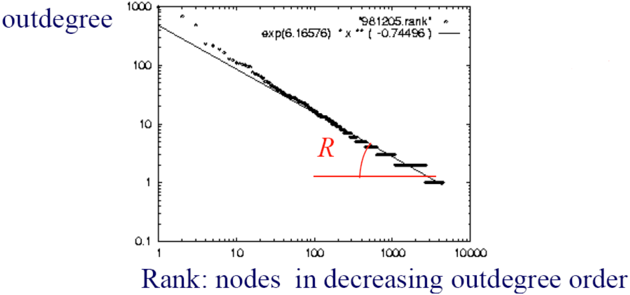
- 80% of Italy's land owned by 20% of the population.
- Richest 20% obtain 82.70% income [world GDP, 1989].
- Bible: rank VS. frequency (log-log)
- Web: hit count VS. volume
- File: count VS. size
- Business
  - 80% of a company's profits come from 20% of its customers.
  - 80% of a company's complaints come from 20% of its customers.
  - 80% of a company's profits come from 20% of the time its staff spent
  - 80% of a company's sales are made by 20% of its sales staff

Power-law II



Rank of out-degrees [ICDE 09]

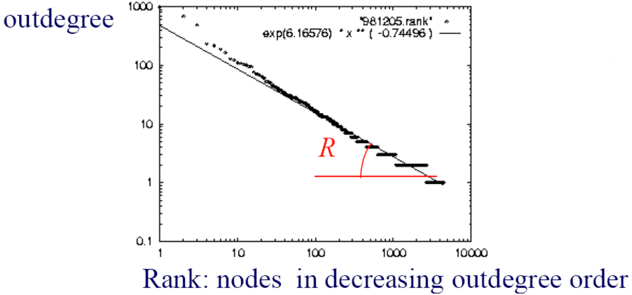
Power-law II



Rank of out-degrees [ICDE 09]

- Vertices are ranked in decreasing out-degree order, and plotted in log-log scale.

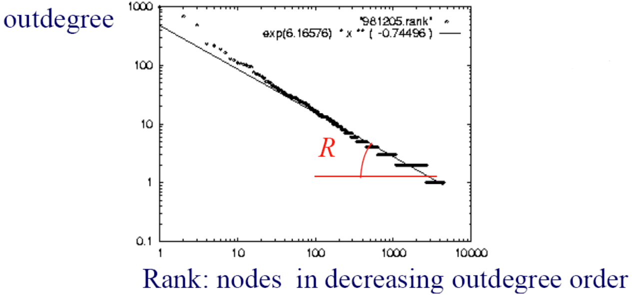
Power-law II



Rank of out-degrees [ICDE 09]

- Vertices are ranked in decreasing out-degree order, and plotted in log-log scale.
- It forms a line with a slope ~ -0.74

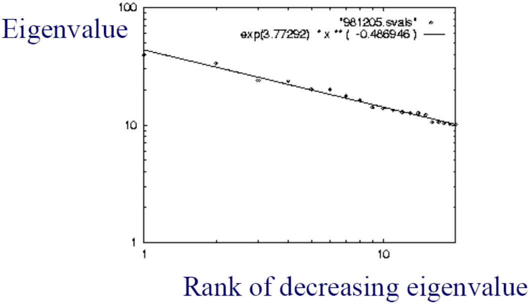
Power-law II



Rank of out-degrees [ICDE 09]

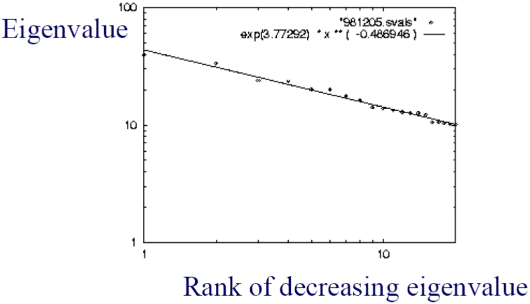
- Vertices are ranked in decreasing out-degree order, and plotted in log-log scale.
- It forms a line with a slope ~ -0.74
- $deg. = rank^{-0.74}$

Power-law III



Rank of eigenvalues [ICDE 09]

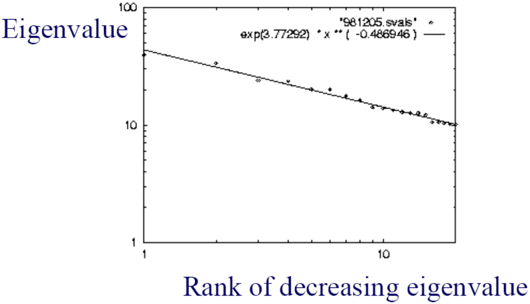
Power-law III



Rank of eigenvalues [ICDE 09]

- Eigenvalues of adjacency matrix (top 20) are ranked in decreasing order, and plotted in log-log scale.

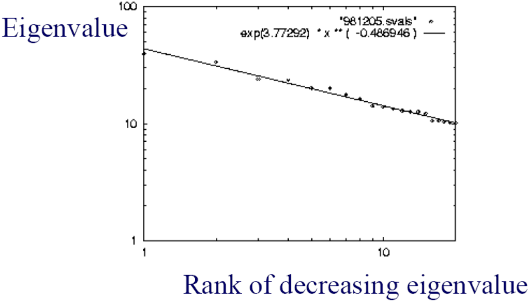
Power-law III



Rank of eigenvalues [ICDE 09]

- Eigenvalues of adjacency matrix (top 20) are ranked in decreasing order, and plotted in log-log scale.
- It forms a line with a slope ~ -0.48

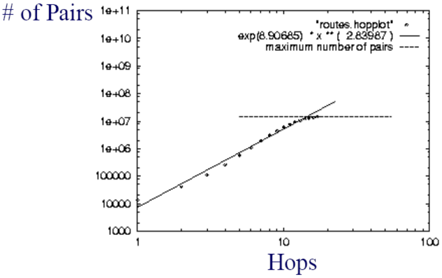
Power-law III

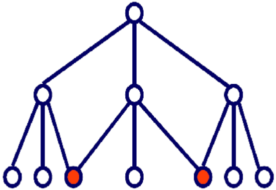


Rank of eigenvalues [ICDE 09]

- Eigenvalues of adjacency matrix (top 20) are ranked in decreasing order, and plotted in log-log scale.
- It forms a line with a slope ~ -0.48
- $eigen. = rank^{-0.48}$

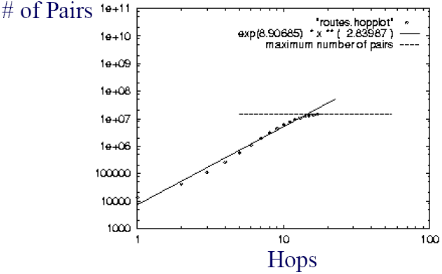
Power-law IV

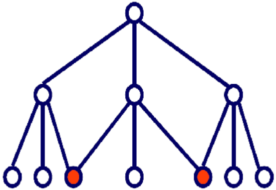




Hop plot [ICDE 09]

Power-law IV

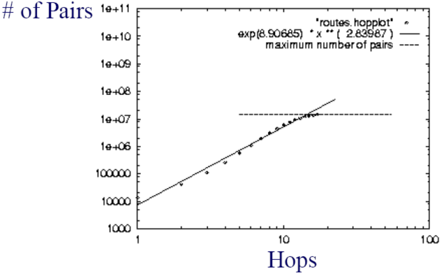


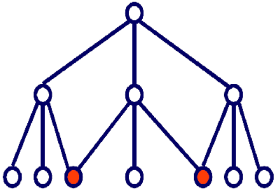


Hop plot [ICDE 09]

- How many neighbors within $1, 2, \cdots, h$ hops? $(\sum_{i=1}^{h} avg.^{i})$

Power-law IV

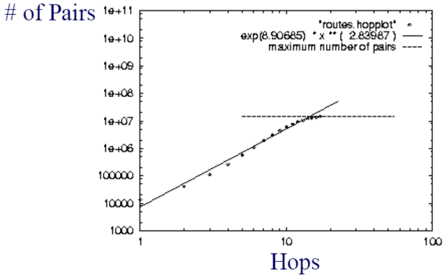


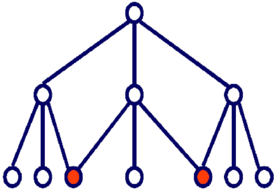


Hop plot [ICDE 09]

- How many neighbors within $1, 2, \cdots, h$ hops? $(\sum_{i=1}^{h} avg.^{i})$
- Pairs of vertices are plotted in log-log scale. It forms a line with a slope ~ 2.83

Power-law IV

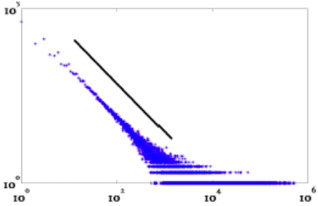




Hop plot [ICDE 09]

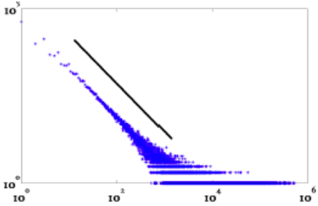
- How many neighbors within $1, 2, \cdots, h$ hops? $(\sum_{i=1}^{h} avg.^{i})$
- Pairs of vertices are plotted in log-log scale. It forms a line with a slope ~ 2.83
- $pairs. = hop^{2.83}$

Power-law V



Counting of triangle [ICDM 08]

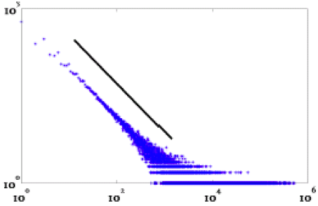
Power-law V



Counting of triangle [ICDM 08]

- X-axis: # of triangles a vertex participates in

Power-law V



Counting of triangle [ICDM 08]

- X-axis: # of triangles a vertex participates in
- Y-axis: count of such vertices
- In log-log scale, the plot is almost linear.

