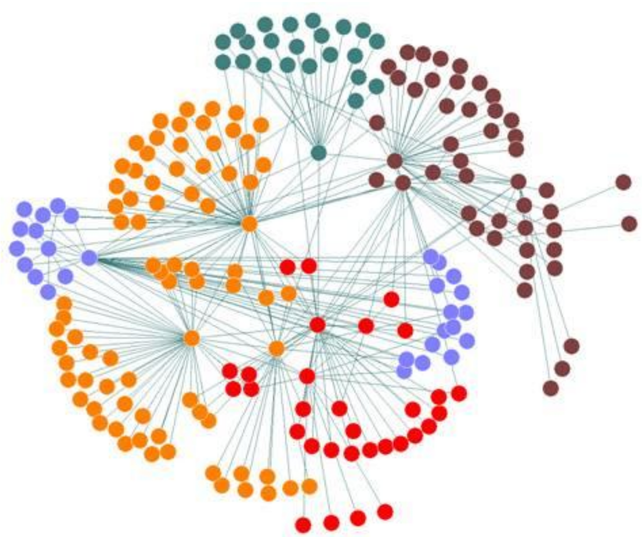
TEL AVIV UNIVERSITY

Algorithms and Applications in Social Networks



2023/2024, Semester A

Slava Novgorodov

Lesson #4

- Similarity in networks
- Network structure
- Communities
- Detection of communities

Similarity in Networks

Properties of Network

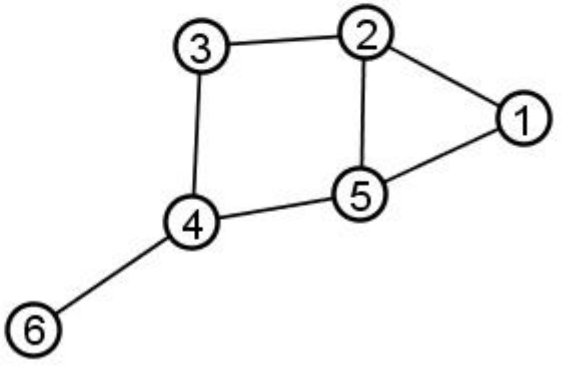
- Global (per network):
  - Average clustering coefficient
  - Degree distribution
  - Diameter
  - …
- Local (per node):
  - Degree
  - Node centrality
  - …
- Pairwise:
  - Similarity
  - …

Similarity

- Jaccard Similarity: [N (v) = Neighbors(v)]

$$J(a, b) = \frac{|N(a) \cap N(b)|}{|N(a) \cup N(b)|}$$

J(2, 4) = |{3, 5, 6} ∩ {1, 3, 5}| /
|{3, 5, 6} ∪ {1, 3, 5}| = 2/4 = 0.5



Similarity

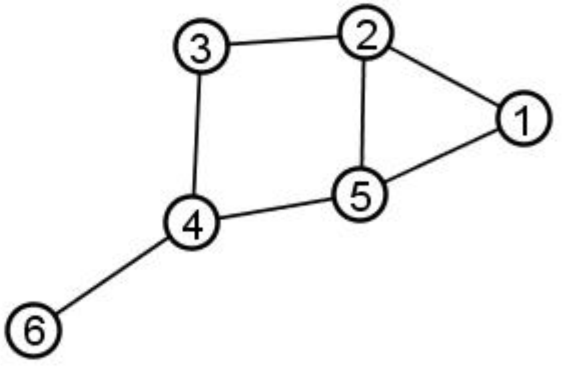
- Cosine Similarity:

cos(a,b) = (a * b) / (|a|*|b|)

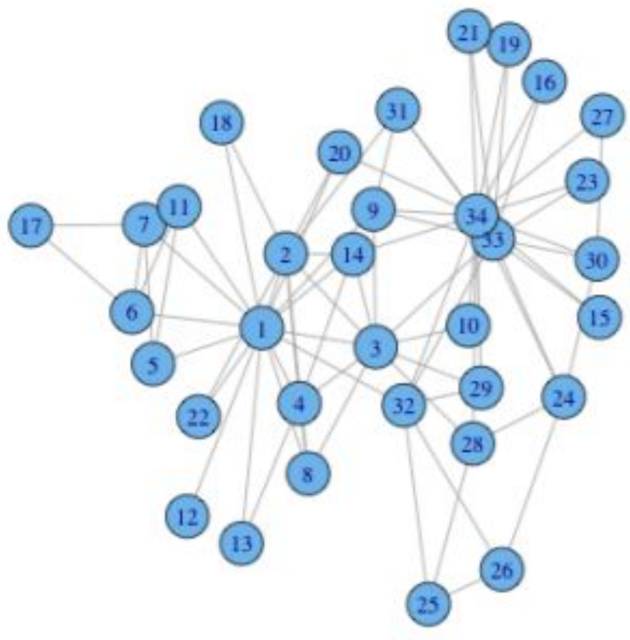
(2) = (1, 0, 1, 0, 1, 0)

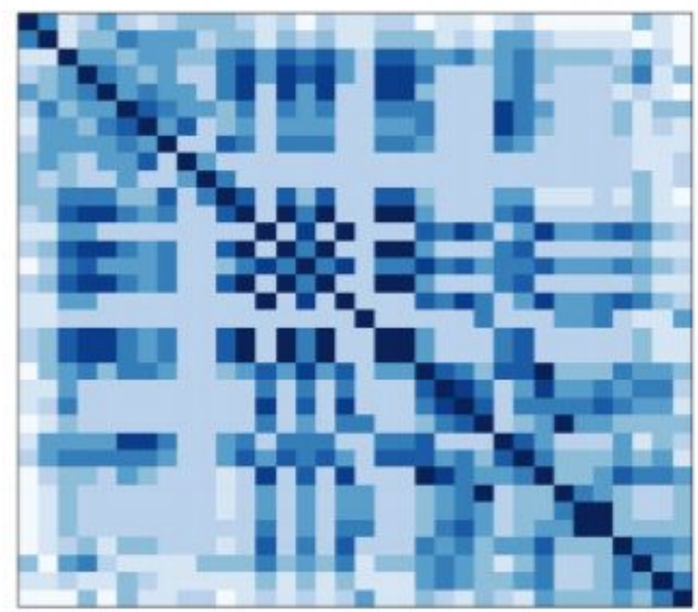
(4) = (0, 0, 1, 0, 1, 1)

cos((2), (4)) = 2 / 3



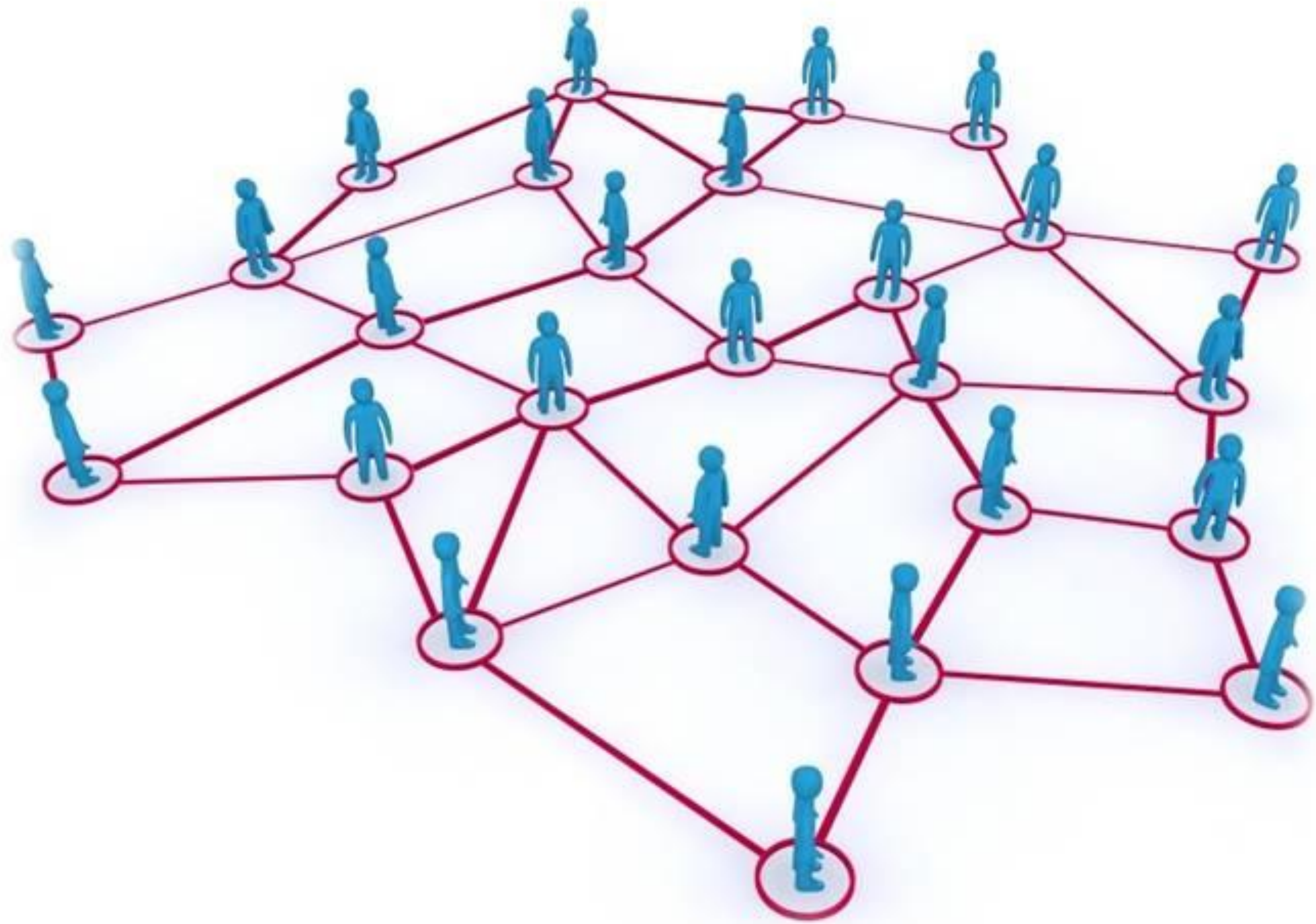
Similarity


21
19
16
31
27
18
20
23
17
7
11
9
34
33
2
14
30
6
1
10
15
3
5
29
24
4
32
22
28
8
12
13
26
25



