Mining Networks

Chapter 5 — Community Detection

Bogumił Kamiński, Paweł Prałat, and François Théberge

Updated: 2025/03/04

Department of Mathematics, Toronto Metropolitan University

File: DS8014-Chapter05

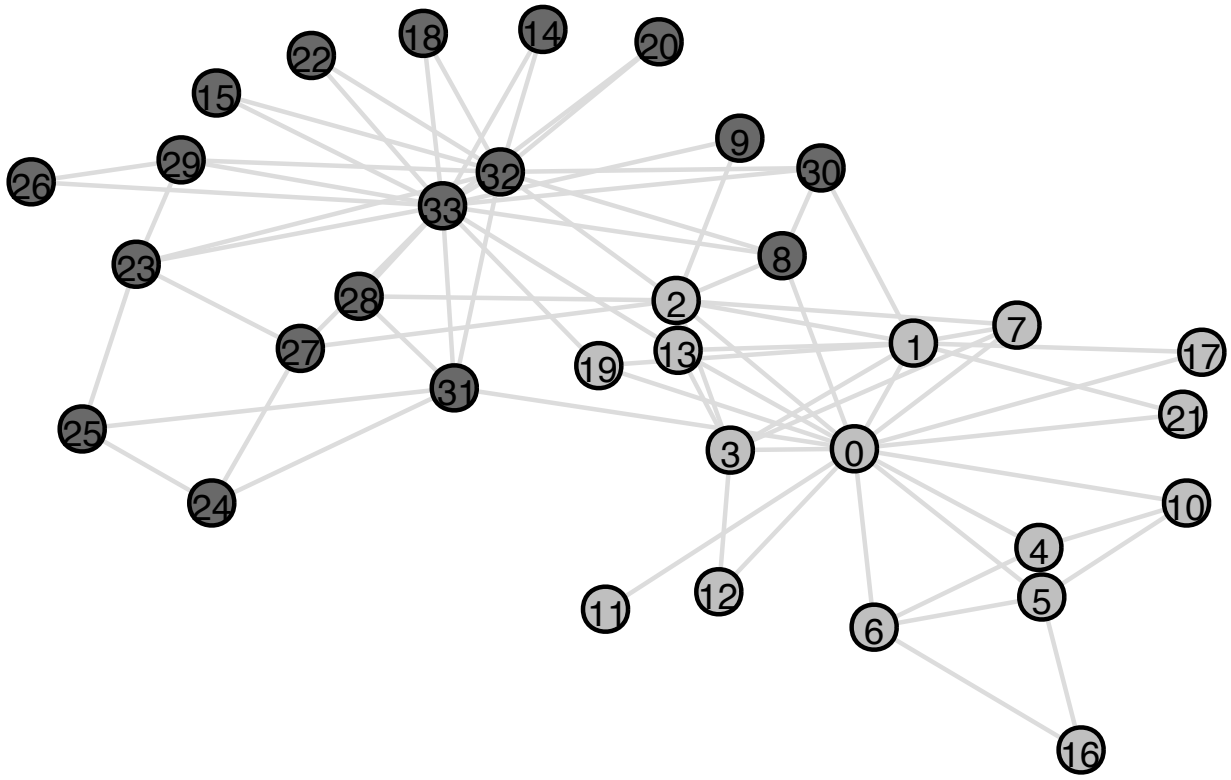
Toronto Metropolitan University

Overview

1. Basic Properties of Communities
2. Synthetic Models with Community Structure
3. Graph Modularity
4. Hierarchical Clustering
5. A Few Other Methods
6. Experiments

Introduction

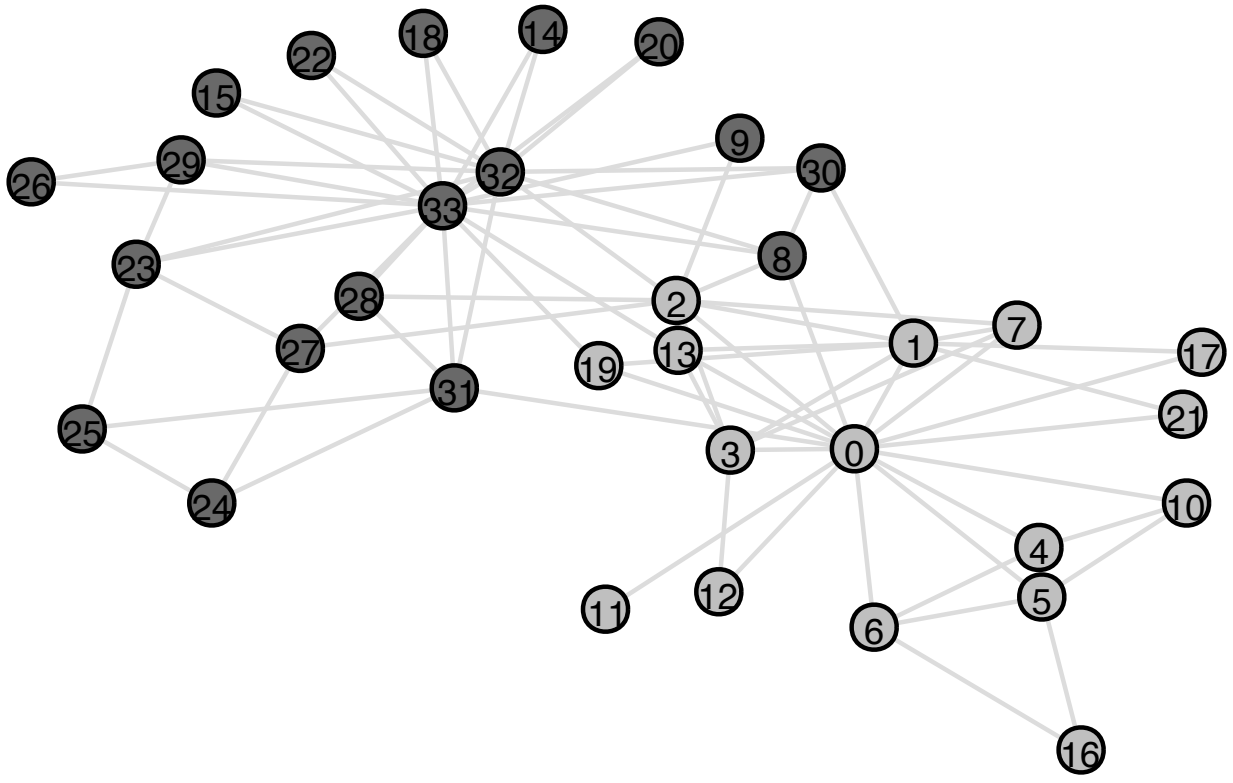
Zachary's Karate Club: 0 – instructor, 33 – administrator



A network has community structure if its set of nodes can be split into a number of subsets such that each subset is densely internally connected.

Introduction

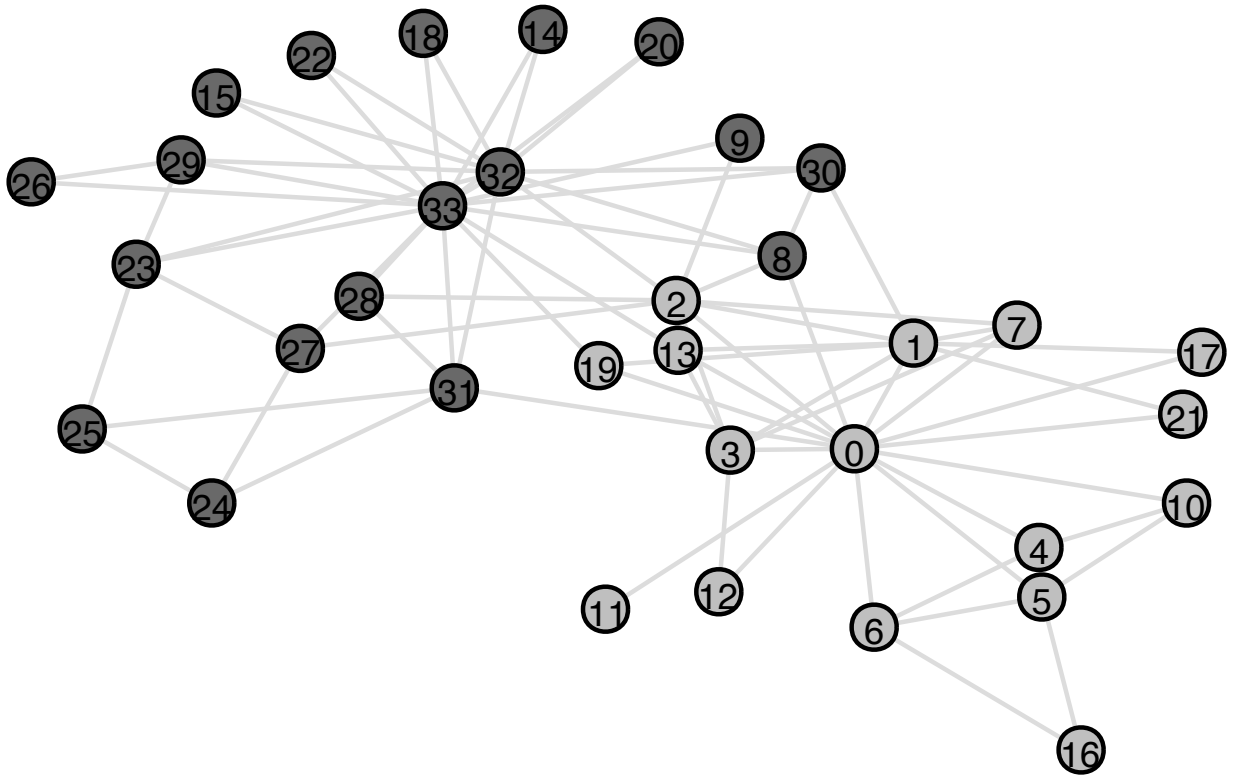
Zachary's Karate Club: 0 – instructor, 33 – administrator



social networks: communities based on common location of their users, their interests, occupation, gender, age, etc.

Introduction

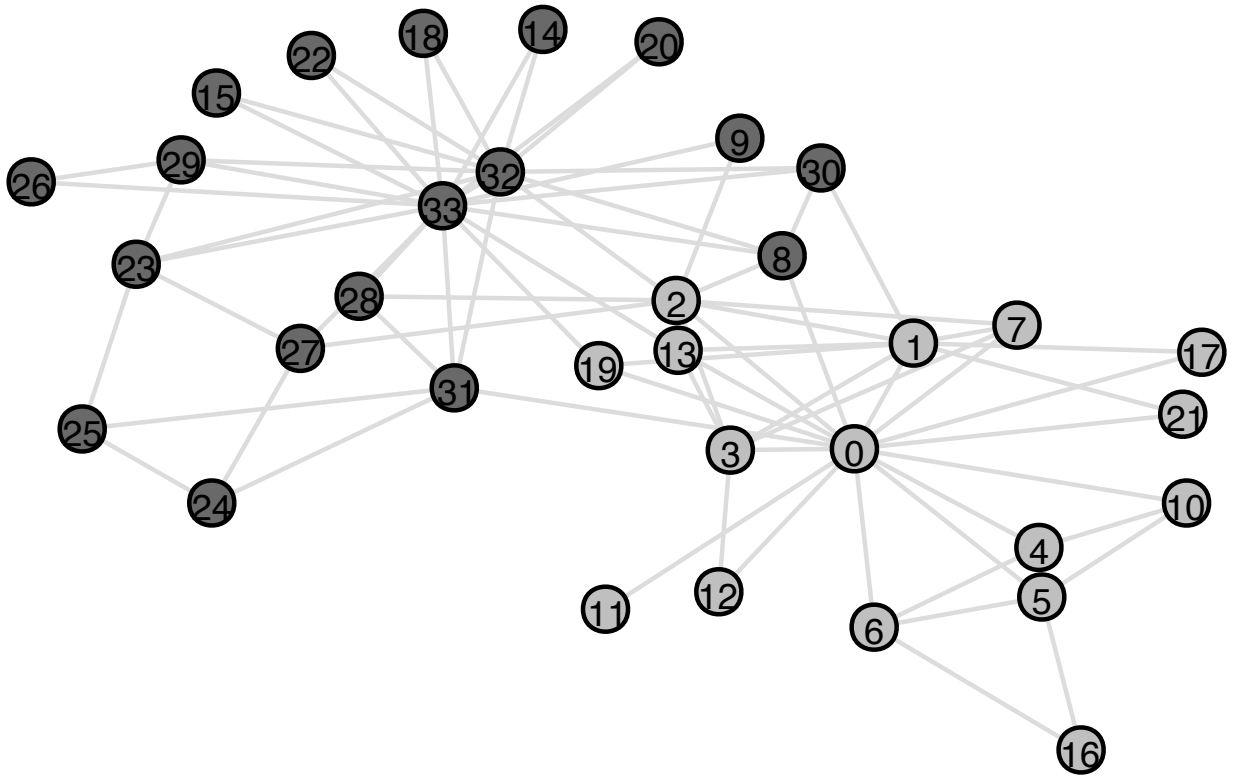
Zachary's Karate Club: 0 – instructor, 33 – administrator



web graph: web pages that belong to the same community are on a similar topic.

Introduction

Zachary's Karate Club: 0 – instructor, 33 – administrator



protein-protein interaction networks: proteins that belong to the same community are often associated with a particular biological function within the organizm.

Introduction

Finding the right partition that represents the community structure is a challenging but important problem for a number of reasons.

Introduction

Finding the right partition that represents the community structure is a challenging but important problem for a number of reasons.

Communities allow us to...
– see a "big picture" (large scale map with individual communities represented as meta-nodes),

Introduction

Finding the right partition that represents the community structure is a challenging but important problem for a number of reasons.

Communities allow us to...
– see a "big picture" (large scale map with individual communities represented as meta-nodes),
– better understand the function of the system represented by the network,

Introduction

Finding the right partition that represents the community structure is a challenging but important problem for a number of reasons.

Communities allow us to...
– see a "big picture" (large scale map with individual communities represented as meta-nodes),
– better understand the function of the system represented by the network,
– classify the nodes based on the position they have in their own clusters and how they are connected to other clusters: roles and importance,
– ...

Basic Properties of Communities

Ground Truth

Collaboration network:

nodes – researchers, edges – collaborations.

Ground Truth

Collaboration network:
nodes – researchers, edges – collaborations.

Mathematicians are interested in doing mathematics and, as a result, tend to work together forming community of mathematicians (there are more edges between mathematicians than between mathematicians and biologists).

Ground Truth

Collaboration network:
nodes – researchers, edges – collaborations.

Mathematicians are interested in doing mathematics and, as a result, tend to work together forming community of mathematicians (there are more edges between mathematicians than between mathematicians and biologists).

We do not know where are mathematicians (ground truth) but we want to uncover this hidden, underlying attributes of nodes based on a graph that is visible to us—reversed engineering process.

Ground Truth

It is extremely rare but sometimes we do have access to ground truth (Zachary's Karate Club).

Ground Truth

It is extremely rare but sometimes we do have access to ground truth (Zachary's Karate Club).

There is need for synthetic networks with community structure (in order to tune community detection algorithms and to benchmark them in a fair and rigorous way).

Ground Truth

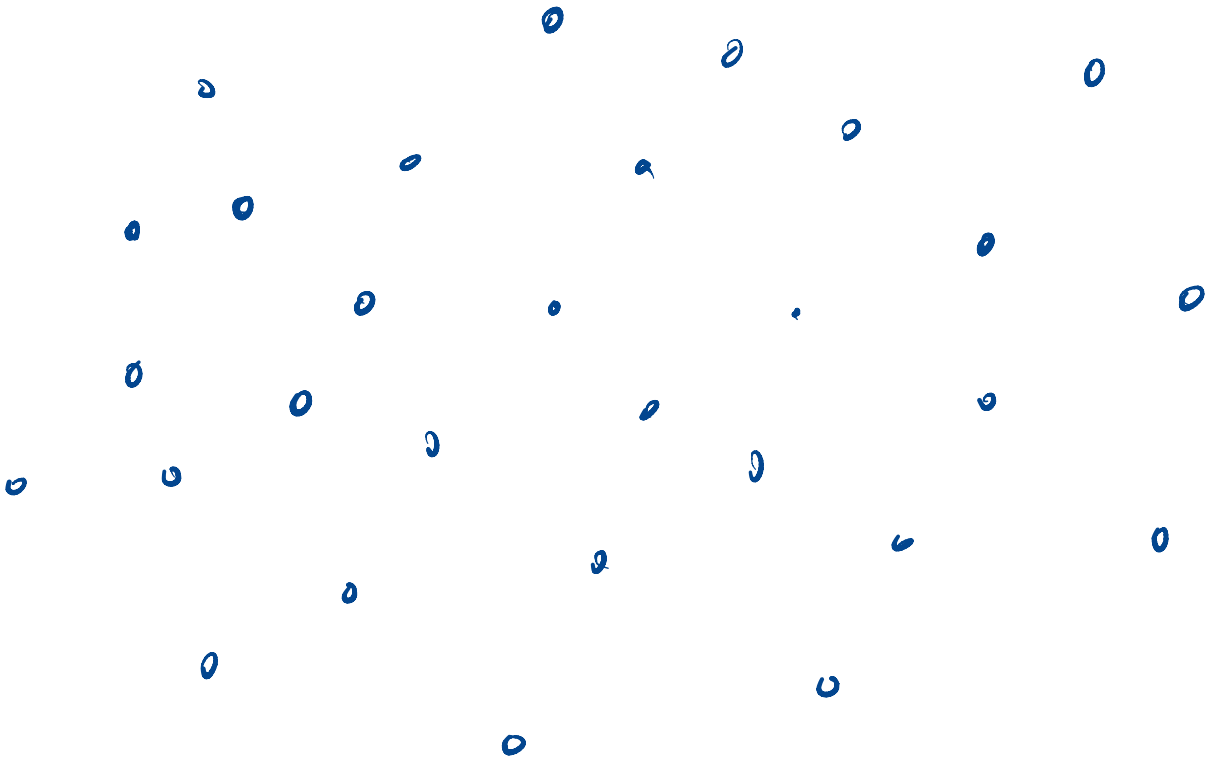
The ground truth could simply be a partition of nodes (the number of parts can be arbitrary).

Partition

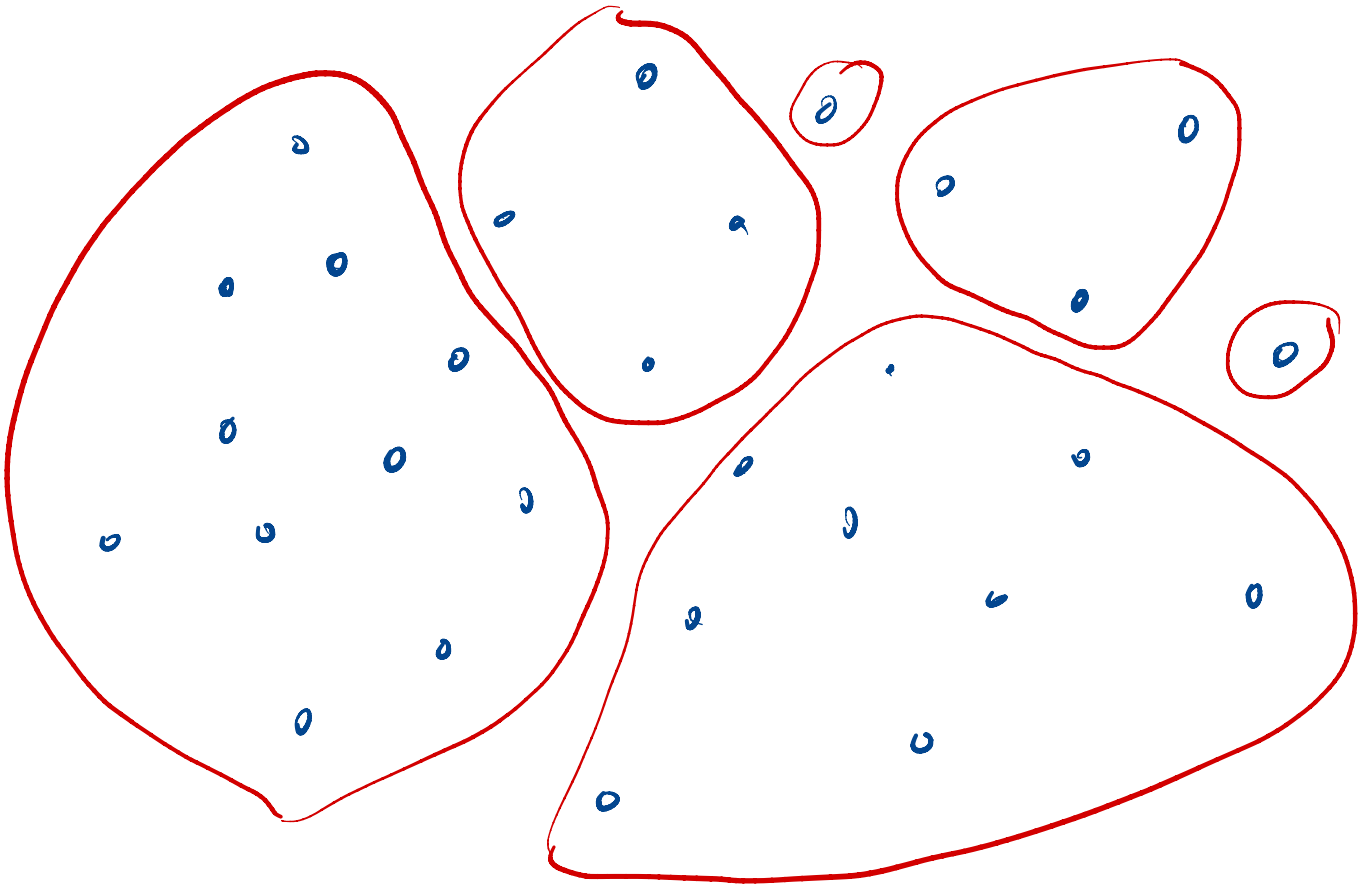
Let V be any finite set (in our scenario, a set of nodes). A partition of V is a grouping of its elements into non-empty subsets, in such a way that every element is included in exactly one subset.

In other words, $V_1, V_2, \ldots, V_\ell$ is a partition if and only if $V_1 \cup V_2 \cup \ldots \cup V_\ell = V$, $V_i \neq \emptyset$ for any $1 \leq i \leq \ell$, and $V_i \cap V_j = \emptyset$ for any $1 \leq i < j \leq \ell$.