Erdos-Renyi model

Erdős-Renyi model is known as the random graph model, which generates undirected random graphs.

Erdos-Renyi model

Erdös-Renyi model is known as the random graph model, which generates undirected random graphs.

- Parameters: N (# vertices) and p (probability of forming an edge)

Erdos-Renyi model

Erdös-Renyi model is known as the random graph model, which generates undirected random graphs.

- Parameters: N (# vertices) and p (probability of forming an edge)
- For each possible node pair, the approach generates an edge with probability p. Thus, # edges $= \frac{pN(N-1)}{2}$.

Erdos-Renyi model

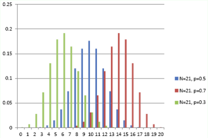
Erdös-Renyi model is known as the random graph model, which generates undirected random graphs.

- Parameters: N (# vertices) and p (probability of forming an edge)
- For each possible node pair, the approach generates an edge with probability p. Thus, # edges $= \frac{pN(N-1)}{2}$.
- Degree distribution:
  - $P(\text{node has degree k}) = \binom{N-1}{k} p^k (1-p)^{N-1-k}$
  - Follows binomial distribution with mean $(N-1)p$ and variance $(N-1)p(1-p)$ (not power-law distribution).

Erdos-Renyi model

Erdös-Renyi model is known as the random graph model, which generates undirected random graphs.

- Parameters: N (# vertices) and p (probability of forming an edge)
- For each possible node pair, the approach generates an edge with probability p. Thus, # edges $= \frac{pN(N-1)}{2}$.
- Degree distribution:
  - P(node has degree k) $= \binom{N-1}{k} p^k (1-p)^{N-1-k}$
  - Follows binomial distribution with mean $(N-1)p$ and variance $(N-1)p(1-p)$ (not power-law distribution).



Scale-free network

Preferential attachment model

The more connected a node is, the more likely it is to receive new links (namely, Rich gets Richer, Matthew Effect or Paretos Law, etc.).

Scale-free network

Preferential attachment model

The more connected a node is, the more likely it is to receive new links (namely, Rich gets Richer, Matthew Effect or Paretos Law, etc.).

- Price model

Scale-free network

Preferential attachment model

The more connected a node is, the more likely it is to receive new links (namely, Rich gets Richer, Matthew Effect or Paretos Law, etc.).

- Price model
- Barabasi Albert model

Scale-free network

Preferential attachment model

The more connected a node is, the more likely it is to receive new links (namely, Rich gets Richer, Matthew Effect or Paretos Law, etc.).

- Price model
- Barabasi Albert model

Price model for citation networks

Scale-free network

Preferential attachment model

The more connected a node is, the more likely it is to receive new links (namely, Rich gets Richer, Matthew Effect or Paretos Law, etc.).

- Price model
- Barabasi Albert model

Price model for citation networks

- Each new paper is generated with m citations (mean).

Scale-free network

Preferential attachment model

The more connected a node is, the more likely it is to receive new links (namely, Rich gets Richer, Matthew Effect or Paretos Law, etc.).

- Price model
- Barabasi Albert model

Price model for citation networks

- Each new paper is generated with m citations (mean).
- New papers cite previous papers with probability proportional to their indegree (citations).

Scale-free network

Preferential attachment model

The more connected a node is, the more likely it is to receive new links (namely, Rich gets Richer, Matthew Effect or Paretos Law, etc.).

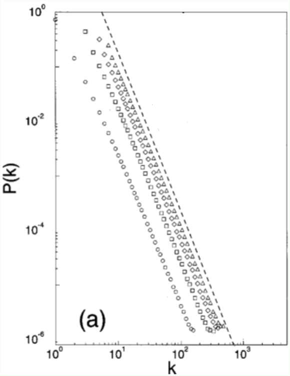
- Price model
- Barabasi Albert model

Price model for citation networks

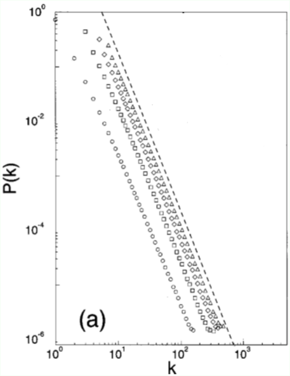
- Each new paper is generated with m citations (mean).
- New papers cite previous papers with probability proportional to their indegree (citations).
- Power law with exponent $\alpha = 2 + \frac{1}{m}$ [Science 1965]

Barabasi Albert model

Model



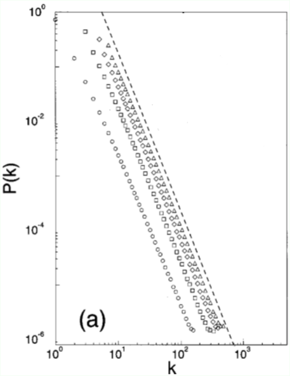
Barabasi Albert model



Model

- Start with an initial network of m_0 (≥ 2) nodes, and the degree of each node ≥ 1, otherwise it will always remain isolated.

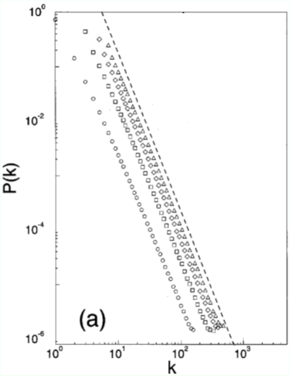
Barabasi Albert model



Model

- Start with an initial network of m_0 (≥ 2) nodes, and the degree of each node ≥ 1, otherwise it will always remain isolated.
- For each new node, connect it to m existing nodes i with a probability p_i, where $p_i = \frac{k_i}{\sum_j k_j}$, where k_i is degree of node i.

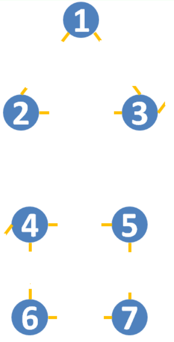
Barabasi Albert model



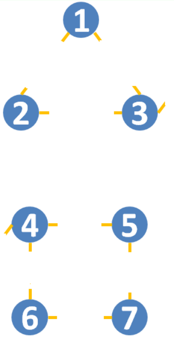
Model

- Start with an initial network of m_0 (≥ 2) nodes, and the degree of each node ≥ 1, otherwise it will always remain isolated.
- For each new node, connect it to m existing nodes i with a probability p_i, where $p_i = \frac{k_i}{\sum_j k_j}$, where k_i is degree of node i.
- Results in a single connected component with power-law degree distribution with $\alpha = 3$ [Reviews of Modern Physics 2003].

Null model

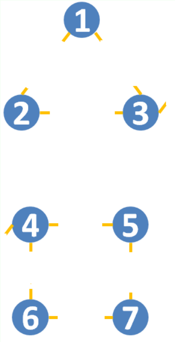


Null model



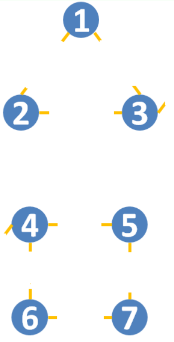
- The expected number of edges between vertices v_i and v_j

Null model



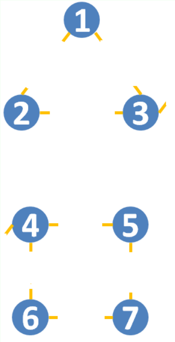
- The expected number of edges between vertices v_i and v_j
  - Suppose that the starting vertices v_j;

Null model



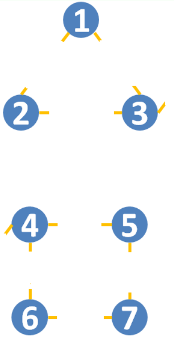
- The expected number of edges between vertices v_i and v_j
  - Suppose that the starting vertices v_j;
  - Define a r.v. $X_{ij} = 1$ if the target vertex is v_j, 0 otherwise.

Null model



- The expected number of edges between vertices v_i and v_j
  - Suppose that the starting vertices v_j;
  - Define a r.v. $X_{ij} = 1$ if the target vertex is v_j, 0 otherwise.
  - Thus, we have $Pr[X_{ij} = 1] = P_{ij}$, which is a Bernoulli r.v..

Null model



- The expected number of edges between vertices v_i and v_j
  - Suppose that the starting vertices v_j;
  - Define a r.v. $X_{ij} = 1$ if the target vertex is v_j, 0 otherwise.
  - Thus, we have $Pr[X_{ij} = 1] = P_{ij}$, which is a Bernoulli r.v..

We have:

$$\sum_{i,j} E(X_{ij}) = \sum_{i,j} P_{ij} = 2m$$

$$\sum_{j} E(X_{ij}) = \sum_{j} P_{ij} = k_i$$

Null model Cont'd

Since P_{ij} is related to k_i and k_j, let $P_{ij} = f(k_i)f(k_j)$.