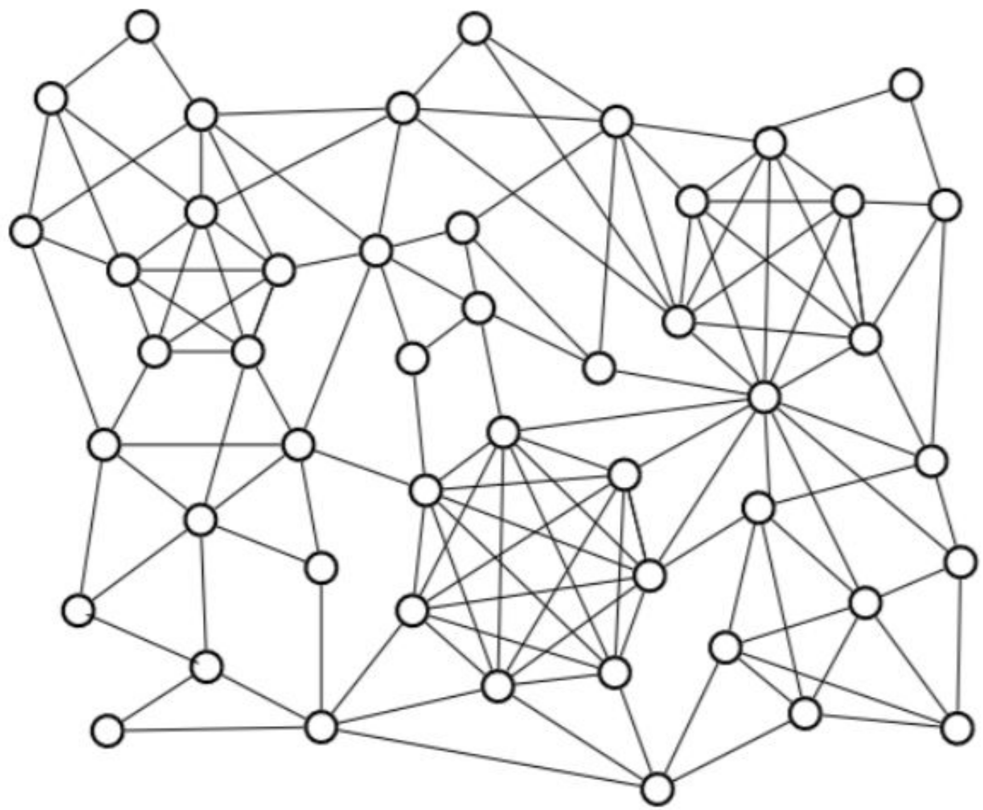
Network Structure

Network Structure



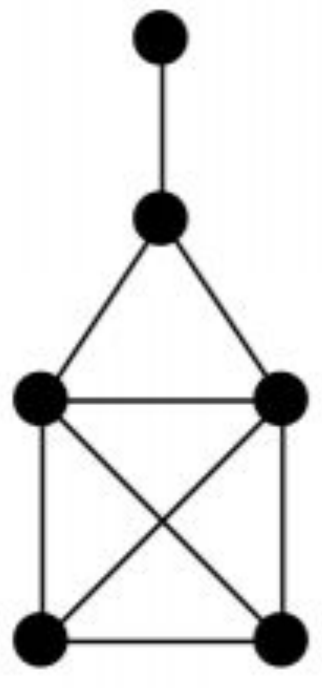
Graph cliques

- Clique is a complete subgraph.
- Cliques can overlap



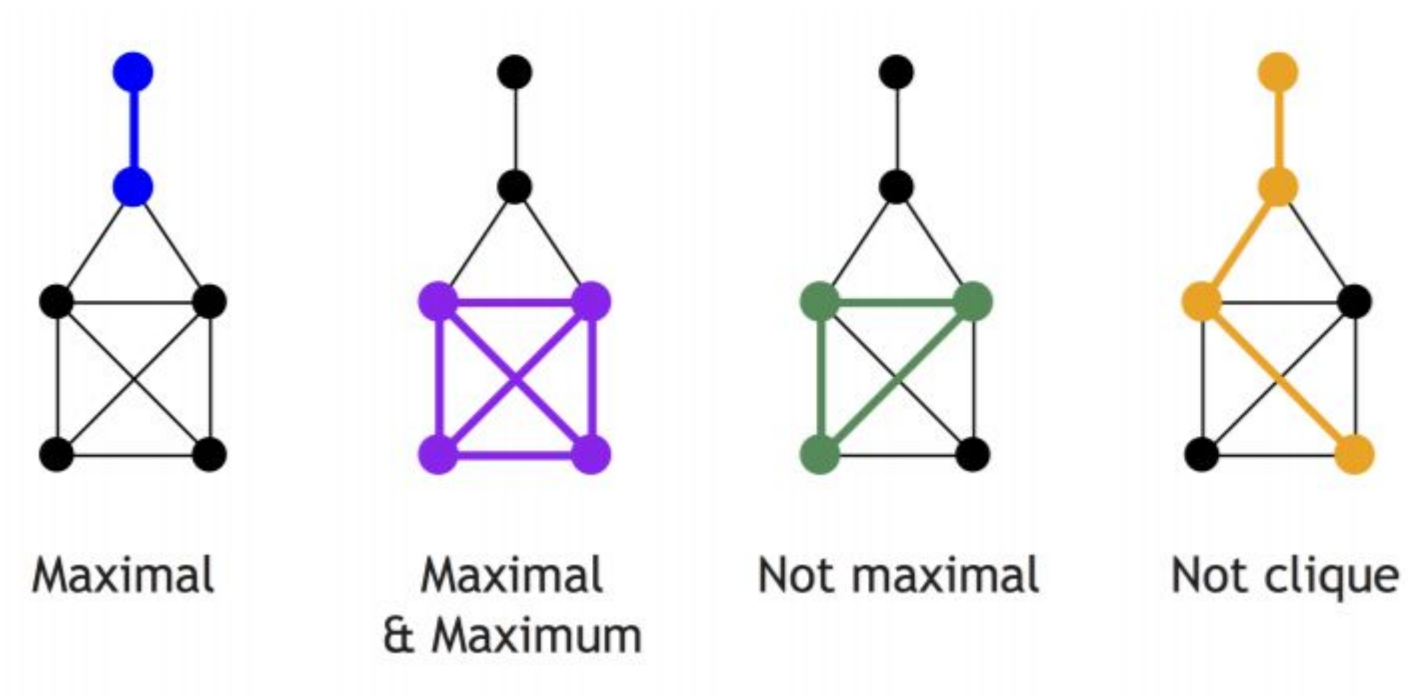
Graph cliques

- **Maximal clique** – a clique that cannot be extended by adding more nodes
- **Maximum clique** – a clique of a maximum size



Graph cliques

- **Maximal clique** – a clique that cannot be extended by adding more nodes
- **Maximum clique** – a clique of a maximum size

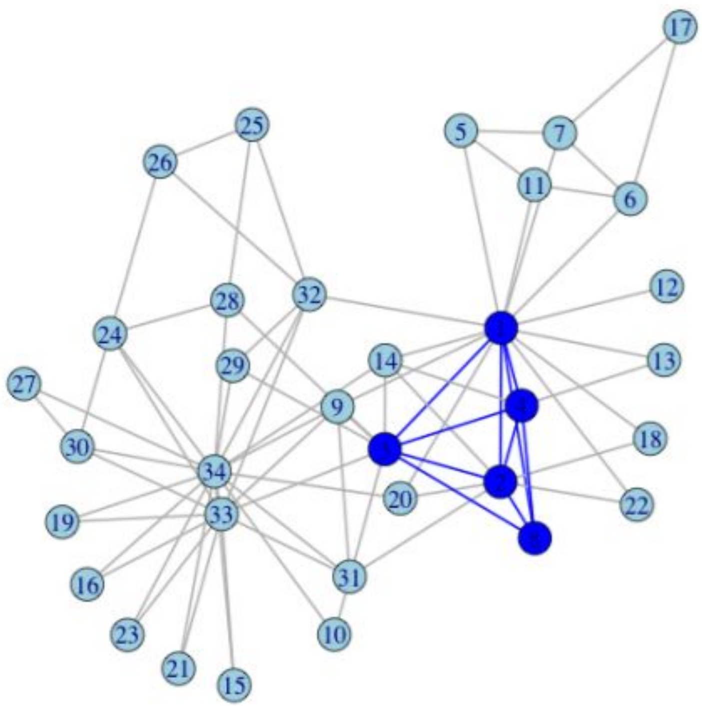


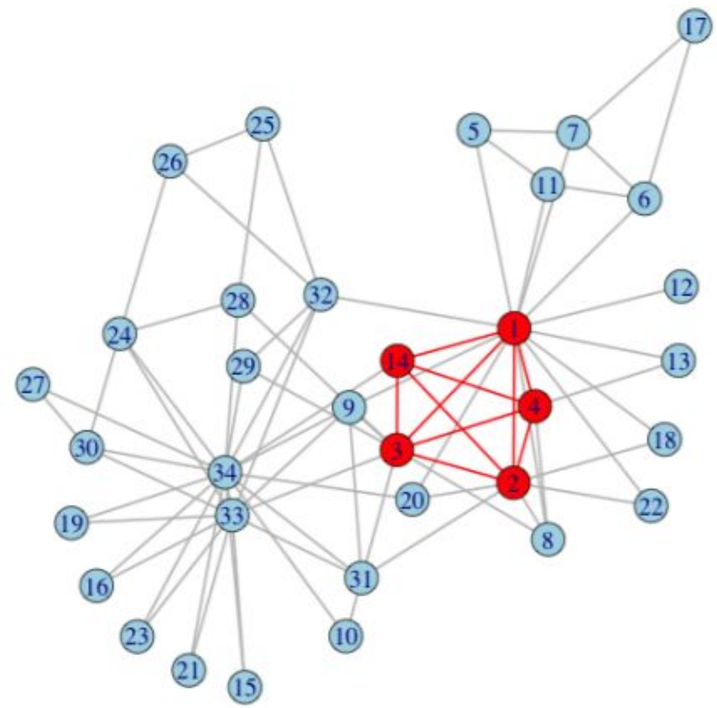
Graph cliques

Maximal cliques:

Size: 2 3 4 5

#: 11 21 2 2





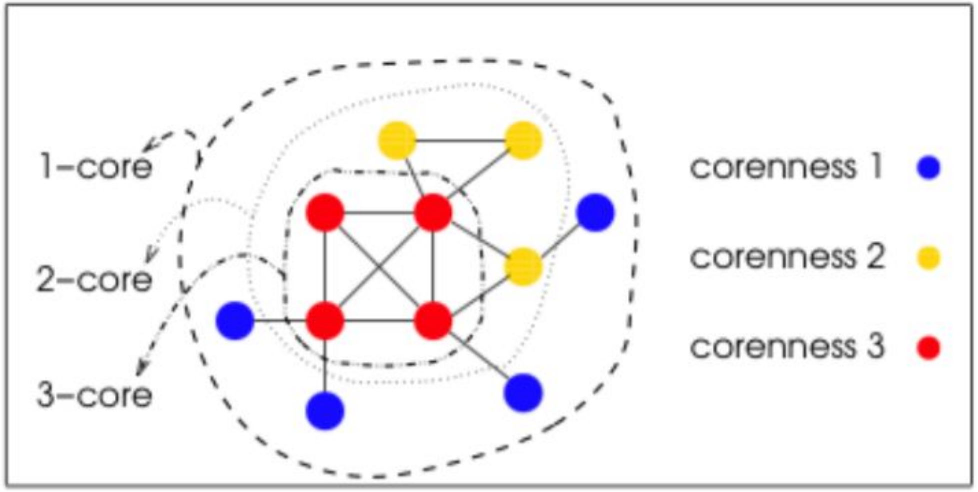
Graph cliques

Computation of cliques:

- Finding a clique of size k: $O(n^k k^2)$
- Finding maximum clique: $O(3^{n/3})$
- Easier in sparse graphs…

