Script generated by TTT

Title: profile1 (11.06.2013)

Date: Tue Jun 11 11:59:36 CEST 2013

Duration: 87:08 min

Pages: 67

Conductance

- Clustering paradigm reformulated: Clusters should be well connected (many edges need to be removed to make it unconnected); few inter cluster edges (ideally none)
- Conductance: Measure for bottlenecks (Bottleneck: Cut that separates V into roughly same size halves and “cuts across” relatively few edges)
- Let **C**={C_1, V \ C_1} be a cut. Conductance φ of **C** is defined as

$$\varphi(\mathbf{C}) = \begin{cases} 1 & \text{if } C\_1 \in \{\emptyset, V\} \\ 0 & \text{if } C\_1 \notin \{\emptyset, V\},\ w(\overline{E(C)})=0 \\ \dfrac{w(\overline{E(\mathbf{C})})}{\min(\sum_{e\in E(C\_1, V)} w(e), \sum_{e\in E(V\setminus C\_1, V)} w(e))} & \text{otherwise} \end{cases}$$

the smaller φ(**C**), the more „bottlenecky“ is **C**

Conductance

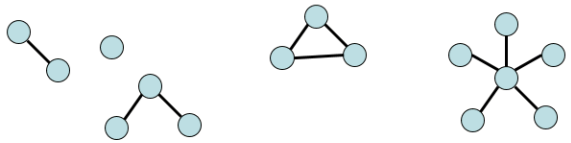
- Clustering paradigm reformulated: Clusters should be well connected (many edges need to be removed to make it unconnected); few inter cluster edges (ideally none)
- Conductance: Measure for bottlenecks (Bottleneck: Cut that separates V into roughly same size halves and “cuts across” relatively few edges)
- Let **C**={C_1, V \ C_1} be a cut. Conductance φ of **C** is defined as

$$\varphi(\mathbf{C}) = \begin{cases} 1 & \text{if } C\_1 \in \{\emptyset, V\} \\ 0 & \text{if } C\_1 \notin \{\emptyset, V\},\ w(\overline{E(C)})=0 \\ \dfrac{w(\overline{E(\mathbf{C})})}{\min(\sum_{e\in E(C\_1, V)} w(e), \sum_{e\in E(V\setminus C\_1, V)} w(e))} & \text{otherwise} \end{cases}$$

the smaller φ(**C**), the more „bottlenecky“ is **C**

Conductance

- Theorem: If G is undirected and positively weighted, G has maximum conductance φ(G)=1 iff G is connected and has at most three nodes or is a star. (Proof: see [1])



- Proof `←‘ $\sum_{e\in E(C\_1, V)} w(e) = w(E(C\_1)) + w(\overline{E(\mathbf{C})})$ →

$$\frac{w(\overline{E(\mathbf{C})})}{\min(\sum_{e\in E(C\_1, V)} w(e), \sum_{e\in E(V\setminus C\_1, V)} w(e))} =$$

$$\frac{w(\overline{E(\mathbf{C})})}{w(\overline{E(\mathbf{C})}) + \min(w(E(C\_1)), w(E(V\setminus C\_1)))} = 1$$

$=0$ if star or at most 3 nodes

Conductance

- With conductance we can define two appropriate quality measures for clusterings:
- First measure: g=0 and $f(\mathbf{C}) = \min_{1 \le i \le k} \varphi(G[\text{C\_i}])$
- If the measure is small: At least one of the clusters (more precisely: the induced subgraph) contains at least one bottleneck → This cluster is too coarse →Use minimum conductance cut to cut this cluster in "halves"
- From theorem before: Only clusterings where the clusters induce subgraphs that are stars or have size at most three have f=1 (f is called *intra cluster conductance*)

Conductance

- With conductance we can define two appropriate quality measures for clusterings:
- First measure: g=0 and $f(\mathbf{C}) = \min_{1 \le i \le k} \varphi(G[\text{C\_i}])$
- If the measure is small: At least one of the clusters (more precisely: the induced subgraph) contains at least one bottleneck → This cluster is too coarse →Use minimum conductance cut to cut this cluster in "halves"
- From theorem before: Only clusterings where the clusters induce subgraphs that are stars or have size at most three have f=1 (f is called *intra cluster conductance*)

Conductance

- Second measure: f=0 and

$$g(\mathbf{C}) = \begin{cases} 1 & \text{if } \mathbf{C} = \{V\} \\ 1 - \max_{1 \le i \le k} \varphi(\text{C\_i}, V \setminus \text{C\_i}) & \text{otherwise} \end{cases}$$

- If the measure is small: At least one of the clusters (more precisely: the induced subgraph) has many connections to outside → The clustering is too fine →Merge clusters
- From theorem before: Only clusterings that have inter cluster edge weight zero have g=1 (g is called *inter cluster conductance*)

Performance

- Main idea: Clustering paradigm → Count "correctly classified pairs of nodes". A pair of nodes is correctly classified if:
  - It is in the same cluster AND connected by an edge → f counts the number of edges within clusters
  - If it is not in the same cluster AND not connected by an edge →g counts the number of non-existent edges between clusters

$$f(\mathbf{C}) = \sum_{i=1}^{k} |E(\text{C\_i})|$$

$$g(\mathbf{C}) = \sum_{u,v \in V} [(u,v) \notin E] * [u \in \text{C\_i}, v \in \text{C\_j}, i \neq j]$$

Iverson-notation: [L]=1 if L is true

Performance

- Main idea: Clustering paradigm → Count "correctly classified pairs of nodes". A pair of nodes is correctly classified if:
  - It is in the same cluster AND connected by an edge → f counts the number of edges within clusters
  - If it is not in the same cluster AND not connected by an edge → g counts the number of non-existent edges between clusters

$$f(\mathbf{C}) = \sum_{i=1}^{k} |E(\text{C\_i})|$$

$$g(\mathbf{C}) = \sum_{u,v \in V} [(u,v) \notin E] * [u \in \text{C\_i}, v \in \text{C\_j}, i \neq j]$$

Iverson-notation: [L]=1 if L is true

Performance

- Main idea: Clustering paradigm → Count "correctly classified pairs of nodes". A pair of nodes is correctly classified if:
  - It is in the same cluster AND connected by an edge → f counts the number of edges within clusters
  - If it is not in the same cluster AND not connected by an edge → g counts the number of non-existent edges between clusters

$$f(\mathbf{C}) = \sum_{i=1}^{k} |E(\text{C\_i})|$$

$$g(\mathbf{C}) = \sum_{u,v \in V} [(u,v) \notin E] * [u \in \text{C\_i}, v \in \text{C\_j}, i \neq j]$$

Iverson-notation: [L]=1 if L is true

Performance

- If using weighted edges → some modifications:
  - For the denominator, we need a maximum for the edge weights; Take the max weight in G → Clusterings over different graphs are not comparable in quality. Better: Use weights normalized to 1 → Max weight M = 1

$$f(\mathbf{C}) = \sum_{i=1}^{k} w(E(\text{C\_i}))$$

$$g(\mathbf{C}) = \sum_{u,v \in V} M * [(u,v) \notin E] * [u \in \text{C\_i}, v \in \text{C\_j}, i \neq j]$$

Performance

- In that version g neglects the individual inter-cluster edges → Introduce g_w

$$g'(\mathbf{C}) = g(\mathbf{C}) + \underbrace{M \,|\, \overline{E(\mathbf{C})} \,| - w(\overline{E(\mathbf{C})})}_{g_w(\mathbf{C})}$$