Null model Cont'd

Since P_{ij} is related to k_i and k_j, let $P_{ij} = f(k_i)f(k_j)$.

- Due to $P_{ij} = P_{ji}$, we have

$$\sum_j P_{ij} = \sum_j f(k_i)f(k_j) = f(k_i)\sum_j f(k_j) = k_i;$$

Null model Cont'd

Since P_{ij} is related to k_i and k_j, let $P_{ij} = f(k_i)f(k_j)$.

- Due to $P_{ij} = P_{ji}$, we have

$$\sum_j P_{ij} = \sum_j f(k_i)f(k_j) = f(k_i)\sum_j f(k_j) = k_i;$$

- Since $\sum_j f(k_j)$ is not related to v_i, $f(k_i) = C \cdot k_i$;

Null model Cont'd

Since P_{ij} is related to k_i and k_j, let $P_{ij} = f(k_i)f(k_j)$.

- Due to $P_{ij} = P_{ji}$, we have

$$\sum_j P_{ij} = \sum_j f(k_i)f(k_j) = f(k_i)\sum_j f(k_j) = k_i;$$

- Since $\sum_j f(k_j)$ is not related to v_i, $f(k_i) = C \cdot k_i$;
- Furthermore,

$$\sum_{i,j} f(k_i)f(k_j) = C^2 \cdot \sum_{i,j} k_i \cdot k_j = C^2 \cdot (2m)^2 = 2m.$$

Null model Cont'd

Since P_{ij} is related to k_i and k_j, let $P_{ij} = f(k_i)f(k_j)$.

- Due to $P_{ij} = P_{ji}$, we have

$$\sum_j P_{ij} = \sum_j f(k_i)f(k_j) = f(k_i)\sum_j f(k_j) = k_i;$$

- Since $\sum_j f(k_j)$ is not related to v_i, $f(k_i) = C \cdot k_i$;
- Furthermore,

$$\sum_{i,j} f(k_i)f(k_j) = C^2 \cdot \sum_{i,j} k_i \cdot k_j = C^2 \cdot (2m)^2 = 2m.$$

- Therefore,

$$P_{ij} = f(k_i)f(k_j) = C^2 \cdot k_i \cdot k_j = \frac{k_i \cdot k_j}{2m}.$$

Binomial explanation for Null model

- Let X_i be i.i.d. $Bernoulli(\frac{k_i}{2m})$, where $X_i = 1$ means that v_i is a starting vertex, otherwise 0.

Binomial explanation for Null model

- Let X_i be i.i.d. $Bernoulli(\frac{k_i}{2m})$, where $X_i = 1$ means that v_i is a starting vertex, otherwise 0.
- Similarly, let Y_i be i.i.d. $Bernoulli(\frac{k_i}{2m})$, where $Y_i = 1$ means that v_i is a targeting vertex, otherwise 0.

Binomial explanation for Null model

- Let X_i be i.i.d. $Bernoulli(\frac{k_i}{2m})$, where $X_i = 1$ means that v_i is a starting vertex, otherwise 0.
- Similarly, let Y_i be i.i.d. $Bernoulli(\frac{k_i}{2m})$, where $Y_i = 1$ means that v_i is a targeting vertex, otherwise 0.

For an edge $e_{ij} \in E$,

Binomial explanation for Null model

- Let X_i be i.i.d. $Bernoulli(\frac{k_i}{2m})$, where $X_i = 1$ means that v_i is a starting vertex, otherwise 0.
- Similarly, let Y_i be i.i.d. $Bernoulli(\frac{k_i}{2m})$, where $Y_i = 1$ means that v_i is a targeting vertex, otherwise 0.

For an edge $e_{ij} \in E$,

- It is formed when the starting and targeting vertices are v_i and v_j, respectively.

Binomial explanation for Null model

- Let X_i be i.i.d. $Bernoulli(\frac{k_i}{2m})$, where $X_i = 1$ means that v_i is a starting vertex, otherwise 0.
- Similarly, let Y_i be i.i.d. $Bernoulli(\frac{k_i}{2m})$, where $Y_i = 1$ means that v_i is a targeting vertex, otherwise 0.

For an edge $e_{ij} \in E$,

- It is formed when the starting and targeting vertices are v_i and v_j, respectively.
- Let

$$Z_{ij} = \begin{cases} 1, & e_{ij} \in E \\ 0, & \text{otherwise.} \end{cases}$$

Binomial explanation for Null model Cont'd

- Note that there are $2m$ edges in the input graph. After inverting $2m$ edges into the graph, $\sum_{i,j} Z_{ij} \sim Binom(2m, p)$.

Binomial explanation for Null model Cont'd

- Note that there are $2m$ edges in the input graph. After inverting $2m$ edges into the graph, $\sum_{i,j} Z_{ij} \sim Binom(2m, p)$.
- Since

$$p = P(Z_{ij} = 1) = P(X_i = 1, Y_j = 1) = P(X_i = 1)P(Y_j = 1)$$
$$= \frac{k_i}{2m} \cdot \frac{k_j}{2m}$$

Binomial explanation for Null model Cont'd

- Note that there are $2m$ edges in the input graph. After inverting $2m$ edges into the graph, $\sum_{i,j} Z_{ij} \sim Binom(2m, p)$.
- Since

$$p = P(Z_{ij} = 1) = P(X_i = 1, Y_j = 1) = P(X_i = 1)P(Y_j = 1)$$
$$= \frac{k_i}{2m} \cdot \frac{k_j}{2m}$$

- Thus, the expectation number of edge between vertices v_i and v_j is

$$E(\sum_{i,j} Z_{ij}) = 2m \cdot P(Z_{ij} = 1) = \frac{k_i \cdot k_j}{2m}.$$

Outline

Modularity

Definition

Modularity is one measure of the structure of networks or graphs.

Modularity

Definition

Modularity is one measure of the structure of networks or graphs. Given an undirected $G = (V, E)$, it's adjacency matrix is A, where

$$A_{ij} = \begin{cases} 1, & \text{if } (v_i, v_j) \in E; \\ 0, & \text{otherwise.} \end{cases}$$

Modularity

Definition

Modularity is one measure of the structure of networks or graphs. Given an undirected $G = (V, E)$, it's adjacency matrix is A, where

$$A_{ij} = \begin{cases} 1, & \text{if } (v_i, v_j) \in E; \\ 0, & \text{otherwise.} \end{cases}$$

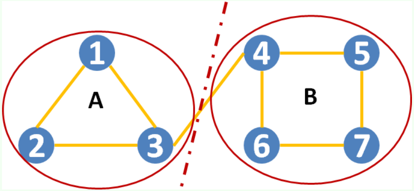
Modularity [Newman 2006]:

$$Q = \frac{1}{2m} \sum_{i,j} (A_{ij} - \frac{k_i k_j}{2m}) \delta(C_i, C_j),$$

where m and C_i denote $\#$ edges and the $i-$th community in the graph, k_i is the degree of vertex v_i, and

$$\delta(C_i, C_j) = \begin{cases} 1, & \text{if } C_i = C_j; \\ 0, & \text{otherwise.} \end{cases}$$

Example of modularity

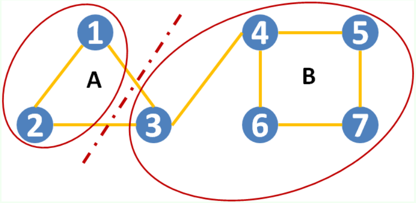


$m = 8, k_1 = 2, k_2 = 2, k_3 = 3,$
$k_4 = 3, k_5 = 2, k_6 = 2, k_7 = 2$

Thus, the modularity of this partition is

$$Q = \frac{1}{2m}\sum_{i,j}(A_{ij} - \frac{k_i k_j}{16})\delta(C_i, C_j) = \frac{1}{16}\Big[(0 - \frac{k_1 k_1}{16}) + 2(1 - \frac{k_1 k_2}{16}) + 2(1 - \frac{k_1 k_3}{16})$$
$$+ (0 - \frac{k_2 k_2}{16}) + 2(1 - \frac{k_2 k_3}{16}) + (0 - \frac{k_3 k_3}{16}) + (0 - \frac{k_4 k_4}{16}) + 2(1 - \frac{k_4 k_5}{16})$$
$$+ 2(1 - \frac{k_4 k_6}{16}) + 2(0 - \frac{k_4 k_7}{16}) + (0 - \frac{k_5 k_5}{16}) + 2(0 - \frac{k_5 k_6}{16}) + 2(1 - \frac{k_5 k_7}{16})$$
$$+ (0 - \frac{k_6 k_6}{16}) + 2(1 - \frac{k_6 k_7}{16}) + (0 - \frac{k_7 k_7}{16})\Big] = \frac{47}{128}$$

Example of modularity Cont'd



$m = 8, k_1 = 2, k_2 = 2, k_3 = 3,$
$k_4 = 3, k_5 = 2, k_6 = 2, k_7 = 2$

Thus, the modularity of this partition is

$$Q = \frac{1}{2m}\sum_{i,j}(A_{ij} - \frac{k_i k_j}{16})\delta(C_i, C_j) = \frac{1}{16}\Big[(0 - \frac{k_1 k_1}{16}) + 2(1 - \frac{k_1 k_2}{16}) + (0 - \frac{k_2 k_2}{16})$$
$$+ (0 - \frac{k_3 k_3}{16}) + 2(1 - \frac{k_3 k_4}{16}) + 2(0 - \frac{k_3 k_5}{16}) + 2(0 - \frac{k_3 k_6}{16}) + 2(0 - \frac{k_3 k_7}{16})$$
$$+ (0 - \frac{k_4 k_4}{16}) + 2(1 - \frac{k_4 k_5}{16}) + 2(1 - \frac{k_4 k_6}{16}) + 2(0 - \frac{k_4 k_7}{16}) + (0 - \frac{k_5 k_5}{16})$$
$$+ 2(0 - \frac{k_5 k_6}{16}) + 2(1 - \frac{k_5 k_7}{16}) + (0 - \frac{k_6 k_6}{16}) + 2(1 - \frac{k_6 k_7}{16}) + (0 - \frac{k_7 k_7}{16})\Big] = \frac{1}{8}$$

Intuition of modularity

A measure of how well a network is partitioned into communities.