Ground Truth

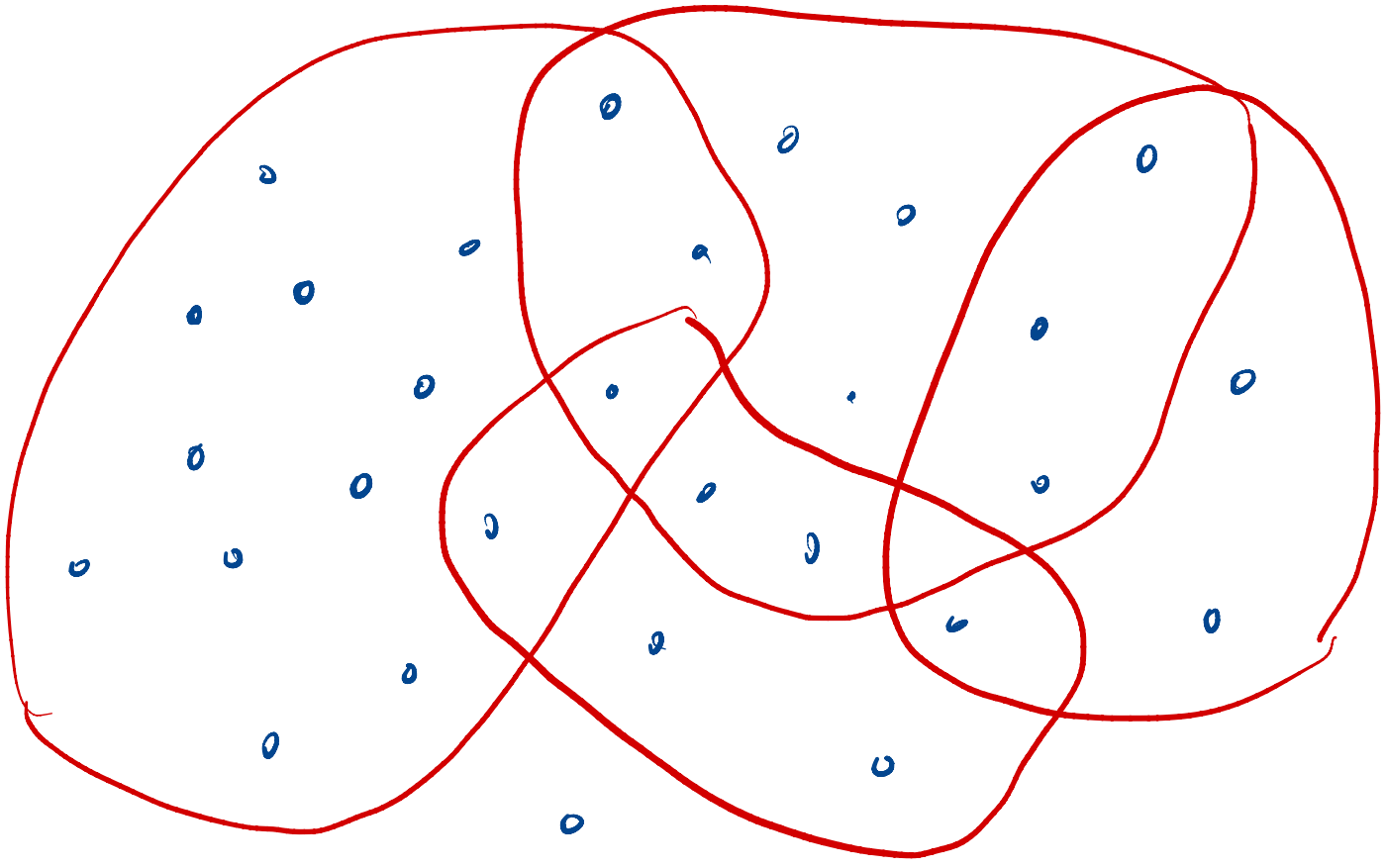


Ground Truth



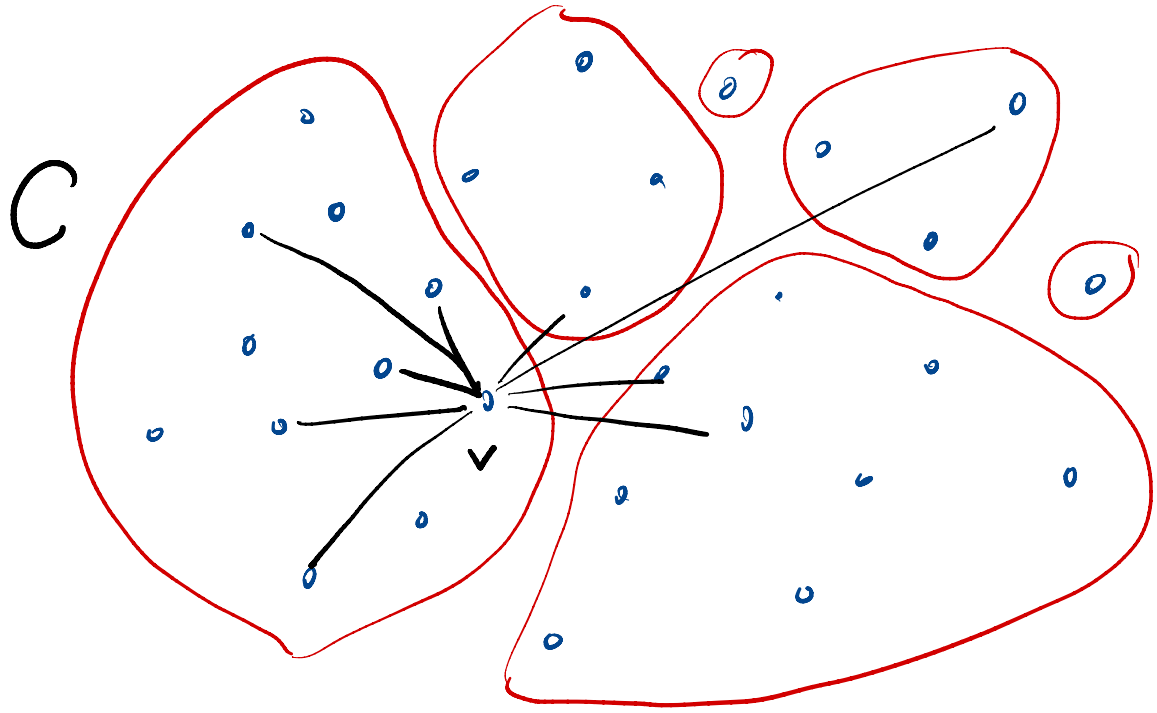
Partition

Ground Truth



Overlapping Communities and/or Outliers — see supplementary material

Communities



Consider any $v \in C$.

internal degree: $\deg^{int}(v) = |N(v) \cap C|$

external degree: $\deg^{ext}(v) = |N(v) \setminus C| = \deg(v) - \deg^{int}(v)$

Strong/Weak Community

Set $C \subseteq V$ forms a strong community if each node in C has more neighbours in C than outside of C, that is, for each $v \in C$ we have

$$\deg^{int}(v) > \deg^{ext}(v).$$

Strong/Weak Community

Set $C \subseteq V$ forms a strong community if each node in C has more neighbours in C than outside of C, that is, for each $v \in C$ we have

$$\deg^{int}(v) > \deg^{ext}(v).$$

Set $C \subseteq V$ forms a weak community if

$$\sum_{v \in C} \deg^{int}(v) > \sum_{v \in C} \deg^{ext}(v).$$

Strong/Weak Community

Set $C \subseteq V$ forms a strong community if each node in C has more neighbours in C than outside of C, that is, for each $v \in C$ we have

$$\deg^{int}(v) > \deg^{ext}(v).$$

Set $C \subseteq V$ forms a weak community if

$$\sum_{v \in C} \deg^{int}(v) > \sum_{v \in C} \deg^{ext}(v).$$

(strong $\Rightarrow$ weak but weak $\not\Rightarrow$ strong)

Communities

"Toy example": two researchers have the same number of friends on Instagram (say, 100) but they belong to two different communities.

Alice (data scientist) is part of a large community whereas Bob (mathematician) belongs to a small community.

Communities

"Toy example": two researchers have the same number of friends on Instagram (say, 100) but they belong to two different communities.

Alice (data scientist) is part of a large community whereas Bob (mathematician) belongs to a small community.

60% of friends of Alice do data science making her a clear member of that community.

Communities

"Toy example": two researchers have the same number of friends on Instagram (say, 100) but they belong to two different communities.

Alice (data scientist) is part of a large community whereas Bob (mathematician) belongs to a small community.

60% of friends of Alice do data science making her a clear member of that community.

What about Bob? Should we expect that more than 50% of his friends to be working in his field of research?

Communities

"Toy example": two researchers have the same number of friends on Instagram (say, 100) but they belong to two different communities.

Alice (data scientist) is part of a large community whereas Bob (mathematician) belongs to a small community.

60% of friends of Alice do data science making her a clear member of that community.

What about Bob? Should we expect that more than 50% of his friends to be working in his field of research?

No! In fact, it might be the case that there are less than 50 people around the world working in his field!

Communities

The number of internal neighbours of a given node should be proportional to the size of the community this node belongs to. But the probability that a node is adjacent to another member of its community should be larger than the probability of being adjacent to a random node in the whole graph.

Communities

The number of internal neighbours of a given node should be proportional to the size of the community this node belongs to. But the probability that a node is adjacent to another member of its community should be larger than the probability of being adjacent to a random node in the whole graph.

Community

Let $C \subseteq V$ ($C \neq \emptyset$ and $C \neq V$). C is a community if each node in C has proportionally more neighbours in C than outside of C, that is, for each $v \in C$ we have

$$\frac{\deg^{int}(v)}{|C|} > \frac{\deg^{ext}(v)}{|V \setminus C|}.$$

Node Roles

Partition $\mathcal{A} = \{A_1, A_2, \ldots, A_\ell\}$ of V; (ground truth or a partition returned by some community detection algorithm).

$\deg_{A_i}(v) = |N(v) \cap A_i|$ (number of neighbours of v in part A_i),
$\deg^{int}(v) = \deg_{A_i}(v)$ for the unique $i \in [\ell]$ for which $v \in A_i$.

Node Roles

Partition $\mathcal{A} = \{A_1, A_2, \ldots, A_\ell\}$ of V; (ground truth or a partition returned by some community detection algorithm).

$\deg_{A_i}(v) = |N(v) \cap A_i|$ (number of neighbours of v in part A_i),
$\deg^{int}(v) = \deg_{A_i}(v)$ for the unique $i \in [\ell]$ for which $v \in A_i$.

Normalized within-module degree

The normalized within-module degree of a node v (with respect to $\mathcal{A}$) is defined as follows:

$$z(v) = \frac{\deg^{int}(v) - \mu(v)}{\sigma(v)},$$

where $\mu(v)/\sigma(v)$ are the mean/standard deviation of $\deg^{int}(u)$ over all nodes in the part v belongs to.

Node Roles

$z(v)$ captures how strongly a particular node is connected to other nodes within its own community, completely ignoring edges between communities.

Node Roles

$z(v)$ captures how strongly a particular node is connected to other nodes within its own community, completely ignoring edges between communities.

$z(v)$ is a familiar Z-score as it measures how many standard deviations the internal degree of v deviates from the mean.

Node Roles

$z(v)$ captures how strongly a particular node is connected to other nodes within its own community, completely ignoring edges between communities.

$z(v)$ is a familiar Z-score as it measures how many standard deviations the internal degree of v deviates from the mean.

$z(v)$ is large and positive: v is tightly connected to other nodes within the community.
$|z(v)|$ is large and $z(v)$ is negative: v is loosely connected to other peers.

Node Roles

$z(v)$ captures how strongly a particular node is connected to other nodes within its own community, completely ignoring edges between communities.

$z(v)$ is a familiar Z-score as it measures how many standard deviations the internal degree of v deviates from the mean.

$z(v)$ is large and positive: v is tightly connected to other nodes within the community.
$|z(v)|$ is large and $z(v)$ is negative: v is loosely connected to other peers.

If $z(v) > 2.5$, then node v is classified as a hub; otherwise, it is a non-hub.

Node Roles

Another important aspect: how edges (both internal and external) are distributed between all parts of the partition $\mathcal{A}$.

Participation coefficient

The participation coefficient of a node v (with respect to $\mathcal{A}$) is defined as follows:

$$p(v) = 1 - \sum_{i=1}^{\ell} \left(\frac{\deg_{A_i}(v)}{\deg(v)} \right)^2 .$$

Node Roles

Another important aspect: how edges (both internal and external) are distributed between all parts of the partition $\mathcal{A}$.

Participation coefficient

The participation coefficient of a node v (with respect to $\mathcal{A}$) is defined as follows:

$$p(v) = 1 - \sum_{i=1}^{\ell} \left(\frac{\deg_{A_i}(v)}{\deg(v)} \right)^2 .$$

If v has neighbours exclusively in one part, then $p(v) = 0$.

If the neighbours of v are homogeneously distributed among all parts, then $p(v) \approx 1 - 1/\ell \approx 1$.

Node Roles

Non-hubs are partitioned into the following four classes:

– ultra-peripheral nodes (almost all edges are internal) if $p(v) < 0.05$,

– peripheral nodes (most edges are internal) if $0.05 \leq p(v) < 0.62$,

– connector nodes (most edges are external) if $0.62 \leq p(v) < 0.80$,

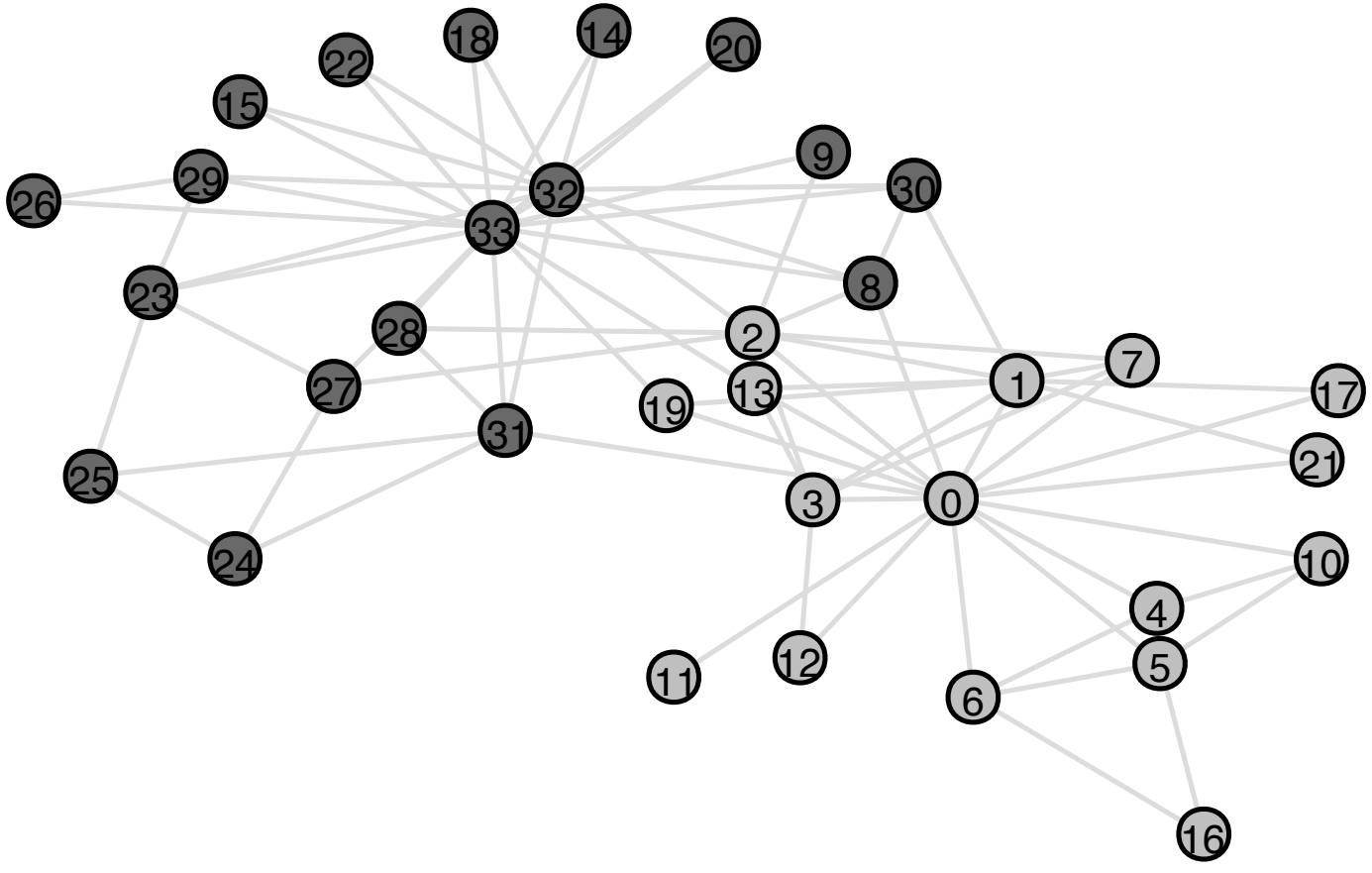
– kinless nodes (neighbours are homogeneously distributed) if $p(v) \geq 0.80$.

Node Roles

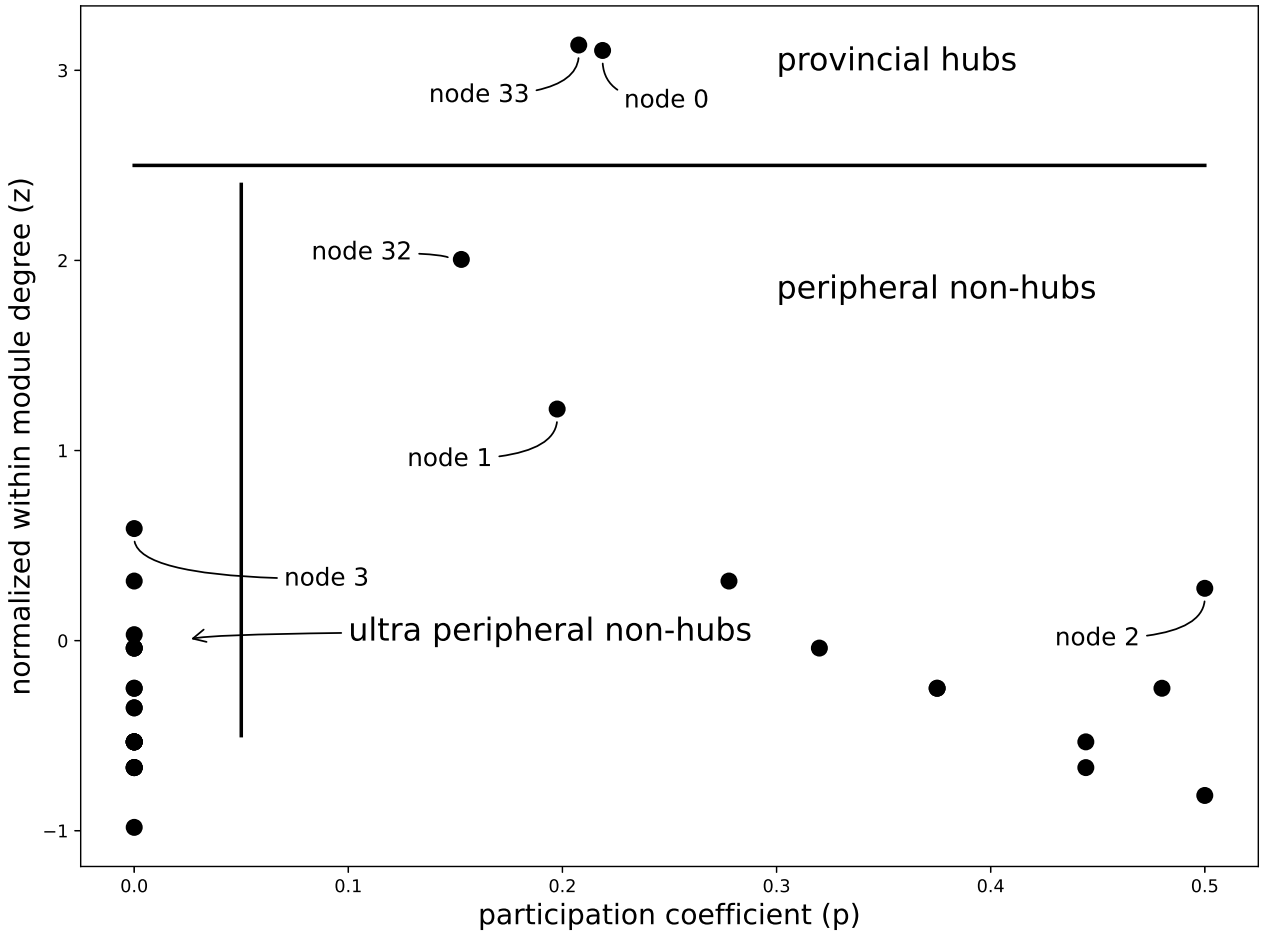
Non-hubs are partitioned into the following four classes:
– ultra-peripheral nodes (almost all edges are internal) if
$p(v) < 0.05$,
– peripheral nodes (most edges are internal) if
$0.05 \leq p(v) < 0.62$,
– connector nodes (most edges are external) if
$0.62 \leq p(v) < 0.80$,
– kinless nodes (neighbours are homogeneously distributed) if
$p(v) \geq 0.80$.

On the other hand, hubs are partitioned into three classes:
– provincial hubs if $p(v) < 0.30$,
– connector hubs if $0.30 \leq p(v) < 0.75$,
– kinless hubs if $p(v) \geq 0.75$.

Node Roles


18
14
20
22
15
9
29
30
26
32
33
8
23
28
2
7
27
13
1
17
19
31
21
25
3
0
24
10
4
11
12
5
6
16

Node Roles


provincial hubs
node 33
node 0
node 32
peripheral non-hubs
node 1
node 3
ultra peripheral non-hubs
node 2
normalized within module degree (z)
participation coefficient (p)
3
2
1
0
−1
0.0
0.1
0.2
0.3
0.4
0.5

Anomaly Score

The anomaly score of a node v (with respect to $\mathcal{A}$) is defined as follows:

$$cd(v) = \frac{\deg(v)}{\deg^{int}(v)}.$$

Node Roles

Anomaly Score

The anomaly score of a node v (with respect to $\mathcal{A}$) is defined as follows:

$$cd(v) = \frac{\deg(v)}{\deg^{int}(v)}.$$

Participation coefficient and anomaly score ignore the sizes of communities. It is natural to expect that if a node belongs to a larger community it should have more neighbours in it by pure luck.

Anomaly Score

The anomaly score of a node v (with respect to $\mathcal{A}$) is defined as follows:

$$cd(v) = \frac{\deg(v)}{\deg^{int}(v)}.$$

Participation coefficient and anomaly score ignore the sizes of communities. It is natural to expect that if a node belongs to a larger community it should have more neighbours in it by pure luck.

We can model "pure luck" with the Chung-Lu model.

Node Roles

Community Association Strength

The community association strength of a node v (with respect to community A_i from $\mathcal{A}$) is defined as follows:

$$cas(v) = \frac{\deg_{A_i}(v)}{\deg(v)} - \frac{\mathrm{vol}(A_i) - \deg(v)}{\mathrm{vol}(V)}.$$

Community Distribution Distance

The community distribution distance of a node v (with respect to $\mathcal{A}$) is defined as follows:

$$cdd(v) = \left(\sum_{i=1}^{\ell} \left(\frac{\deg_{A_i}(v)}{\deg(v)} - \frac{\mathrm{vol}(A_i)}{\mathrm{vol}(V)} \right)^2 \right)^{1/2}.$$

The Number of Partitions

Bell number = the number of partitions of a set of size n.

The Number of Partitions

Bell number = the number of partitions of a set of size n.

Clearly, $B_n \geq 2^n$ (the number of partitions into 2 parts) but, in fact, B_n grows even faster!

The Number of Partitions

Bell number = the number of partitions of a set of size n.

Clearly, $B_n \geq 2^n$ (the number of partitions into 2 parts) but, in fact, B_n grows even faster!

Asymptotic behaviour:

$$\frac{\ln B_n}{n} = \ln n - \ln \ln n - 1 + o(1),$$

which implies that

$$B_n = \left(\frac{n}{(e + o(1)) \ln n} \right)^n.$$

The Number of Partitions

Bell number = the number of partitions of a set of size n.

Clearly, $B_n \geq 2^n$ (the number of partitions into 2 parts) but, in fact, B_n grows even faster!

Asymptotic behaviour:

$$\frac{\ln B_n}{n} = \ln n - \ln \ln n - 1 + o(1),$$

which implies that

$$B_n = \left(\frac{n}{(e + o(1)) \ln n} \right)^n.$$

Important implication: we will not be able to investigate all partitions even if, say, $n = 20$. We aim for heuristic algorithms.

Synthetic Models with Community Structure

Synthetic Models with Community Structure

In order to evaluate algorithms in a fair and rigorous way, one need a good benchmark: large synthetic network with an engineered ground truth.