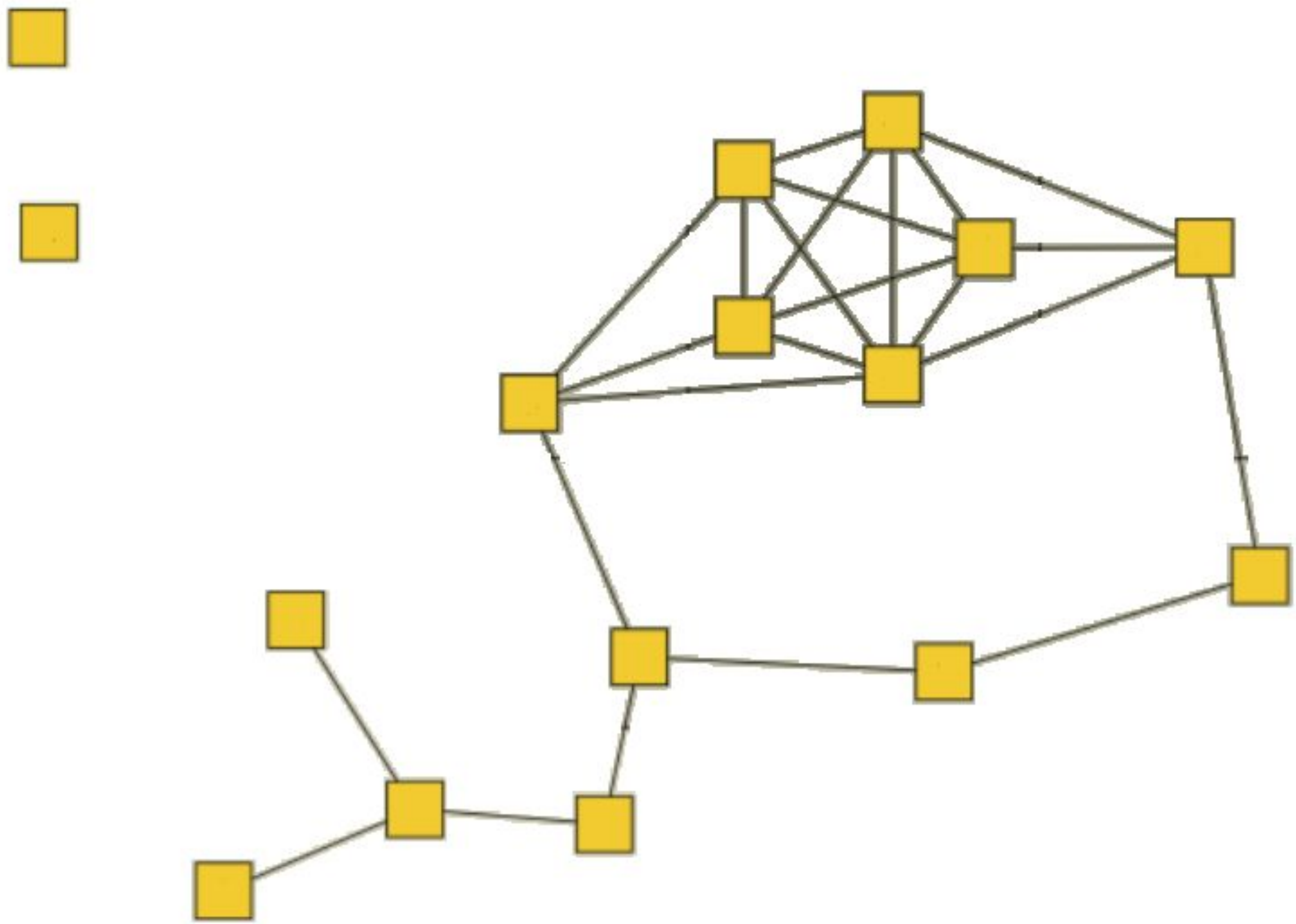
Graph Core

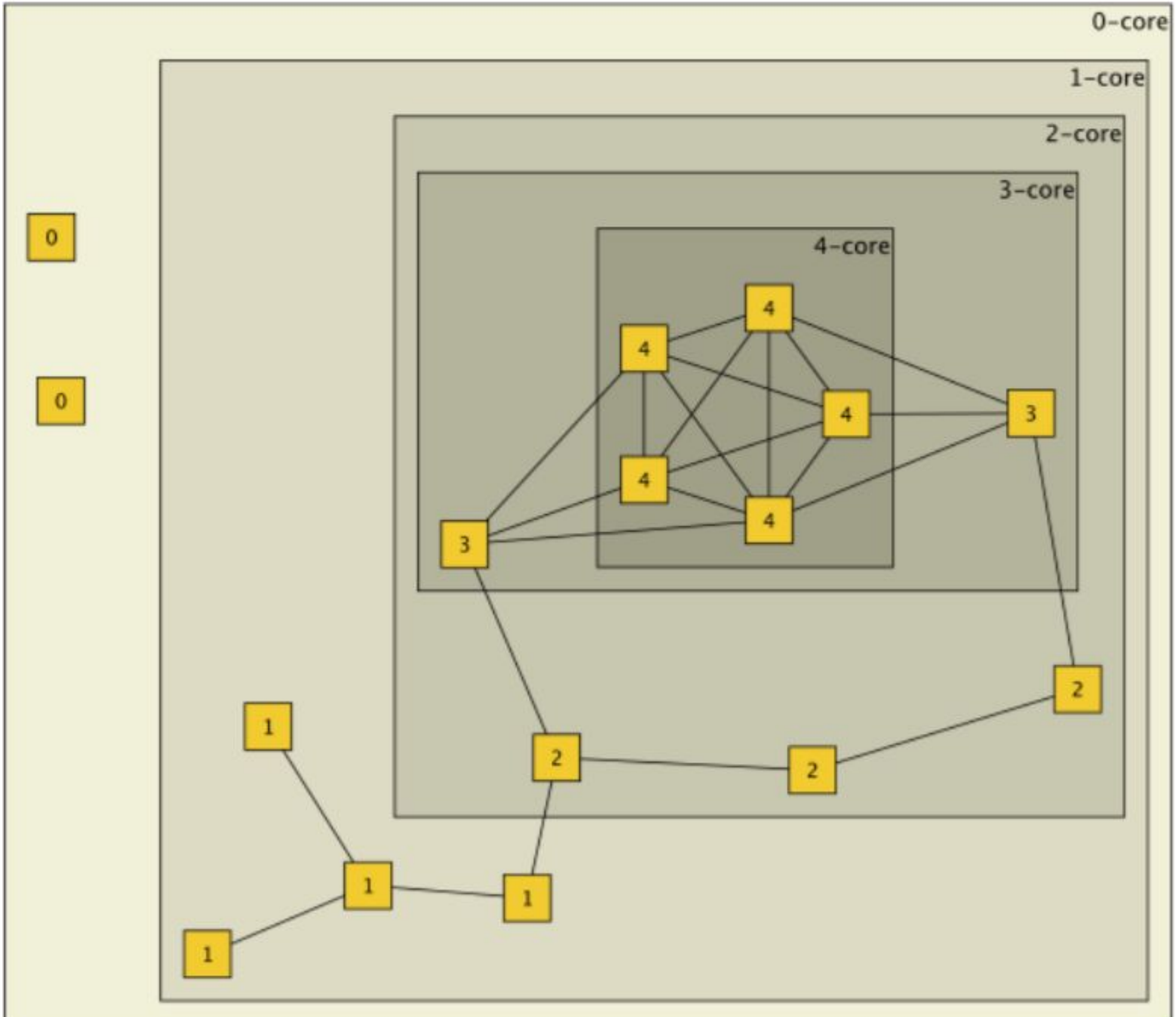
- A **k-core** is the largest subgraph S such as each node is connected to at least k nodes in S



- Every node in k-core has degree >= k
- (k+1)-core is always a subgraph of k-core
- Core number of node is the highest “k” of the k-core that contains this node

Graph Core - Example




0-core
1-core
2-core
3-core
4-core
0
0
4
4
4
4
4
3
3
2
2
2
1
1
1
1

k-core decomposition

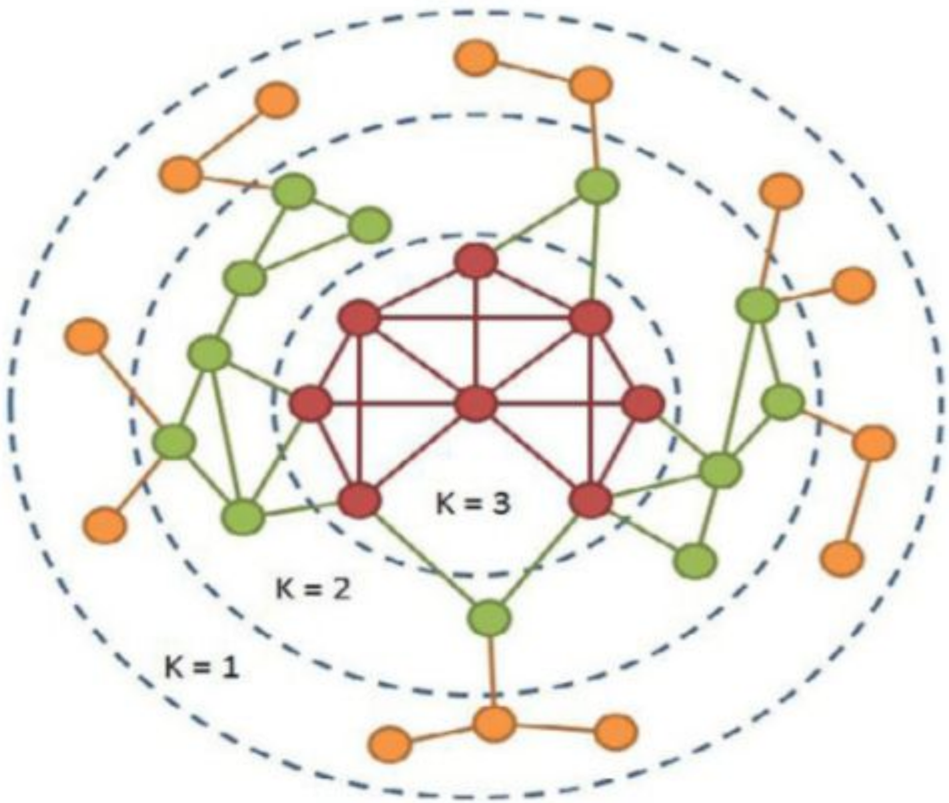
Input: Graph G(V, E)