- Overall index is then:

$$\text{perf}_w(\mathbf{C}) = \frac{f(\mathbf{C}) + g(\mathbf{C}) + \vartheta\, g_w(\mathbf{C})}{M(|V|(|V|-1)}$$

- other possibility: minimize incorrectly classified edges (dual problem)

Performance

- In that version g neglects the individual inter-cluster edges → Introduce g_w

$$g'(\mathbf{C}) = g(\mathbf{C}) + \underbrace{M \mid \overline{E(\mathbf{C})} \mid -w(\overline{E(\mathbf{C})})}_{g_w(\mathbf{C})}$$

- Overall index is then:

$$\text{perf}_w(\mathbf{C}) = \frac{f(\mathbf{C}) + g(\mathbf{C}) + \vartheta\, g_w(\mathbf{C})}{M(\mid V \mid (\mid V \mid -1)}$$

- other possibility: minimize incorrectly classified edges (dual problem)

Performance

- In that version g neglects the individual inter-cluster edges → Introduce g_w

$$g'(\mathbf{C}) = g(\mathbf{C}) + \underbrace{M \mid \overline{E(\mathbf{C})} \mid -w(\overline{E(\mathbf{C})})}_{g_w(\mathbf{C})}$$

- Overall index is then:

$$\text{perf}_w(\mathbf{C}) = \frac{f(\mathbf{C}) + g(\mathbf{C}) + \vartheta\, g_w(\mathbf{C})}{M(\mid V \mid (\mid V \mid -1)}$$

- other possibility: minimize incorrectly classified edges (dual problem)

Performance

- Calculating the maximum of f+g is NP-hard (In fact calculating the maximum of f+g would in essence be calculating the optimal clustering)
- → Since there are 1/2 |V| (|V|-1) node pairs →upper bound for f+g is |V| (|V|-1) →use |V| (|V|-1) as denominator in quality measure
- The performance index is then:

$$\text{perf}(\mathbf{C}) = \frac{f(\mathbf{C}) + g(\mathbf{C})}{\mid V \mid (\mid V \mid -1)}$$

- Problems with Performance: when graph is sparse (example: planar graphs: |E| is linear in |V|). Tendency: Performance delivers many small clusters

Other Indices

- If density measure π on graphs is available:

$$\text{worst case: } \min_i \{\pi(G[C_1]), \ldots, \pi(G[C_k])\}$$

$$\text{average case: } \frac{1}{k}\sum_i \pi(G[C_i])$$

$$\text{best case: } \max_i \{\pi(G[C_1]), \ldots, \pi(G[C_k])\}$$

- (especially suitable in metric spaces)

Graph Clustering Algorithms

- What have we seen so far? Measures for cluster quality
- But how do we compute such clusters?
- First group of methods: Greedy approaches

greedy minimization:

```
let L0 be a feasible solution;
i ←0;
while({L | L∈N(Li), c(L)<c(Li)} ≠ Ø) {
        Li+1 ← argminL∈N(Li) c(L);
        i ← i+1;
}
```

Space of all solutions L that can be constructed from solution L_i (annotating $N(L_i)$)

c(L) is the cost of solution L (annotating $c(L)$)

Graph Clustering Algorithms

- What have we seen so far? Measures for cluster quality
- But how do we compute such clusters?
- First group of methods: Greedy approaches

greedy minimization:

```
let L0 be a feasible solution;
i ←0;
while({L | L∈N(Li), c(L)<c(Li)} ≠ Ø) {
        Li+1 ← argminL∈N(Li) c(L);
        i ← i+1;
}
```

Space of all solutions L that can be constructed from solution L_i

c(L) is the cost of solution L

Graph Clustering Algorithms

- What have we seen so far? Measures for cluster quality
- But how do we compute such clusters?
- First group of methods: Greedy approaches

greedy minimization:

```
let L0 be a feasible solution;
i ←0;
while({L | L∈N(Li), c(L)<c(Li)} ≠ Ø) {
        Li+1 ← argminL∈N(Li) c(L);
        i ← i+1;
}
```

Space of all solutions L that can be constructed from solution L_i

c(L) is the cost of solution L

Graph Clustering Algorithms

- What have we seen so far? Measures for cluster quality
- But how do we compute such clusters?
- First group of methods: Greedy approaches

greedy minimization:

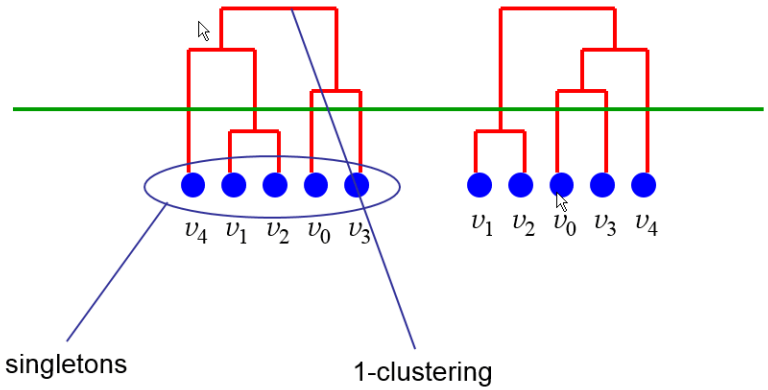
```
let L0 be a feasible solution;
i ←0;
while({L | L∈N(Li), c(L)<c(Li)} ≠ Ø) {
        Li+1 ← argminL∈N(Li) c(L);
        i ← i+1;
}
```

Space of all solutions L that can be constructed from solution L_i

c(L) is the cost of solution L

Graph Clustering Algorithms

- Advantage of Dendrograms: Can be "**cut**" at any desired number of clusters.



Graph Clustering Algorithms

- Linkage (Agglomeration): Iteratively coarsens a given clustering by merging two clusters until 1-clustering is reached ("bottom up")
- Splitting (Division): Iteratively refines a given clustering by splitting one cluster until slingleton clustering is reached ("top down").

- Linkage:
  - Given: G=(V,E,w); initial clustering $\mathbf{C}_1$;
  - Given: Either c_{global}: A(G) → $\mathbb{R}^+$ or c_{local}: P(V) x P(V) → $\mathbb{R}^+$ (for merging operations)
  - i→i+1: Either merge those two clusters where resulting clustering yields the minimum global cost
    or merge those two clusters with the minimum local merging cost

Linkage

Variants / realizations of Linkage:

- Let d(u,v) denote the minimal path length between nodes u and v then local cost function:

$$c_{local}(C_i, C_j) = \begin{matrix}\max \\ \min\end{matrix} \{d(u,v) \mid u \in C_i, v \in C_j\}$$

Complete Linkage Single Linkage

Graph Clustering Algorithms

- Linkage (Agglomeration): Iteratively coarsens a given clustering by merging two clusters until 1-clustering is reached ("bottom up")
- Splitting (Division): Iteratively refines a given clustering by splitting one cluster until slingleton clustering is reached ("top down").

- Linkage:
  - Given: G=(V,E,w); initial clustering $\mathbf{C}_1$;
  - Given: Either c_{global}: A(G) → $\mathbb{R}^+$ or c_{local}: P(V) x P(V) → $\mathbb{R}^+$ (for merging operations)
  - i→i+1: Either merge those two clusters where resulting clustering yields the minimum global cost
    or merge those two clusters with the minimum local merging cost

Graph Clustering Algorithms

- Linkage (Agglomeration): Iteratively coarsens a given clustering by merging two clusters until 1-clustering is reached ("bottom up")
- Splitting (Division): Iteratively refines a given clustering by splitting one cluster until slingleton clustering is reached ("top down").