Intuition of modularity

A measure of how well a network is partitioned into communities.

Intuition of modularity

A measure of how well a network is partitioned into communities.

- Given a partitioning of the network into groups $c \in C$:

$$Q \propto \sum_{i,j}(A_{ij} - \frac{k_i k_j}{2m})\delta(C_i, C_j) = \big[\sum_{i,j} A_{ij} - \sum_{i,j} \frac{k_i k_j}{2m}\big]\delta(C_i, C_j)$$

$$= \sum_{c \in C} \big[(\#\text{ edges within group } c)$$

$$- (\text{expected } \#\text{ edges within group } c)\big]$$

Intuition of modularity

A measure of how well a network is partitioned into communities.

- Given a partitioning of the network into groups $c \in C$:

$$Q \propto \sum_{i,j} (A_{ij} - \frac{k_i k_j}{2m}) \delta(C_i, C_j) = \big[\sum_{i,j} A_{ij} - \sum_{i,j} \frac{k_i k_j}{2m} \big] \delta(C_i, C_j)$$

$$= \sum_{c \in C} \big[(\# \text{ edges within group } c)$$

$$- (\text{expected } \# \text{ edges within group } c) \big]$$

- Given real G on n vertices and m edges, construct a random network G';

Intuition of modularity

A measure of how well a network is partitioned into communities.

- Given a partitioning of the network into groups $c \in C$:

$$Q \propto \sum_{i,j} (A_{ij} - \frac{k_i k_j}{2m}) \delta(C_i, C_j) = \big[\sum_{i,j} A_{ij} - \sum_{i,j} \frac{k_i k_j}{2m} \big] \delta(C_i, C_j)$$

$$= \sum_{c \in C} \big[(\#\text{ edges within group } c)$$

$$- (\text{expected } \#\text{ edges within group } c) \big]$$

- Given real G on n vertices and m edges, construct a random network G';
  - Same degree distribution but random connections.

Intuition of modularity

A measure of how well a network is partitioned into communities.

- Given a partitioning of the network into groups $c \in C$:

$$Q \propto \sum_{i,j}(A_{ij} - \frac{k_i k_j}{2m})\delta(C_i, C_j) = \big[\sum_{i,j} A_{ij} - \sum_{i,j} \frac{k_i k_j}{2m}\big]\delta(C_i, C_j)$$

$$= \sum_{c \in C} \big[(\# \text{ edges within group } c)$$

$$- (\text{expected } \# \text{ edges within group } c)\big]$$

- Given real G on n vertices and m edges, construct a random network G';
  - Same degree distribution but random connections.
  - Consider G' as a multigraph.

Modularity: community VS. Null model

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

Modularity: community VS. Null model

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

Modularity can be viewed as a score that computes the difference between the observed structure of the network from an expected structure under a random "null" network.

Modularity: community VS. Null model

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

Modularity can be viewed as a score that computes the difference between the observed structure of the network from an expected structure under a random "null" network.

- The null network creates connections between vertices at random without any special structure of interest.

Modularity: community VS. Null model

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

Modularity can be viewed as a score that computes the difference between the observed structure of the network from an expected structure under a random "null" network.

- The null network creates connections between vertices at random without any special structure of interest.
- Compares the number of edges within a community with the expected such number in a corresponding random graph.

Modularity: community VS. Null model

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

Modularity can be viewed as a score that computes the difference between the observed structure of the network from an expected structure under a random "null" network.

- The null network creates connections between vertices at random without any special structure of interest.
- Compares the number of edges within a community with the expected such number in a corresponding random graph.
- The community structure is strong if this score, with a proper normalization, is high.

Modularity: community VS. Null model

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

Modularity can be viewed as a score that computes the difference between the observed structure of the network from an expected structure under a random "null" network.

- The null network creates connections between vertices at random without any special structure of interest.
- Compares the number of edges within a community with the expected such number in a corresponding random graph.
- The community structure is strong if this score, with a proper normalization, is high.
- It can be used as a measure to evaluate the communities quality.

Modularity computation

$$Q = \frac{1}{2m} \sum_{c \in C} \sum_{i \in c} \sum_{j \in c} (A_{ij} - \frac{k_i k_j}{2m})$$

Modularity computation

$$Q = \frac{1}{2m}\sum_{c \in C}\sum_{i \in c}\sum_{j \in c}(A_{ij} - \frac{k_i k_j}{2m})$$

- Modularity values take range $[-1, 1]$;

Modularity computation

$$Q = \frac{1}{2m} \sum_{c \in C} \sum_{i \in c} \sum_{j \in c} (A_{ij} - \frac{k_i k_j}{2m})$$

- Modularity values take range $[-1, 1]$;
- It is positive if the number of edges within groups exceeds the expected number;

Modularity computation

$$Q = \frac{1}{2m} \sum_{c \in C} \sum_{i \in c} \sum_{j \in c} (A_{ij} - \frac{k_i k_j}{2m})$$

- Modularity values take range $[-1, 1]$;
- It is positive if the number of edges within groups exceeds the expected number;
- M is greater than $0.3 - 0.7$ means significant community structure in the input network;

Modularity computation

$$Q = \frac{1}{2m} \sum_{c \in C} \sum_{i \in c} \sum_{j \in c} (A_{ij} - \frac{k_i k_j}{2m})$$

- Modularity values take range $[-1, 1]$;
- It is positive if the number of edges within groups exceeds the expected number;
- M is greater than $0.3 - 0.7$ means significant community structure in the input network;
- It can be worked as a metric to evaluate how well a community structure is.

Outline

Modularity for weighted networks

Definition

Modularity is one measure of the structure of networks or graphs.

Modularity for weighted networks

Definition

Modularity is one measure of the structure of networks or graphs. Given an undirected graph $G = (V, E)$, it's adjacency matrix is A associated with a weighted matrix W, where

$$W_{ij} = \begin{cases} w_{ij}, & \text{if } A_{ij} = 1; \\ 0, & \text{otherwise.} \end{cases}$$

Modularity for weighted networks

Definition

Modularity is one measure of the structure of networks or graphs. Given an undirected graph $G = (V, E)$, it's adjacency matrix is A associated with a weighted matrix W, where

$$W_{ij} = \begin{cases} w_{ij}, & \text{if } A_{ij} = 1; \\ 0, & \text{otherwise.} \end{cases}$$

Modularity can be defined as:

$$Q = \frac{1}{2m} \sum_{i,j} (W_{ij} - \frac{w_i w_j}{2m}) \delta(C_i, C_j),$$

where m and C_i denote total edge weights and the $i-$th community in the graph, w_i is the degree of vertex v_i, and

$$\delta(C_i, C_j) = \begin{cases} 1, & \text{if } C_i = C_j; \\ 0, & \text{otherwise.} \end{cases}$$

Modularity for directed networks

Definition

Modularity is one measure of the structure of networks or graphs.

Modularity for directed networks

Definition

Modularity is one measure of the structure of networks or graphs. Given a directed graph $G = (V, E)$, it's adjacency matrix is A, where

$$A_{ij} = \begin{cases} A_{ij}, & \text{if } (v_i, v_j) \in E; \\ 0, & \text{otherwise.} \end{cases}$$

Modularity for directed networks

Definition

Modularity is one measure of the structure of networks or graphs. Given a directed graph $G = (V, E)$, it's adjacency matrix is A, where

$$A_{ij} = \begin{cases} A_{ij}, & \text{if } (v_i, v_j) \in E; \\ 0, & \text{otherwise.} \end{cases}$$

Modularity can be defined as:

$$Q = \frac{1}{m} \sum_{i,j} (A_{ij} - \frac{k_i^{out} k_j^{in}}{m}) \delta(C_i, C_j),$$

where m and C_i denote total number of edges and the $i-$th community in the graph, k_i^{out} and k_i^{in} are the out-degree and in-degree of vertex v_i, and

$$\delta(C_i, C_j) = \begin{cases} 1, & \text{if } C_i = C_j; \\ 0, & \text{otherwise.} \end{cases}$$

Outline

Modularity Matrix

Suppose our network contains n vertices.

Modularity Matrix

Suppose our network contains n vertices.

Modularity Matrix

Suppose our network contains n vertices.

- For a particular division of the network into two groups let

$$s_i = \begin{cases} 1, & \text{if vertex } i \text{ belongs to group 1;} \\ -1, & \text{otherwise.} \end{cases}$$

Modularity Matrix

Suppose our network contains n vertices.