Output: core[v] – core number for each node

```
Store degree of v in degree[v]
Sort V by degree[v] (ascending)
For each v in V:
    core[v] = degree[v]
    For each u in neighbors(v):
        If degree[u] > degree[v]:
            degree[u] -= 1
            Sort V by degree[v]
```

k-core decomposition

Recursively delete all nodes with degree less then k, the remaining graph is k-core



k-core decomposition

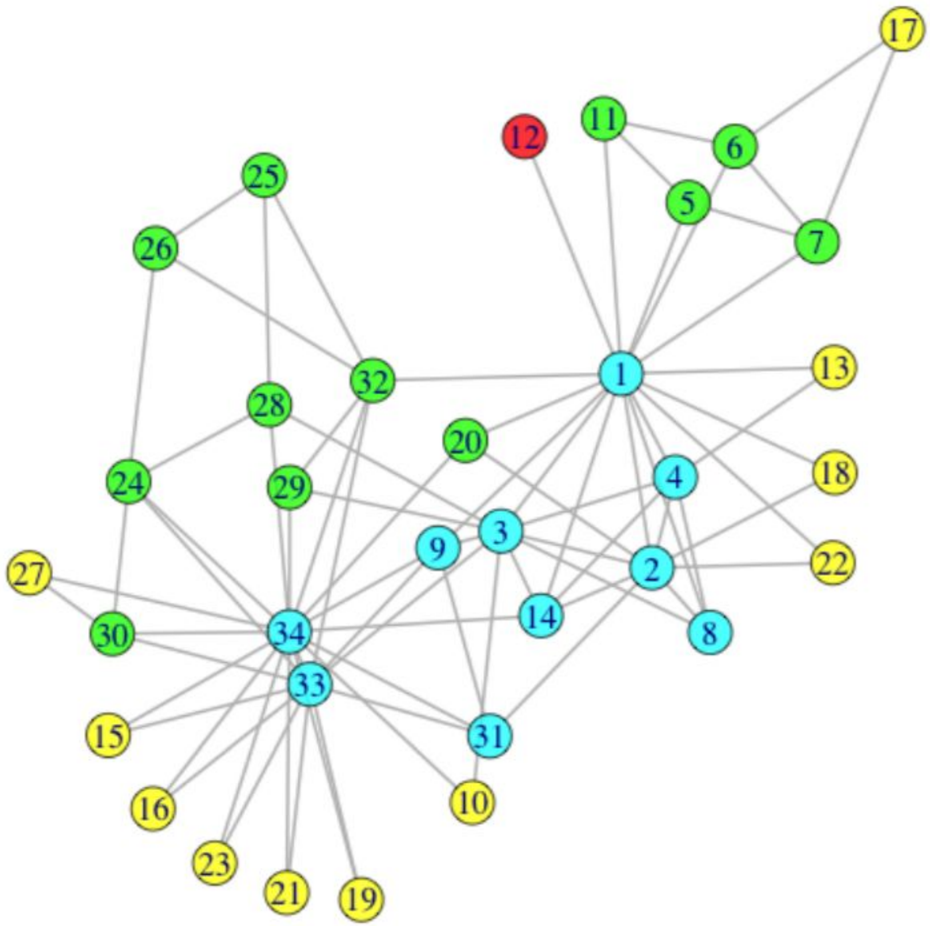
Zachary karate club k-core decomposition:

Red – 1

Yellow – 2

Green – 3

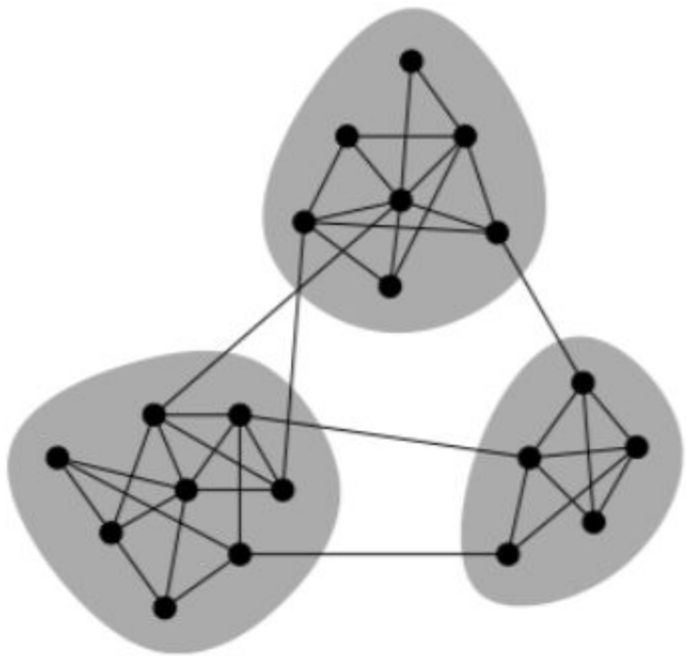
Blue – 4



Network Communities

Community

Network Communities are group of vertices such that vertices inside the group connected with many more edges than between groups

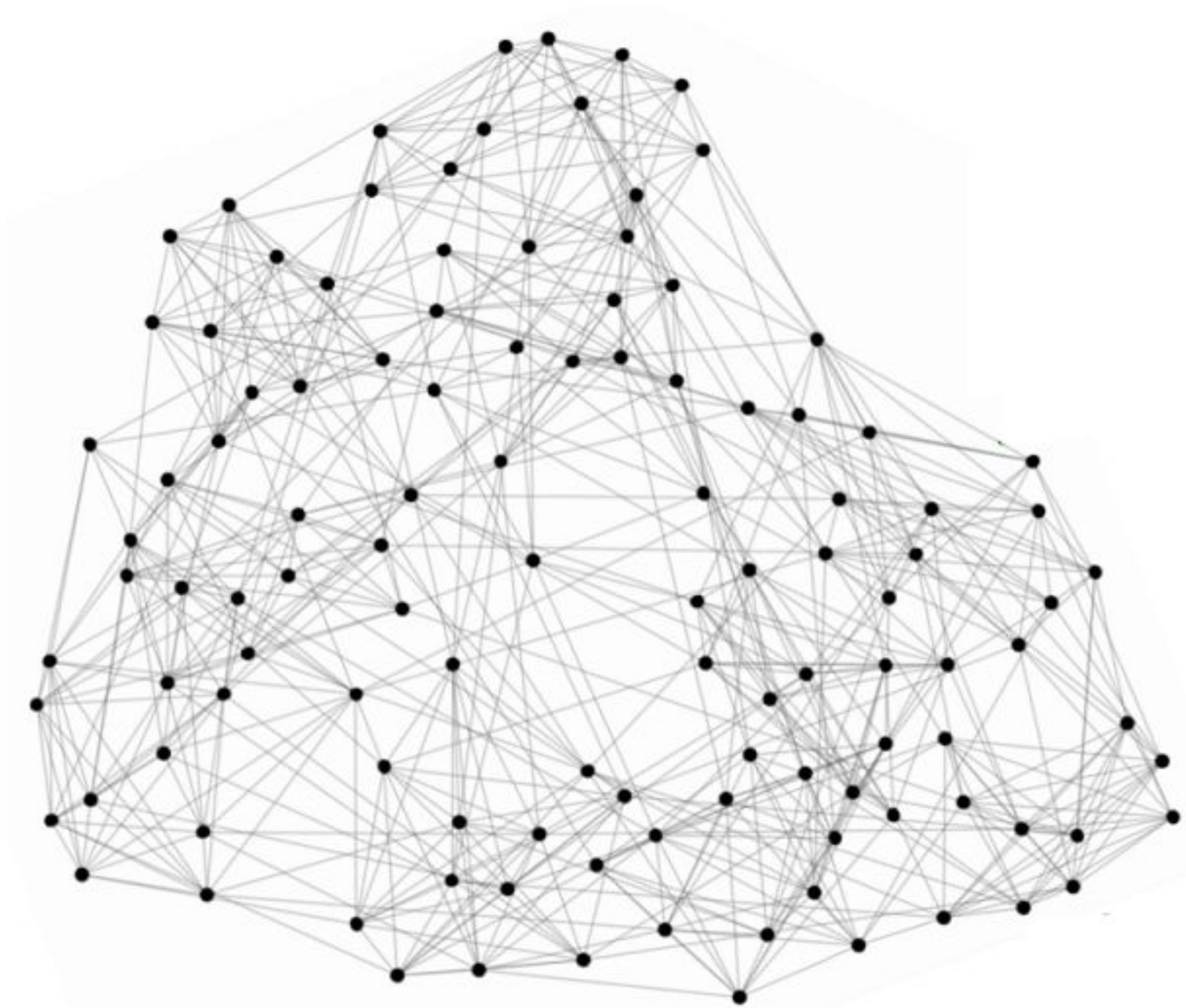


Community Types

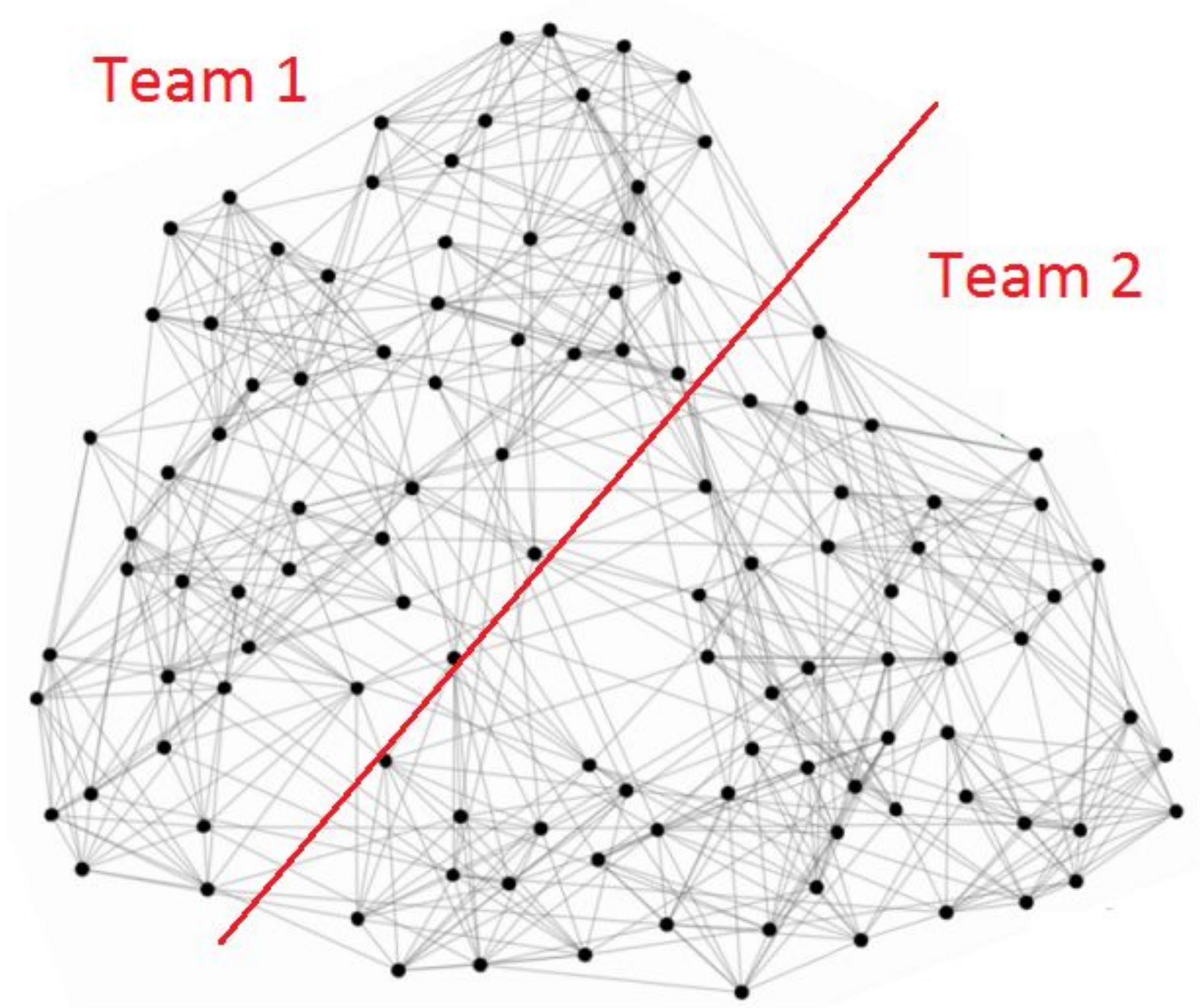
Communities can be either:

- Non-Overlapping
  - Country of residence
  - Working place
  - Favorite football club
- Overlapping
  - Types of friends (friends from school, army, work)
  - Groups of interest