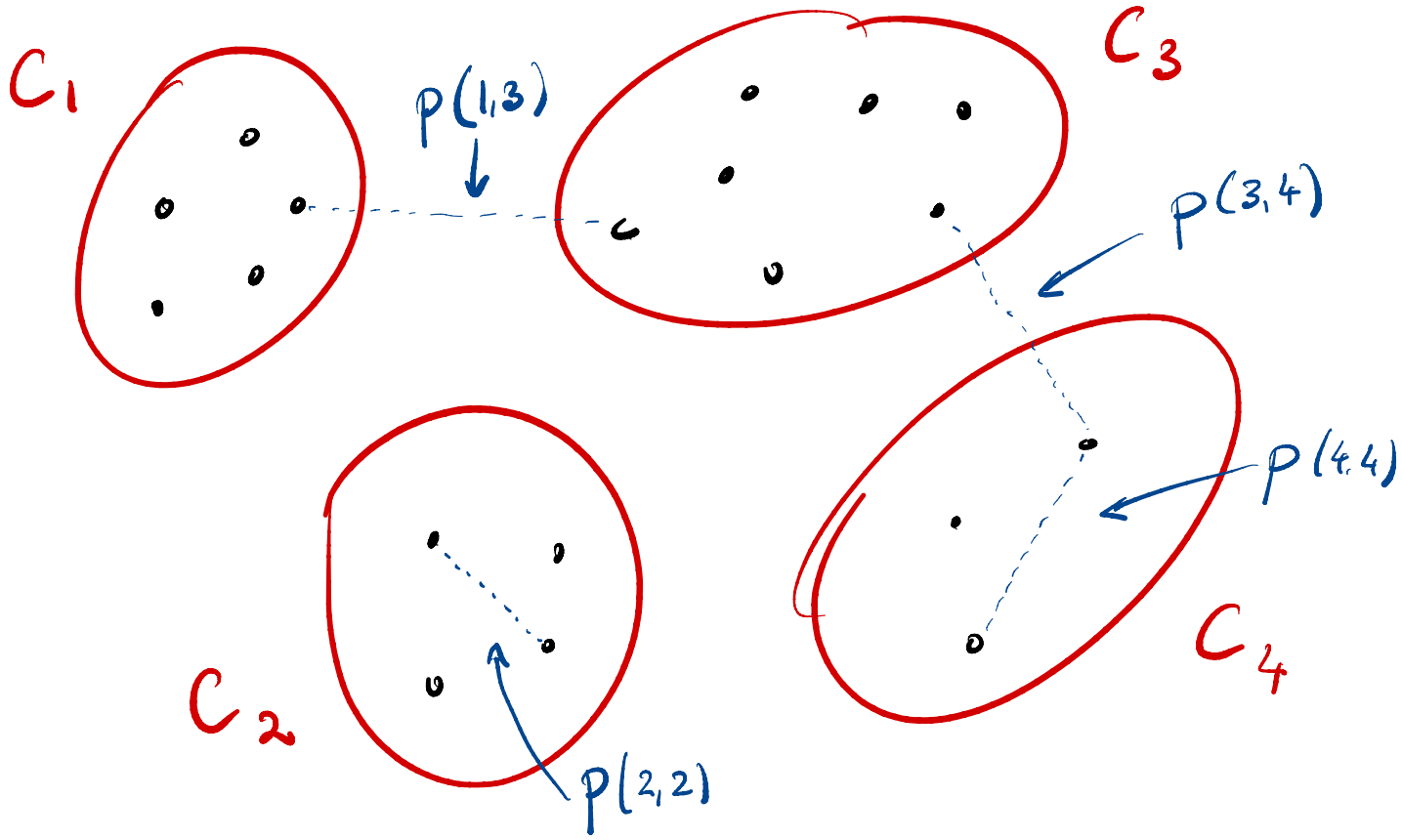
Synthetic Models with Community Structure

In order to evaluate algorithms in a fair and rigorous way, one need a good benchmark: large synthetic network with an engineered ground truth.

We present a few random graph models with community structure and then show how to use them to make such evaluation.

Stochastic Block Model

Simple and natural generalization of $\mathcal{G}(n,p)$.



Stochastic Block Model

Simple and natural generalization of $\mathcal{G}(n,p)$.

Stochastic Block Model

Let $\mathbf{P} = (p(i,j))_{i,j\in[\ell]}$ be a symmetric $\ell \times \ell$ matrix with all entries in $[0,1]$ and let $\mathbf{N} = (n_i)_{i\in[\ell]}$ be a vector of ℓ natural numbers. The stochastic block model $\mathcal{S}(\mathbf{P},\mathbf{N})$ can be generated by starting with an empty graph on the set of nodes $[n] = \{1,2,\ldots,n\}$ with $n = \sum_{i\in[\ell]} n_i$ that is partitioned into ℓ subsets $C_1, C_2, \ldots, C_\ell$, called communities, with $|C_i| = n_i$, $i \in [\ell]$. For each pair of nodes $u, v \in [n]$ with $u < v$, $u \in C_i$, and $v \in C_j$, we independently introduce an edge uv in $\mathcal{S}(\mathbf{P},\mathbf{N})$ with probability $p(i,j)$.

If $\ell = 1$, then we recover $\mathcal{G}(n,p)$.

Lancichinetti–Fortunato–Radicchi (LFR) Model

Problem with $\mathcal{S}(\mathbf{P}, \mathbf{N})$: all nodes that belong to the same community have exactly the same distributions of their degrees—not a realistic assumption.

Lancichinetti–Fortunato–Radicchi (LFR) Model

Problem with $\mathcal{S}(\mathbf{P}, \mathbf{N})$: all nodes that belong to the same community have exactly the same distributions of their degrees—not a realistic assumption.

LFR Model

Let $\gamma \in [2, 3]$, $\tau \in [1, 2]$, and $\mu \in [0, 1]$ (μ is called the mixing parameter). The LFR model $\mathcal{L}(\gamma, \tau, \mu)$ generates random graphs with community sizes and node degrees following a truncated power law distributions with exponents τ and, respectively, γ. Each node aims to have a μ fraction of its neighbours outside of its own community.

Lancichinetti–Fortunato–Radicchi (LFR) Model

Problem with $\mathcal{S}(\mathbf{P}, \mathbf{N})$: all nodes that belong to the same community have exactly the same distributions of their degrees—not a realistic assumption.

LFR Model

Let $\gamma \in [2, 3]$, $\tau \in [1, 2]$, and $\mu \in [0, 1]$ (μ is called the mixing parameter). The LFR model $\mathcal{L}(\gamma, \tau, \mu)$ generates random graphs with community sizes and node degrees following a truncated power law distributions with exponents τ and, respectively, γ. Each node aims to have a μ fraction of its neighbours outside of its own community.

LFR is a standard benchmark for experimental studies.
See the book for details of the process of generating LFR.

Some potential issues of **LFR**

Scalability:

It uses the edge switching Markov chain algorithm to obtain the desired community structure once the stationary distribution is reached. The convergence process is inherently slow and so the model has clear scalability limitations that are known to both academics and practitioners.

Some potential issues of **LFR**

Many Variants and Lack of Theoretical Foundations:

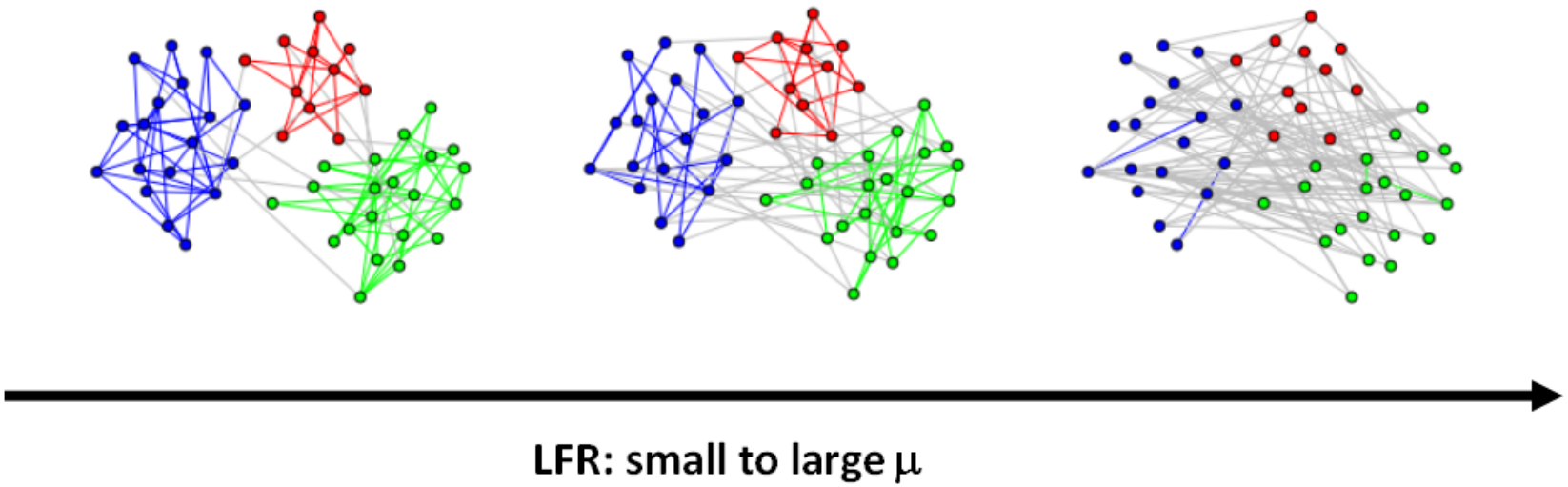
Many fast implementations stop the process before the stationary distribution is reached.

As a result,
a) there are many variants of this benchmark;
b) it is challenging to analyze it theoretically.

Some potential issues of **LFR**

Communities are Unnaturally-defined:

The mixing parameter μ, the main parameter guiding the strength of the communities, has a non-obvious interpretation and so can lead to unnaturally-defined networks.



Some potential issues of **LFR**

Densities of Communities:

It generates a graph in which $(1 - \mu)$ fraction of edges adjacent to a given node stays within the community of that node, regardless whether the community is large or small community. (Do you remember the story about Alice and Bob?)

As a result, small communities will become much denser!

Artificial Benchmark for Community Detection (ABCD)

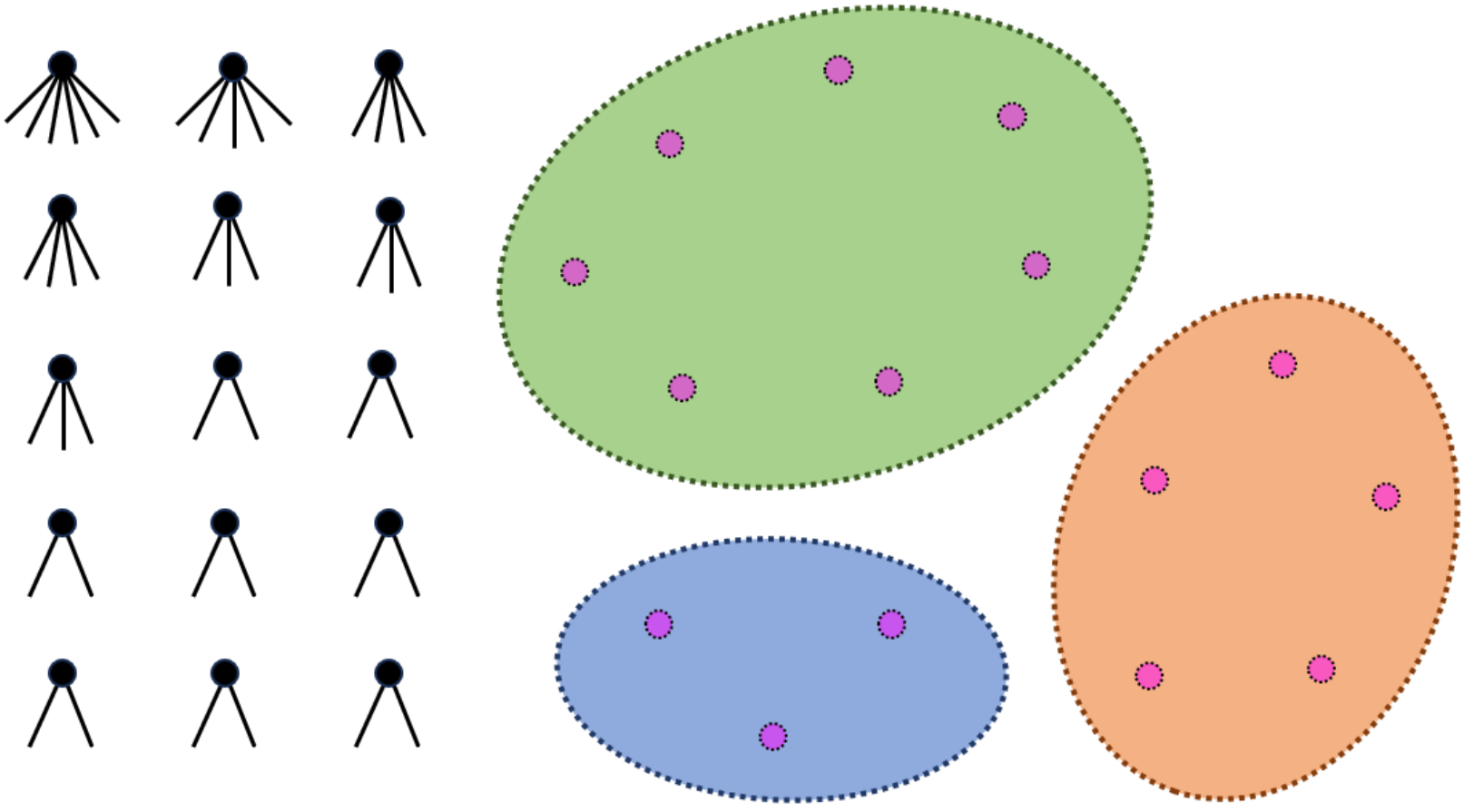
Here is an alternative.

ABCD model

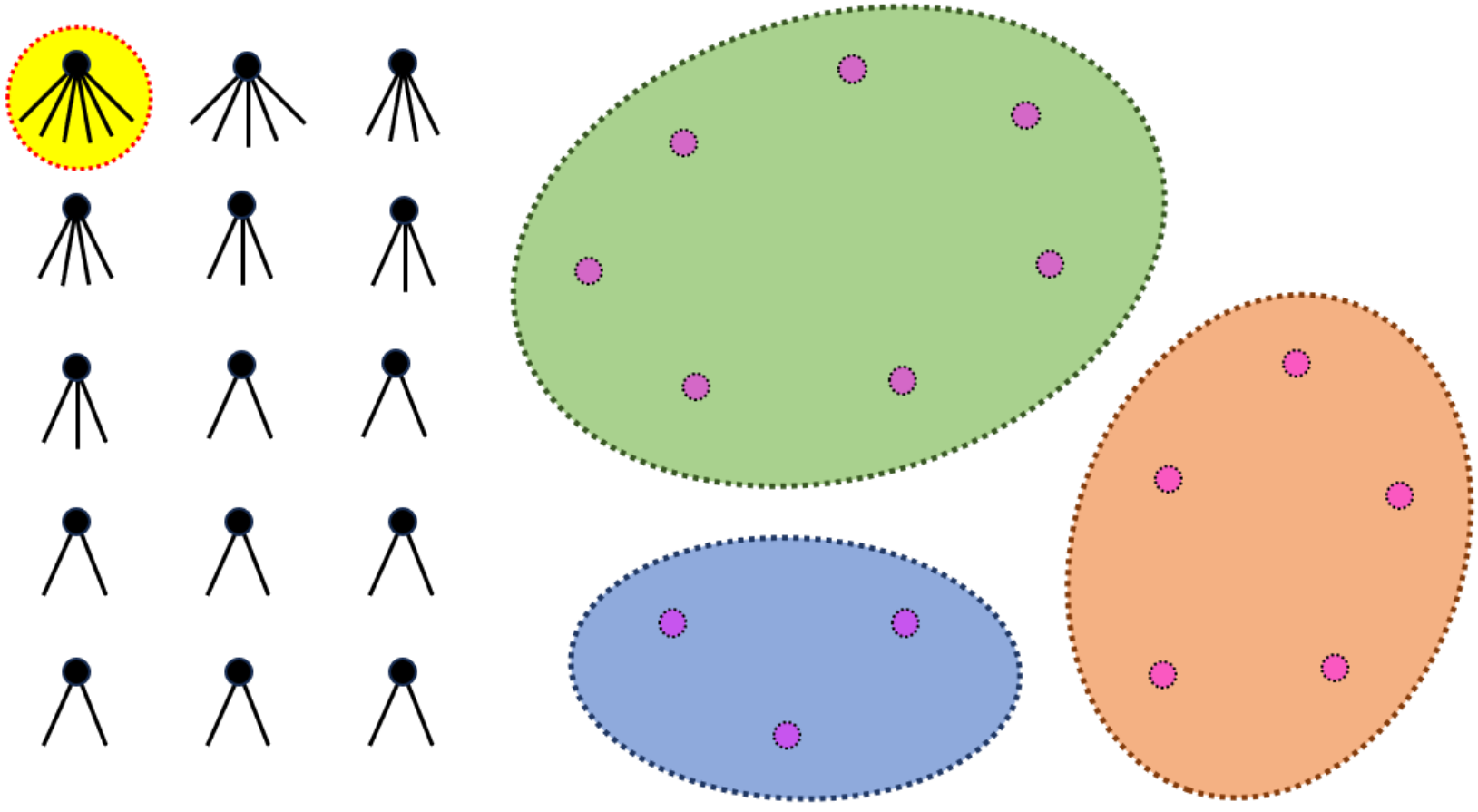
Let $\gamma \in [2,3]$, $\tau \in [1,2]$, and $\xi \in [0,1]$ (ξ is called the mixing parameter). The ABCD model $\mathscr{A}(\gamma, \tau, \mu)$ generates random graphs with community sizes and node degrees following truncated power law distributions with exponents τ and, respectively, γ. The mixing parameter ξ controls the fraction of edges that are between communities.

The high level description is almost the same as for LFR but, of course, details differ—see the book.

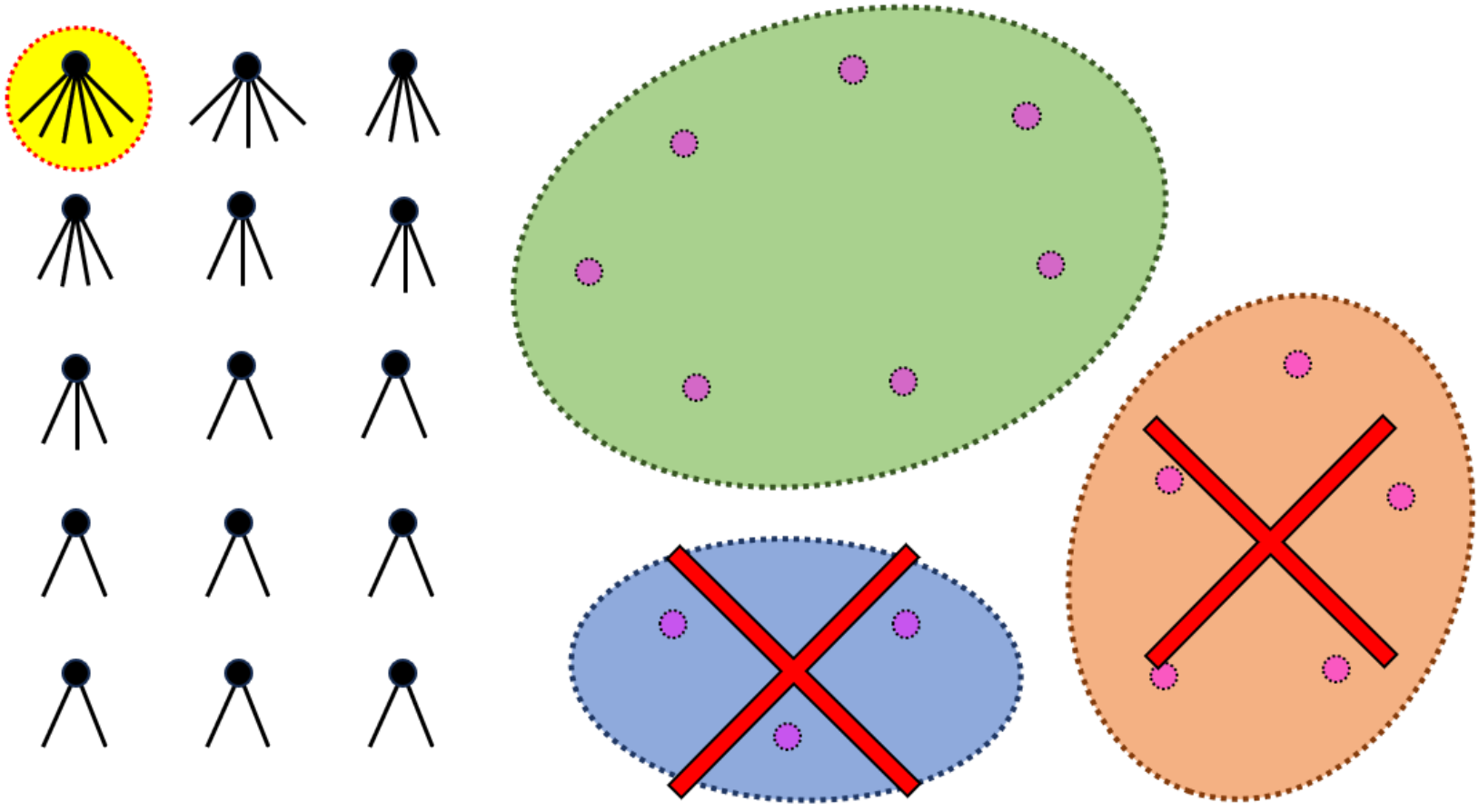
Artificial **B**enchmark for **C**ommunity **D**etection (**ABCD**)



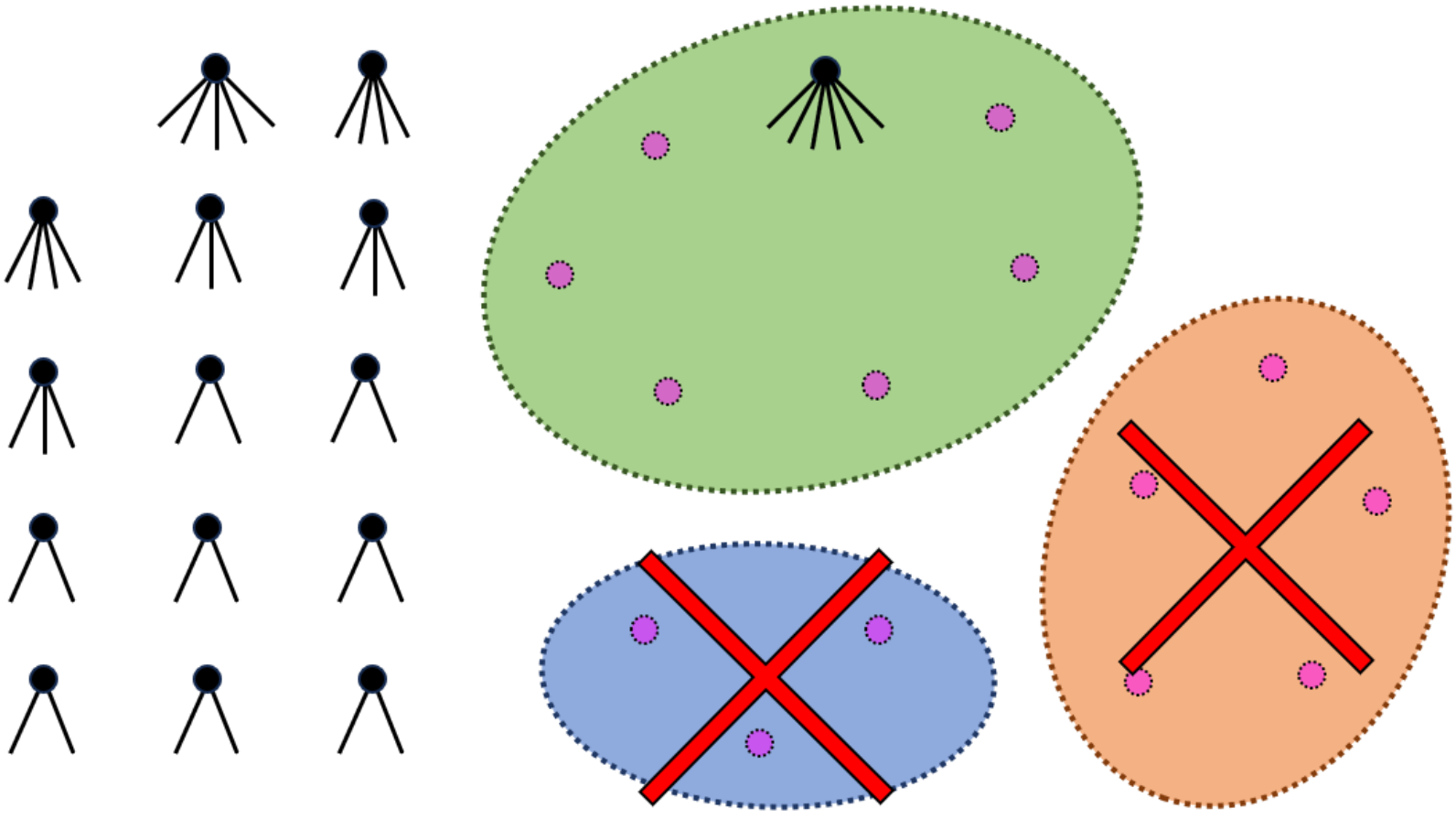
Artificial **B**enchmark for **C**ommunity **D**etection (**ABCD**)