- For a particular division of the network into two groups let

$$s_i = \begin{cases} 1, & \text{if vertex } i \text{ belongs to group 1;} \\ -1, & \text{otherwise.} \end{cases}$$

- The modularity can be rewritten as

$$\begin{aligned} Q &= \frac{1}{4m} \sum_{i,j} (A_{ij} - \frac{k_i k_j}{2m})(s_i s_j + 1) \\ &= \frac{1}{4m} \sum_{i,j} (A_{ij} - \frac{k_i k_j}{2m}) s_i s_j = \frac{1}{4m} \mathbf{s}^T B \mathbf{s}, \end{aligned}$$

where $B_{ij} = A_{ij} - \frac{k_i k_j}{2m}$ is called modularity matrix.

Partitioning

Note that the elements of each of its rows and columns sum to zero

Partitioning

Note that the elements of each of its rows and columns sum to zero

- It always has an eigenvector $(1, 1, \cdots, 1)$ with eigenvalue zero;

Partitioning

Note that the elements of each of its rows and columns sum to zero

- It always has an eigenvector $(1, 1, \cdots, 1)$ with eigenvalue zero;
- This observation is reminiscent of the best-known methods of graph partitioning, spectral partitioning.

Partitioning

Note that the elements of each of its rows and columns sum to zero

- It always has an eigenvector $(1, 1, \cdots, 1)$ with eigenvalue zero;
- This observation is reminiscent of the best-known methods of graph partitioning, spectral partitioning.
- We proceed by writing $\mathbf{s}$ as a linear combination of the normalized eigenvectors $\mathbf{u}_i$ of B s.t., $\mathbf{s} = \sum_{i=1}^{n} a_i \mathbf{u}_i$ with $a_i = \mathbf{u}_i^T \cdot \mathbf{s}$. Then

$$Q = \frac{1}{4m} \sum_i a_i \mathbf{u}_i^T B \sum_j a_j \mathbf{u}_j = \frac{1}{4m} \sum_i^n (\mathbf{u}_i^T \cdot \mathbf{s})^2 \beta_i,$$

  where β_i is the eigenvalue of B corresponding to eigenvector $\mathbf{u}_i$.

Partitioning Cont'd

Let us assume that the eigenvalues are labeled in decreasing order, $\beta_1 \geq \beta_2 \geq \cdots \geq \beta_n$.

Partitioning Cont'd

Let us assume that the eigenvalues are labeled in decreasing order, $\beta_1 \geq \beta_2 \geq \cdots \geq \beta_n$.

- We want to maximize the modularity by choosing an appropriate division of the network, or equivalently by choosing the value of the index vector **s**, i.e., choosing **s** so as to concentrate as much weight as possible in terms of the sum in Eq. 4 involving the largest (most positive) eigenvalues;

Partitioning Cont'd

Let us assume that the eigenvalues are labeled in decreasing order, $\beta_1 \geq \beta_2 \geq \cdots \geq \beta_n$.

- We want to maximize the modularity by choosing an appropriate division of the network, or equivalently by choosing the value of the index vector $\mathbf{s}$, i.e., choosing $\mathbf{s}$ so as to concentrate as much weight as possible in terms of the sum in Eq. 4 involving the largest (most positive) eigenvalues;
- If there were no other constraints on $\mathbf{s}$, we would simply chose $\mathbf{s}$ proportional to the eigenvector $\mathbf{u}_1$ since the eigenvectors are orthogonal.

Partitioning Cont'd

Let us assume that the eigenvalues are labeled in decreasing order, $\beta_1 \geq \beta_2 \geq \cdots \geq \beta_n$.

- We want to maximize the modularity by choosing an appropriate division of the network, or equivalently by choosing the value of the index vector $\mathbf{s}$, i.e., choosing $\mathbf{s}$ so as to concentrate as much weight as possible in terms of the sum in Eq. 4 involving the largest (most positive) eigenvalues;
- If there were no other constraints on $\mathbf{s}$, we would simply chose $\mathbf{s}$ proportional to the eigenvector $\mathbf{u}_1$ since the eigenvectors are orthogonal.
- Unfortunately, there is another constraint on the problem imposed by the restriction of the elements of $\mathbf{s}$ to the values ± 1, which means $\mathbf{s}$ cannot normally be chosen parallel to $\mathbf{u}_1$.

Partitioning Cont'd

Let us do our best

Partitioning Cont'd

Let us do our best

- Make it as close to parallel as possible, which is equivalent to maximizing the dot product $\mathbf{u}_1^T \cdot \mathbf{s}$;

Partitioning Cont'd

Let us do our best

- Make it as close to parallel as possible, which is equivalent to maximizing the dot product $\mathbf{u}_1^T \cdot \mathbf{s}$;
- It is straightforward to see that the maximum is achieved by setting

Partitioning Cont'd

Let us do our best

- Make it as close to parallel as possible, which is equivalent to maximizing the dot product $\mathbf{u}_1^T \cdot \mathbf{s}$;
- It is straightforward to see that the maximum is achieved by setting

$$s_i = \begin{cases} 1, & \text{if } \mathbf{u}_{1i} > 0 \\ -1, & \text{otherwise.} \end{cases} \quad 1$$

  i.e., all vertices whose corresponding elements are positive go in one group and all of the rest in the other;

Partitioning Cont'd

Let us do our best

- Make it as close to parallel as possible, which is equivalent to maximizing the dot product $\mathbf{u}_1^T \cdot \mathbf{s}$;
- It is straightforward to see that the maximum is achieved by setting

$$s_i = \begin{cases} 1, & \text{if } \mathbf{u}_{1i} > 0 \\ -1, & \text{otherwise.} \end{cases} \quad 1$$

  i.e., all vertices whose corresponding elements are positive go in one group and all of the rest in the other;
- As a result, we compute the leading eigenvector of the modularity matrix and divide the vertices into two groups according to the signs of the elements in this vector.

Outline

Extension to multiple community partitioning

However, contain more than two communities, so we would like to extend the method to find good divisions of networks into larger numbers of parts.

Extension to multiple community partitioning

However, contain more than two communities, so we would like to extend the method to find good divisions of networks into larger numbers of parts.

- The standard approach to this problem, and the one adopted here, is repeated division into two;

Extension to multiple community partitioning

However, contain more than two communities, so we would like to extend the method to find good divisions of networks into larger numbers of parts.

- The standard approach to this problem, and the one adopted here, is repeated division into two;
- We use the algorithm of the previous section first to divide the network into two parts, then divide those parts, and so forth. That is, after first dividing a network in two, to simply delete the edges falling between the two parts and then apply the algorithm again to each subgraph

Extension to multiple community partitioning

However, contain more than two communities, so we would like to extend the method to find good divisions of networks into larger numbers of parts.

- The standard approach to this problem, and the one adopted here, is repeated division into two;
- We use the algorithm of the previous section first to divide the network into two parts, then divide those parts, and so forth. That is, after first dividing a network in two, to simply delete the edges falling between the two parts and then apply the algorithm again to each subgraph
- In doing this it is crucial to note that it is not correct.

Extension to multiple community partitioning

However, contain more than two communities, so we would like to extend the method to find good divisions of networks into larger numbers of parts.

- The standard approach to this problem, and the one adopted here, is repeated division into two;
- We use the algorithm of the previous section first to divide the network into two parts, then divide those parts, and so forth. That is, after first dividing a network in two, to simply delete the edges falling between the two parts and then apply the algorithm again to each subgraph
- In doing this it is crucial to note that it is not correct.
  - The modularity will change if edges are deleted;

Extension to multiple community partitioning

However, contain more than two communities, so we would like to extend the method to find good divisions of networks into larger numbers of parts.

- The standard approach to this problem, and the one adopted here, is repeated division into two;
- We use the algorithm of the previous section first to divide the network into two parts, then divide those parts, and so forth. That is, after first dividing a network in two, to simply delete the edges falling between the two parts and then apply the algorithm again to each subgraph
- In doing this it is crucial to note that it is not correct.
  - The modularity will change if edges are deleted;
  - Any subsequent maximization of modularity would thus maximize the wrong quantity.

Extension Cont'd

Instead, the correct approach is to write the additional contribution $\triangle Q$ to the modularity upon further dividing a group g of size n_g in two as

$$\begin{aligned}
\triangle Q &= \frac{1}{2m}\Big[\frac{1}{2}\sum_{i,j\in g} B_{ij}(s_i s_j + 1) - \sum_{i,j\in g} B_{ij}\Big] \\
&= \frac{1}{4m}\Big[\sum_{i,j\in g} B_{ij} s_i s_j - \sum_{i,j\in g} B_{ij}\Big] \\
&= \frac{1}{4m}\sum_{i,j\in g}\Big[B_{ij} - \delta_{ij}\sum_{k\in g} B_{ik}\Big] s_i s_j \\
&= \frac{1}{4m}\sum_{i,j\in g}\Big[B_{ij} - \delta_{ij}\sum_{k\in g} B_{ik}\Big] s_i s_j = \frac{1}{4m}\mathbf{s}^T B^{(g)}\mathbf{s}
\end{aligned}$$

where $B_{ij}^{(g)} = B_{ij} - \delta_{ij}\sum_{k\in g} B_{ik}$ is the new modularity matrix.