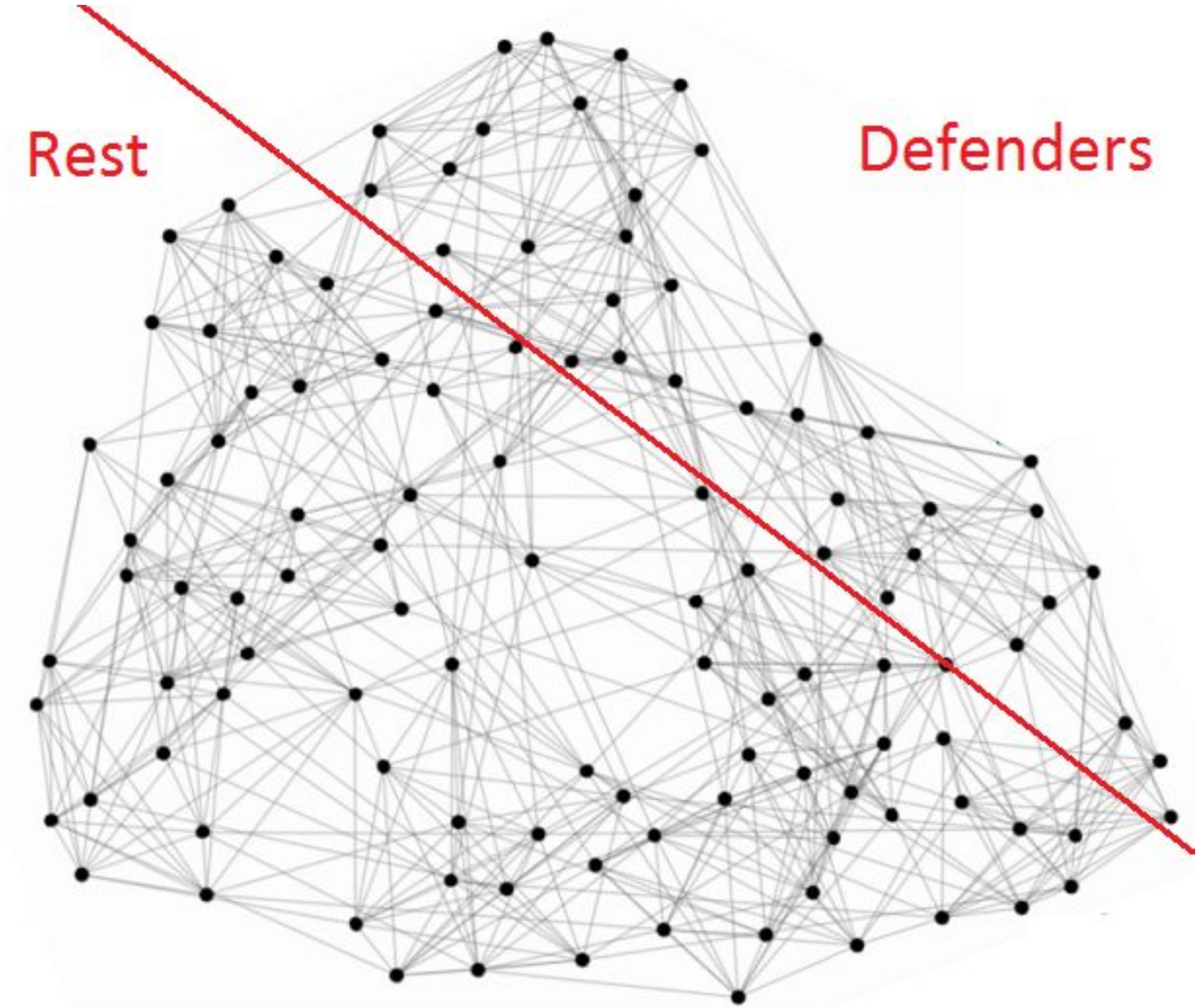
Example – Soccer teams



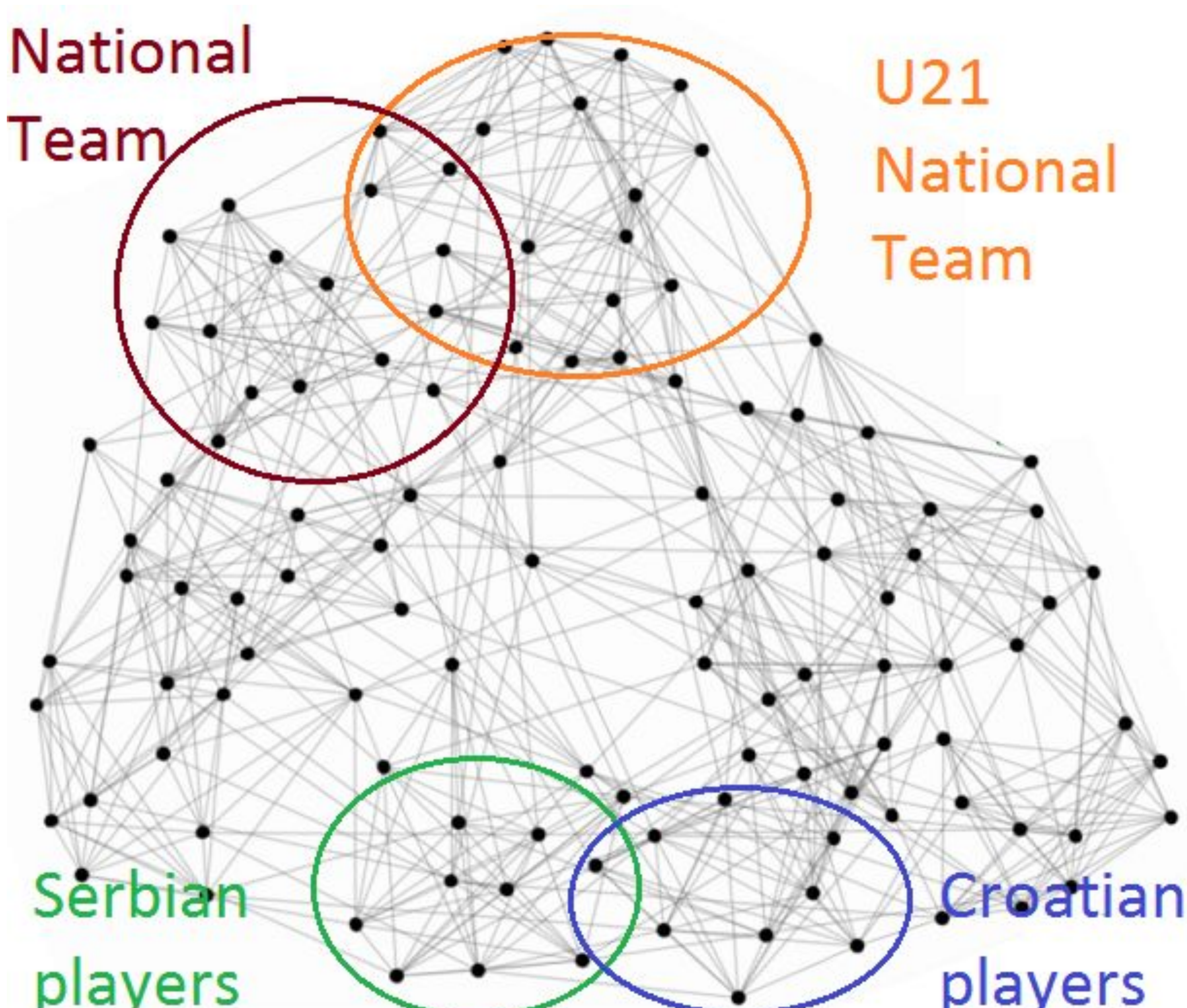
Example – Non-overlapping



Example – Non-overlapping



Example – Overlapping



Example

Example

What makes community?

- Mutuality of edges.
  - Almost everyone in the group has edge to one another
- Compactness.
  - Closeness or reachability of group members in small number of steps, not necessarily adjacency
- Density of edges.
  - High frequency of edges within the group
- Separation.
  - Higher frequency of ties among group members compared to non-members

Community density

- Network density: $$\rho = \frac{m}{n(n-1)/2}$$
- Inter-community density: $$\delta_{int}(C) = \frac{m_s}{n_s(n_s-1)/2}$$
- External edges density: $$\delta_{ext}(C) = \frac{m_{ext}}{n_c(n-n_c)}$$

Community has: $\delta_{int} > \rho,\ \delta_{ext} < \rho$

n_s – nodes in S , m_s – edges in S, m_{ext} – edges from S outside

Modularity

Modularity:

$$Q = \frac{1}{2m}\sum_{ij}\left(A_{ij} - \frac{k_i k_j}{2m}\right)\delta(c_i, c_j), = \sum_u (e_{uu} - a_u^2)$$

e_{uu} - fraction of edges within community u

$a_u = \sum_u e_{uv}$ - fraction of ends of edges attached to nodes in u

$$\delta_{ij} = \begin{cases} 0 & \text{if } i \neq j, \\ 1 & \text{if } i = j. \end{cases}$$ - Kronecker Delta

Higher modularity score – better community

Single community – Q = 0

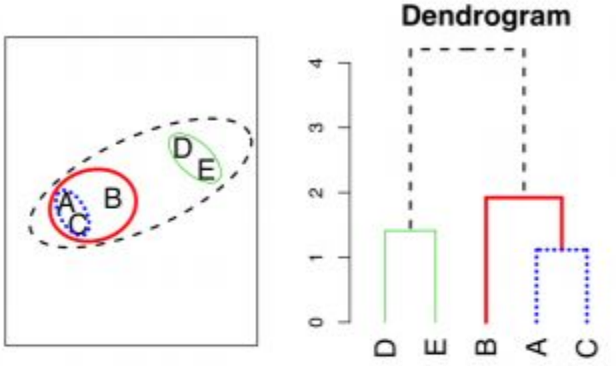
Detection of Communities

Similarity Based Clustering

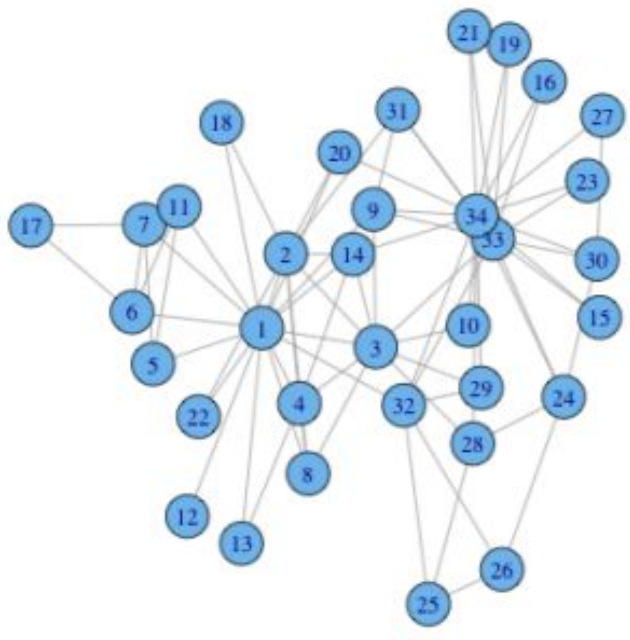
- Define similarity between every two nodes:
  - Jaccard
  - Cosine
  - …
- Build similarity matrix (compute all-pairs similarity)
- Group together nodes with high similarity