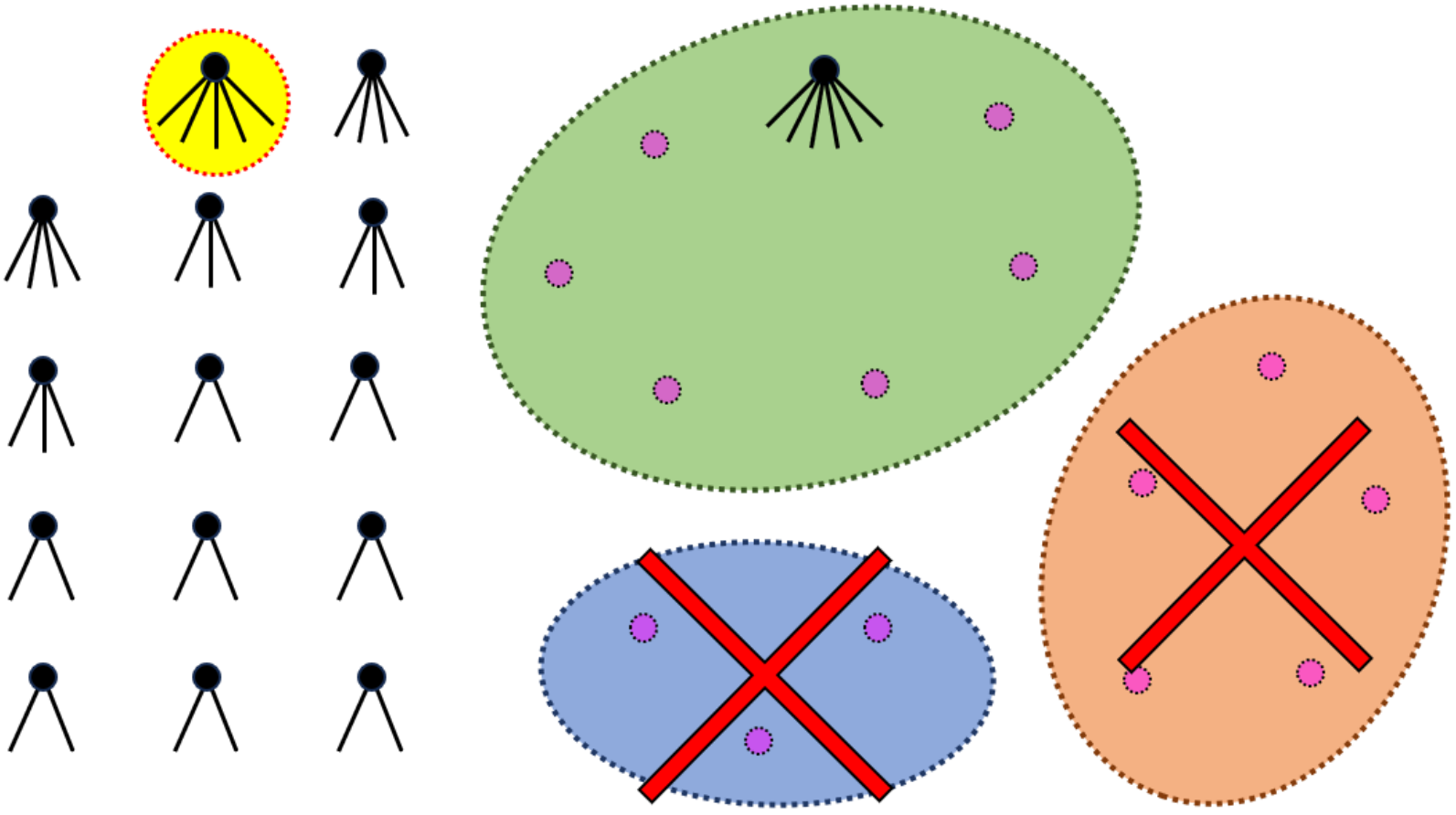
Artificial **B**enchmark for **C**ommunity **D**etection (**ABCD**)



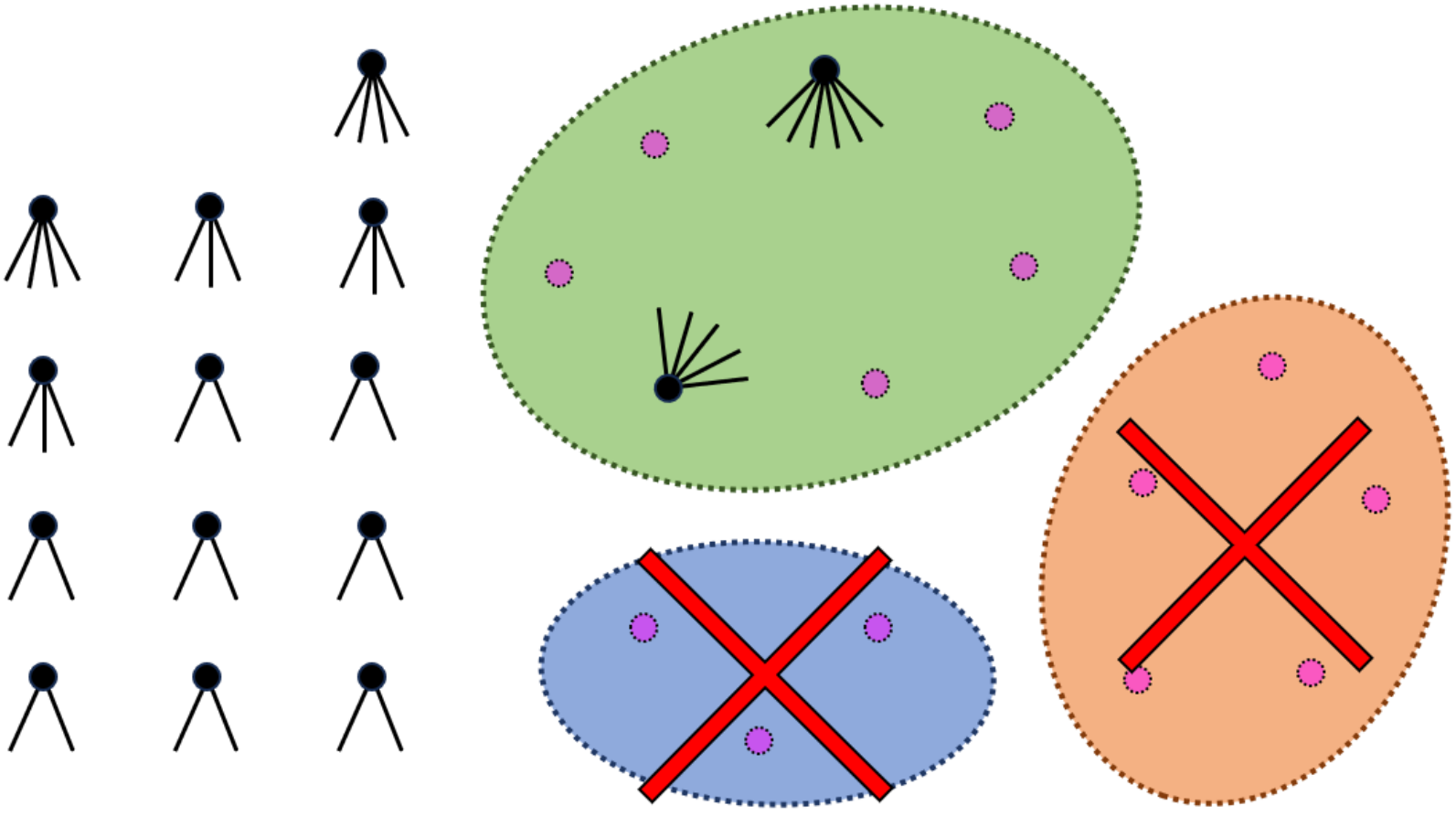
Artificial **B**enchmark for **C**ommunity **D**etection (**ABCD**)



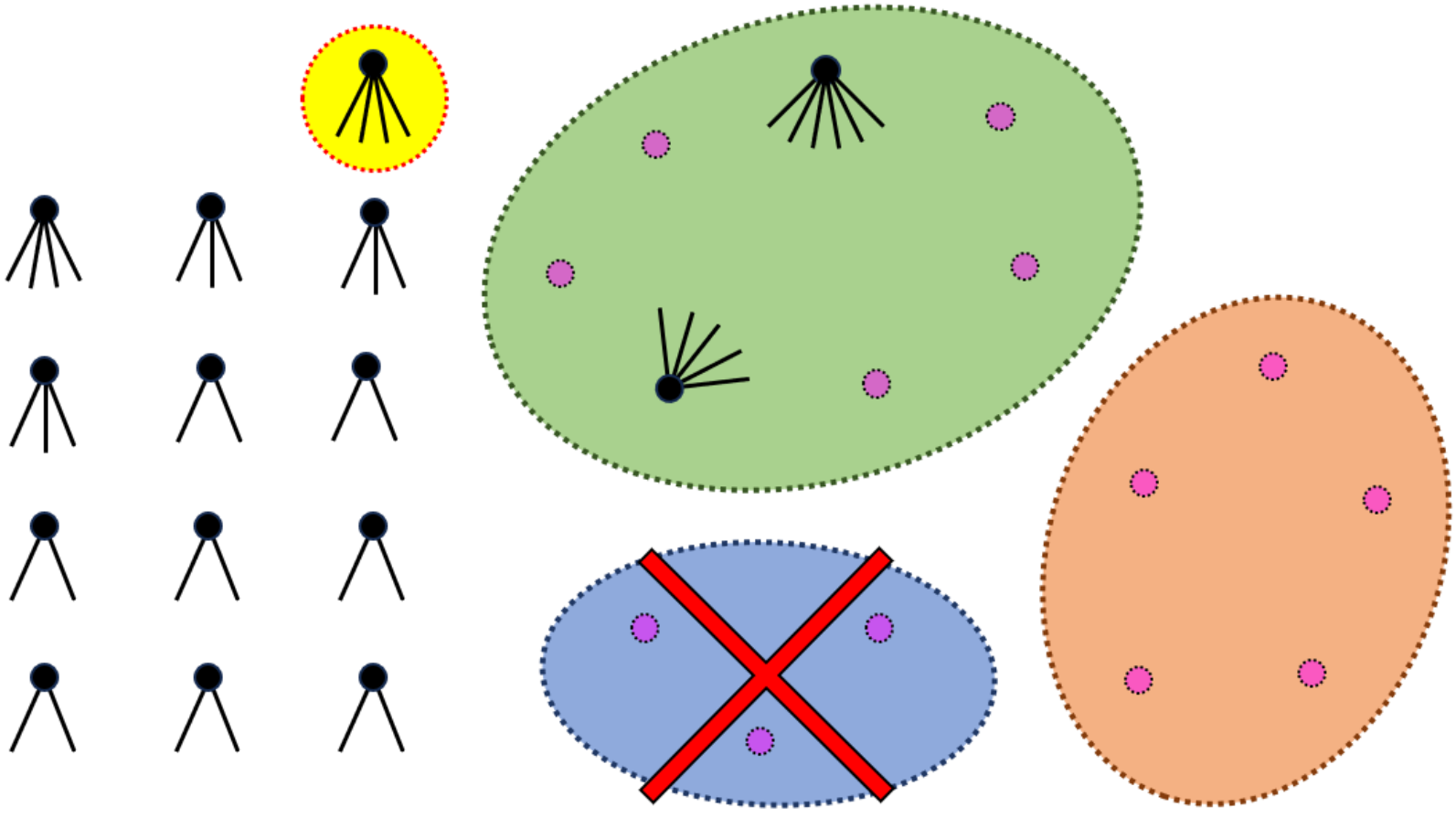
Artificial **B**enchmark for **C**ommunity **D**etection (**ABCD**)



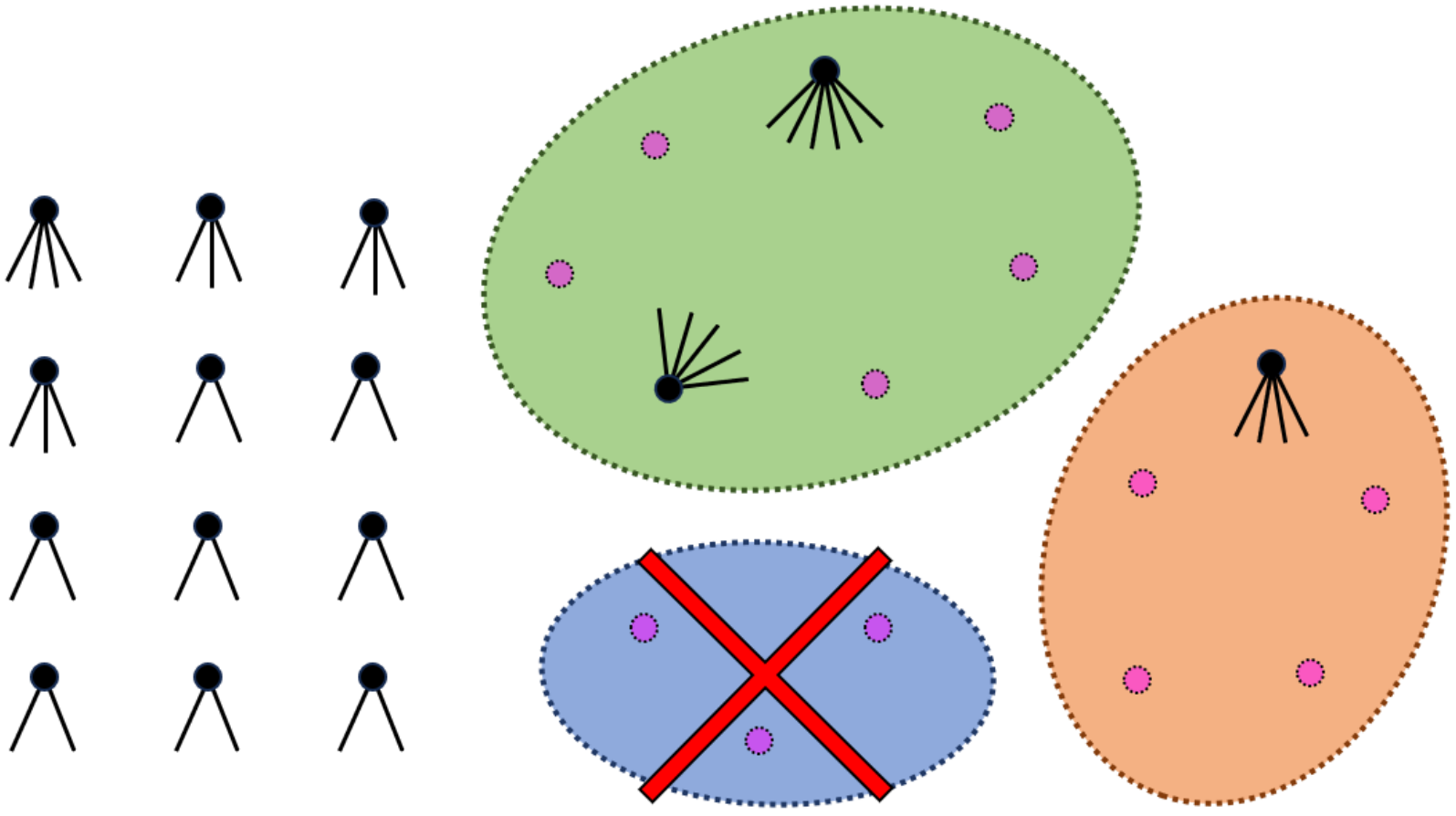
Artificial **B**enchmark for **C**ommunity **D**etection (**ABCD**)



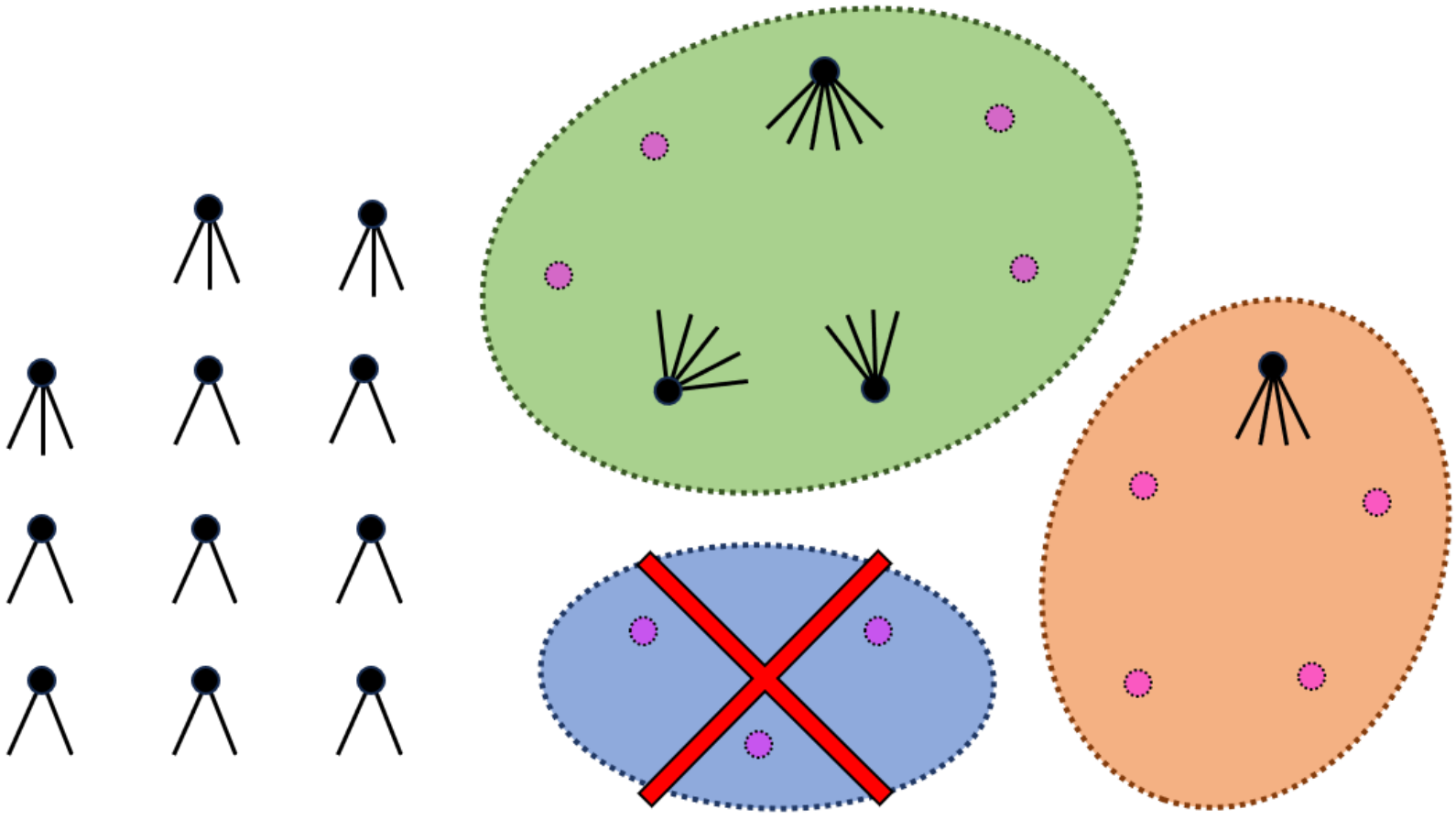
Artificial **B**enchmark for **C**ommunity **D**etection (**ABCD**)



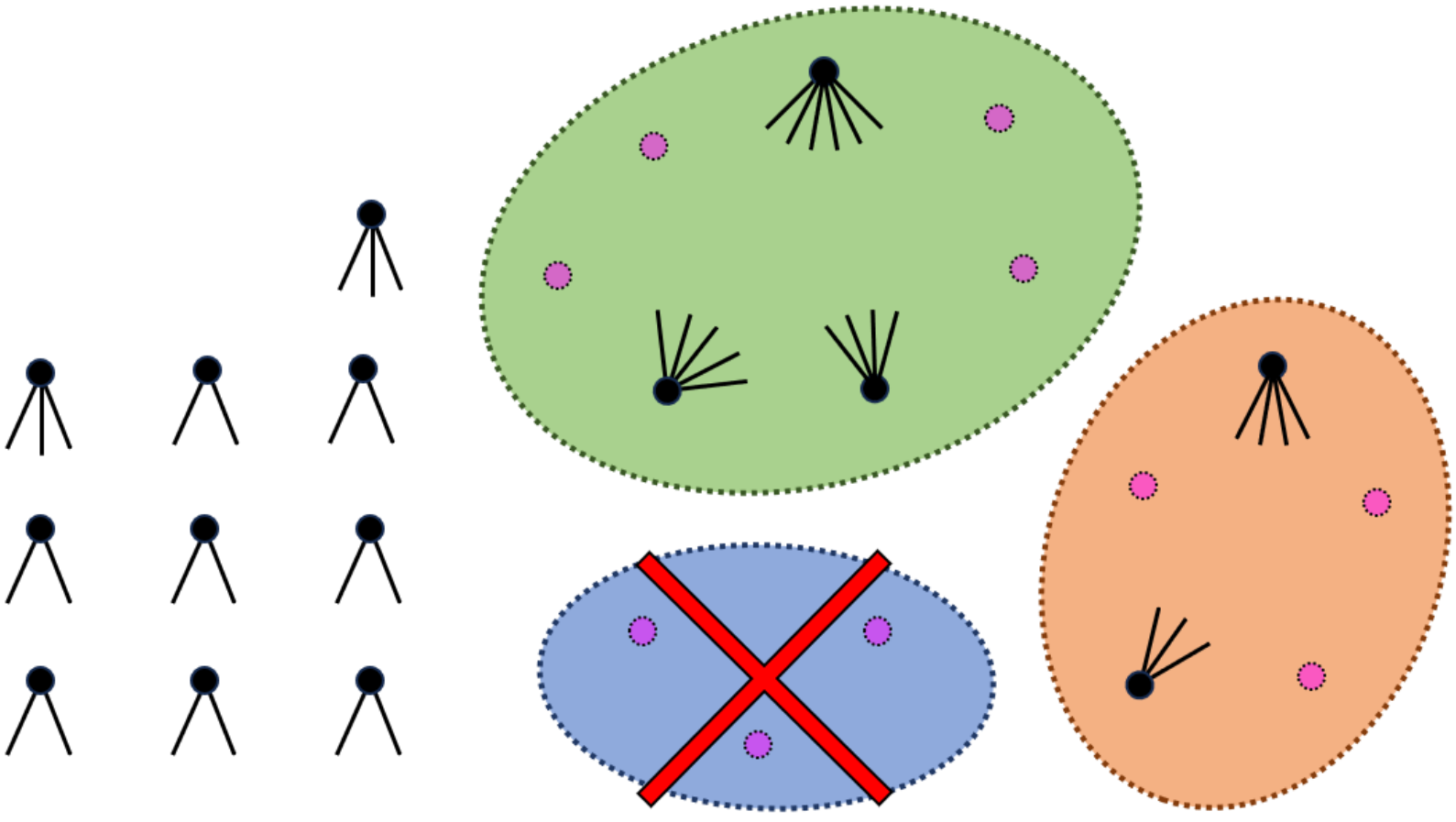
Artificial **B**enchmark for **C**ommunity **D**etection (**ABCD**)