Outline

Louvain method

Greedy algorithm for community detection

- Supports directed and weighted graphs;

Louvain method

Greedy algorithm for community detection

- Supports directed and weighted graphs;
- Provides hierarchical partitions;

Louvain method

Greedy algorithm for community detection

- Supports directed and weighted graphs;
- Provides hierarchical partitions;
- The number of communities is not a hyper-parameter;

Louvain method

Greedy algorithm for community detection

- Supports directed and weighted graphs;
- Provides hierarchical partitions;
- The number of communities is not a hyper-parameter;
- Widely utilized to study large networks because;

Louvain method

Greedy algorithm for community detection

- Supports directed and weighted graphs;
- Provides hierarchical partitions;
- The number of communities is not a hyper-parameter;
- Widely utilized to study large networks because;
  - Fast: running time is only $O(|E|)$;

Louvain method

Greedy algorithm for community detection

- Supports directed and weighted graphs;
- Provides hierarchical partitions;
- The number of communities is not a hyper-parameter;
- Widely utilized to study large networks because;
  - Fast: running time is only $O(|E|)$;
  - Rapid convergence properties;

Louvain method

Greedy algorithm for community detection

- Supports directed and weighted graphs;
- Provides hierarchical partitions;
- The number of communities is not a hyper-parameter;
- Widely utilized to study large networks because;
  - Fast: running time is only $O(|E|)$;
  - Rapid convergence properties;
  - High modularity output (i.e., "better communities").

Louvain algorithm greedily maximizes modularity

Louvain method

Greedy algorithm for community detection

- Supports directed and weighted graphs;
- Provides hierarchical partitions;
- The number of communities is not a hyper-parameter;
- Widely utilized to study large networks because;
  - Fast: running time is only $O(|E|)$;
  - Rapid convergence properties;
  - High modularity output (i.e., "better communities").

Louvain algorithm greedily maximizes modularity

- Each pass is made of 2 phases;

Louvain method

Greedy algorithm for community detection

- Supports directed and weighted graphs;
- Provides hierarchical partitions;
- The number of communities is not a hyper-parameter;
- Widely utilized to study large networks because;
  - Fast: running time is only $O(|E|)$;
  - Rapid convergence properties;
  - High modularity output (i.e., "better communities").

Louvain algorithm greedily maximizes modularity

- Each pass is made of 2 phases;
- The passes are repeated iteratively until no increase of modularity is possible!

Modularity rewriting

$$
\begin{aligned}
M &= \frac{1}{2m}\sum_{i,j}(A_{ij} - \frac{k_i k_j}{2m})\delta(C_i, C_j) \\
&= [\sum_{i,j}\frac{A_{ij}}{2m} - \frac{\sum_i k_i \sum_j k_j}{4m^2}]\delta(C_i, C_j) \\
&= \sum_{c \in C}[\frac{\sum_{in}^{c}}{2m} - (\frac{\sum_{tot}^{c}}{2m})^2]
\end{aligned}
$$

Modularity rewriting

$$
\begin{aligned}
M &= \frac{1}{2m}\sum_{i,j}(A_{ij} - \frac{k_i k_j}{2m})\delta(C_i, C_j) \\
&= \Big[\sum_{i,j}\frac{A_{ij}}{2m} - \frac{\sum_i k_i \sum_j k_j}{4m^2}\Big]\delta(C_i, C_j) \\
&= \sum_{c \in C}\Big[\frac{\sum_{in}^{c}}{2m} - \Big(\frac{\sum_{tot}^{c}}{2m}\Big)^2\Big]
\end{aligned}
$$

- $\sum_{in}^{c}$ is the sum of link weights between vertices in c;

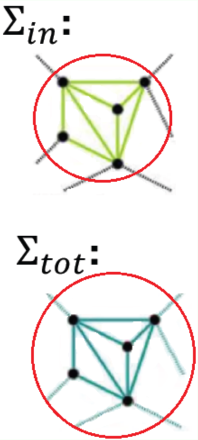
Modularity rewriting

$$M = \frac{1}{2m}\sum_{i,j}(A_{ij} - \frac{k_i k_j}{2m})\delta(C_i, C_j)$$

$$= [\sum_{i,j}\frac{A_{ij}}{2m} - \frac{\sum_i k_i \sum_j k_j}{4m^2}]\delta(C_i, C_j)$$

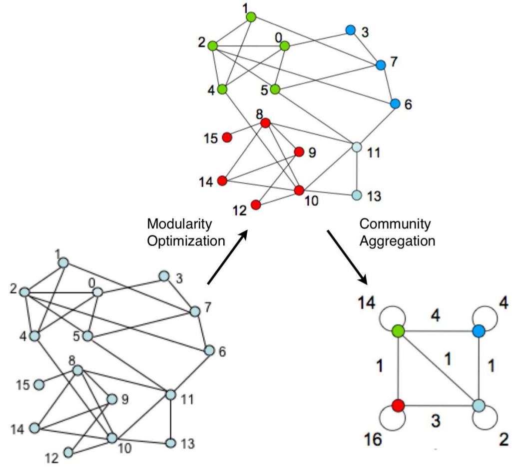
$$= \sum_{c \in C}[\frac{\sum_{in}^{c}}{2m} - (\frac{\sum_{tot}^{c}}{2m})^2]$$

- $\sum_{in}^{c}$ is the sum of link weights between vertices in c;
- $\sum_{tot}^{c}$ is the sum of all link weights of vertices in c;



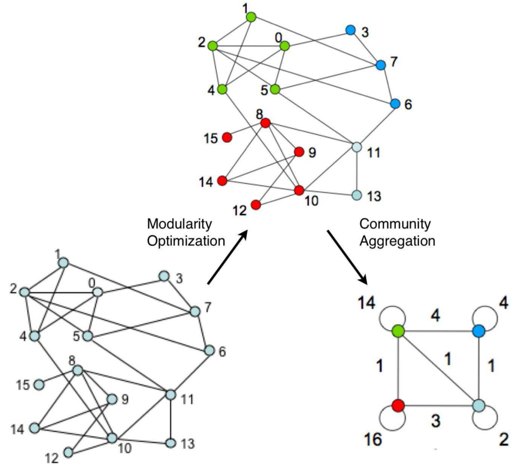
Outline

Louvain method Cont'd



Each iteration

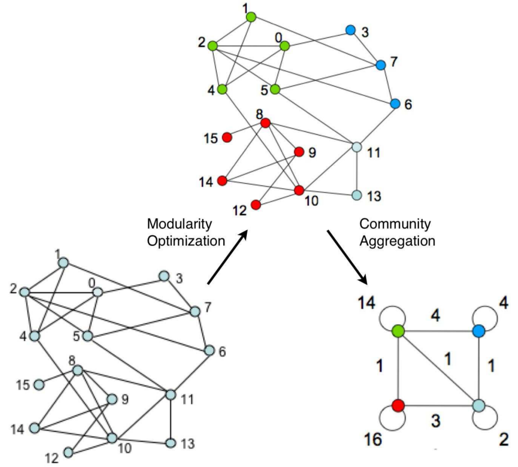
Louvain method Cont'd



Each iteration

- Phase 1: Modularity is optimized by allowing only local changes of communities;

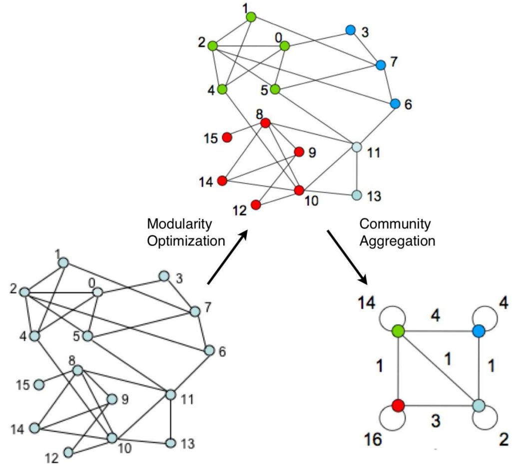
Louvain method Cont'd



Each iteration

- Phase 1: Modularity is optimized by allowing only local changes of communities;
- Phase 2: The identified communities are aggregated in order to build a new network of communities;

Louvain method Cont'd



Each iteration

- Phase 1: Modularity is optimized by allowing only local changes of communities;
- Phase 2: The identified communities are aggregated in order to build a new network of communities;
- Goto Phase 1.

Louvain: Phase 1 (Partitioning)

- Put each vertex in a graph into a distinct community (one vertex per community);

Louvain: Phase 1 (Partitioning)

- Put each vertex in a graph into a distinct community (one vertex per community);
- For each vertex v_i, the algorithm performs two calculations:

Louvain: Phase 1 (Partitioning)

- Put each vertex in a graph into a distinct community (one vertex per community);
- For each vertex v_i, the algorithm performs two calculations:
  - Compute the modularity gain ($\triangle Q$) when putting vertex v_i from its current community into the community of some neighbor v_j of v_i;

Louvain: Phase 1 (Partitioning)

- Put each vertex in a graph into a distinct community (one vertex per community);
- For each vertex v_i, the algorithm performs two calculations:
  - Compute the modularity gain ($\triangle Q$) when putting vertex v_i from its current community into the community of some neighbor v_j of v_i;
  - Move v_i to a community that yields the largest modularity gain $\triangle Q$;

Louvain: Phase 1 (Partitioning)

- Put each vertex in a graph into a distinct community (one vertex per community);
- For each vertex v_i, the algorithm performs two calculations:
  - Compute the modularity gain ($\triangle Q$) when putting vertex v_i from its current community into the community of some neighbor v_j of v_i;
  - Move v_i to a community that yields the largest modularity gain $\triangle Q$;
  - The loop runs until no movement yields a gain.