Agglomerative clustering

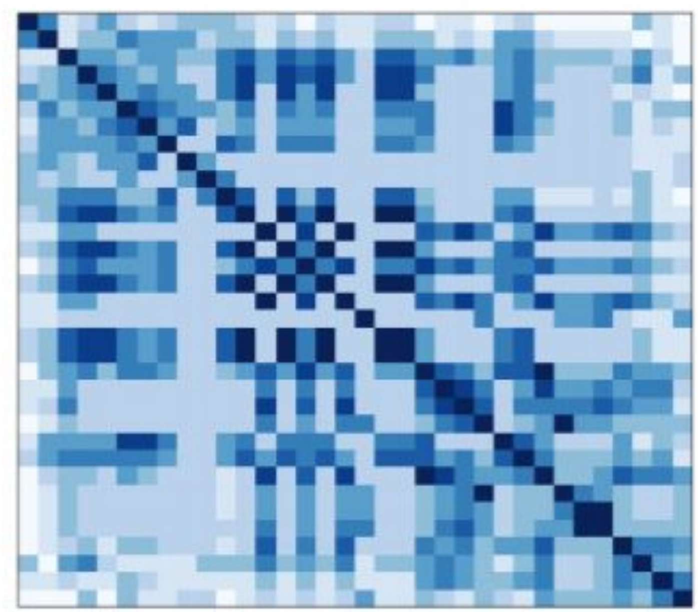
- Assign each vertex to a group of its own
- Find two groups with the highest similarity and join them in a single group
- Calculate similarity between groups:
  - single-linkage clustering (most similar in the group)
  - complete-linkage clustering (least similar in the group)
  - average-linkage clustering (mean similarity between groups)
- Repeat until all joined into single group



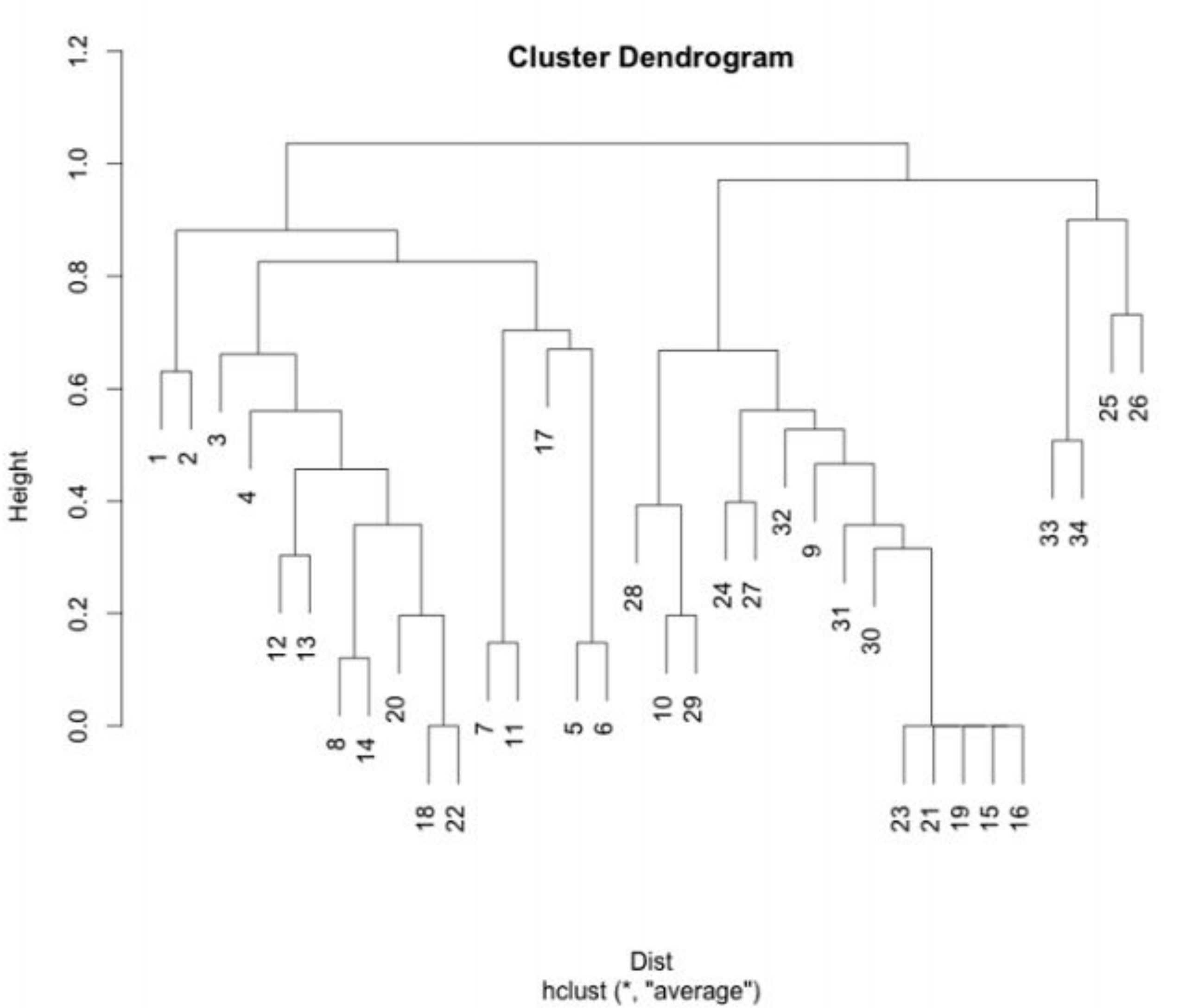
Similarity



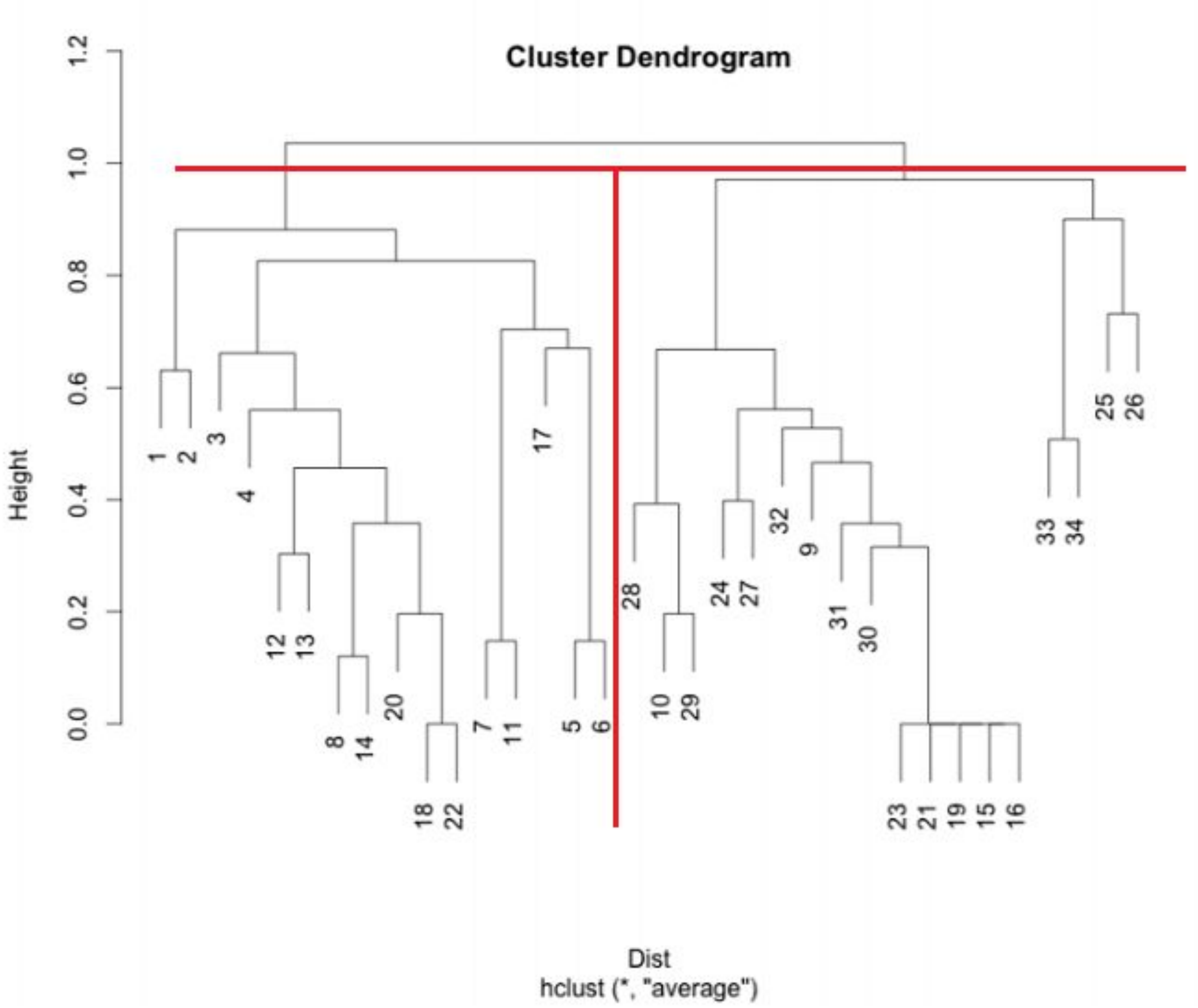
17
7
11
18
6
5
1
2
14
20
31
9
4
22
12
13
8
3
32
10
29
28
34
33
21
19
16
27
23
30
15
24
26
25