- Linkage:
  - Given: G=(V,E,w); initial clustering $\mathbf{C}_1$;
  - Given: Either c_{global}: A(G) → $\mathbb{R}^+$ or c_{local}: P(V) x P(V) → $\mathbb{R}^+$ (for merging operations)
  - i→i+1: Either merge those two clusters where resulting clustering yields the minimum global cost
    or merge those two clusters with the minimum local merging cost

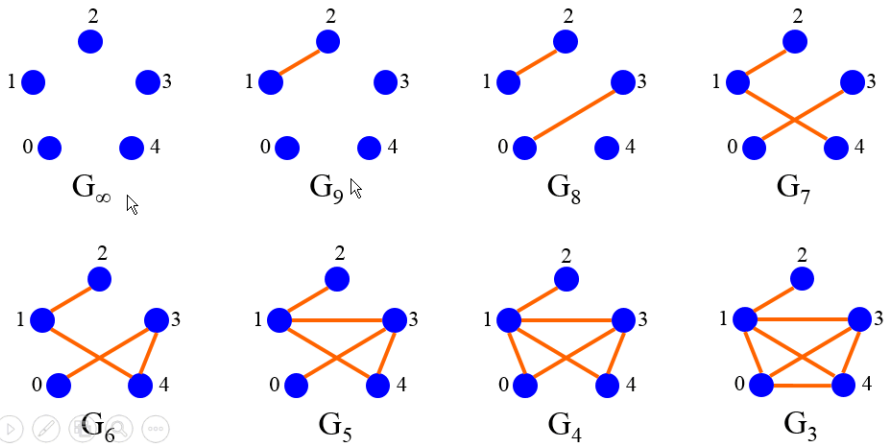
Linkage

- Example

weight matrix:

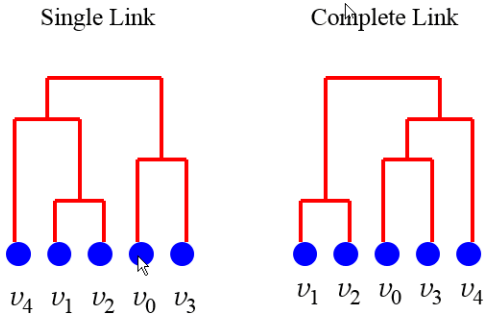
$$\begin{array}{c} \\ v_0 \\ v_1 \\ v_2 \\ v_3 \\ v_4 \end{array} \begin{array}{c} \begin{array}{ccccc} v_0 & v_1 & v_2 & v_3 & v_4 \end{array} \\ \begin{pmatrix} \infty & 4 & 2 & 8 & 3 \\ 4 & \infty & 9 & 5 & 7 \\ 2 & 9 & \infty & 0 & 1 \\ 8 & 5 & 0 & \infty & 6 \\ 3 & 7 & 1 & 6 & \infty \end{pmatrix} \end{array}$$

Threshold graphs:



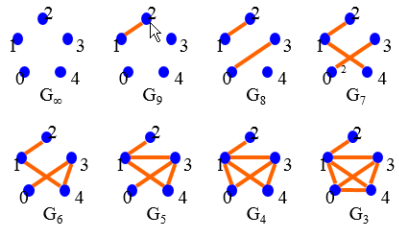
Linkage

Resulting dendrograms:



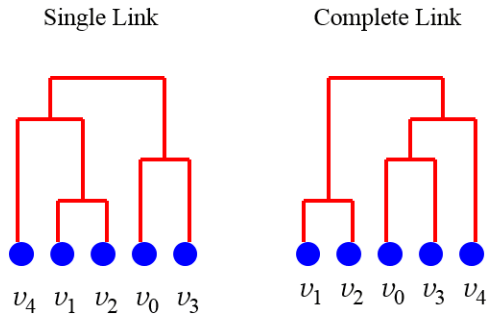
weight matrix:

$$\begin{array}{c} \\ v_0 \\ v_1 \\ v_2 \\ v_3 \\ v_4 \end{array} \begin{array}{c} \begin{array}{ccccc} v_0 & v_1 & v_2 & v_3 & v_4 \end{array} \\ \begin{pmatrix} \infty & 4 & 2 & 8 & 3 \\ 4 & \infty & 9 & 5 & 7 \\ 2 & 9 & \infty & 0 & 1 \\ 8 & 5 & 0 & \infty & 6 \\ 3 & 7 & 1 & 6 & \infty \end{pmatrix} \end{array}$$



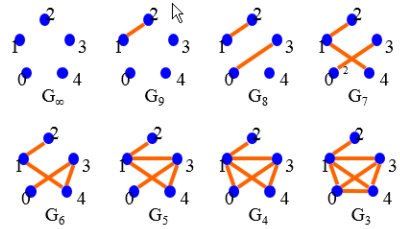
Linkage

Resulting dendrograms:



weight matrix:

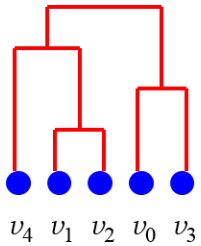
$$\begin{array}{c} \\ v_0 \\ v_1 \\ v_2 \\ v_3 \\ v_4 \end{array} \begin{array}{c} \begin{array}{ccccc} v_0 & v_1 & v_2 & v_3 & v_4 \end{array} \\ \begin{pmatrix} \infty & 4 & 2 & 8 & 3 \\ 4 & \infty & 9 & 5 & 7 \\ 2 & 9 & \infty & 0 & 1 \\ 8 & 5 & 0 & \infty & 6 \\ 3 & 7 & 1 & 6 & \infty \end{pmatrix} \end{array}$$



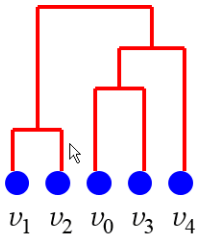
Linkage

Resulting dendrograms:

Single Link

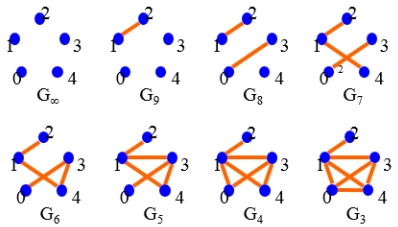


Complete Link



weight matrix:

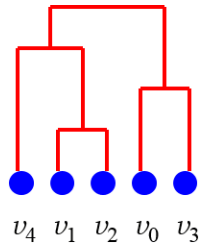
$$\begin{array}{c} \begin{matrix} v_0 & v_1 & v_2 & v_3 & v_4 \end{matrix} \\ \begin{pmatrix} \infty & 4 & 2 & 8 & 3 \\ 4 & \infty & 9 & 5 & 7 \\ 2 & 9 & \infty & 0 & 1 \\ 8 & 5 & 0 & \infty & 6 \\ 3 & 7 & 1 & 6 & \infty \end{pmatrix} \begin{matrix} v_0 \\ v_1 \\ v_2 \\ v_3 \\ v_4 \end{matrix} \end{array}$$