Louvain: Phase 1 (Partitioning)

- Put each vertex in a graph into a distinct community (one vertex per community);
- For each vertex v_i, the algorithm performs two calculations:
  - Compute the modularity gain ($\triangle Q$) when putting vertex v_i from its current community into the community of some neighbor v_j of v_i;
  - Move v_i to a community that yields the largest modularity gain $\triangle Q$;
  - The loop runs until no movement yields a gain.

This first phase stops when a local maxima of the modularity is attained, i.e., when no individual move can improve the modularity.

Louvain: Phase 1 (Partitioning)

- Put each vertex in a graph into a distinct community (one vertex per community);
- For each vertex v_i, the algorithm performs two calculations:
  - Compute the modularity gain ($\triangle Q$) when putting vertex v_i from its current community into the community of some neighbor v_j of v_i;
  - Move v_i to a community that yields the largest modularity gain $\triangle Q$;
  - The loop runs until no movement yields a gain.

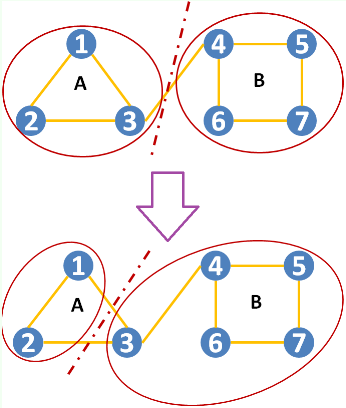
This first phase stops when a local maxima of the modularity is attained, i.e., when no individual move can improve the modularity. One should also note that the output of the algorithm depends on the order in which the vertices are considered.

Louvain: Phase 1 (Partitioning)

- Put each vertex in a graph into a distinct community (one vertex per community);
- For each vertex v_i, the algorithm performs two calculations:
  - Compute the modularity gain ($\triangle Q$) when putting vertex v_i from its current community into the community of some neighbor v_j of v_i;
  - Move v_i to a community that yields the largest modularity gain $\triangle Q$;
  - The loop runs until no movement yields a gain.

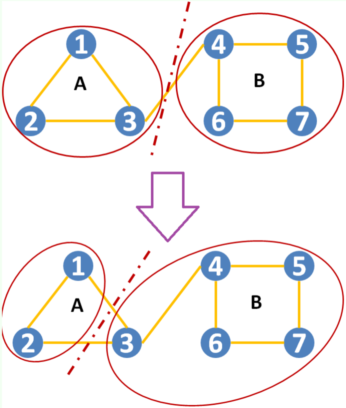
This first phase stops when a local maxima of the modularity is attained, i.e., when no individual move can improve the modularity. One should also note that the output of the algorithm depends on the order in which the vertices are considered. Research indicates that the ordering of the vertices does not have a significant influence on the modularity.

Modularity gain $\triangle Q$



Modularity change is two parts

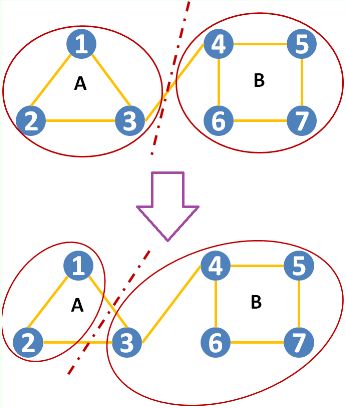
Modularity gain $\triangle Q$



Modularity change is two parts

- Gain: $\triangle Q(v_i \rightarrow C)$ means the gain of Q if we move vertex v_i to community C;

Modularity gain $\triangle Q$



Modularity change is two parts

- Gain: $\triangle Q(v_i \rightarrow C)$ means the gain of Q if we move vertex v_i to community C;
- Loss: $\triangle Q(D \rightarrow v_i)$ means the loss of Q if we take vertex v_i out of community D.

Computing $\triangle Q(v_i \to C)$

What is $\triangle Q$ if we move vertex v_i to community C?

Computing $\triangle Q(v_i \to C)$

What is $\triangle Q$ if we move vertex v_i to community C?

$$
\begin{aligned}
\triangle Q(v_i \to C) &= \Big[\frac{\sum_{in}^{C} + k_{i,in}^{C}}{2m} - \big(\frac{\sum_{tot}^{C} + k_i}{2m}\big)^2\Big] \\
&\quad - \Big[\frac{\sum_{in}^{C}}{2m} - \big(\frac{\sum_{tot}^{C}}{2m}\big)^2 - \big(\frac{k_i}{2m}\big)^2\Big] \\
&= \Big[\frac{k_{i,in}^{C}}{2m} - \frac{\sum_{tot}^{C} \cdot k_i}{2m^2}\Big]
\end{aligned}
$$

Computing $\triangle Q(v_i \to C)$

What is $\triangle Q$ if we move vertex v_i to community C?

$$\begin{aligned}
\triangle Q(v_i \to C) &= \Big[\frac{\sum_{in}^{C} + k_{i,in}^{C}}{2m} - \big(\frac{\sum_{tot}^{C} + k_i}{2m}\big)^2\Big] \\
&\quad - \Big[\frac{\sum_{in}^{C}}{2m} - \big(\frac{\sum_{tot}^{C}}{2m}\big)^2 - \big(\frac{k_i}{2m}\big)^2\Big] \\
&= \Big[\frac{k_{i,in}^{C}}{2m} - \frac{\sum_{tot}^{C} \cdot k_i}{2m^2}\Big]
\end{aligned}$$

- $\sum_{in}^{C}$ is the sum of link weights between vertices in C;

Computing $\triangle Q(v_i \to C)$

What is $\triangle Q$ if we move vertex v_i to community C?

$$\begin{aligned}
\triangle Q(v_i \to C) = &\Big[\frac{\sum_{in}^{C} + k_{i,in}^{C}}{2m} - \Big(\frac{\sum_{tot}^{C} + k_i}{2m}\Big)^2\Big] \\
&- \Big[\frac{\sum_{in}^{C}}{2m} - \Big(\frac{\sum_{tot}^{C}}{2m}\Big)^2 - \Big(\frac{k_i}{2m}\Big)^2\Big] \\
= &\Big[\frac{k_{i,in}^{C}}{2m} - \frac{\sum_{tot}^{C} \cdot k_i}{2m^2}\Big]
\end{aligned}$$

- $\sum_{in}^{C}$ is the sum of link weights between vertices in C;
- $\sum_{tot}^{C}$ is the sum of all link weights of vertices in C;

Computing $\triangle Q(v_i \to C)$

What is $\triangle Q$ if we move vertex v_i to community C?

$$
\begin{aligned}
\triangle Q(v_i \to C) &= \Big[\frac{\sum_{in}^{C} + k_{i,in}^{C}}{2m} - \big(\frac{\sum_{tot}^{C} + k_i}{2m}\big)^2\Big] \\
&\quad - \Big[\frac{\sum_{in}^{C}}{2m} - \big(\frac{\sum_{tot}^{C}}{2m}\big)^2 - \big(\frac{k_i}{2m}\big)^2\Big] \\
&= \Big[\frac{k_{i,in}^{C}}{2m} - \frac{\sum_{tot}^{C} \cdot k_i}{2m^2}\Big]
\end{aligned}
$$

- $\sum_{in}^{C}$ is the sum of link weights between vertices in C;
- $\sum_{tot}^{C}$ is the sum of all link weights of vertices in C;
- $k_{i,in}^{C}$ is the sum of link weights between vertex v_i and C;

Computing $\triangle Q(v_i \to C)$

What is $\triangle Q$ if we move vertex v_i to community C?

$$\triangle Q(v_i \to C) = \Big[\frac{\sum_{in}^{C} + k_{i,in}^{C}}{2m} - \big(\frac{\sum_{tot}^{C} + k_i}{2m}\big)^2\Big]$$
$$- \Big[\frac{\sum_{in}^{C}}{2m} - \big(\frac{\sum_{tot}^{C}}{2m}\big)^2 - \big(\frac{k_i}{2m}\big)^2\Big]$$
$$= \Big[\frac{k_{i,in}^{C}}{2m} - \frac{\sum_{tot}^{C} \cdot k_i}{2m^2}\Big]$$

- $\sum_{in}^{C}$ is the sum of link weights between vertices in C;
- $\sum_{tot}^{C}$ is the sum of all link weights of vertices in C;
- $k_{i,in}^{C}$ is the sum of link weights between vertex v_i and C;
- k_i is the sum of all link weights (i.e., degree) of vertex v_i;

Computing $\triangle Q(D \to v_i)$

What is $\triangle Q$ if we take node v_i out of community D, i.e., $\triangle Q(D \to v_i)$?

Computing $\triangle Q(D \to v_i)$

What is $\triangle Q$ if we take node v_i out of community D, i.e., $\triangle Q(D \to v_i)$?

- Let community D' be the community after taking v_i out of community D;

Computing $\triangle Q(D \to v_i)$

What is $\triangle Q$ if we take node v_i out of community D, i.e., $\triangle Q(D \to v_i)$?