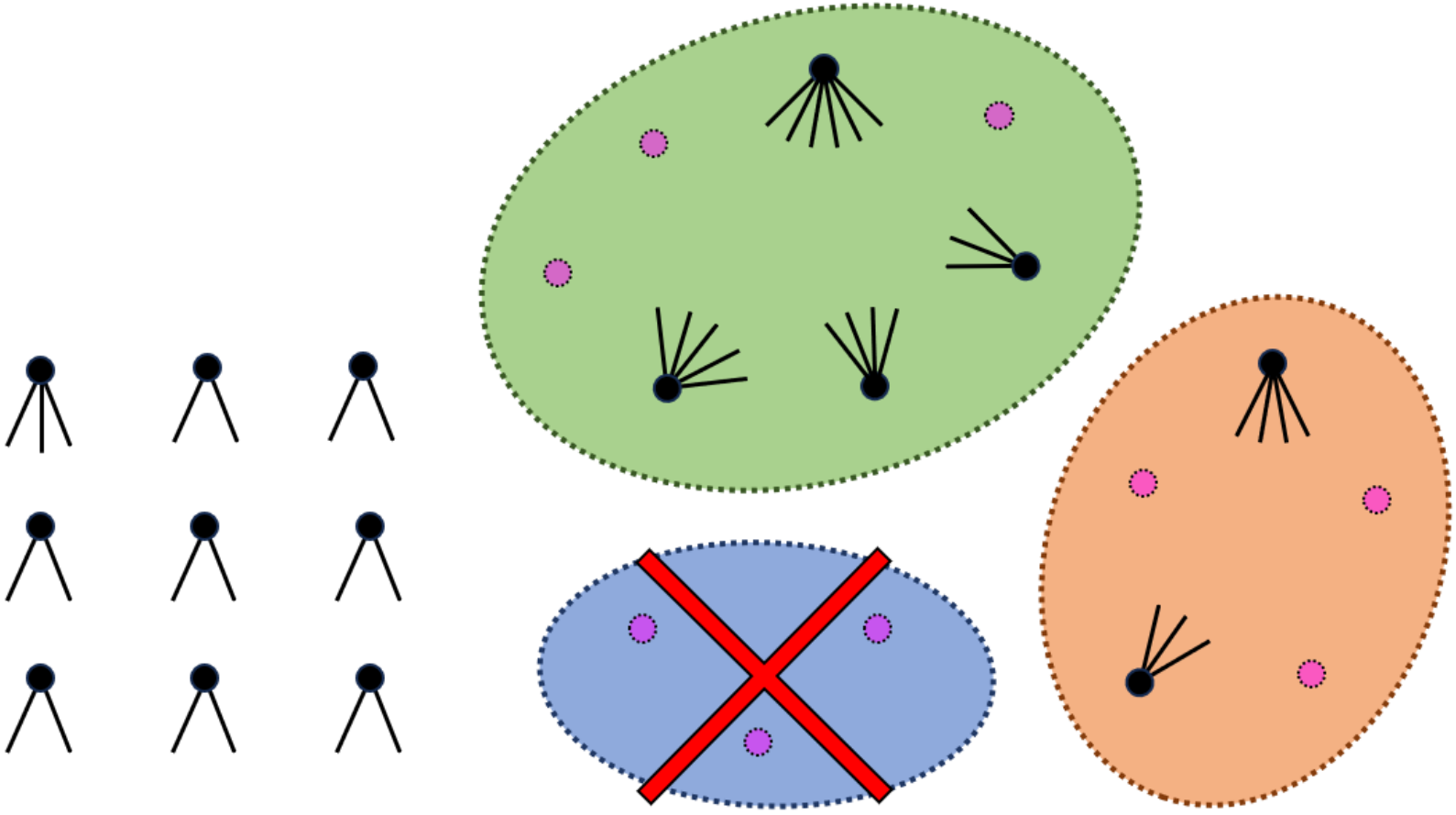
Artificial **B**enchmark for **C**ommunity **D**etection (**ABCD**)



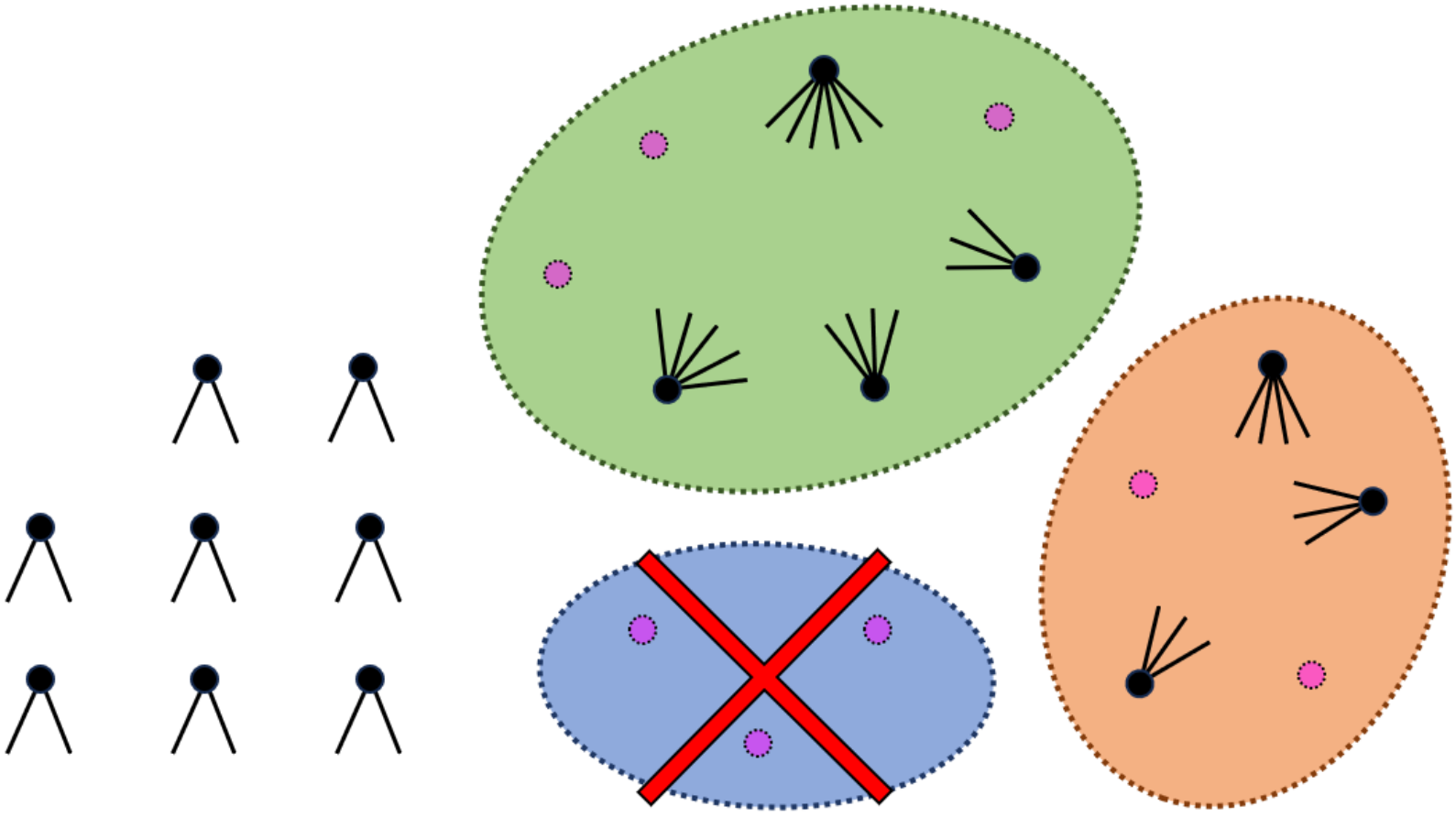
Artificial **B**enchmark for **C**ommunity **D**etection (**ABCD**)



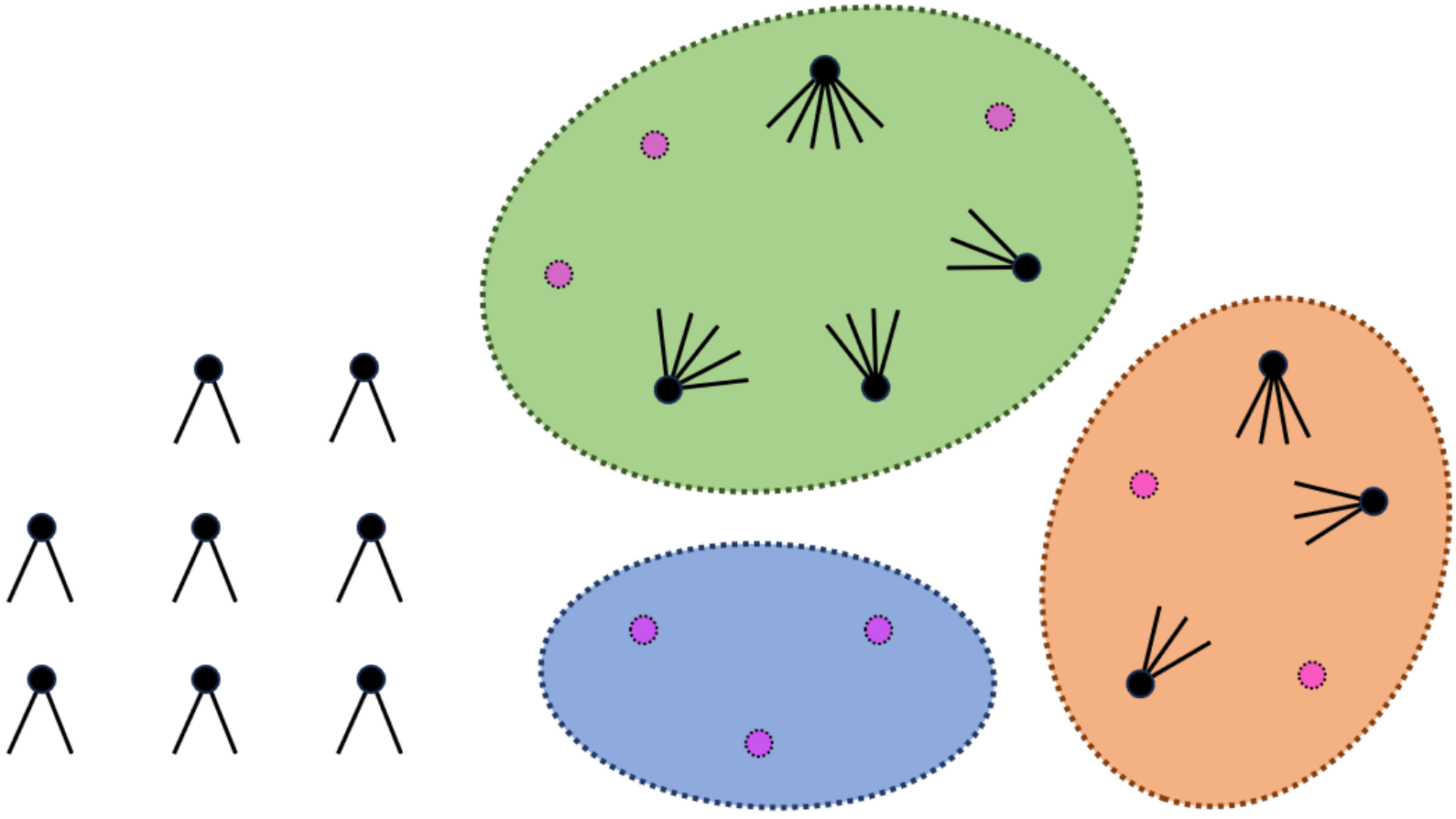
Artificial **B**enchmark for **C**ommunity **D**etection (**ABCD**)



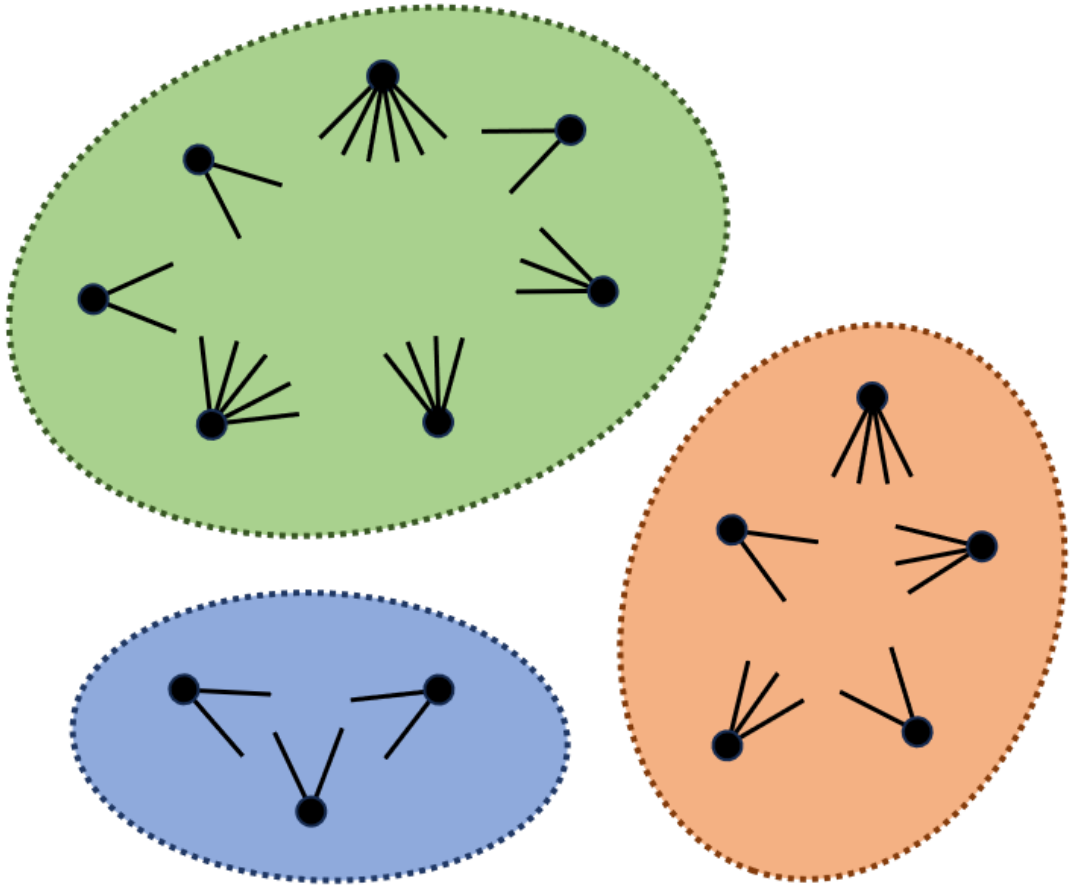
Artificial **B**enchmark for **C**ommunity **D**etection (**ABCD**)



Artificial **B**enchmark for **C**ommunity **D**etection (**ABCD**)

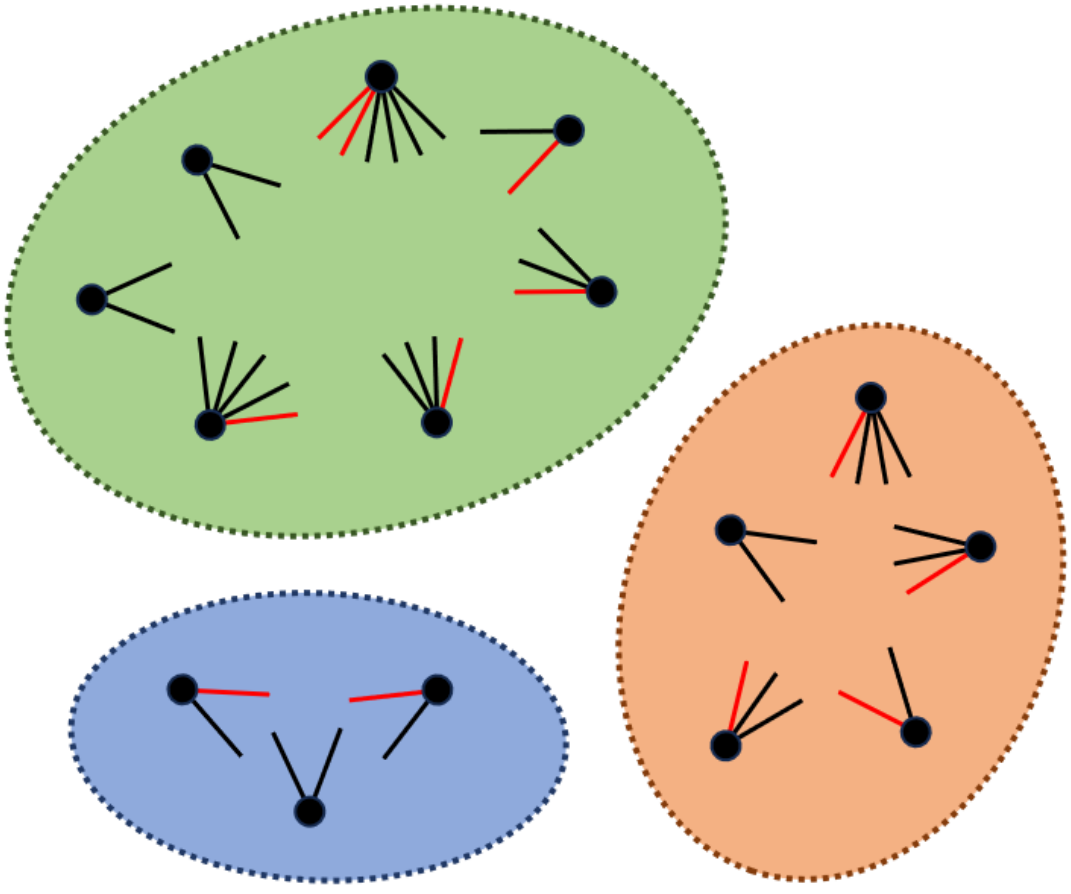


Artificial **B**enchmark for **C**ommunity **D**etection (**ABCD**)

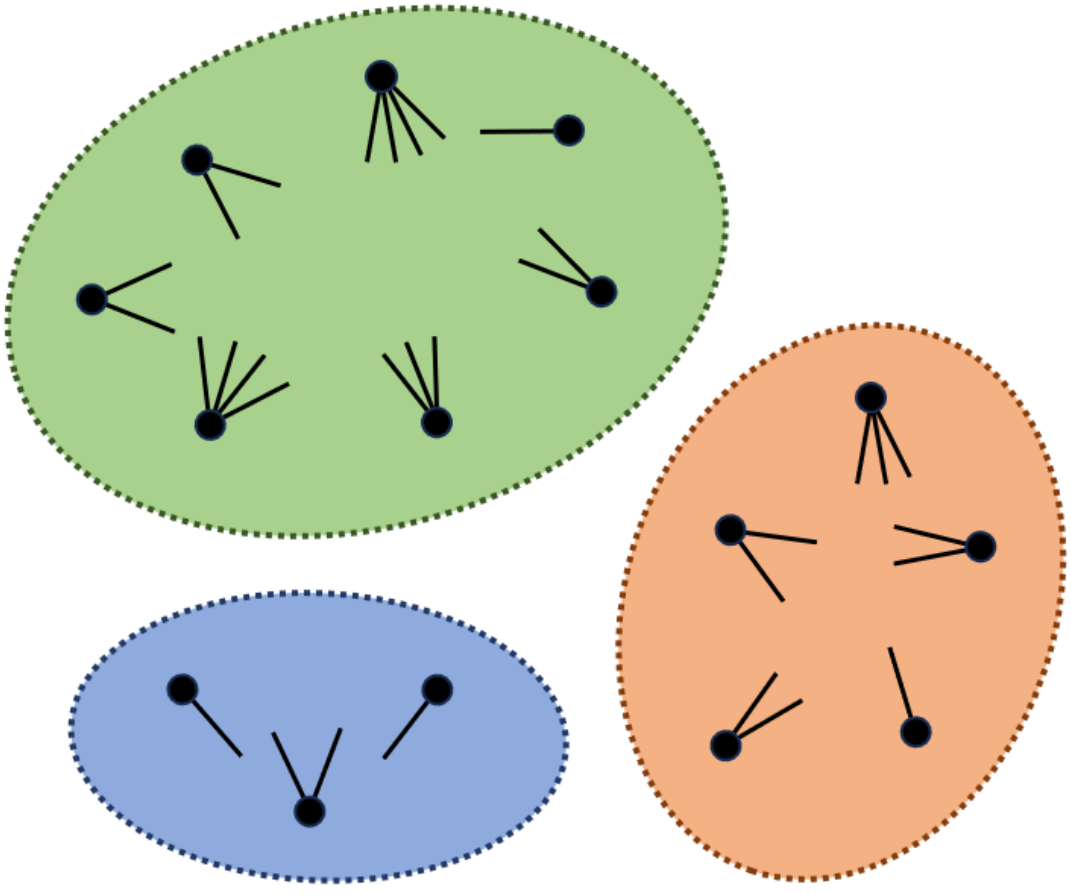


Artificial **B**enchmark for **C**ommunity **D**etection (**ABCD**)