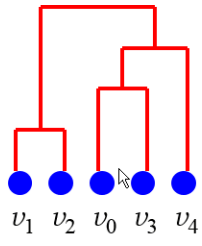
Linkage

Resulting dendrograms:

Single Link

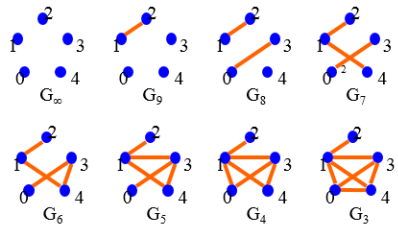


Complete Link



weight matrix:

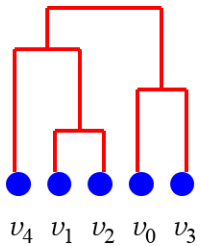
$$\begin{array}{c} \begin{matrix} v_0 & v_1 & v_2 & v_3 & v_4 \end{matrix} \\ \begin{pmatrix} \infty & 4 & 2 & 8 & 3 \\ 4 & \infty & 9 & 5 & 7 \\ 2 & 9 & \infty & 0 & 1 \\ 8 & 5 & 0 & \infty & 6 \\ 3 & 7 & 1 & 6 & \infty \end{pmatrix} \begin{matrix} v_0 \\ v_1 \\ v_2 \\ v_3 \\ v_4 \end{matrix} \end{array}$$



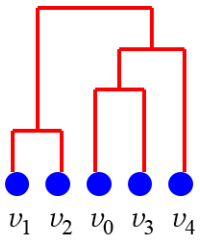
Linkage

Resulting dendrograms:

Single Link

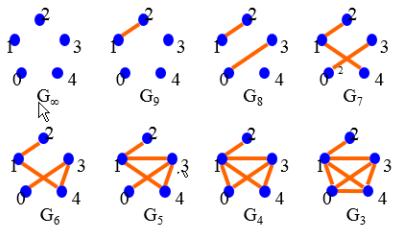


Complete Link



weight matrix:

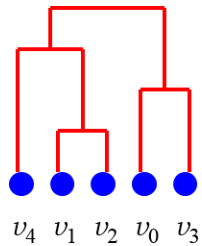
$$\begin{array}{c} \begin{matrix} v_0 & v_1 & v_2 & v_3 & v_4 \end{matrix} \\ \begin{pmatrix} \infty & 4 & 2 & 8 & 3 \\ 4 & \infty & 9 & 5 & 7 \\ 2 & 9 & \infty & 0 & 1 \\ 8 & 5 & 0 & \infty & 6 \\ 3 & 7 & 1 & 6 & \infty \end{pmatrix} \begin{matrix} v_0 \\ v_1 \\ v_2 \\ v_3 \\ v_4 \end{matrix} \end{array}$$



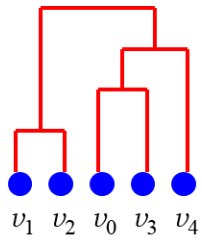
Linkage

Resulting dendrograms:

Single Link

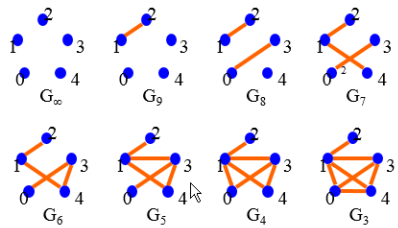


Complete Link



weight matrix:

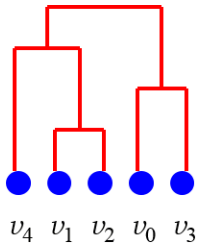
$$\begin{array}{c} \begin{matrix} v_0 & v_1 & v_2 & v_3 & v_4 \end{matrix} \\ \begin{pmatrix} \infty & 4 & 2 & 8 & 3 \\ 4 & \infty & 9 & 5 & 7 \\ 2 & 9 & \infty & 0 & 1 \\ 8 & 5 & 0 & \infty & 6 \\ 3 & 7 & 1 & 6 & \infty \end{pmatrix} \begin{matrix} v_0 \\ v_1 \\ v_2 \\ v_3 \\ v_4 \end{matrix} \end{array}$$



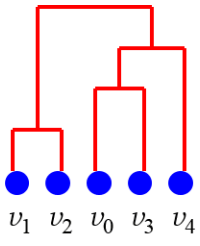
Linkage

Resulting dendrograms:

Single Link

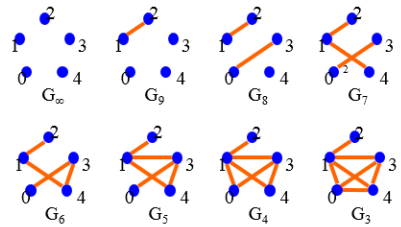


Complete Link



weight matrix:

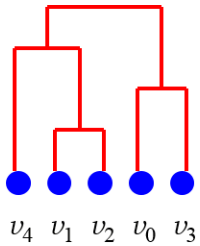
$$\begin{array}{c} \\ v_0 \\ v_1 \\ v_2 \\ v_3 \\ v_4 \end{array} \begin{array}{c} \begin{array}{ccccc} v_0 & v_1 & v_2 & v_3 & v_4 \end{array} \\ \begin{pmatrix} \infty & 4 & 2 & 8 & 3 \\ 4 & \infty & 9 & 5 & 7 \\ 2 & 9 & \infty & 0 & 1 \\ 8 & 5 & 0 & \infty & 6 \\ 3 & 7 & 1 & 6 & \infty \end{pmatrix} \end{array}$$



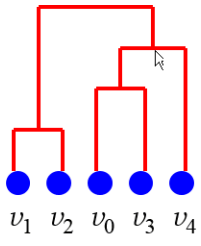
Linkage

Resulting dendrograms:

Single Link

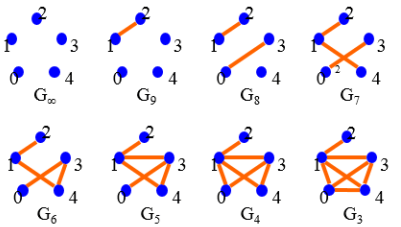


Complete Link



weight matrix:

$$\begin{array}{c} \\ v_0 \\ v_1 \\ v_2 \\ v_3 \\ v_4 \end{array} \begin{array}{c} \begin{array}{ccccc} v_0 & v_1 & v_2 & v_3 & v_4 \end{array} \\ \begin{pmatrix} \infty & 4 & 2 & 8 & 3 \\ 4 & \infty & 9 & 5 & 7 \\ 2 & 9 & \infty & 0 & 1 \\ 8 & 5 & 0 & \infty & 6 \\ 3 & 7 & 1 & 6 & \infty \end{pmatrix} \end{array}$$



Linkage

Variants / realizations of Linkage:

- Let d(u,v) denote the minimal path length between nodes u and v then local cost function:

$$c_{local}(C_i, C_j) = \begin{matrix} \max \\ \min \end{matrix} \{d(u,v) \mid u \in C_i, v \in C_j\}$$

Complete Linkage

Single Linkage

Splitting

- Given: G=(V,E,w); initial clustering $\mathbf{C}_1$;
- Given: Either c_{global}: A(G) → $\mathbb{R}^+$
  c_{local}: P(V) x P(V) → $\mathbb{R}^+$ (for splitting operations) or
  c_{global}: A(G) → $\mathbb{R}^+$ and cut function S: P(V) → P(V) or
  c_{local}: P(V) x P(V) → $\mathbb{R}^+$ and cut function S: P(V) → P(V)
- i→i+1: Split that cluster where the resulting clustering yields the minimum global cost or
  split the cluster with the minimum local splitting cost or
  split that cluster (according to cut S) where the resulting clustering yields the minimum global cost or
  split the cluster (according to cut S) with the minimum local splitting cost