- Let community D' be the community after taking v_i out of community D;
- Thus, we have

$$\triangle Q(D \to v_i) = -\triangle Q(v_i \to D');$$

Computing $\triangle Q(D \rightarrow v_i)$

What is $\triangle Q$ if we take node v_i out of community D, i.e., $\triangle Q(D \rightarrow v_i)$?

- Let community D' be the community after taking v_i out of community D;
- Thus, we have

$$\triangle Q(D \rightarrow v_i) = -\triangle Q(v_i \rightarrow D');$$

- Furthermore,

$$\begin{aligned}\triangle Q(D \rightarrow v_i) &= -\triangle Q(v_i \rightarrow D') \\ &= \frac{\sum_{tot}^{D'} \cdot k_i}{2m^2} - \frac{k_{i,in}^{D'}}{2m}\end{aligned}$$

Computing $\triangle Q(D \rightarrow v_i)$

What is $\triangle Q$ if we take node v_i out of community D, i.e., $\triangle Q(D \rightarrow v_i)$?

- Let community D' be the community after taking v_i out of community D;
- Thus, we have

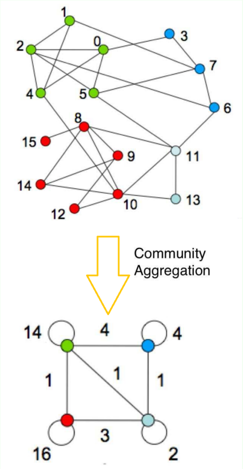
$$\triangle Q(D \rightarrow v_i) = -\triangle Q(v_i \rightarrow D');$$

- Furthermore,

$$\begin{aligned} \triangle Q(D \rightarrow v_i) &= -\triangle Q(v_i \rightarrow D') \\ &= \frac{\sum_{tot}^{D'} \cdot k_i}{2m^2} - \frac{k_{i,in}^{D'}}{2m} \end{aligned}$$

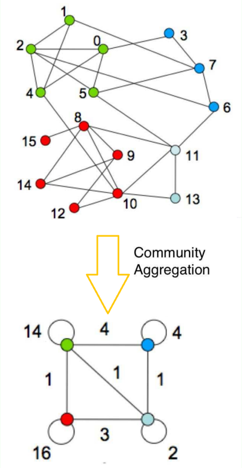
Finally, $\triangle Q = \triangle Q(v_i \rightarrow C) + \triangle Q(D \rightarrow v_i)$.

Louvain: Phase 2 (Reconstructing)



The partitions obtained in the first phase are contracted into super-nodes, and the weighted network is created as follows

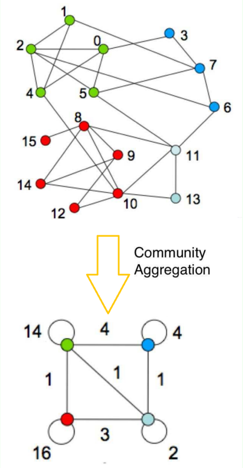
Louvain: Phase 2 (Reconstructing)



The partitions obtained in the first phase are contracted into super-nodes, and the weighted network is created as follows

- Super-nodes are connected if there is at least one edge between vertices of the corresponding communities;

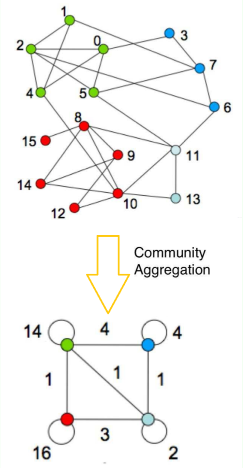
Louvain: Phase 2 (Reconstructing)



The partitions obtained in the first phase are contracted into super-nodes, and the weighted network is created as follows

- Super-nodes are connected if there is at least one edge between vertices of the corresponding communities;
- The weight of the edge between the two supernodes is the sum of the weights from all edges between their corresponding partitions;

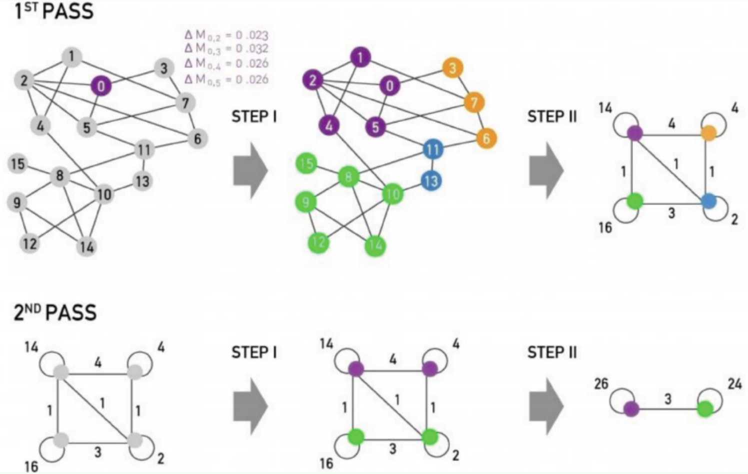
Louvain: Phase 2 (Reconstructing)



The partitions obtained in the first phase are contracted into super-nodes, and the weighted network is created as follows

- Super-nodes are connected if there is at least one edge between vertices of the corresponding communities;
- The weight of the edge between the two supernodes is the sum of the weights from all edges between their corresponding partitions;
- After aggregation, the graph becomes a weighted graph.

Louvain method



Outline

Prons of Louvain method

Prons of Louvain method:

Prons of Louvain method

Prons of Louvain method:

- This algorithm is extremely fast since the number of communities reduces drastically after the first pass, making the amount of computations less in the later passes.

Prons of Louvain method

Prons of Louvain method:

- This algorithm is extremely fast since the number of communities reduces drastically after the first pass, making the amount of computations less in the later passes.
- Also the gains in modularity are very easy to compute;

Prons of Louvain method

Prons of Louvain method:

- This algorithm is extremely fast since the number of communities reduces drastically after the first pass, making the amount of computations less in the later passes.
- Also the gains in modularity are very easy to compute;
- The time complexity of this algorithm is $O(|E|)$, e.g., the time is less than 1 minutes for finding the community from a graph of 1 million vertices.

Prons of Louvain method

Prons of Louvain method:

- This algorithm is extremely fast since the number of communities reduces drastically after the first pass, making the amount of computations less in the later passes.
- Also the gains in modularity are very easy to compute;
- The time complexity of this algorithm is $O(|E|)$, e.g., the time is less than 1 minutes for finding the community from a graph of 1 million vertices.
- The number of communities is not a hyper-parameter.

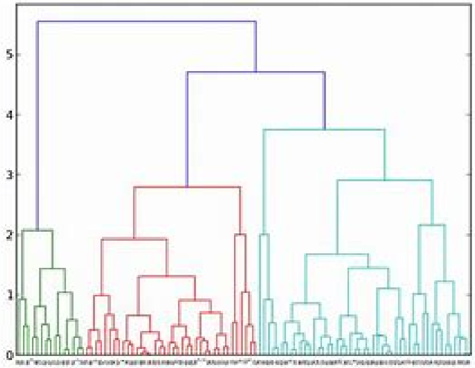
Prons of Louvain method

Prons of Louvain method:

- This algorithm is extremely fast since the number of communities reduces drastically after the first pass, making the amount of computations less in the later passes.
- Also the gains in modularity are very easy to compute;
- The time complexity of this algorithm is $O(|E|)$, e.g., the time is less than 1 minutes for finding the community from a graph of 1 million vertices.
- The number of communities is not a hyper-parameter.
- It can be used to evaluate the quality of community structure;

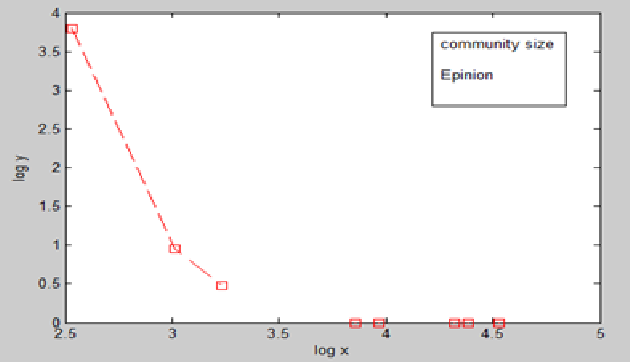
Prons of Louvain method Cont'd

Hierarchical partitions



Cons of Louvain method

The sizes of communities follow the rule of Power-law.



One of the major drawbacks of the Louvain algorithm is resolution limit.

Experimental result

	Karate	Arxiv	Internet	Web nd.edu
Nodes/links	34/77	9k/24k	70k/351k	325k/1M
CNM	.38/0s	.772/3.6s	.692/799s	.927/5034s
PL	.42/0s	.757/3.3s	.729/575s	.895/6666s
WT	.42/0s	.761/0.7s	.667/62s	.898/248s
Our algorithm	.42/0s	.813/0s	.781/1s	.935/3s

	Phone	Web uk-2005	Web WebBase 2001
Nodes/links	2.6M/6.3M	39M/783M	118M/1B
CNM	-/-	-/-	-/-
PL	-/-	-/-	-/-
WT	.56/464s	-/-	-/-
Our algorithm	.769/134s	.979/738s	.984/152mn

Take-home messages

- Motivation
- Modularity
  - Graph Model
  - Definition
  - Variants
- Modularity Matrix
  - Two Communities
  - Multiple Community Partitioning
- Louvain Method
  - Introduction
  - Algorithm
  - Analysis