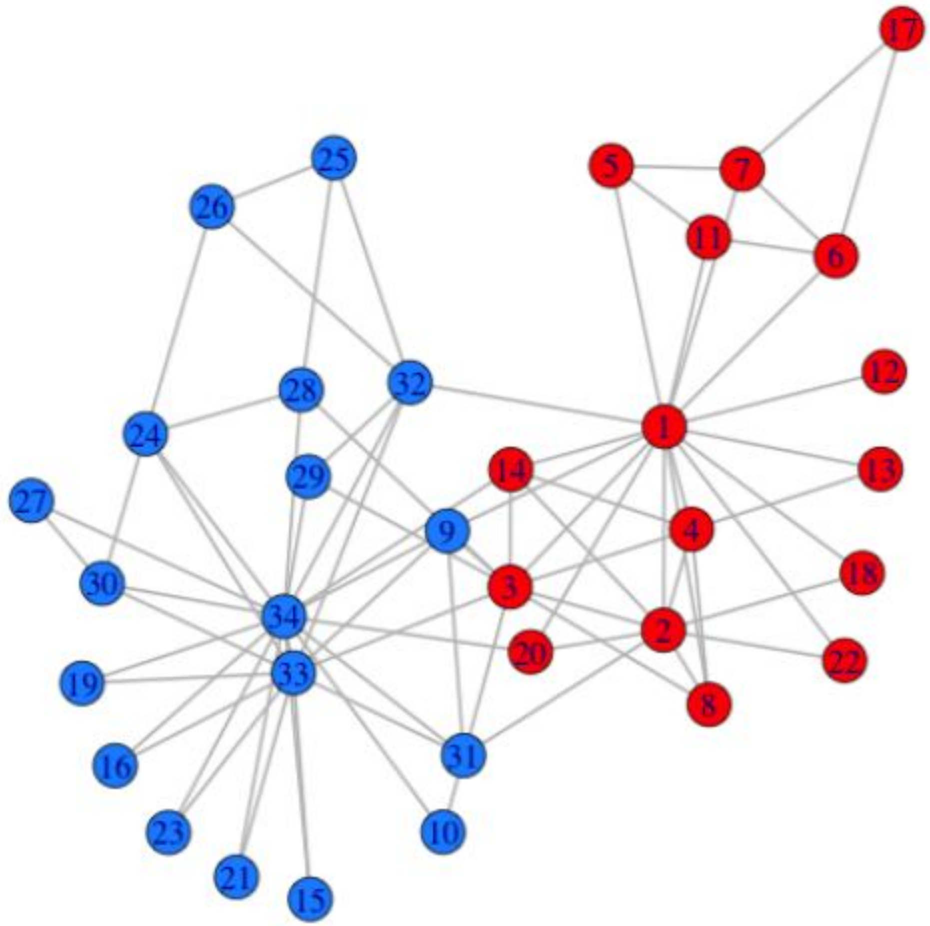
Karate club example



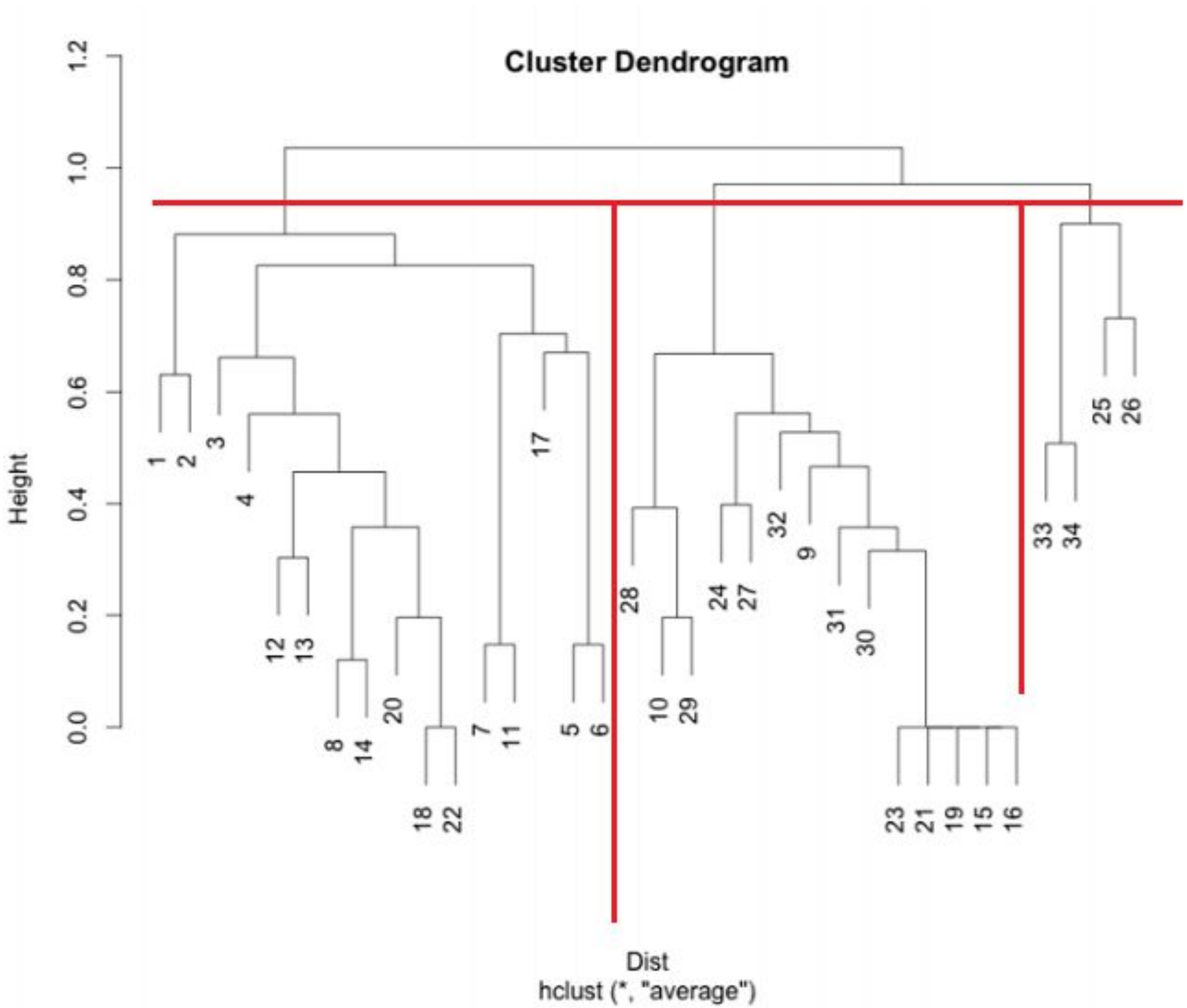
Karate club example



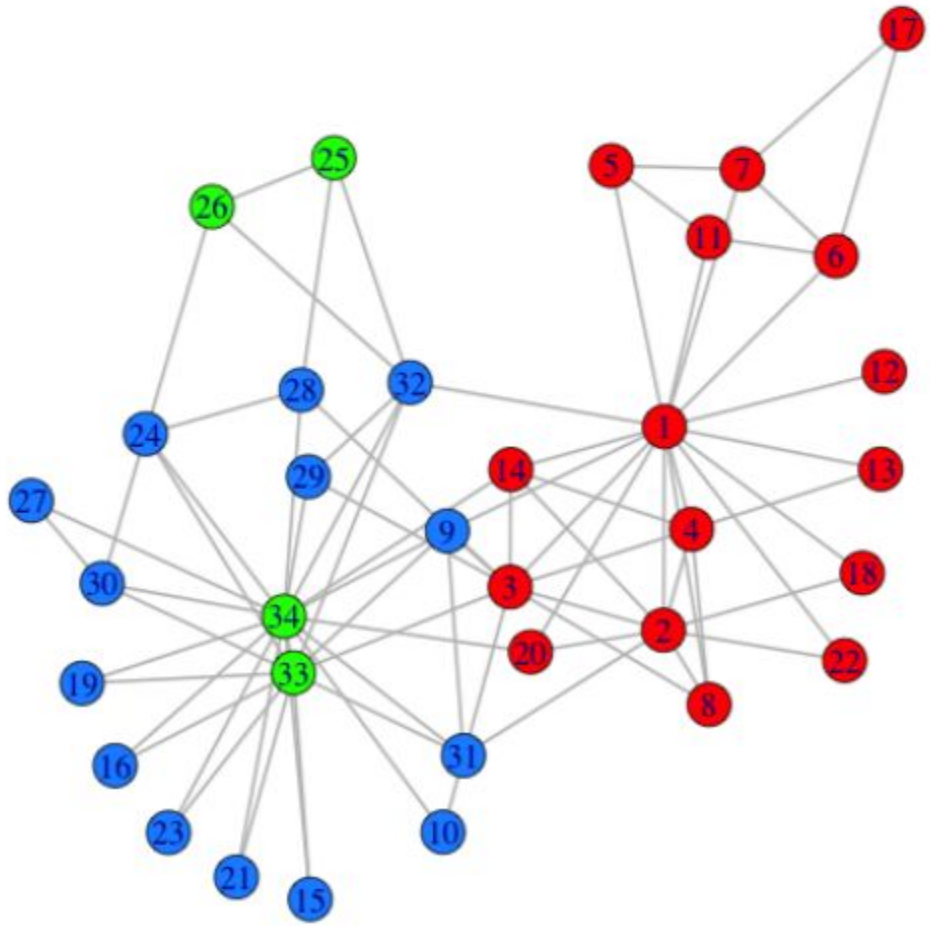
Karate club example



Karate club example



Karate club example



Graph Partitioning

Combinatorial problem:

- Number of ways to divide n nodes into n1, n2 nodes (n1+n2 = n), bi-partitioning

# of combinations = n! / (n1! * n2!)

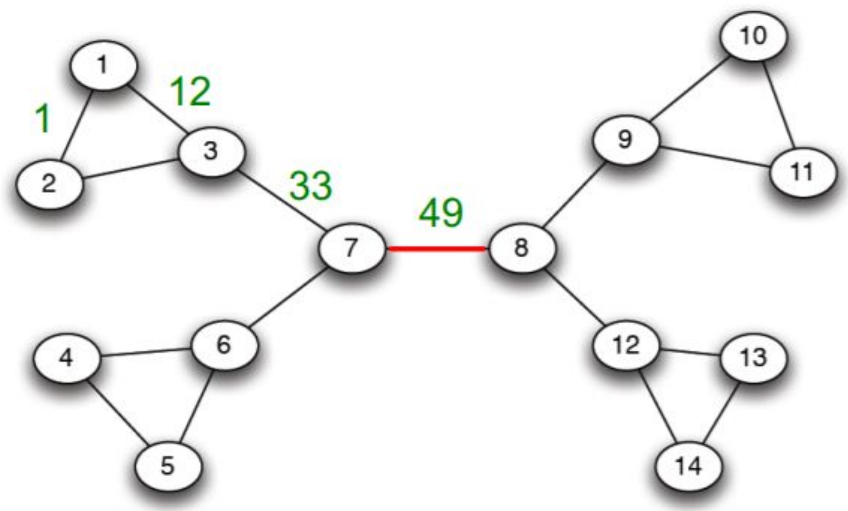
Community Detection

- Setup:
  - Undirected sparse graph (m << n^2)
  - Non-overlapping communities
  - Each community should be connected
  - Exact solution NP-hard, hence we are using heuristics / approximate algorithms

Edge Betweenness

- Number of shortest paths going via edge e

$$C_B(e) = \sum_{s \neq t} \frac{\sigma_{st}(e)}{\sigma_{st}}$$



Newman-Girvan algorithm

Algorithm: Newman-Girvan, 2004

Input: graph G(V,E)

Output: Dendrogram

repeat

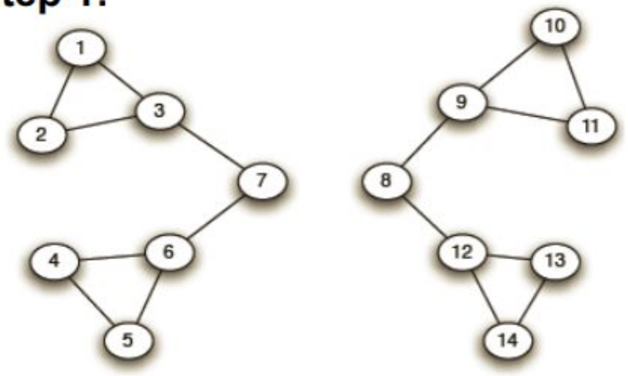
For all $e \in E$ compute edge betweenness $C_B(e)$;

remove edge e_i with largest $C_B(e_i)$;

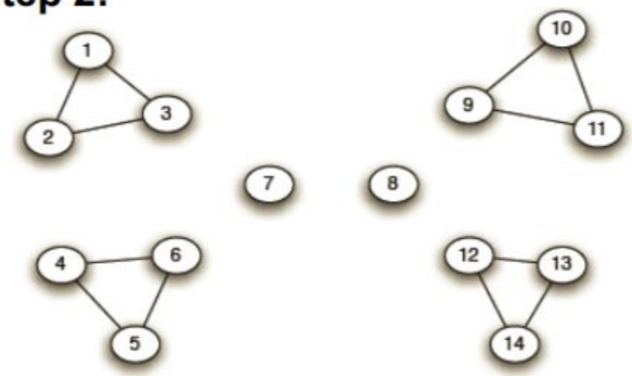
until *edges left*;