Splitting

C_x: one of the clusters

one "half" of the split of C_x

- Given: G=(V,E,w); initial clustering $\mathbf{C}_1$,
- Given: Either c_{global}: A(G) → $\mathbb{R}^+$

strictly global

c_{local}: P(V) x P(V) → $\mathbb{R}^+$ (for splitting operations) or

strictly local

c_{global}: A(G) → $\mathbb{R}^+$ and cut function S: P(V) → P(V) or

semi global

c_{local}: P(V) x P(V) → $\mathbb{R}^+$ and cut function S: P(V) → P(V)

one "half" of the cut (split) of some C_x, defining the cut

- i→i+1: Split that cluster where the

semi local

global cost or

split the cluster with the minimum local splitting cost or

split that cluster (according to cut S) where the resulting clustering yields the minimum global cost or

split the cluster (according to cut S) with the minimum local splitting cost

Splitting

- Cut function avoids having to test all possible splits
  - Variants of Cut functions:

$$S(V) := \operatorname*{argmin}_{\emptyset \neq V' \subset V} \omega(E(V', V \setminus V'))$$

$$S_{\text{ratio}}(V) := \operatorname*{argmin}_{\emptyset \neq V' \subset V} \frac{\omega(E(V', V \setminus V'))}{|V'| \cdot (|V| - |V'|)}$$

$$S_{\text{balanced}}(V) := \operatorname*{argmin}_{\emptyset \neq V' \subset V} \frac{\omega(E(V', V \setminus V'))}{\min(|V'|, (|V| - |V'|))}$$

$$S_{\text{conductance}}(V) := \operatorname*{argmax}_{\emptyset \neq V' \subset V} \delta(V') = \operatorname*{argmin}_{V' \subset V} \varphi(V', V \setminus V')$$

[1]

inter cluster conductance (slide 14):

$$g(\mathbf{C} = \{V', V \setminus V'\}) = \delta(V') = \begin{cases} 1 & \text{if } \mathbf{C} = \{V, \{\}\} \\ 1 - \varphi(V', V \setminus V') & \text{otherwise} \end{cases}$$

Splitting

- Cut function avoids having to test all possible splits
  - Variants of Cut functions:

$$S(V) := \operatorname*{argmin}_{\emptyset \neq V' \subset V} \omega(E(V', V \setminus V'))$$

$$S_{\text{ratio}}(V) := \operatorname*{argmin}_{\emptyset \neq V' \subset V} \frac{\omega(E(V', V \setminus V'))}{|V'| \cdot (|V| - |V'|)}$$

$$S_{\text{balanced}}(V) := \operatorname*{argmin}_{\emptyset \neq V' \subset V} \frac{\omega(E(V', V \setminus V'))}{\min(|V'|, (|V| - |V'|))}$$

$$S_{\text{conductance}}(V) := \operatorname*{argmax}_{\emptyset \neq V' \subset V} \delta(V') = \operatorname*{argmin}_{V' \subset V} \varphi(V', V \setminus V')$$

[1]

inter cluster conductance (slide 14):

$$g(\mathbf{C} = \{V', V \setminus V'\}) = \delta(V') = \begin{cases} 1 & \text{if } \mathbf{C} = \{V, \{\}\} \\ 1 - \varphi(V', V \setminus V') & \text{otherwise} \end{cases}$$

Shifting

-

shifting minimization:

```
let L_0 be a feasible solution;
i ←0;
while({L | L∈N(L_i)} ≠ ∅) {
      choose L_{i+1} from N(L_i) according to ☺;
      i ← i+1;
}
```

- Choosing schema ☺ can be either based on potential function φ, on random selection or based on genetic algorithms with fitness function etc.
- Potential function φ: A(G) x A(G) → $\mathbb{R}$ based: Chose a new clustering $\mathbf{C}_{i+1}$ so that $\varphi(\mathbf{C}_i, \mathbf{C}_{i+1}) > 0$

Shifting

shifting minimization:

```
let L0 be a feasible solution;
i ←0;
while({L | L∈N(Li)} ≠ ∅) {
      choose Li+1 from N(Li)according to ☺;
      i ← i+1;
}
```

- Choosing schema ☺ can be either based on potential function φ, on random selection or based on genetic algorithms with fitness function etc.
- Potential function φ: A(G) x A(G) → $\mathbb{R}$ based: Chose a new clustering $\mathbf{C}_{i+1}$ so that $\varphi(\mathbf{C}_i, \mathbf{C}_{i+1}) > 0$

Newman Girvan Method: Centrality-based Splitting + Modularity

Last example of this part: bringing it all together (see [3]):

- Observations → critique on agglomerative methods: fail to cluster peripheral nodes correctly [3] → Newman Girvan method: Divisive hierarchical clustering (splitting) + Modularity:

1. Calculate edge betweenness for all edges
2. Remove edge with highest edge betweenness
3. Recalculate edge betweennes, goto 1.

→ dendrogram

- Use Modularity as intra cluster coherence (f) cluster validity measure (g=0) to optimally cut dendrogram:

$$Q = \sum_i (e_{ii} - a_i^2) = \mathrm{Tr}\,\mathbf{e} - \| \mathbf{e}^2 \|$$

Newman Girvan Method: Centrality-based Splitting + Modularity

Last example of this part: bringing it all together (see [3]):

- Observations → critique on agglomerative methods: fail to cluster peripheral nodes correctly [3] → Newman Girvan method: Divisive hierarchical clustering (splitting) + Modularity:

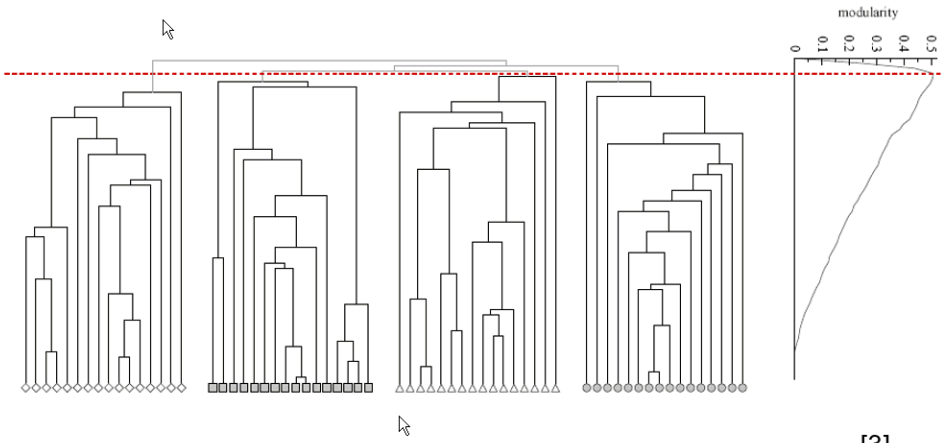
1. Calculate edge betweenness for all edges
2. Remove edge with highest edge betweenness
3. Recalculate edge betweennes, goto 1.

