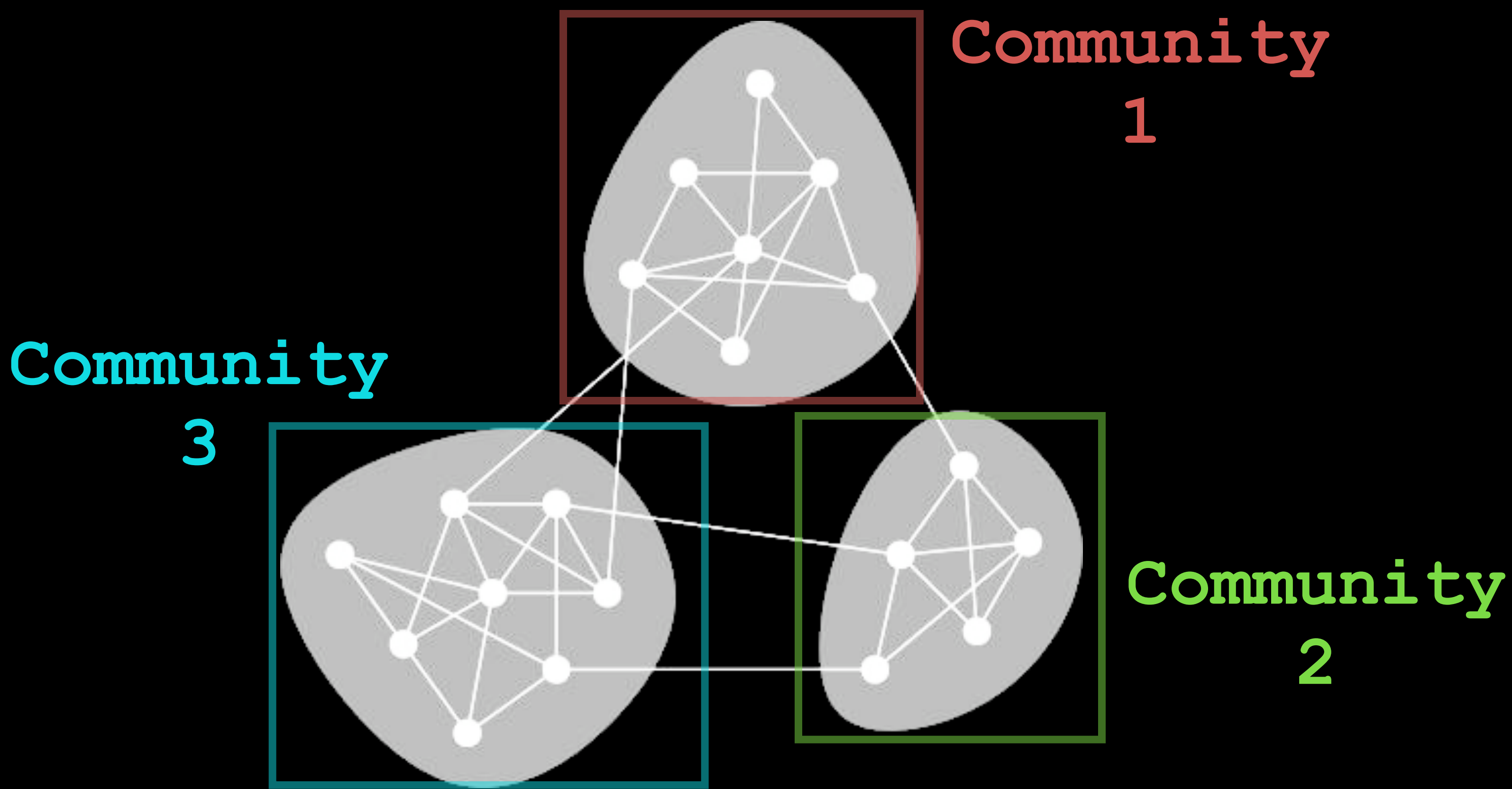
Community Detection


Community
1
Community
3
Community
2

Goals

- **Tools and Datasets**
- **Community Detection Applications**
- **Measure the quality of a community**
  - ★ Modularity
  - ★ Node Similarity
- **Divide a graph into communities**
  - ★ Girvan and Newman (Divisive Algorithm)
  - ★ Hierarchical Clustering (Agglomerative Algorithm)