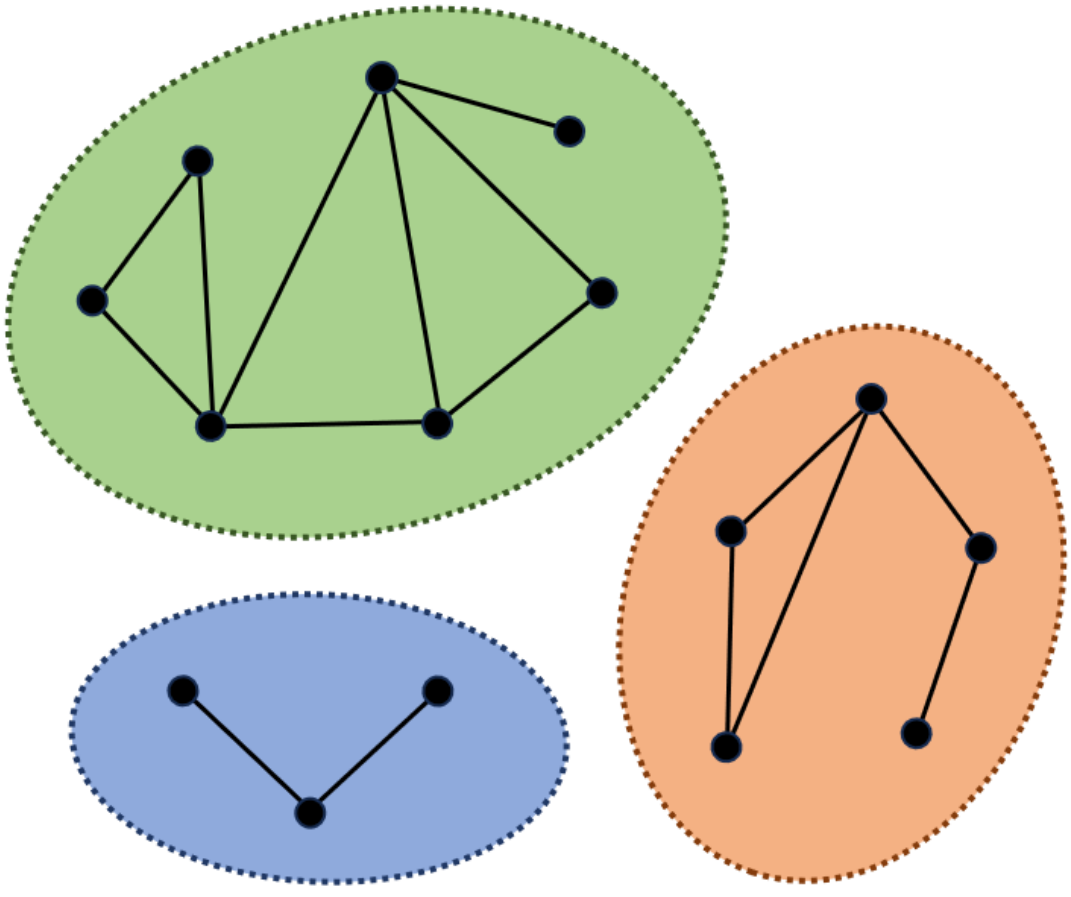
$\xi = 0.25$



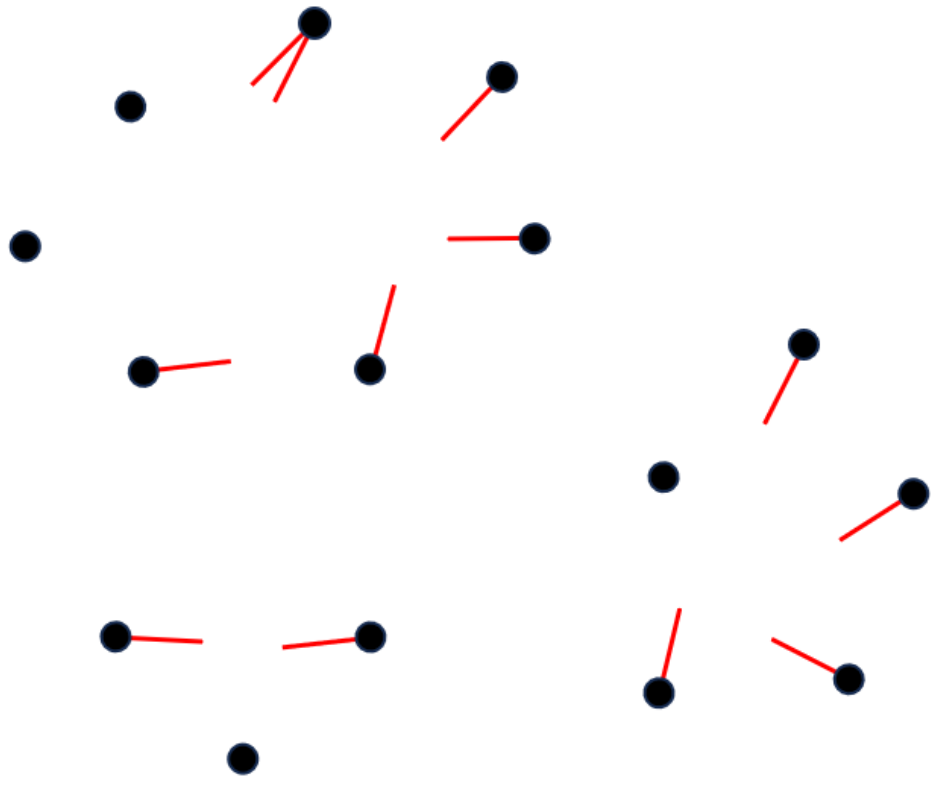
Artificial **B**enchmark for **C**ommunity **D**etection (**ABCD**)



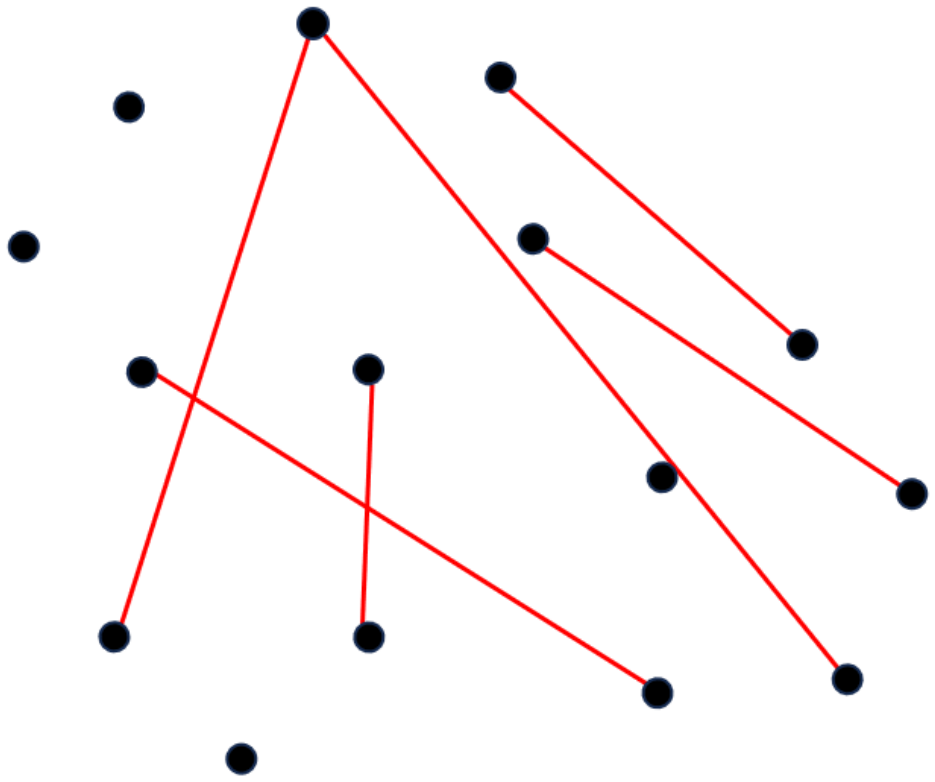
Artificial **B**enchmark for **C**ommunity **D**etection (**ABCD**)



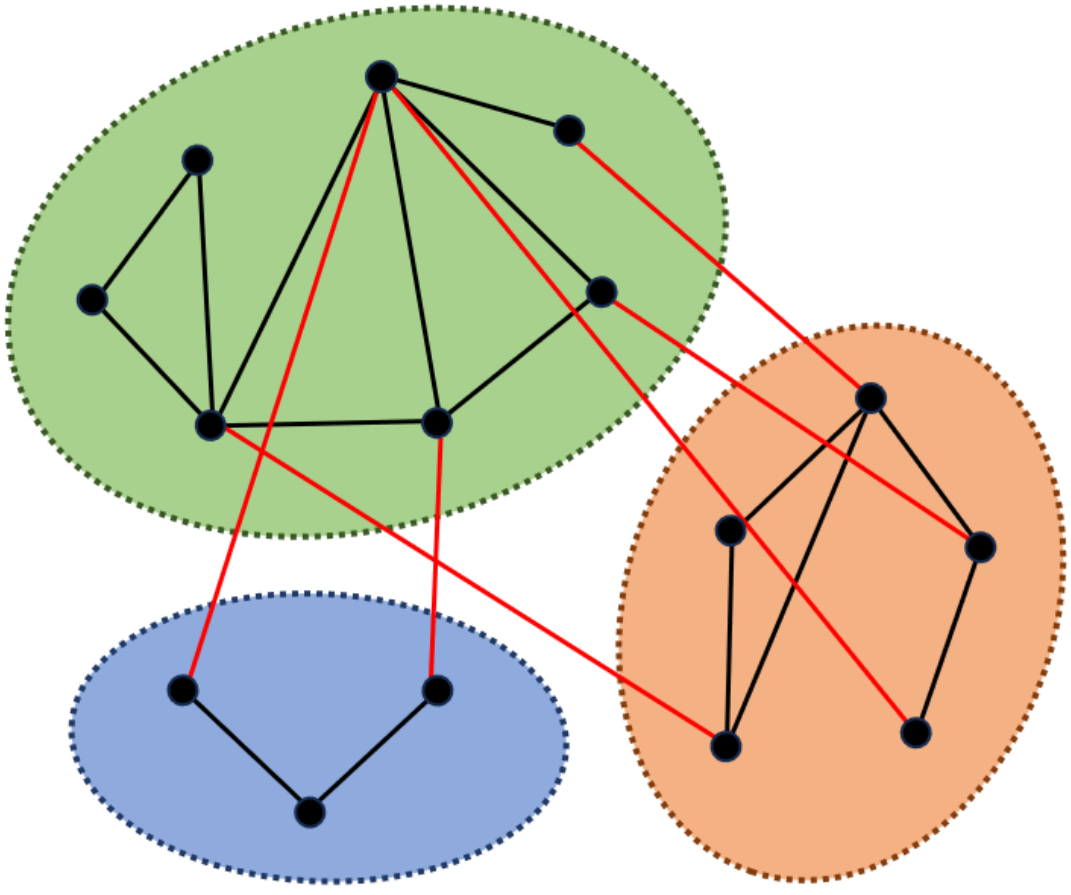
Artificial **B**enchmark for **C**ommunity **D**etection (**ABCD**)



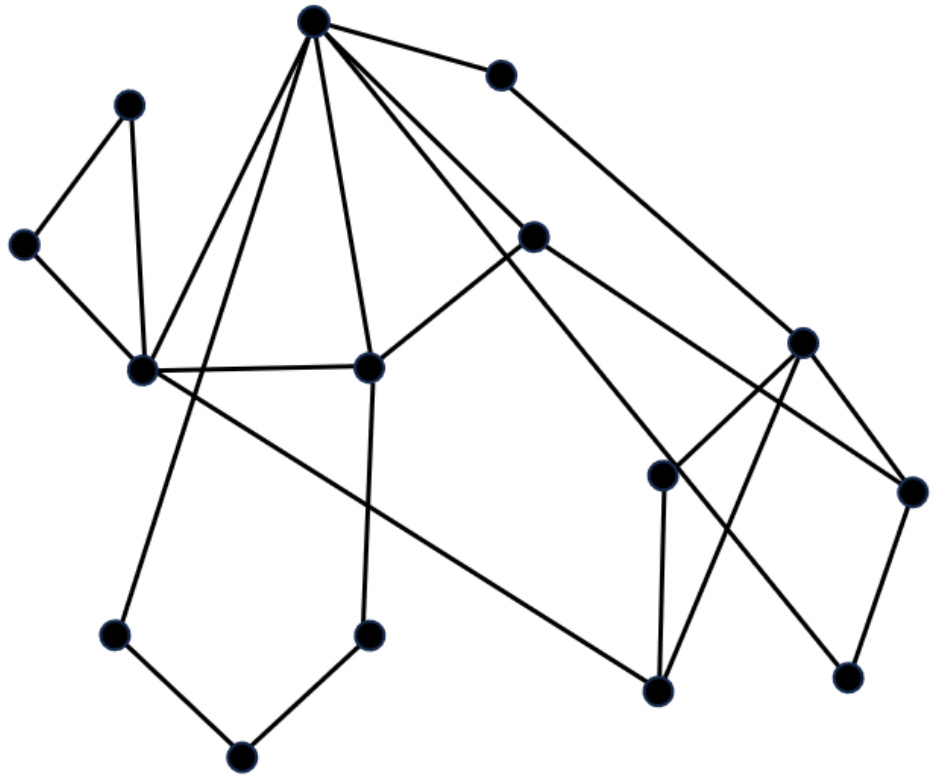
Artificial **B**enchmark for **C**ommunity **D**etection (**ABCD**)



Artificial **B**enchmark for **C**ommunity **D**etection (**ABCD**)



Artificial **B**enchmark for **C**ommunity **D**etection (**ABCD**)



Artificial **B**enchmark for **C**ommunity **D**etection (**ABCD**)

The building blocks in the model are flexible.

- Outliers: ABCD+o
- Hypergraphs: h-ABCD
- Outliers + Overlapping: ABCD+o^2
- Multilayered networks: mABCD
- Fast implementation that uses multiple threads: ABCDe

Evaluating the Quality of Clustering Algorithms

$\mathcal{U}$ $(U_1 \cup U_2 \cup \ldots \cup U_u)$ and $\mathcal{W}$ $(W_1 \cup W_2 \cup \ldots \cup W_w)$ are two partitions of the same set of nodes V. Say, $\mathcal{U}$ is ground truth (typically from synthetic model such as ABCD) and $\mathcal{W}$ is a partition returned by the algorithm we wish to evaluate.

Evaluating the Quality of Clustering Algorithms

$\mathcal{U}$ $(U_1 \cup U_2 \cup \ldots \cup U_u)$ and $\mathcal{W}$ $(W_1 \cup W_2 \cup \ldots \cup W_w)$ are two partitions of the same set of nodes V. Say, $\mathcal{U}$ is ground truth (typically from synthetic model such as ABCD) and $\mathcal{W}$ is a partition returned by the algorithm we wish to evaluate.

Two approaches:
1) based on information theory, such as the Adjusted Mutual Information (AMI),
2) based on considering pairs of nodes, such as the Adjusted Rand Index (ARI).

Evaluating the Quality of Clustering Algorithms

$\mathcal{U}$ $(U_1 \cup U_2 \cup \ldots \cup U_u)$ and $\mathcal{W}$ $(W_1 \cup W_2 \cup \ldots \cup W_w)$ are two partitions of the same set of nodes V. Say, $\mathcal{U}$ is ground truth (typically from synthetic model such as ABCD) and $\mathcal{W}$ is a partition returned by the algorithm we wish to evaluate.

Two approaches:
1) based on information theory, such as the Adjusted Mutual Information (AMI),
2) based on considering pairs of nodes, such as the Adjusted Rand Index (ARI).

They are graph-agnostic as they ignore the graph structure. There are more sophisticated graph-aware measures.

Evaluating the Quality of Clustering Algorithms

The mutual information can be summarized in the form of contingency table, $u \times w$ matrix $N = (n_{ij})_{i \in [u], j \in [w]}$, where $n_{ij} = |U_i \cap W_j|$ (the number of nodes in both U_i and W_j).

Evaluating the Quality of Clustering Algorithms

The mutual information can be summarized in the form of contingency table, $u \times w$ matrix $N = (n_{ij})_{i \in [u], j \in [w]}$, where $n_{ij} = |U_i \cap W_j|$ (the number of nodes in both U_i and W_j).

$P_U(i) = \frac{|U_i|}{n}$ (the probability that a random node belongs to U_i), $P_W(j) = \frac{|W_j|}{n}$, and $P_{UW}(i,j) = \frac{|U_i \cap W_j|}{n} = \frac{n_{ij}}{n}$.

Evaluating the Quality of Clustering Algorithms

The mutual information can be summarized in the form of contingency table, $u \times w$ matrix $N = (n_{ij})_{i \in [u], j \in [w]}$, where $n_{ij} = |U_i \cap W_j|$ (the number of nodes in both U_i and W_j).

$P_U(i) = \frac{|U_i|}{n}$ (the probability that a random node belongs to U_i), $P_W(j) = \frac{|W_j|}{n}$, and $P_{UW}(i,j) = \frac{|U_i \cap W_j|}{n} = \frac{n_{ij}}{n}$.

The Mutual Information (MI) measures the information shared by the two community assignments:

$$MI(\mathcal{U}, \mathcal{W}) = \sum_{i \in [u], j \in [w] : P_{UW}(i,j) > 0} P_{UW}(i,j) \log_2 \left(\frac{P_{UW}(i,j)}{P_U(i) P_W(j)} \right).$$

It tells us how much knowing one of these partitions reduces our uncertainty about the other one.

Evaluating the Quality of Clustering Algorithms

Note that $MI(\mathcal{U}, \mathcal{W}) = 0$ if $\mathcal{U}$ and $\mathcal{W}$ are independent of each other, that is, $P_{UW}(i, j) = P_U(i)P_W(j)$ for all i, j.

Evaluating the Quality of Clustering Algorithms

Note that $MI(\mathcal{U}, \mathcal{W}) = 0$ if $\mathcal{U}$ and $\mathcal{W}$ are independent of each other, that is, $P_{UW}(i, j) = P_U(i)P_W(j)$ for all i, j.

At the other extreme, when the two partitions are identical, $MI(\mathcal{U}, \mathcal{W}) = H(\mathcal{U}) = H(\mathcal{W})$, where

$$H(\mathcal{U}) = -\sum_{i \in [u]} P_U(i) \log_2 \Big(P_U(i) \Big)$$

is the Shannon entropy associated with partition $\mathcal{U}$.

Evaluating the Quality of Clustering Algorithms

Note that $MI(\mathcal{U}, \mathcal{W}) = 0$ if $\mathcal{U}$ and $\mathcal{W}$ are independent of each other, that is, $P_{UW}(i, j) = P_U(i)P_W(j)$ for all i, j.

At the other extreme, when the two partitions are identical, $MI(\mathcal{U}, \mathcal{W}) = H(\mathcal{U}) = H(\mathcal{W})$, where

$$H(\mathcal{U}) = -\sum_{i \in [u]} P_U(i) \log_2 \Big(P_U(i) \Big)$$

is the Shannon entropy associated with partition $\mathcal{U}$.

In any case, $MI(\mathcal{U}, \mathcal{W})$ is upper bounded by the entropies $H(\mathcal{U})$ and $H(\mathcal{W})$. This observation justifies the next definition.

Evaluating the Quality of Clustering Algorithms

Normalized mutual information

Let $\mathcal{U}$ and $\mathcal{W}$ be two partitions of the same set of nodes V. For $i \in [u]$ and $j \in [w]$, let $n_i = |U_i|$, $n'_j = |W_j|$, and $n_{ij} = |U_i \cap W_j|$. The normalized mutual information between $\mathcal{U}$ and $\mathcal{W}$ is defined as follows:

$$NMI(\mathcal{U}, \mathcal{W}) = \frac{MI(\mathcal{U}, \mathcal{W})}{(H(\mathcal{U}) + H(\mathcal{W}))/2}$$

$$= \frac{-2 \sum_{i \in [u]} \sum_{j \in [w]} n_{ij} \log_2 \left(\frac{n \cdot n_{ij}}{n_i \cdot n'_j} \right)}{\sum_{i \in [u]} n_i \log_2(n_i/n) + \sum_{j \in [w]} n'_j \log_2(n'_j/n)}.$$

Evaluating the Quality of Clustering Algorithms

Normalized mutual information

Let $\mathcal{U}$ and $\mathcal{W}$ be two partitions of the same set of nodes V. For $i \in [u]$ and $j \in [w]$, let $n_i = |U_i|$, $n'_j = |W_j|$, and $n_{ij} = |U_i \cap W_j|$. The normalized mutual information between $\mathcal{U}$ and $\mathcal{W}$ is defined as follows:

$$NMI(\mathcal{U}, \mathcal{W}) = \frac{MI(\mathcal{U}, \mathcal{W})}{(H(\mathcal{U}) + H(\mathcal{W}))/2}$$

Alternatively, one could normalize it by $\min\{H(\mathcal{U}), H(\mathcal{W})\}$ or $\max\{H(\mathcal{U}), H(\mathcal{W})\}$.

Evaluating the Quality of Clustering Algorithms

Normalized mutual information

Let $\mathcal{U}$ and $\mathcal{W}$ be two partitions of the same set of nodes V. For $i \in [u]$ and $j \in [w]$, let $n_i = |U_i|$, $n'_j = |W_j|$, and $n_{ij} = |U_i \cap W_j|$. The normalized mutual information between $\mathcal{U}$ and $\mathcal{W}$ is defined as follows:

$$NMI(\mathcal{U}, \mathcal{W}) = \frac{MI(\mathcal{U}, \mathcal{W})}{(H(\mathcal{U}) + H(\mathcal{W}))/2}$$

For all variants, $NMI(\mathcal{U}, \mathcal{W})$ is zero when the algorithm completely fails and one when the algorithm works perfectly. Moreover, they maintain the symmetry.

Evaluating the Quality of Clustering Algorithms

The Rand index (pair counting based measures) measures the percentage of correct decisions made by the algorithm:

$$RI(\mathcal{U}, \mathcal{W}) = \frac{N_{11} + N_{00}}{N_{00} + N_{10} + N_{01} + N_{11}} = \frac{N_{11} + N_{00}}{\binom{n}{2}},$$

where N_{11} is the number of true positives (pairs of nodes that are in the same community in both $\mathcal{W}$ and $\mathcal{U}$), N_{10} is the number of false negatives, N_{01} is the number of false positives, and N_{00} is the number of true negatives.

Evaluating the Quality of Clustering Algorithms

$$
\begin{aligned}
N_{11} &= \sum_{i \in [u]} \sum_{j \in [w]} \binom{n_{ij}}{2} \\
N_{10} &= \sum_{i \in [u]} \binom{n_i}{2} - N_{11} \\
N_{01} &= \sum_{j \in [w]} \binom{n'_j}{2} - N_{11} \\
N_{00} &= \binom{n}{2} - N_{11} - N_{10} - N_{01} \\
&= \binom{n}{2} + N_{11} - \sum_{i \in [u]} \binom{n_i}{2} - \sum_{j \in [w]} \binom{n'_j}{2}
\end{aligned}
$$

Evaluating the Quality of Clustering Algorithms

Rand index

Let $\mathcal{U}$ and $\mathcal{W}$ be two partitions of the same set of nodes V. For $i \in [u]$ and $j \in [w]$, let $n_i = |U_i|$, $n'_j = |W_j|$, and $n_{ij} = |U_i \cap W_j|$. The Rand index between $\mathcal{U}$ and $\mathcal{W}$ is defined as follows:

$$RI(\mathcal{U}, \mathcal{W}) = \frac{2\sum_{i\in[u]}\sum_{j\in[w]}\binom{n_{ij}}{2} - \sum_{i\in[u]}\binom{n_i}{2} - \sum_{j\in[w]}\binom{n'_j}{2} + \binom{n}{2}}{\binom{n}{2}}.$$

Evaluating the Quality of Clustering Algorithms

Both measures fail to satisfy the following, rather intuitive and desired, property known as the constant baseline property.

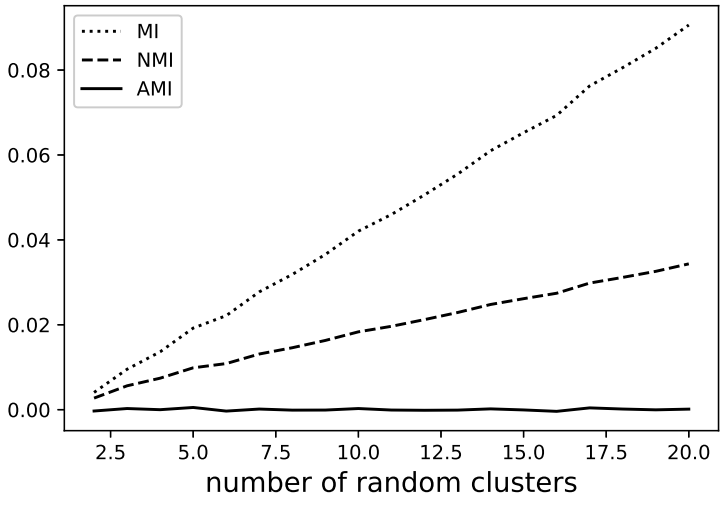
Let $\mathcal{W}_i$ be a partition sampled independently and uniformly at random from the set of partitions into i parts. One would expect that the similarity measure between $\mathcal{U}$ and $\mathcal{W}_i$ should be equal to zero.