→ dendrogram

- Use Modularity as intra cluster coherence (f) cluster validity measure (g=0) to optimally cut dendrogram:

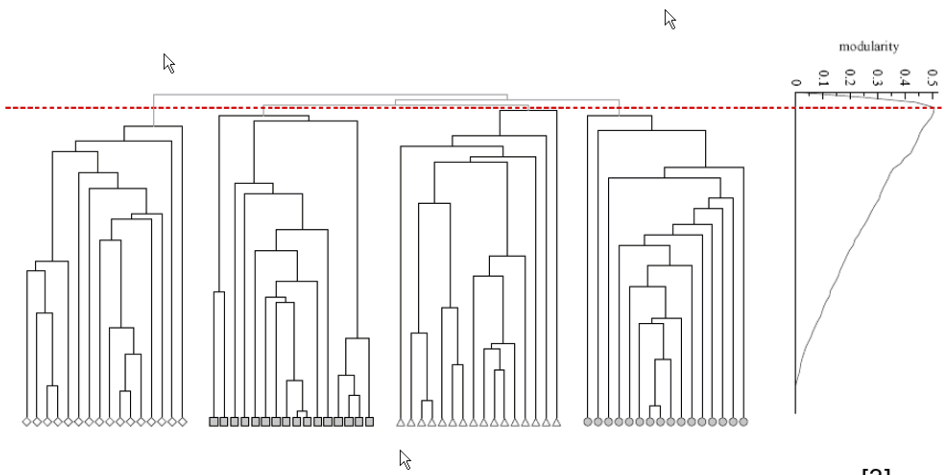
$$Q = \sum_i (e_{ii} - a_i^2) = \mathrm{Tr}\,\mathbf{e} - \| \mathbf{e}^2 \|$$

Newman Girvan Method: Centrality-based Splitting + Modularity



[3]

Newman Girvan Method: Centrality-based Splitting + Modularity



[3]

Newman Girvan Method: Centrality-based Splitting + Modularity

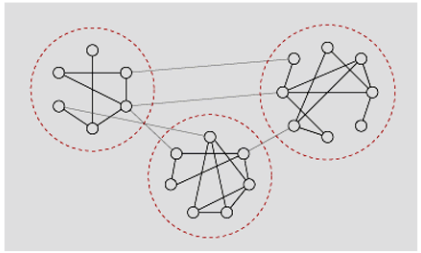
Which edge centrality?

- Shortest Path Betweeness (works best for most cases [3])
  (naive: $O(n^2m)$ (breadth first (O(m)) for each pair of vertices) → better: O(nm) Alg. by Brandes or Newman [3])
- Electric Network based == Random Walk based (see [3])

Newman Girvan Method: Centrality-based Splitting + Modularity

Modularity:

- k clusters → k x k symmetric matrix **e**: e_{ij} = |E(C_i,C_j)| / |E| : fraction of edges between communities



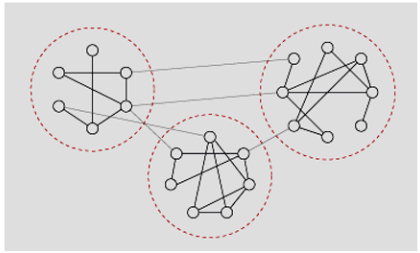
[3]

- $\mathrm{Tr}\,\mathbf{e} = \sum_i e_{ii}$: fraction of edges within communities
- $a_i = \sum_j e_{ij}$: fraction of edges that connect to cluster C_i
- Random network (keep a_i fixed): $e_{ij}^{\mathrm{rnd}} = a_i a_j \rightarrow e_{ii}^{\mathrm{rnd}} = a_i^2$
- f: Compare (→difference)
  real with rnd → $Q = \sum_i (e_{ii} - a_i^2) = \mathrm{Tr}\,\mathbf{e} - \| \mathbf{e}^2 \|$

Newman Girvan Method: Centrality-based Splitting + Modularity

Modularity:

- k clusters → k x k symmetric matrix **e**: e_{ij} = |E(C_i,C_j)| / |E| : fraction of edges between communities

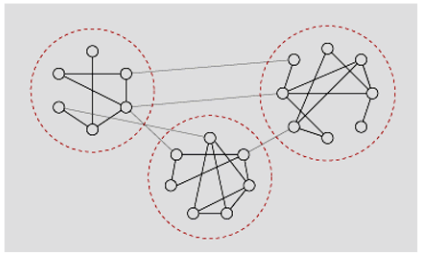


[3]

- $\mathrm{Tr}\,\mathbf{e} = \sum_i e_{ii}$: fraction of edges within communities
- $a_i = \sum_j e_{ij}$: fraction of edges that connect to cluster C_i
- Random network (keep a_i fixed): $e_{ij}^{\mathrm{rnd}} = a_i a_j \rightarrow e_{ii}^{\mathrm{rnd}} = a_i^2$
- f: Compare (→difference)
  real with rnd → $Q = \sum_i (e_{ii} - a_i^2) = \mathrm{Tr}\,\mathbf{e} - \| \mathbf{e}^2 \|$

Newman Girvan Method: Centrality-based Splitting + Modularity

Modularity:



[3]

- k clusters → k x k symmetric matrix **e**: e_{ij} = |E(C_i,C_j)| / |E| : fraction of edges between communities
- $\mathrm{Tr}\,\mathbf{e} = \sum_i e_{ii}$: fraction of edges within communities
- $a_i = \sum_j e_{ij}$: fraction of edges that connect to cluster C_i
- Random network (keep a_i fixed): $e_{ij}^{rnd} = a_i a_j \rightarrow e_{ii}^{rnd} = a_i^2$
- f: Compare (→difference) real with rnd → $Q = \sum_i (e_{ii} - a_i^2) = \mathrm{Tr}\,\mathbf{e} - \| \mathbf{e}^2 \|$

Newman Girvan Method: Centrality-based Splitting + Modularity

Modularity:

[3]

- k clusters → k x k symmetric matrix **e**: e_{ij} = |E(C_i,C_j)| / |E| : fraction of edges between communities
- $\mathrm{Tr}\,\mathbf{e} = \sum_i e_{ii}$: fraction of edges within communities
- $a_i = \sum_j e_{ij}$: fraction of edges that connect to cluster C_i
- Random network (keep a_i fixed): $e_{ij}^{rnd} = a_i a_j \rightarrow e_{ii}^{rnd} = a_i^2$
- f: Compare (→difference) real with rnd → $Q = \sum_i (e_{ii} - a_i^2) = \mathrm{Tr}\,\mathbf{e} - \| \mathbf{e}^2 \|$

Newman Girvan Method: Centrality-based Splitting + Modularity

Modularity:

[3]

- k clusters → k x k symmetric matrix **e**: e_{ij} = |E(C_i,C_j)| / |E| : fraction of edges between communities
- $\mathrm{Tr}\,\mathbf{e} = \sum_i e_{ii}$: fraction of edges within communities
- $a_i = \sum_j e_{ij}$: fraction of edges that connect to cluster C_i
- Random network (keep a_i fixed): $e_{ij}^{rnd} = a_i a_j \rightarrow e_{ii}^{rnd} = a_i^2$
- f: Compare (→difference) real with rnd → $Q = \sum_i (e_{ii} - a_i^2) = \mathrm{Tr}\,\mathbf{e} - \| \mathbf{e}^2 \|$