Goals

- **Tools and Datasets**
- **Community Detection Applications**
- **Measure the quality of a community**
  - ⋆ Modularity
  - ⋆ Node Similarity
- **Divide a graph into communities**
  - ⋆ Girvan and Newman (Divisive Algorithm)
  - ⋆ Hierarchical Clustering (Agglomerative Algorithm)

Python Library
Small Graphs

- **networkx**
  - https://networkx.github.io/
  - Suitable for small graphs (~10,000 nodes)
  - Too slow for big graphs

Python Library
Large Graphs

- **igraph** (C code)
  - http://igraph.org/python/doc/igraph.clustering-module.html
- **graph-tool** (Heavily optimized C code)
  - Takes ~1/2 hour to install
  - https://graph-tool.skewed.de/static/doc/community.html

Benchmarks

```
N = 39796 vertices
E = 301498 edges
```

Algorithm	graph-tool (4 cores)	graph-tool (1 core)	igraph	NetworkX
Single-source shortest path	0.0064 s	0.0063 s	0.012 s	0.127 s
PageRank	0.193 s	0.555 s	0.781 s	34.26 s
K-core	0.0205 s	0.0250 s	0.0181 s	0.9586 s
Minimum spanning tree	0.0268 s	0.0296 s	0.0397 s	0.413 s
Betweenness	579.7 s (~9.6 mins)	1977.6 s (~33 mins)	1182.6 s (~19.7 mins)	53716.692 s (~14.9 hours)

Visualization and Database

- **Neo4j (Database)**
  - http://console.neo4j.org/

- **Gephi (Visualization)**
  - http://gephi.github.io/

Network Data Sources

- http://snap.stanford.edu/class/cs224w-2012/resources.html
- http://konect.uni-koblenz.de/

Goals

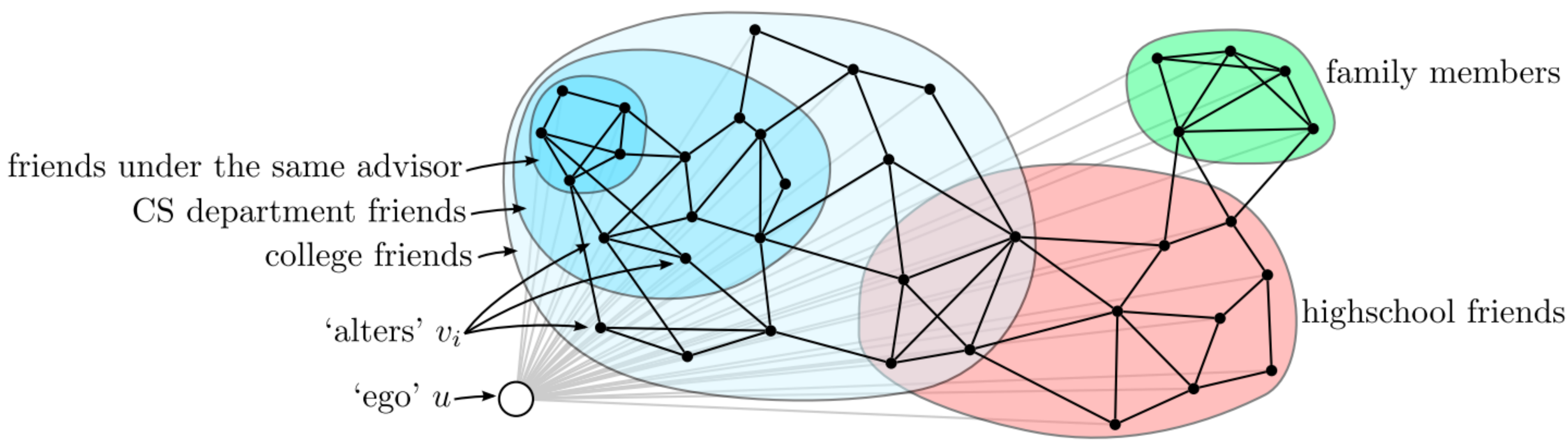
- **Tools and Datasets**
- **Community Detection Applications**
- **Measure the quality of a community**
  - ⋆ Modularity
  - ⋆ Node Similarity
- **Divide a graph into communities**
  - ⋆ Girvan and Newman (Divisive Algorithm)
  - ⋆ Hierarchical Clustering (Agglomerative Algorithm)