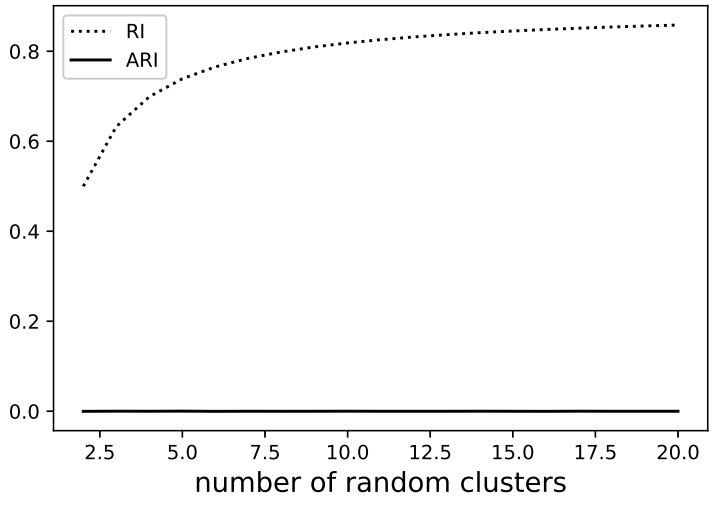
Evaluating the Quality of Clustering Algorithms

Both measures fail to satisfy the following, rather intuitive and desired, property known as the constant baseline property.

Let $\mathcal{W}_i$ be a partition sampled independently and uniformly at random from the set of partitions into i parts. One would expect that the similarity measure between $\mathcal{U}$ and $\mathcal{W}_i$ should be equal to zero.

However, all measures increase with i!





Evaluating the Quality of Clustering Algorithms

This is the main reason why these measures are mainly used in its adjusted form.

Evaluating the Quality of Clustering Algorithms

This is the main reason why these measures are mainly used in its adjusted form.

Generalized hypergeometric model: partitions are generated randomly, subject to having a fixed number of parts and the number of elements in each part.

Evaluating the Quality of Clustering Algorithms

This is the main reason why these measures are mainly used in its adjusted form.

Generalized hypergeometric model: partitions are generated randomly, subject to having a fixed number of parts and the number of elements in each part.

$$\mathbb{E}\left[MI(\mathcal{U},\mathcal{W})\right] = \sum_{i\in[u]}\sum_{j\in[w]}\sum_{n_{ij}=\max\{n_i+n'_j-n,1\}}^{\min\{n_i,n'_j\}} \frac{n_{ij}}{n}\log_2\left(\frac{n\cdot n_{ij}}{n_i\cdot n'_j}\right) \times \frac{n_i!\,n'_j!\,(n-n_i)!\,(n-n'_j)!}{n!\,n_{ij}!\,(n_i-n_{ij})!\,(n'_j-n_{ij})!\,(n-n_i-n'_j+n_{ij})!}.$$

Evaluating the Quality of Clustering Algorithms

Adjusted Mutual Information (AMI):

$$AMI(\mathcal{U}, \mathcal{W}) = \frac{MI(\mathcal{U}, \mathcal{W}) - \mathbb{E}\left[MI(\mathcal{U}, \mathcal{W})\right]}{\max\{H(\mathcal{U}), H(\mathcal{W})\} - \mathbb{E}\left[MI(\mathcal{U}, \mathcal{W})\right]}.$$

AMI is equal to one when the two partitions are identical and equal to zero when the MI between two partitions equals the value expected due to chance alone.

Evaluating the Quality of Clustering Algorithms

Adjusted Mutual Information (AMI):

$$AMI(\mathcal{U}, \mathcal{W}) = \frac{MI(\mathcal{U}, \mathcal{W}) - \mathbb{E}\left[MI(\mathcal{U}, \mathcal{W})\right]}{\max\{H(\mathcal{U}), H(\mathcal{W})\} - \mathbb{E}\left[MI(\mathcal{U}, \mathcal{W})\right]}.$$

AMI is equal to one when the two partitions are identical and equal to zero when the MI between two partitions equals the value expected due to chance alone.

Adjusted Rand Index (ARI) is defined similarly.

Graph Modularity

Definition

Modularity for graphs is based on the comparison between:

a) the actual density of edges inside a community (the edge contribution), and

b) the expected density if nodes of the graph were wired randomly, regardless of community structure (the degree tax).

Definition

Modularity for graphs is based on the comparison between:

a) the actual density of edges inside a community (the edge contribution), and

b) the expected density if nodes of the graph were wired randomly, regardless of community structure (the degree tax).

Such reference random graph is known in this context as the null-model. We will use the Chung-Lu model.

Definition

Modularity for graphs is based on the comparison between:

a) the actual density of edges inside a community (the edge contribution), and

b) the expected density if nodes of the graph were wired randomly, regardless of community structure (the degree tax).

Such reference random graph is known in this context as the null-model. We will use the Chung-Lu model.

Despite some known issues with the modularity function (such as the "resolution limit"), many popular algorithms use it.

Definition

Modularity for graphs is based on the comparison between:

a) the actual density of edges inside a community (the edge contribution), and

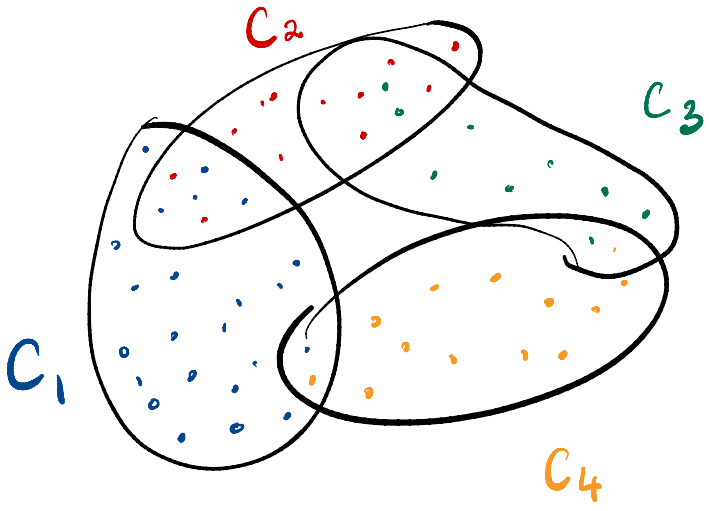
b) the expected density if nodes of the graph were wired randomly, regardless of community structure (the degree tax).

Such reference random graph is known in this context as the null-model. We will use the Chung-Lu model.

Despite some known issues with the modularity function (such as the "resolution limit"), many popular algorithms use it.

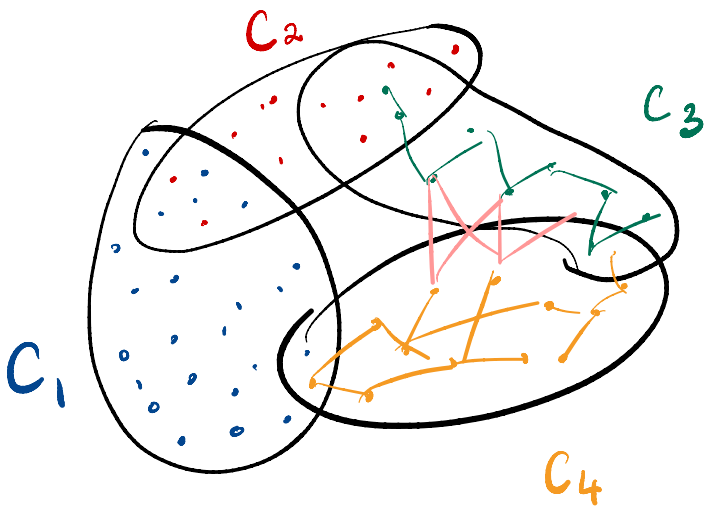
Can be easily generalized to directed graphs.

Definition



Let $\mathcal{A} = \{A_1, A_2, \ldots, A_\ell\}$ be a given partition of V.

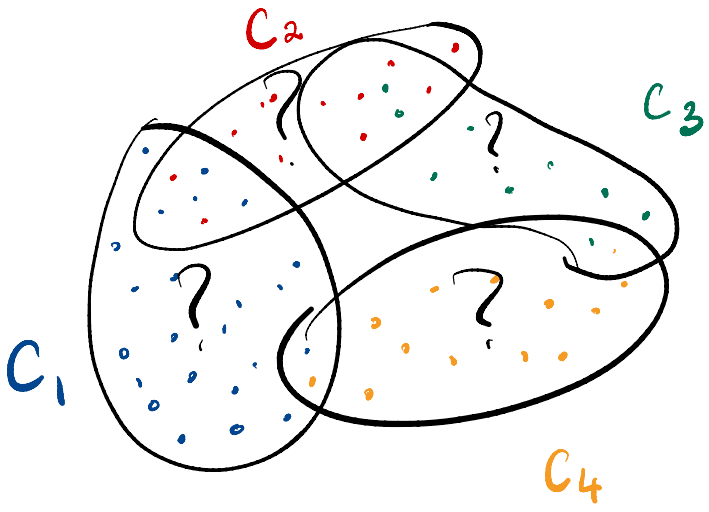
Definition



Let $\mathscr{A} = \{A_1, A_2, \ldots, A_\ell\}$ be a given partition of V.

This partition captured $\sum_{A_i \in \mathscr{A}} e_G(A_i)/|E|$ fraction of edges (edge contribution). Should we be happy with this?

Definition



Let $\mathscr{A} = \{A_1, A_2, \ldots, A_\ell\}$ be a given partition of V.

This partition captured $\sum_{A_i \in \mathscr{A}} e_G(A_i)/|E|$ fraction of edges (edge contribution). Should we be happy with this?

No! Compare it to the expected fraction of edges captured by this partition in the Chung-Lu model with the (expected) degree distribution $\mathbf{d} = \big(\deg(v_1), \deg(v_2), \ldots, \deg(v_n)\big)$.

Definition

The expected fraction of edges is equal to

$$\frac{1}{|E|}\left(\sum_{v_j v_k \in \binom{A_i}{2}} \frac{\deg(v_j)\deg(v_k)}{2|E|} + \sum_{v_j \in A_i} \frac{\deg^2(v_j)}{4|E|}\right)$$

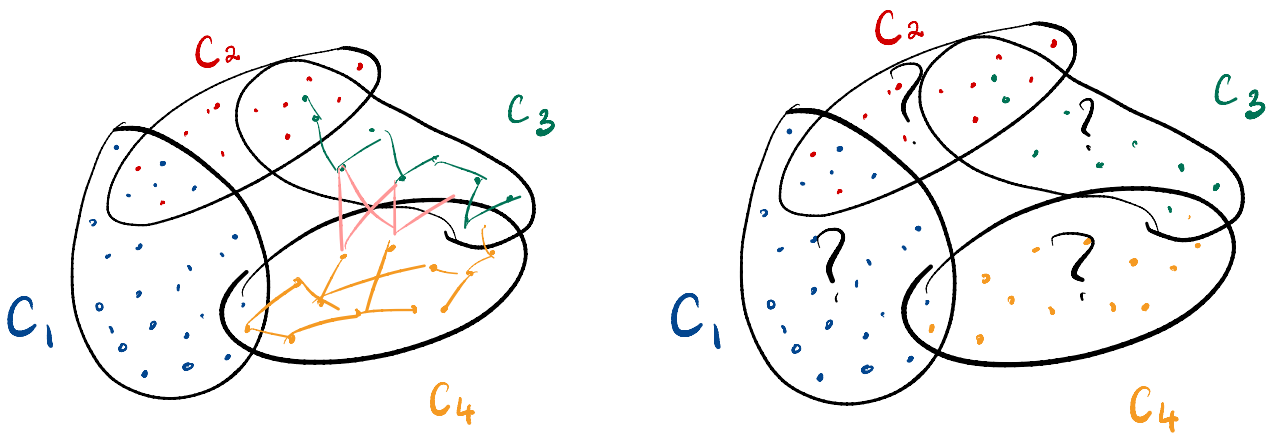
$$= \frac{1}{4|E|^2} \sum_{v_j \in A_i} \sum_{v_k \in A_i} \deg(v_j)\deg(v_k)$$

$$= \frac{1}{4|E|^2}\left(\sum_{v_j \in A_i} \deg(v_j)\right)^2 = \frac{(\mathrm{vol}(A_i))^2}{(\mathrm{vol}(V))^2}.$$

Definition

Modularity function:

$$q_G(\mathcal{A}) = \sum_{A_i \in \mathcal{A}} \frac{e_G(A_i)}{|E|} - \sum_{A_i \in \mathcal{A}} \frac{(\mathrm{vol}(A_i))^2}{(\mathrm{vol}(V))^2}$$



Definition

Modularity function:

$$q_G(\mathcal{A}) = \sum_{A_i \in \mathcal{A}} \frac{e_G(A_i)}{|E|} - \sum_{A_i \in \mathcal{A}} \frac{(\mathrm{vol}(A_i))^2}{(\mathrm{vol}(V))^2}$$

Some properties:

– $q_G(\mathcal{A}) \leq 1$,

– If $\mathcal{A} = \{V\}$, then $q_G(\mathcal{A}) = 0$,

– If $\mathcal{A} = \{\{v_1\}, \ldots, \{v_n\}\}$, then $q_G(\mathcal{A}) = -\frac{\sum \deg^2(v)}{4|E|^2} < 0$,

– $q_G(\mathcal{A}) \geq -1/2$.

Definition

Modularity function:

$$q_G(\mathscr{A}) = \sum_{A_i \in \mathscr{A}} \frac{e_G(A_i)}{|E|} - \sum_{A_i \in \mathscr{A}} \frac{(\mathrm{vol}(A_i))^2}{(\mathrm{vol}(V))^2}$$

Some properties:

– $q_G(\mathscr{A}) \leq 1$,

– If $\mathscr{A} = \{V\}$, then $q_G(\mathscr{A}) = 0$,

– If $\mathscr{A} = \{\{v_1\}, \ldots, \{v_n\}\}$, then $q_G(\mathscr{A}) = -\frac{\sum \deg^2(v)}{4|E|^2} < 0$,

– $q_G(\mathscr{A}) \geq -1/2$.

$$q^*(G) = \max_{\mathscr{A}} q_G(\mathscr{A})$$

– Well defined but impossible to find in practice *unless* G is small (then you may try Bayan algorithm).

– Used to guide a heuristic algorithms that try to maximize it.

Louvain algorithm

Initially, each node is assigned to its own community.

Louvain algorithm

Initially, each node is assigned to its own community.

Phase 1: modularity is locally optimized.

– consider nodes in a random order,

– for each node v, move v to community of some of its neighbour if it improves the modularity (can be easily and efficiently calculated!),

– if no increase is possible after considering all nodes, a local maximum value is achieved and the first phase ends.

Louvain algorithm

Phase 2: grouping nodes.

– contract nodes from each community into a single node (super-node),

– all edges within that community are replaced by a single weighted loop,

– all edges between two communities are replaced by a single weighted edge,

– typically, the resulting graph becomes much smaller.

Louvain algorithm

Phase 2: grouping nodes.

– contract nodes from each community into a single node (super-node),

– all edges within that community are replaced by a single weighted loop,

– all edges between two communities are replaced by a single weighted edge,

– typically, the resulting graph becomes much smaller.

Repeat the two phases until no improvement on the modularity function can be further achieved. The first pass is typically the most time consuming part of the algorithm.

Leiden algorithm

One problem with Louvain algorithm is that once super-nodes are created, the associated nodes will stay together forever.

Leiden algorithm

One problem with Louvain algorithm is that once super-nodes are created, the associated nodes will stay together forever.

Solution: Leiden algorithm—periodically randomly breaking down communities into smaller well-connected ones.

Ensemble Clustering algorithm for Graphs (**ECG**)

Louvain is a randomized algorithm: can be unstable, that is, the results of independent runs of the same algorithm on the very same data can vary a lot.

Ensemble Clustering algorithm for Graphs (**ECG**)

Louvain is a randomized algorithm: can be unstable, that is, the results of independent runs of the same algorithm on the very same data can vary a lot.

Solution: use ensemble clustering often referred to as consensus clustering.

Ensemble Clustering algorithm for Graphs (**ECG**)

Start with ℓ randomized level-1 partitions $\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_\ell$ obtained by ℓ independent runs of the first phase of the Louvain algorithm and only once.

Ensemble Clustering algorithm for Graphs (**ECG**)

Start with ℓ randomized level-1 partitions $\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_\ell$ obtained by ℓ independent runs of the first phase of the Louvain algorithm and only once.

Assign new weights to edges as follows: for each $uv \in E$,

$$w(uv) = \begin{cases} w_* + (1 - w_*) \cdot \frac{1}{\ell} \sum_{i=1}^{\ell} \alpha_i(u, v), & \text{if } uv \text{ is in the 2-core} \\ w_* & \text{otherwise,} \end{cases}$$

$w_* \in [0, 1]$ is the parameter of the algorithm, and $\alpha_i(u, v) = 1$ if u and v appear together in some part of $\mathcal{P}_i$; otherwise, $\alpha_i(u, v) = 0$. (Default values are $\ell = 16$ and $w_* = 0.05$.)

Ensemble Clustering algorithm for Graphs (**ECG**)

Start with ℓ randomized level-1 partitions $\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_\ell$ obtained by ℓ independent runs of the first phase of the Louvain algorithm and only once.

Assign new weights to edges as follows: for each $uv \in E$,

$$w(uv) = \begin{cases} w_* + (1 - w_*) \cdot \frac{1}{\ell} \sum_{i=1}^{\ell} \alpha_i(u, v), & \text{if } uv \text{ is in the 2-core} \\ w_* & \text{otherwise,} \end{cases}$$

$w_* \in [0, 1]$ is the parameter of the algorithm, and $\alpha_i(u, v) = 1$ if u and v appear together in some part of $\mathcal{P}_i$; otherwise, $\alpha_i(u, v) = 0$. (Default values are $\ell = 16$ and $w_* = 0.05$.)

Perform Louvain (till the very end, *not* just level-1) on a weighted version of the initial graph.

Resolution Limit

Optimizing modularity function in large networks cannot find small communities, even if they are well defined.

Resolution Limit

Optimizing modularity function in large networks cannot find small communities, even if they are well defined.

The expected number of edges between part A_i and part $A_j \neq A_i$ in the corresponding Chung-Lu model:

$$\sum_{v_i \in A_i} \sum_{v_j \in A_j} \frac{\deg(v_i)\deg(v_j)}{2|E|} = \frac{\mathrm{vol}(A_i)\,\mathrm{vol}(A_j)}{2|E|} < 1$$

if $\mathrm{vol}(A_i) \leq \mathrm{vol}(A_j) < \sqrt{2|E|}$.

Resolution Limit

Optimizing modularity function in large networks cannot find small communities, even if they are well defined.

The expected number of edges between part A_i and part $A_j \neq A_i$ in the corresponding Chung-Lu model:

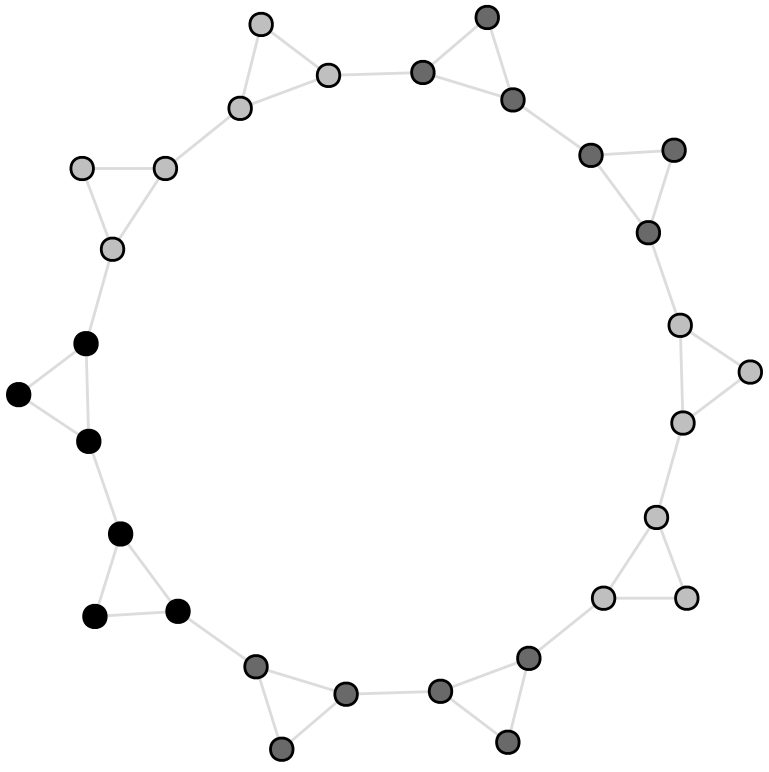
$$\sum_{v_i \in A_i} \sum_{v_j \in A_j} \frac{\deg(v_i)\deg(v_j)}{2|E|} = \frac{\mathrm{vol}(A_i)\,\mathrm{vol}(A_j)}{2|E|} < 1$$

if $\mathrm{vol}(A_i) \leq \mathrm{vol}(A_j) < \sqrt{2|E|}$.

Implication: a single edge between them would lead to merging the two parts!

Resolution Limit

Conclusion: even weakly interconnected complete graphs ("the ring of cliques") would be merged together, provided the network is sufficiently large.



Resolution Limit

Potential solutions:

– Try to subdivide some large communities A_i by independently running the algorithm on $G[A_i]$, the graph induced by part A_i.

Resolution Limit

Potential solutions:

– Try to subdivide some large communities A_i by independently running the algorithm on $G[A_i]$, the graph induced by part A_i.

– Run ECG that tends to deal better with this issue.

Resolution Limit

Potential solutions:

– Try to subdivide some large communities A_i by independently running the algorithm on $G[A_i]$, the graph induced by part A_i.

– Run ECG that tends to deal better with this issue.

– Add a resistance $r \in \mathbb{R}$ to every node that can be viewed as a loop of weight r (multiresolution method). Positive value of r increases the aversion of nodes to form communities whereas negative value of r does the opposite. Independently, one may multiply the degree tax by a universal constant $\gamma \in \mathbb{R}_+$.

– γ is called the resolution parameter.

Hierarchical Clustering

Hierarchical Clustering

Graph clustering algorithms are unsupervised machine learning tools. The algorithm has to decide about the number of clusters (in contrast to, for example, classical k-means).

Hierarchical Clustering

Graph clustering algorithms are unsupervised machine learning tools. The algorithm has to decide about the number of clusters (in contrast to, for example, classical k-means).

Alternatively, one may use hierarchical clustering algorithms to build a hierarchy of n clusters, ranging between:
a) trivial partition into n clusters (each cluster = singe node),
b) trivial partition into 1 cluster.

Hierarchical Clustering

Graph clustering algorithms are unsupervised machine learning tools. The algorithm has to decide about the number of clusters (in contrast to, for example, classical k-means).

Alternatively, one may use hierarchical clustering algorithms to build a hierarchy of n clusters, ranging between:
a) trivial partition into n clusters (each cluster = singe node),
b) trivial partition into 1 cluster.

Agglomerative ("bottom-up") algorithms start with a) and at each step merge two parts with the largest similarity.

Hierarchical Clustering

Graph clustering algorithms are unsupervised machine learning tools. The algorithm has to decide about the number of clusters (in contrast to, for example, classical k-means).