Newman Girvan Method: Centrality-based Splitting + Modularity

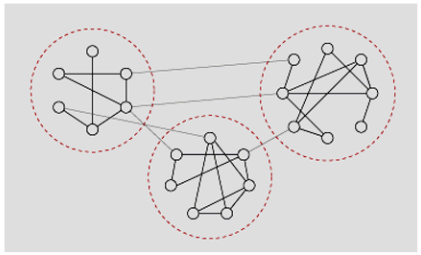
Modularity:

[3]

- k clusters → k x k symmetric matrix **e**: e_{ij} = |E(C_i,C_j)| / |E| : fraction of edges between communities
- $\mathrm{Tr}\,\mathbf{e} = \sum_i e_{ii}$: fraction of edges within communities
- $a_i = \sum_j e_{ij}$: fraction of edges that connect to cluster C_i
- Random network (keep a_i fixed): $e_{ij}^{rnd} = a_i a_j \rightarrow e_{ii}^{rnd} = a_i^2$
- f: Compare (→difference) real with rnd → $Q = \sum_i (e_{ii} - a_i^2) = \mathrm{Tr}\,\mathbf{e} - \| \mathbf{e}^2 \|$

Newman Girvan Method: Centrality-based Splitting + Modularity

Modularity:

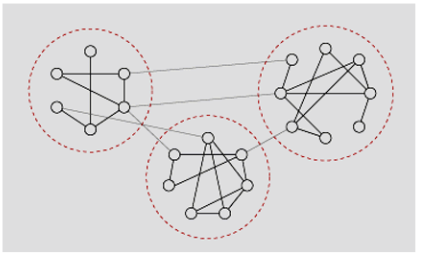


[3]

- k clusters → k x k symmetric matrix **e**: e_{ij} = |E(C_i,C_j)| / |E| : fraction of edges between communities
- $\mathrm{Tr}\,\mathbf{e} = \sum_i e_{ii}$: fraction of edges within communities
- $a_i = \sum_j e_{ij}$: fraction of edges that connect to cluster C_i
- Random network (keep a_i fixed): $e_{ij}^{\mathrm{rnd}} = a_i a_j \rightarrow e_{ii}^{\mathrm{rnd}} = a_i^2$
- f: Compare (→difference)
  real with rnd → $Q = \sum_i (e_{ii} - a_i^2) = \mathrm{Tr}\,\mathbf{e} - \| \mathbf{e}^2 \|$

Newman Girvan Method: Centrality-based Splitting + Modularity

Modularity:

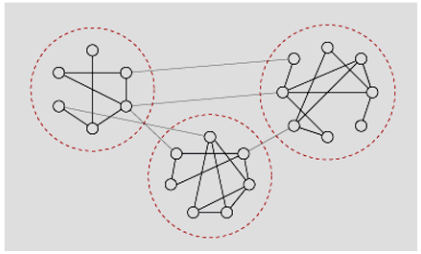


[3]

- k clusters → k x k symmetric matrix **e**: e_{ij} = |E(C_i,C_j)| / |E| : fraction of edges between communities
- $\mathrm{Tr}\,\mathbf{e} = \sum_i e_{ii}$: fraction of edges within communities
- $a_i = \sum_j e_{ij}$: fraction of edges that connect to cluster C_i
- Random network (keep a_i fixed): $e_{ij}^{\mathrm{rnd}} = a_i a_j \rightarrow e_{ii}^{\mathrm{rnd}} = a_i^2$
- f: Compare (→difference)
  real with rnd → $Q = \sum_i (e_{ii} - a_i^2) = \mathrm{Tr}\,\mathbf{e} - \| \mathbf{e}^2 \|$

Newman Girvan Method: Centrality-based Splitting + Modularity

Modularity:

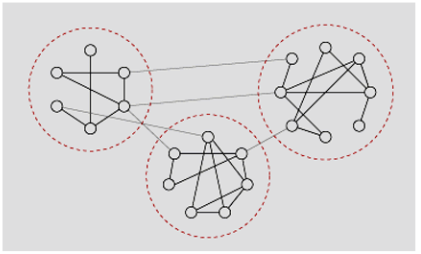


[3]

- k clusters → k x k symmetric matrix **e**: e_{ij} = |E(C_i,C_j)| / |E| : fraction of edges between communities
- $\mathrm{Tr}\,\mathbf{e} = \sum_i e_{ii}$: fraction of edges within communities
- $a_i = \sum_j e_{ij}$: fraction of edges that connect to cluster C_i
- Random network (keep a_i fixed): $e_{ij}^{\mathrm{rnd}} = a_i a_j \rightarrow e_{ii}^{\mathrm{rnd}} = a_i^2$
- f: Compare (→difference)
  real with rnd → $Q = \sum_i (e_{ii} - a_i^2) = \mathrm{Tr}\,\mathbf{e} - \| \mathbf{e}^2 \|$

Newman Girvan Method: Centrality-based Splitting + Modularity

Modularity:

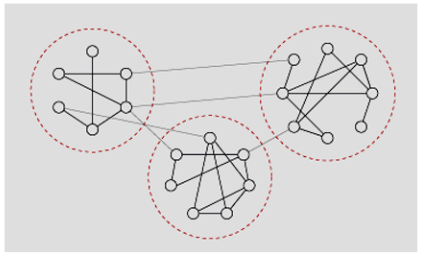


[3]

- k clusters → k x k symmetric matrix **e**: e_{ij} = |E(C_i,C_j)| / |E| : fraction of edges between communities
- $\mathrm{Tr}\,\mathbf{e} = \sum_i e_{ii}$: fraction of edges within communities
- $a_i = \sum_j e_{ij}$: fraction of edges that connect to cluster C_i
- Random network (keep a_i fixed): $e_{ij}^{\mathrm{rnd}} = a_i a_j \rightarrow e_{ii}^{\mathrm{rnd}} = a_i^2$
- f: Compare (→difference)
  real with rnd → $Q = \sum_i (e_{ii} - a_i^2) = \mathrm{Tr}\,\mathbf{e} - \| \mathbf{e}^2 \|$

Newman Girvan Method: Centrality-based Splitting + Modularity

Modularity:

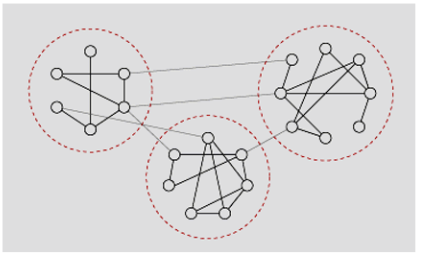


[3]

- k clusters → k x k symmetric matrix **e**: e_{ij} = |E(C_i,C_j)| / |E| : fraction of edges between communities
- $\mathrm{Tr}\,\mathbf{e} = \sum_i e_{ii}$: fraction of edges within communities
- $a_i = \sum_j e_{ij}$: fraction of edges that connect to cluster C_i
- Random network (keep a_i fixed): $e_{ij}^{rnd} = a_i a_j \rightarrow e_{ii}^{rnd} = a_i^2$
- f: Compare (→difference)
  real with rnd → $Q = \sum_i (e_{ii} - a_i^2) = \mathrm{Tr}\,\mathbf{e} - \| \mathbf{e}^2 \|$

Newman Girvan Method: Centrality-based Splitting + Modularity

Modularity:



[3]