Community Detection

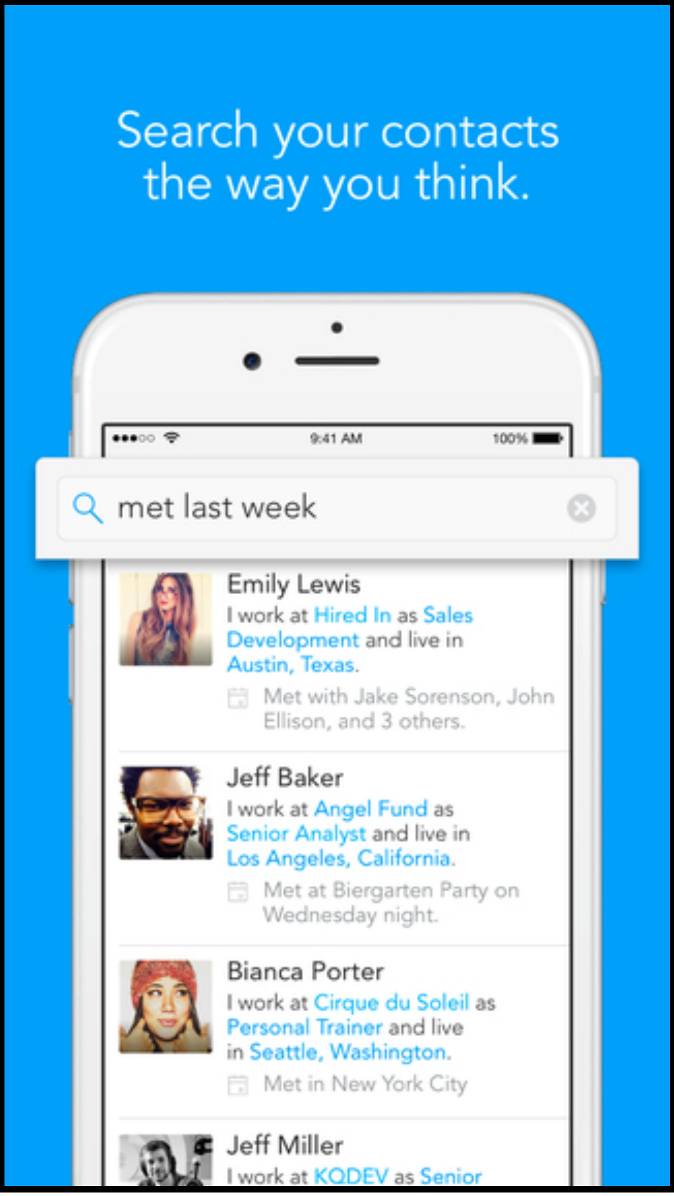
Learning to Discover Social Circles in Ego Networks

Julian McAuley
Stanford, USA
jmcauley@cs.stanford.edu

Jure Leskovec
Stanford, USA
jure@cs.stanford.edu



Community Detection Cont.