Step-by-step

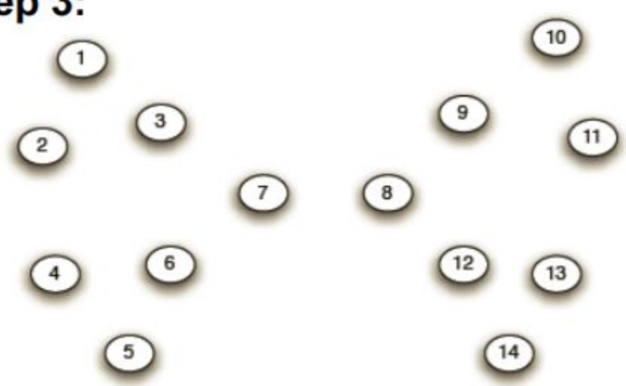
Step 1:



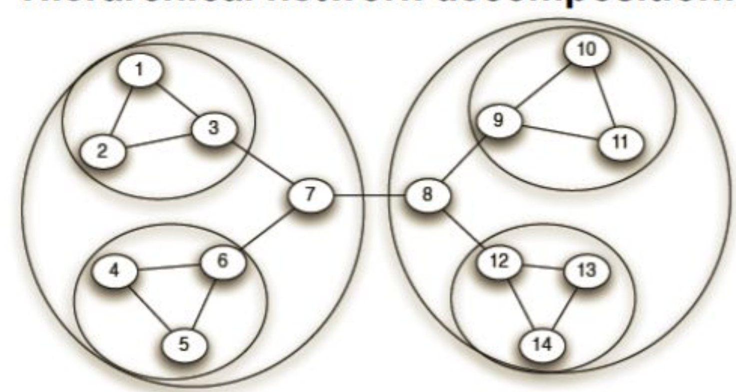
Step 2:



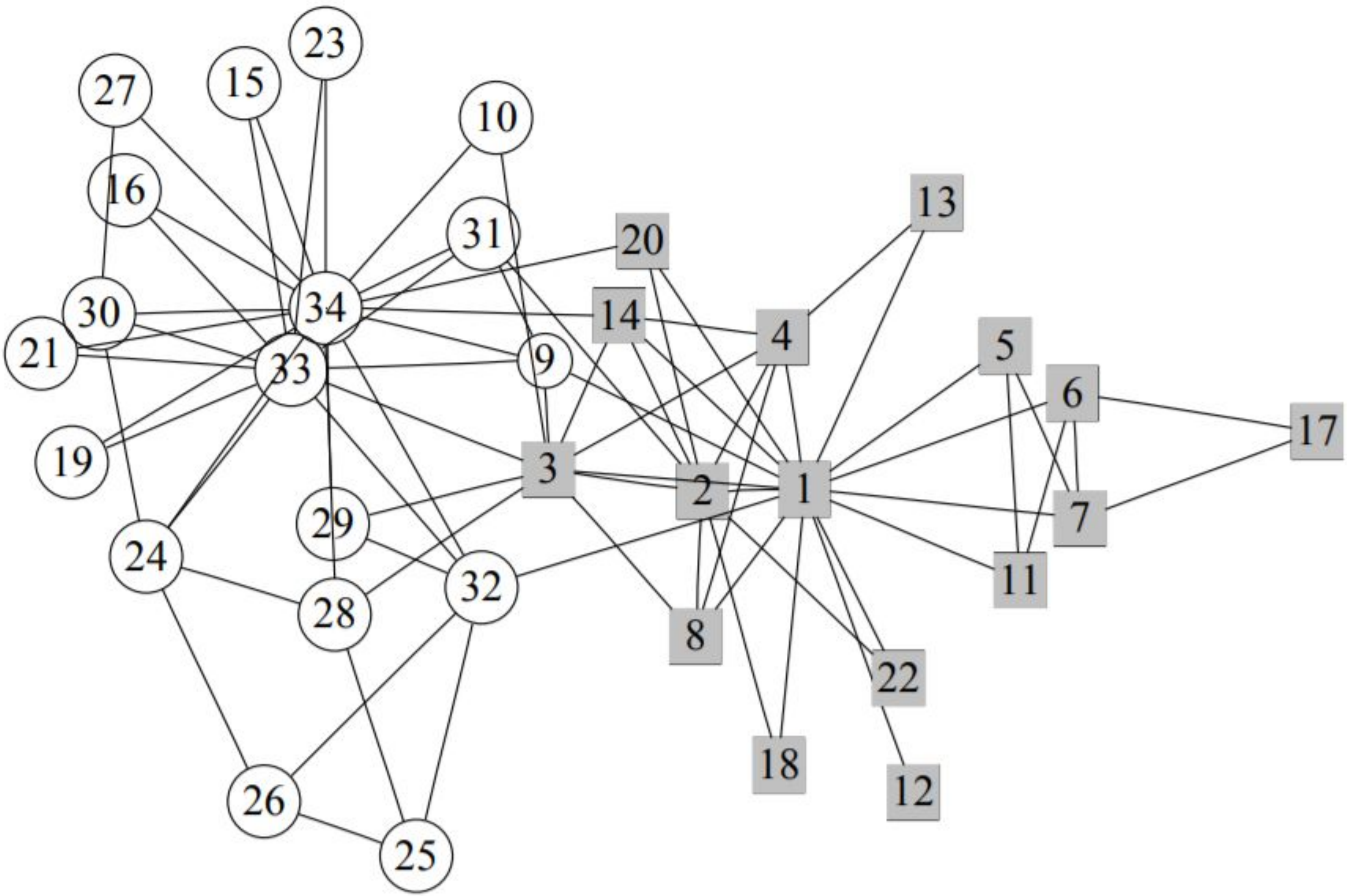
Step 3:



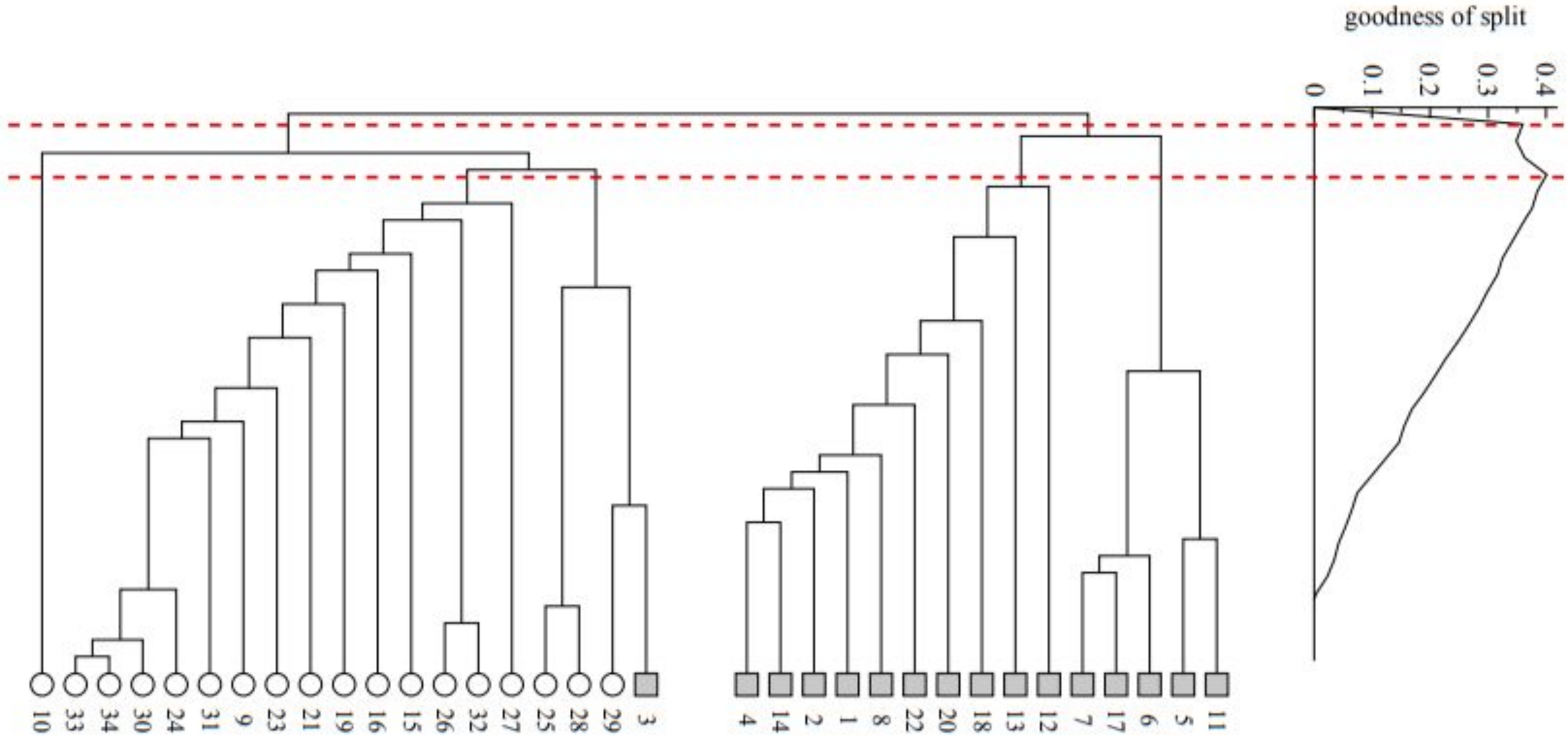
Hierarchical network decomposition:



Karate club example

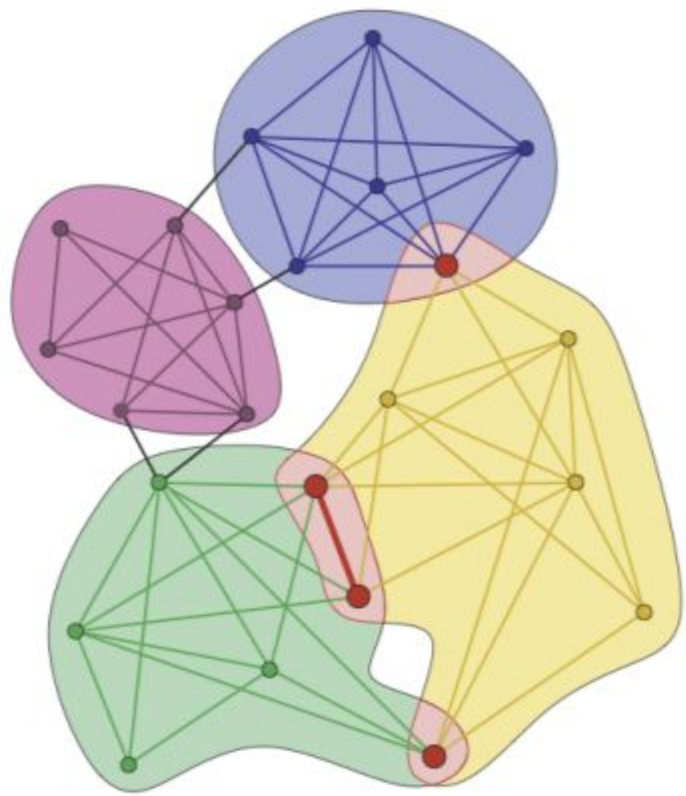


Karate club example



Overlapping Communities

- Next lesson we will extend the community detection task to find overlapping communities



Thank you!

Questions?