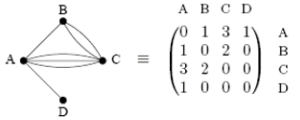
- k clusters → k x k symmetric matrix **e**: e_{ij} = |E(C_i,C_j)| / |E| : fraction of edges between communities
- $\mathrm{Tr}\,\mathbf{e} = \sum_i e_{ii}$: fraction of edges within communities
- $a_i = \sum_j e_{ij}$: fraction of edges that connect to cluster C_i
- Random network (keep a_i fixed): $e_{ij}^{rnd} = a_i a_j \rightarrow e_{ii}^{rnd} = a_i^2$
- f: Compare (→difference)
  real with rnd → $Q = \sum_i (e_{ii} - a_i^2) = \mathrm{Tr}\,\mathbf{e} - \| \mathbf{e}^2 \|$

Newman Girvan Method: Centrality-based Splitting + Modularity

Modularity:

- In [1]: different notion (not keeping a_i fixed): $\sum_{i=1}^{k} \left(|E(C_i)| - m \frac{|C_i| \cdot (|C_i| - 1)|}{n \cdot (n-1)} \right)$

- In [4]: Newman's version for weighted graphs:
  idea: use multiple edges to model weights

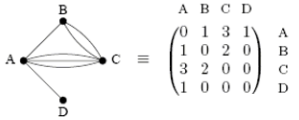


Newman Girvan Method: Centrality-based Splitting + Modularity

Modularity:

- In [1]: different notion (not keeping a_i fixed): $\sum_{i=1}^{k} \left(|E(C_i)| - m \frac{|C_i| \cdot (|C_i| - 1)|}{n \cdot (n-1)} \right)$

- In [4]: Newman's version for weighted graphs:
  idea: use multiple edges to model weights

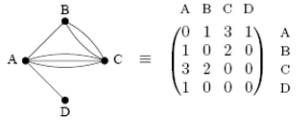


Newman Girvan Method: Centrality-based Splitting + Modularity

Modularity:

- In [1]: different notion (not keeping a_i fixed): $\sum_{i=1}^{k} \left(|E(C_i)| - m \frac{|C_i| \cdot (|C_i| - 1)|}{n \cdot (n-1)} \right)$

- In [4]: Newman's version for weighted graphs:
idea: use multiple edges to model weights

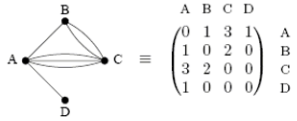


Newman Girvan Method: Centrality-based Splitting + Modularity

Modularity:

- In [1]: different notion (not keeping a_i fixed): $\sum_{i=1}^{k} \left(|E(C_i)| - m \frac{|C_i| \cdot (|C_i| - 1)|}{n \cdot (n-1)} \right)$

- In [4]: Newman's version for weighted graphs:
idea: use multiple edges to model weights

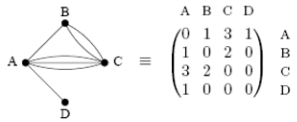


Newman Girvan Method: Centrality-based Splitting + Modularity

Modularity:

- In [1]: different notion (not keeping a_i fixed): $\sum_{i=1}^{k} \left(|E(C_i)| - m \frac{|C_i| \cdot (|C_i| - 1)|}{n \cdot (n-1)} \right)$

- In [4]: Newman's version for weighted graphs:
idea: use multiple edges to model weights



Newman Girvan Method: Centrality-based Splitting + Modularity

How to compute Modularity with a given (weighted) adjacency matrix?

- Real graph: Fraction of edges within clusters:

|E(C_i)| / |E| = $\frac{\sum_{ij} A_{ij} \delta(c_i, c_j)}{\sum_{ij} A_{ij}} = \frac{1}{2m} \sum_{ij} A_{ij} \delta(c_i, c_j)$ $\qquad m = \frac{1}{2} \sum_{ij} A_{ij}$

- Random graph (keep degrees k_i of vertices fixed): prob of edge between vertices i and j is $k_i k_j / (2m)^2$

$$k_v = \sum_w A_{vw}.$$

- → modularity: $Q = \frac{1}{2m} \sum_{ij} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \delta(c_i, c_j)$

which is equal (as before) $= \sum_i (e_{ii} - a_i^2) = \mathrm{Tr}\, \mathbf{e} - \| \mathbf{e}^2 \|$

- → with the new formulation we can compute modularity for weighted graphs

Newman Girvan Method: Centrality-based Splitting + Modularity

How to compute Modularity with a given (weighted) adjacency matrix?

- Real graph: Fraction of edges within clusters:

$|E(C_i)| / |E| = \frac{\sum_{ij} A_{ij}\delta(c_i,c_j)}{\sum_{ij} A_{ij}} = \frac{1}{2m}\sum_{ij} A_{ij}\delta(c_i,c_j)$ $\qquad m = \frac{1}{2}\sum_{ij} A_{ij}$

- Random graph (keep degrees k_i of vertices fixed): prob of edge between vertices i and j is $k_i k_j/(2m)^2$

$k_v = \sum_w A_{vw}.$

- → modularity: $Q = \frac{1}{2m}\sum_{ij}\left[A_{ij} - \frac{k_i k_j}{2m}\right]\delta(c_i,c_j)$

which is equal (as before) $= \sum_i (e_{ii} - a_i^2) = \mathrm{Tr}\,\mathbf{e} - \|\,\mathbf{e}^2\,\|$

- → with the new formulation we can compute modularity for weighted graphs

Newman Girvan Method: Centrality-based Splitting + Modularity

How to compute Modularity with a given (weighted) adjacency matrix?

- Real graph: Fraction of edges within clusters:

$|E(C_i)| / |E| = \frac{\sum_{ij} A_{ij}\delta(c_i,c_j)}{\sum_{ij} A_{ij}} = \frac{1}{2m}\sum_{ij} A_{ij}\delta(c_i,c_j)$ $\qquad m = \frac{1}{2}\sum_{ij} A_{ij}$

- Random graph (keep degrees k_i of vertices fixed): prob of edge between vertices i and j is $k_i k_j/(2m)^2$

$k_v = \sum_w A_{vw}.$

- → modularity: $Q = \frac{1}{2m}\sum_{ij}\left[A_{ij} - \frac{k_i k_j}{2m}\right]\delta(c_i,c_j)$

which is equal (as before) $= \sum_i (e_{ii} - a_i^2) = \mathrm{Tr}\,\mathbf{e} - \|\,\mathbf{e}^2\,\|$

- → with the new formulation we can compute modularity for weighted graphs

Newman Girvan Method: Centrality-based Splitting + Modularity

How to compute Modularity with a given (weighted) adjacency matrix?

- Real graph: Fraction of edges within clusters:

$|E(C_i)| / |E| = \frac{\sum_{ij} A_{ij}\delta(c_i,c_j)}{\sum_{ij} A_{ij}} = \frac{1}{2m}\sum_{ij} A_{ij}\delta(c_i,c_j)$ $\qquad m = \frac{1}{2}\sum_{ij} A_{ij}$

- Random graph (keep degrees k_i of vertices fixed): prob of edge between vertices i and j is $k_i k_j/(2m)^2$

$k_v = \sum_w A_{vw}.$

- → modularity: $Q = \frac{1}{2m}\sum_{ij}\left[A_{ij} - \frac{k_i k_j}{2m}\right]\delta(c_i,c_j)$

which is equal (as before) $= \sum_i (e_{ii} - a_i^2) = \mathrm{Tr}\,\mathbf{e} - \|\,\mathbf{e}^2\,\|$

- → with the new formulation we can compute modularity for weighted graphs