Goals

- **Tools and Datasets**
- **Community Detection Applications**
- **Measure the quality of a community**
  - ⋆ Modularity
  - ⋆ Node Similarity
- **Divide a graph into communities**
  - ⋆ Girvan and Newman (Divisive Algorithm)
  - ⋆ Hierarchical Clustering (Agglomerative Algorithm)

Community Quality

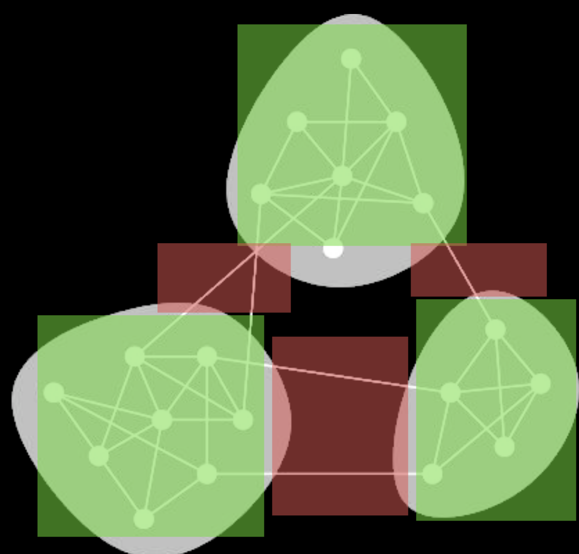
- Quantitative definition of a community
- Used to guide the division of communities

A General Idea

\# Edges
“inside”
community A

>

\# Edges
“between”
community A
and others



Modularity

- Density of edges within a subgraph
- Minus baseline edge density of the same subgraph randomized