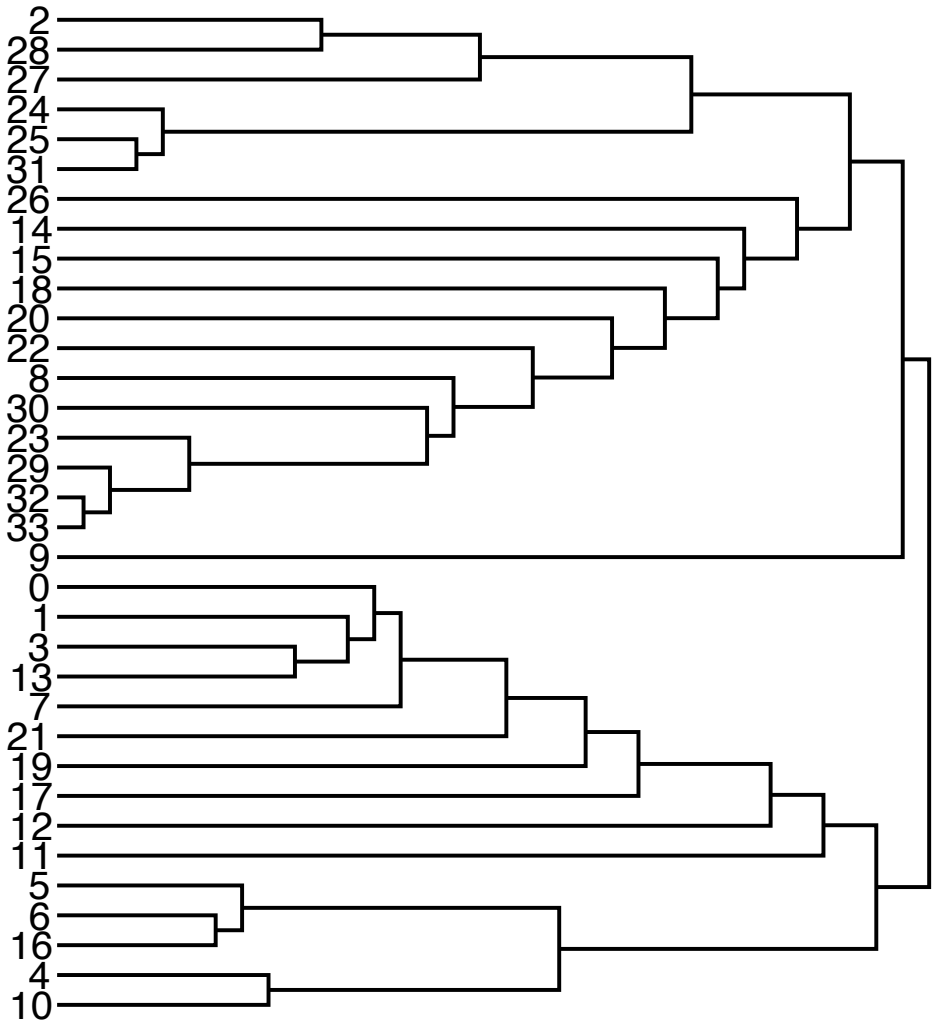
Alternatively, one may use hierarchical clustering algorithms to build a hierarchy of n clusters, ranging between:
a) trivial partition into n clusters (each cluster = singe node),
b) trivial partition into 1 cluster.

Agglomerative ("bottom-up") algorithms start with a) and at each step merge two parts with the largest similarity.

Divisive ("top-down") algorithms start b) then refine the partition by splitting one of the clusters into two.

Dendrograms

Outcomes of such algorithms can be conveniently represented by a dendrogram known also as a hierarchical tree.



Hierarchical Clustering

Known limitations:

– They provide n partitions. Which one should be used?

Hierarchical Clustering

Known limitations:

– They provide n partitions. Which one should be used?
(Benchmark these partitions using the modularity function.)

Hierarchical Clustering

Known limitations:

– They provide n partitions. Which one should be used?
(Benchmark these partitions using the modularity function.)

– These algorithms are rather slow.

Hierarchical Clustering

Known limitations:

– They provide n partitions. Which one should be used?
(Benchmark these partitions using the modularity function.)

– These algorithms are rather slow.

The outcome of the process heavily depends on the similarity measure (for agglomerative algorithms) or dissimilarity measure (for divisive ones) between clusters.
(We will present some specific implementations next.)

Ravasz (Agglomerative) Algorithm

Ravasz algorithm uses the following similarity measure (we want this to be large for pairs of nodes that belong to the same community and small otherwise).

Topological overlap matrix

The topological overlap matrix $\mathbf{S} = (s(v_i, v_j))_{i,j\in[n]}$ is a quadratic and symmetric matrix in which

$$s(v_i, v_j) = \frac{|N(v_i) \cap N(v_j)| + \delta_{v_i v_j \in E}}{\min\{\deg(v_i), \deg(v_j)\} + \delta_{v_i v_j \notin E}}.$$

Ravasz (Agglomerative) Algorithm

Ravasz algorithm uses the following similarity measure (we want this to be large for pairs of nodes that belong to the same community and small otherwise).

Topological overlap matrix

The topological overlap matrix $\mathbf{S} = (s(v_i, v_j))_{i,j\in[n]}$ is a quadratic and symmetric matrix in which

$$s(v_i, v_j) = \frac{|N(v_i) \cap N(v_j)| + \delta_{v_i v_j \in E}}{\min\{\deg(v_i), \deg(v_j)\} + \delta_{v_i v_j \notin E}}.$$

$s(v_i, v_j) = 1$ if and only if v_i and v_j are adjacent and neighbours of v_i (other than v_j) are also adjacent to v_j (or vice versa).
$s(v_i, v_j) = 0$ if and only if the corresponding nodes have no common neighbours and are not adjacent.

Ravasz (Agglomerative) Algorithm

To evaluate how similar two non-overlapping communities A and B are, we use the average cluster similarity:

$$s_a(A, B) = \frac{1}{|A||B|} \sum_{a \in A, b \in B} s(a, b).$$

Ravasz (Agglomerative) Algorithm

To evaluate how similar two non-overlapping communities A and B are, we use the average cluster similarity:

$$s_a(A, B) = \frac{1}{|A||B|} \sum_{a \in A, b \in B} s(a, b).$$

A pair of communities with the largest similarity is merged (in case of a tie, decision is made randomly).

Ravasz (Agglomerative) Algorithm

To evaluate how similar two non-overlapping communities A and B are, we use the average cluster similarity:

$$s_a(A, B) = \frac{1}{|A||B|} \sum_{a \in A, b \in B} s(a, b).$$

A pair of communities with the largest similarity is merged (in case of a tie, decision is made randomly).

Both the time and the space complexity are $\Theta(n^2)$, provided that the average degree is $\Theta(1)$ (not scalable).

Girvan–Newman (Divisive) Algorithm

Start from the original graph.

We systematically remove edges between nodes that we suspect that they belong to different communities.

Girvan–Newman (Divisive) Algorithm

Start from the original graph.

We systematically remove edges between nodes that we suspect that they belong to different communities.

We will use the edge betweenness $\ell(uv)$ that is the number of shortest paths between pairs of nodes that run along edge uv. (If there is more than one shortest path between a pair of nodes, each path is independently considered.)

Girvan–Newman (Divisive) Algorithm

Start from the original graph.

We systematically remove edges between nodes that we suspect that they belong to different communities.

We will use the edge betweenness $\ell(uv)$ that is the number of shortest paths between pairs of nodes that run along edge uv. (If there is more than one shortest path between a pair of nodes, each path is independently considered.)

From time to time, this operation disconnects some connected component, resulting in another branch in the corresponding dendrogram.

Girvan–Newman (Divisive) Algorithm

Start from the original graph.

We systematically remove edges between nodes that we suspect that they belong to different communities.

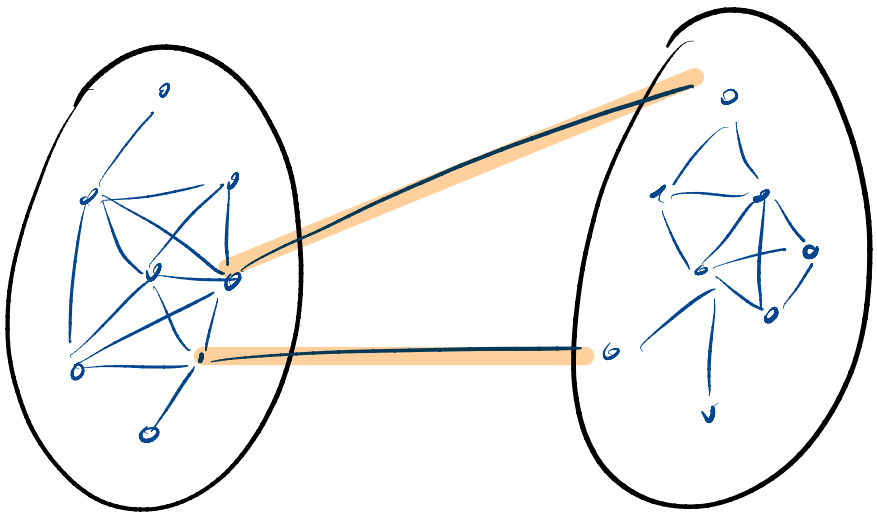
We will use the edge betweenness $\ell(uv)$ that is the number of shortest paths between pairs of nodes that run along edge uv. (If there is more than one shortest path between a pair of nodes, each path is independently considered.)

From time to time, this operation disconnects some connected component, resulting in another branch in the corresponding dendrogram.

The worst-case complexity is $\Theta(m^2 n)$ (not scalable).

Girvan–Newman (Divisive) Algorithm

If a graph consists of communities that are only loosely connected by a few edges between them, then all shortest paths between nodes in different communities must go along one of these few edges. One of them must have large edge betweenness. (Consider the George Washington Bridge, the world's busiest bridge in terms of vehicular traffic.)



A Few Other Methods

Label Propagation Algorithm

Start with a trivial partition into n communities. Each node will keep a label representing its community.

Label Propagation Algorithm

Start with a trivial partition into n communities. Each node will keep a label representing its community.

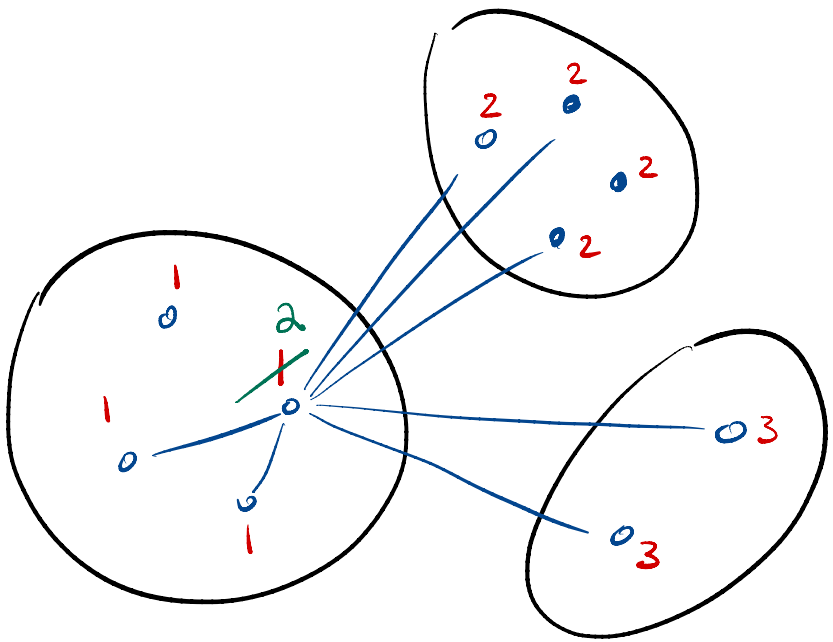
Investigate nodes in random order.

Label Propagation Algorithm

Start with a trivial partition into n communities. Each node will keep a label representing its community.

Investigate nodes in random order.

Each investigated node v adjusts (if needed) its label to the label that majority of neighbours of v have (resolve ties randomly).



Label Propagation Algorithm

Some labels quickly propagate throughout the network whereas some other disappear.

Label Propagation Algorithm

Some labels quickly propagate throughout the network whereas some other disappear.

The algorithm stops if at the end of a given phase each node has a label that is consistent with the majority label of its neighbours (stationary state is reached) or we reach some predefined number of phases (say, 100).

Label Propagation Algorithm

Some labels quickly propagate throughout the network whereas some other disappear.

The algorithm stops if at the end of a given phase each node has a label that is consistent with the majority label of its neighbours (stationary state is reached) or we reach some predefined number of phases (say, 100).

The results are unstable (common problem) but one may use some variant of a consensus clustering to make it more stable.

Spectral Bisection Method

Assume that our goal is to partition the set of nodes $V = \{v_1, v_2, \ldots, v_n\}$ into two subsets. We may repeat this procedure recursively to find a more fine grained clustering.

Spectral Bisection Method

Assume that our goal is to partition the set of nodes $V = \{v_1, v_2, \ldots, v_n\}$ into two subsets. We may repeat this procedure recursively to find a more fine grained clustering.

Use $x_i \in \{-1, 1\}$ to describe such partition. Let $\mathbf{x}$ denote a column vector consisting of x_i.

Spectral Bisection Method

Assume that our goal is to partition the set of nodes $V = \{v_1, v_2, \ldots, v_n\}$ into two subsets. We may repeat this procedure recursively to find a more fine grained clustering.

Use $x_i \in \{-1, 1\}$ to describe such partition. Let $\mathbf{x}$ denote a column vector consisting of x_i.

The graph Laplacian is a matrix $\mathbf{L} = \mathbf{D} - \mathbf{A}$, where $\mathbf{A} = (a(v_i, v_j))_{i,j\in[n]}$ is the (symmetric) adjacency matrix and $\mathbf{D} = (d(v_i, v_j))_{i,j\in[n]} = \mathrm{diag}(\mathbf{A1})$ is the diagonal matrix with $d(v_i, v_i) = \deg(v_i) = \sum_{j=1}^{n} a(v_i, v_j)$.

Spectral Bisection Method

Assume that our goal is to partition the set of nodes $V = \{v_1, v_2, \ldots, v_n\}$ into two subsets. We may repeat this procedure recursively to find a more fine grained clustering.

Use $x_i \in \{-1, 1\}$ to describe such partition. Let $\mathbf{x}$ denote a column vector consisting of x_i.

The graph Laplacian is a matrix $\mathbf{L} = \mathbf{D} - \mathbf{A}$, where $\mathbf{A} = (a(v_i, v_j))_{i,j \in [n]}$ is the (symmetric) adjacency matrix and $\mathbf{D} = (d(v_i, v_j))_{i,j \in [n]} = \mathrm{diag}(\mathbf{A1})$ is the diagonal matrix with $d(v_i, v_i) = \deg(v_i) = \sum_{j=1}^{n} a(v_i, v_j)$.

Objective. Find a partition (vector $\mathbf{x}$) that minimizes the number of edges between the two clusters: $L(\mathbf{x})$.

Spectral Bisection Method

Since $(x_i - x_j)^2 = 4$ if the two corresponding nodes belong to two different clusters and otherwise $(x_i - x_j)^2 = 0$,

$$L(\mathbf{x}) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} a(v(i), v(j))(x_i - x_j)^2/4 = \mathbf{x}^T \mathbf{L}\mathbf{x}/8.$$

Spectral Bisection Method

Since $(x_i - x_j)^2 = 4$ if the two corresponding nodes belong to two different clusters and otherwise $(x_i - x_j)^2 = 0$,

$$L(\mathbf{x}) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} a(v(i), v(j))(x_i - x_j)^2/4 = \mathbf{x}^T \mathbf{L}\mathbf{x}/8.$$

We insist that both clusters have as equal sizes as possible, which can be expressed as $|\sum_{i=1}^{n} x_i| \leq 1$.

Spectral Bisection Method

Since $(x_i - x_j)^2 = 4$ if the two corresponding nodes belong to two different clusters and otherwise $(x_i - x_j)^2 = 0$,

$$L(\mathbf{x}) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} a(v(i), v(j))(x_i - x_j)^2/4 = \mathbf{x}^T \mathbf{L}\mathbf{x}/8.$$

We insist that both clusters have as equal sizes as possible, which can be expressed as $|\sum_{i=1}^{n} x_i| \leq 1$.

It is a difficult combinatorial problem!

Spectral Bisection Method

Since $(x_i - x_j)^2 = 4$ if the two corresponding nodes belong to two different clusters and otherwise $(x_i - x_j)^2 = 0$,

$$L(\mathbf{x}) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} a(v(i), v(j))(x_i - x_j)^2/4 = \mathbf{x}^T \mathbf{L}\mathbf{x}/8.$$

We insist that both clusters have as equal sizes as possible, which can be expressed as $|\sum_{i=1}^{n} x_i| \leq 1$.

It is a difficult combinatorial problem!

Relax the assumption that $x \in \{-1, 1\}$ to $x \in \mathbb{R}$, and then round the solution to the nearest value from $\{-1, 1\}$. This gives us an approximated solution.

Infomap

It uses concepts taken from the information theory.

The optimization framework is similar to Louvain: start with a trivial partition and try to optimize some quality function: the entropy function instead of the modularity function.

Infomap

It uses concepts taken from the information theory.

The optimization framework is similar to Louvain: start with a trivial partition and try to optimize some quality function: the entropy function instead of the modularity function.

Main idea: long random walk on nodes should preserve important information about the structure and the topology of the network.

Infomap

We benchmark a given partition $\mathcal{A} = \{A_1, A_2, \ldots, A_\ell\}$ of the set of nodes V as follows.

Infomap

We benchmark a given partition $\mathcal{A} = \{A_1, A_2, \ldots, A_\ell\}$ of the set of nodes V as follows.

Uniquely represent the sequence of nodes visited by the random walk by a sequence of zeros and ones.

Infomap

We benchmark a given partition $\mathcal{A} = \{A_1, A_2, \ldots, A_\ell\}$ of the set of nodes V as follows.

Uniquely represent the sequence of nodes visited by the random walk by a sequence of zeros and ones.

We need to label nodes but we are also allowed to use two additional labels assigned to each community A_i:
– the entry label indicating that the walk entered A_i
– the exit label indicating that the walk left A_i.

Infomap

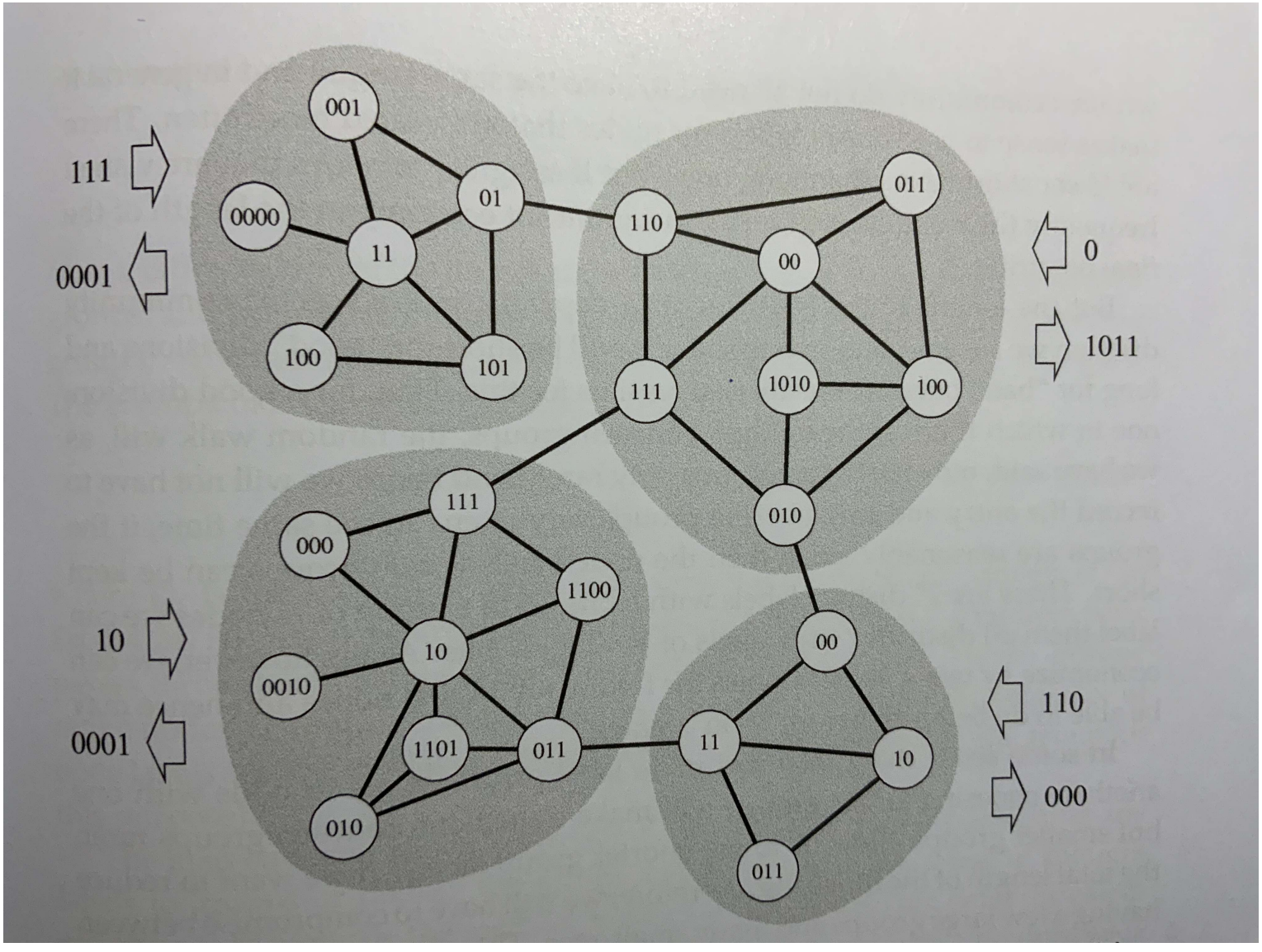
We benchmark a given partition $\mathcal{A} = \{A_1, A_2, \ldots, A_\ell\}$ of the set of nodes V as follows.

Uniquely represent the sequence of nodes visited by the random walk by a sequence of zeros and ones.

We need to label nodes but we are also allowed to use two additional labels assigned to each community A_i:
– the entry label indicating that the walk entered A_i
– the exit label indicating that the walk left A_i.

We may reuse labels as long as there is no ambiguity in reconstructing the random walk. Labels do not have to have the same length (commonly visited nodes should get a shorter label).

Infomap


111
0001
10
0001
0
1011
110
000
001
0000
01
11
100
101
011
110
00
111
1010
100
010
111
000
1100
10
0010
1101
011
010
00
11
10
011

Infomap

The encoding of the walk should be short, provided that there is some strong community structure in the network.

Infomap

The encoding of the walk should be short, provided that there is some strong community structure in the network.

The length of a shortest representation is well defined but finding it might be computationally challenging.

Infomap

The encoding of the walk should be short, provided that there is some strong community structure in the network.

The length of a shortest representation is well defined but finding it might be computationally challenging.

Fortunately, we may use the fundamental and classic result known as the Shannon's source coding theorem.

Infomap

A shortest possible bit-string representing the random walk has the average number of bits per one step of the walk equal to the entropy function (given by the following map equation):

$$L = L(\mathcal{A}) = q \cdot H(Q) + \sum_{i=1}^{\ell} p_i \cdot H(P_i).$$

q is the fraction of steps the walk spends moving between parts, p_i is the fraction of steps it spends within part A_i and leaving it. The entropy of a sequence R of r objects is given by

$$H(R) = -\sum_{i=1}^{r} R_i \log_2(R_i),$$

where R_i is the fraction of steps that object i appears. $H(Q)$ is the entropy of the sequence of entry labels and $H(P_i)$ is the entropy of the sequence of node labels and the exit label of A_i.

Hence, one may benchmark a given partition $\mathcal{A}$ without actually assigning any labels to nodes. Similarly, we do not even need to generate a random walk as one may compute q and p_i's simply by analyzing the structure of graph G.

Infomap

Hence, one may benchmark a given partition $\mathcal{A}$ without actually assigning any labels to nodes. Similarly, we do not even need to generate a random walk as one may compute q and p_i's simply by analyzing the structure of graph G.

The running time of Infomap is in practice comparable to the one of Louvain.

Experiments

Experiments

Jupyter notebooks and datasets that accompany the textbook/slides are available on GitHub (`Python_Notebooks_2nd` directory):

`https://github.com/ftheberge/GraphMiningNotebooks`

Tutorials for the Python notebooks can be found on YouTube:

`https://www.youtube.com/@MiningComplexNetworks`

THE END