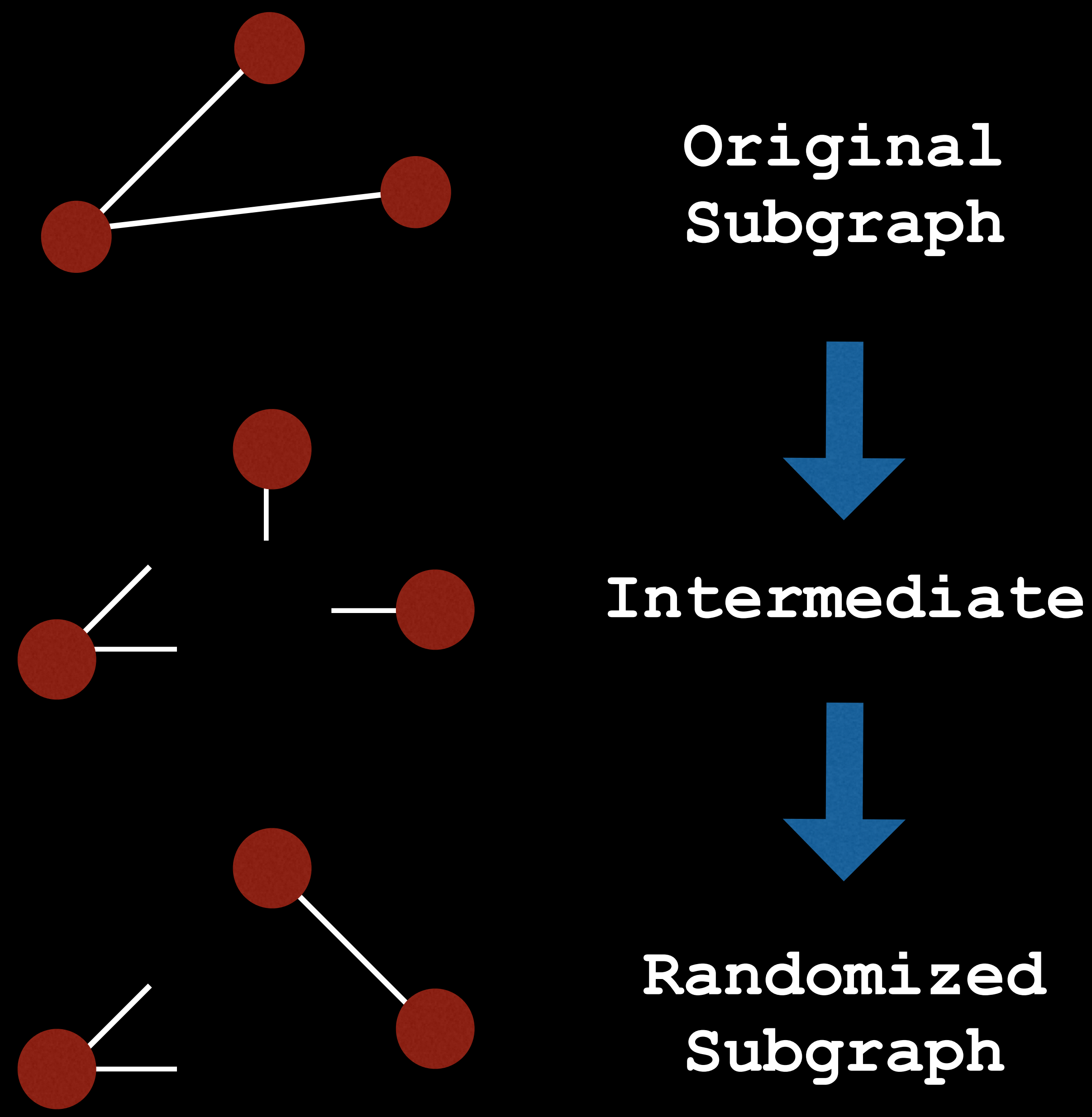
High modularity = “Better” community

`Fraction of edges within a subgraph` − `Fraction of edges if edges were randomly distributed in the subgraph`

- Edge density must be ***higher than expected at random*** to be regarded as ***"high"***

Randomizing Subgraph

1. Halve each edge (stub) in the subgraph
2. Rewire stubs randomly to other nodes / self node
3. Degree distribution of randomized subgraph remains the same


Original
Subgraph
Intermediate
Randomized
Subgraph

Normalization

Adjacency between i and j (0/1)

Prob. of Edge between i and j

$$Q = \boxed{\frac{1}{4m}} \sum_{\substack{i,j \\ \text{in same} \\ \text{module}}} \left(\boxed{A_{ij}} - \boxed{\frac{k_i k_j}{2m}} \right)$$

k_i = Degree of Node i (Randomized Graph)

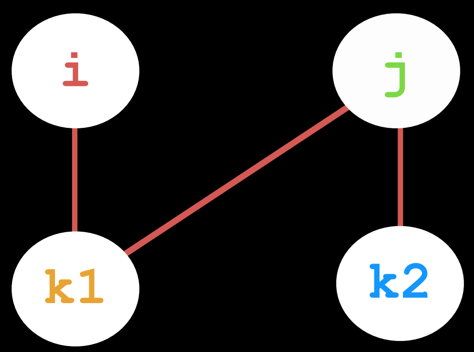
m = # of edges

Node similarity

- The number of neighbors 2 nodes share
- A lot of shared neighbors = High similarity

$$n_{ij} = \sum_k A_{ik} A_{kj}$$

Node similarity Example



$A_{ik1} = 1$ $A_{ik2} = 0$

$A_{jk1} = 1$ $A_{jk2} = 1$

$$n_{ij} = (A_{ik1} * A_{jk1}) + (A_{ik2} * A_{jk2})$$
$$= (1 * 1) + (0 * 1)$$
$$= 1$$

Remarks about Node similarity

- Node similarity is not normalized
- According to the degree of the nodes
- Usually measure **similarity** with **Cosine Similarity**
- Or **(dis)similarity** with **Euclidean Distance**

Node (Dis)Similarity:
Euclidean Distance

$$d_{ij} = \sum_k (A_{ik} - A_{jk})^2$$

$$normal(d_{ij}) = \frac{d_{ij}}{\boxed{k_i} + \boxed{k_j}}$$

Degree of node *i*

Degree of node *j*

Node Similarity: Cosine Similarity

# of shared neighbors

$$\sigma_{ij} = \frac{\boxed{n_{ij}}}{\sqrt{\boxed{k_i}\boxed{k_j}}}$$

Degree of node *i*

Degree of node *j*

Goals

- **Tools and Datasets**
- **Community Detection Applications**
- **Measure the quality of a community**
  - ⋆ Modularity
  - ⋆ Node Similarity
- $\boxed{\text{• Divide a graph into communities}}$
  - ⋆ Girvan and Newman (Divisive Algorithm)
  - ⋆ Hierarchical Clustering (Agglomerative Algorithm)

How to divide a graph

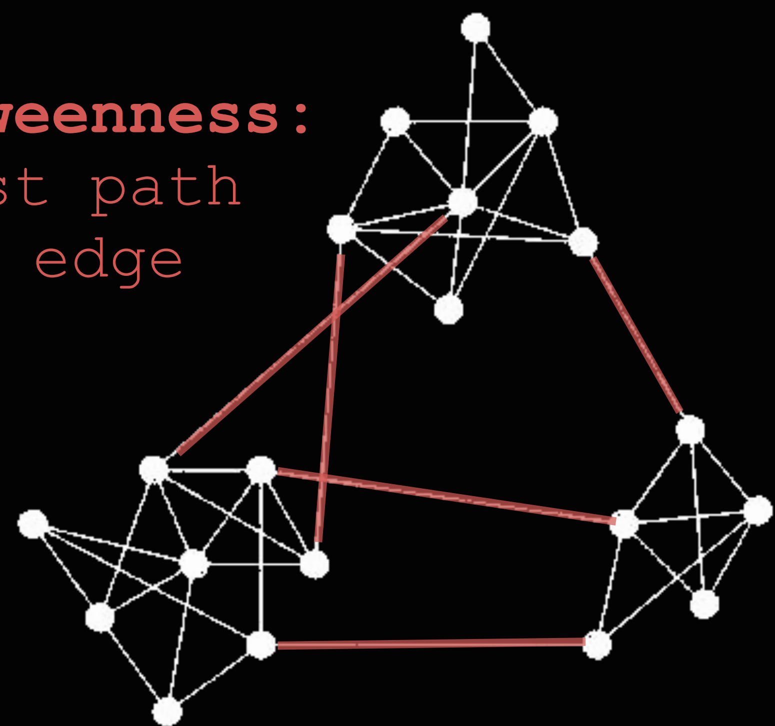
	Divisive	Agglomerative
Approach	Top-down	Bottom-up
Starting Point	Graph	Individual Nodes
Community Formation	Removing edges	Iteratively merging
Technique	Girvan and Newman	Hierarchical Clustering
More popular	Yes	No

Divisive Algorithm

Girvan and Newman

High edge betweenness:

Many shortest path
passes the edge



Girvan and Newman

1. Compute betweenness for all edges
2. Remove edge with largest betweenness
3. Recalculate betweenness

4. Calculate modularity if new communities formed

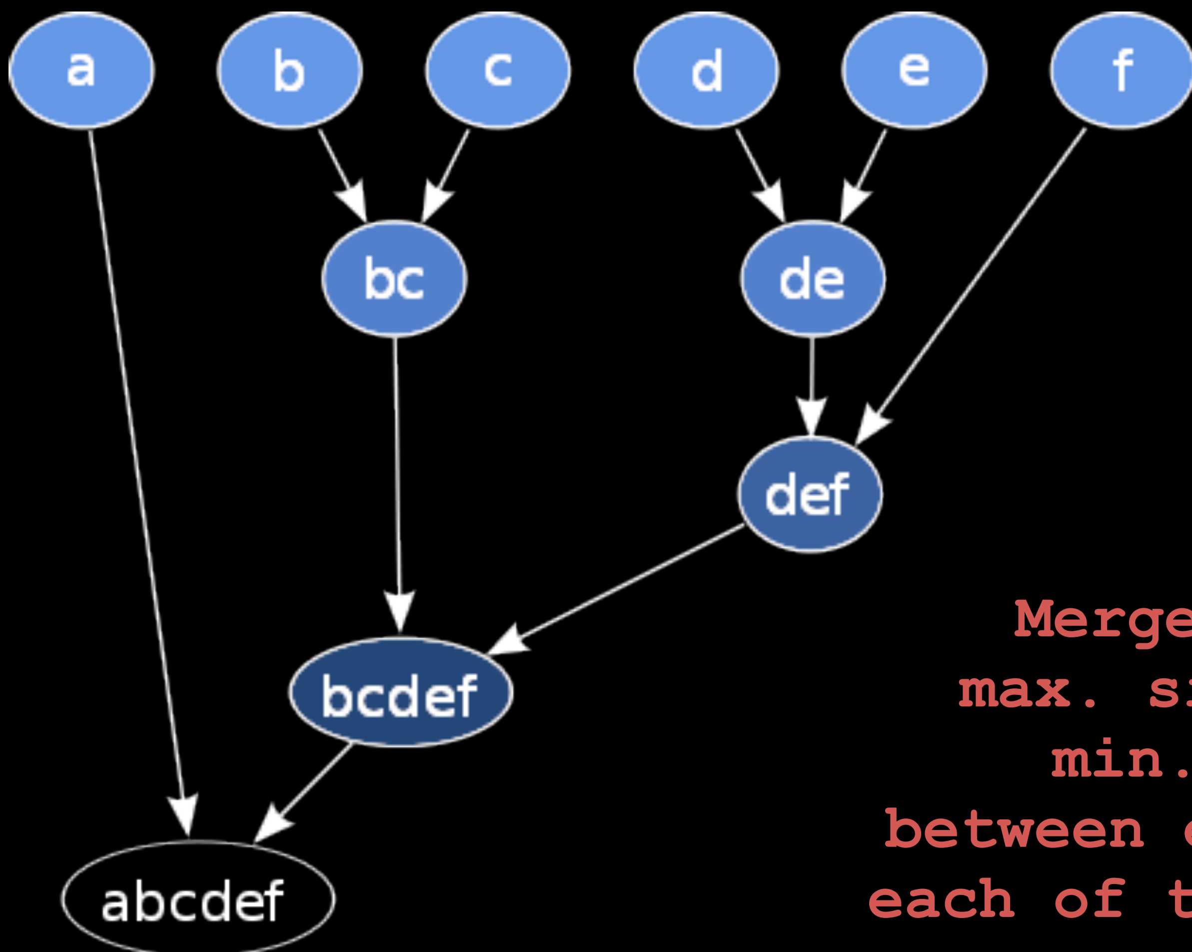
5. Stop if average modularity is maximized

   (i.e. Further iteration would reduce modularity)

6. Otherwise repeat from step 2

Agglomerative Algorithm

Hierarchical Clustering



Merge based on max. similarity / min. distance between elements from each of the 2 clusters

Single Linkage Clustering

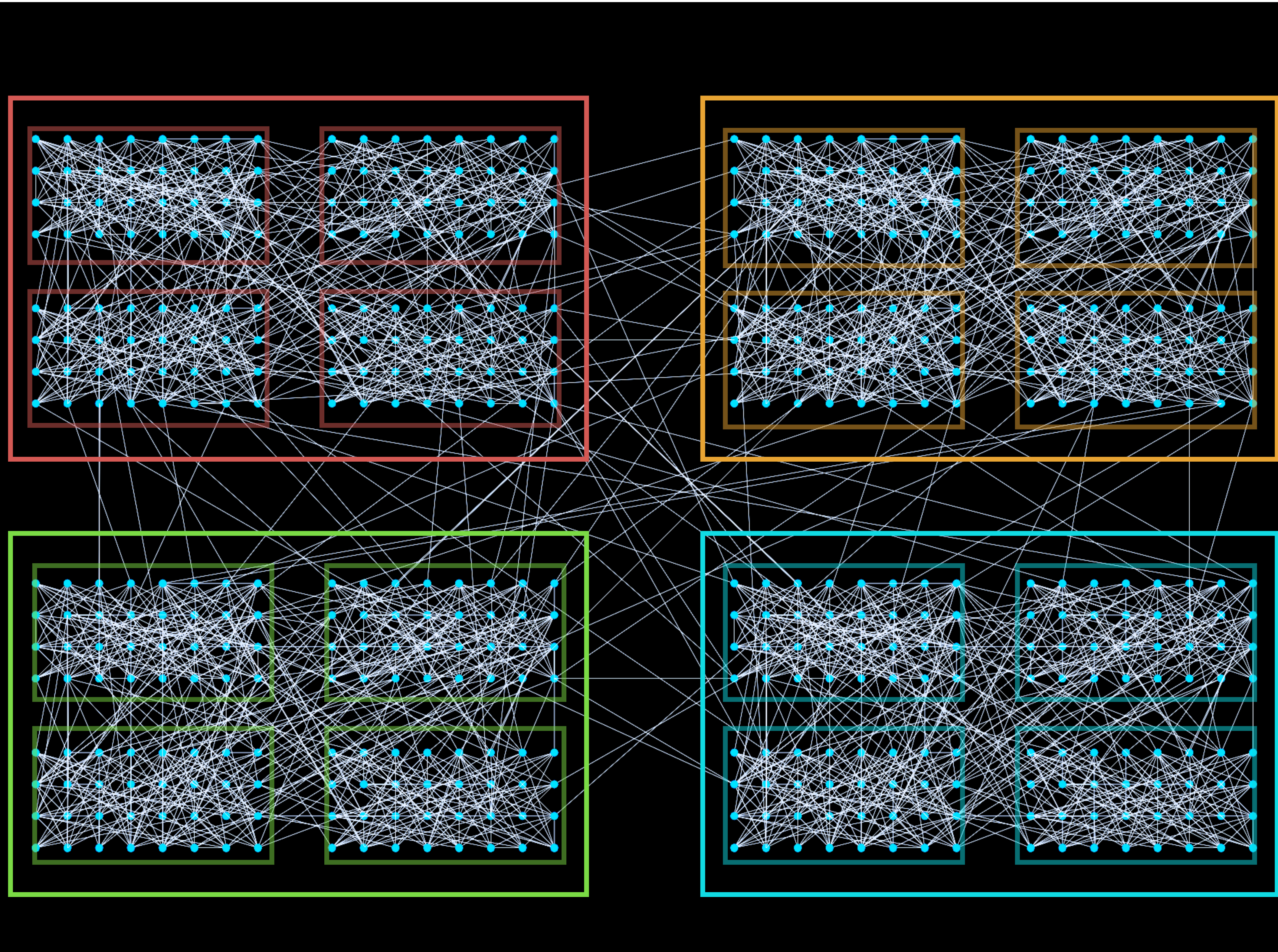
Merge based on the minimum distance between elements of 2 clusters

$$\min\{\, d(x,y) : x \in \mathcal{A},\, y \in \mathcal{B} \,\}.$$

`A and B are 2 separate clusters`

Remarks about Hierarchical Clustering

- Have to decide cut-off
- Hierarchy is by construction, not always sensible
- Good for networks that are hierarchical (social / biological)



Summary

- Applications to detect social community
- Different metric to evaluate communies
- Different algorithms to create communities from graph

Next Steps

- **Cliques** and **k-core** as a measure of similarity
  - ⋆ `p. 10 - 11` (http://arxiv.org/pdf/0906.0612.pdf)
- **Kernighan-Lin (Minimum bisection)** algorithm
  - ⋆ `p. 17 - 18` (http://arxiv.org/pdf/0906.0612.pdf)
- **Partitional Clustering**
  - ⋆ `p. 19 - 20` (http://arxiv.org/pdf/0906.0612.pdf)
- **igraph** and **graph-tool**