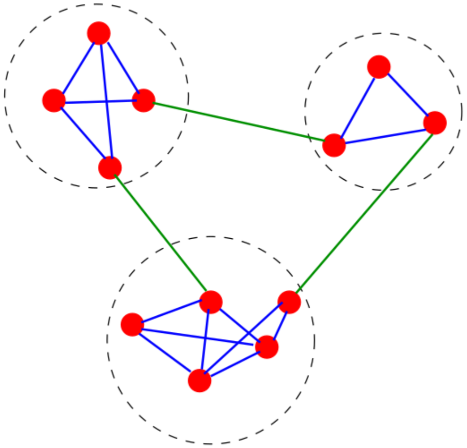
ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

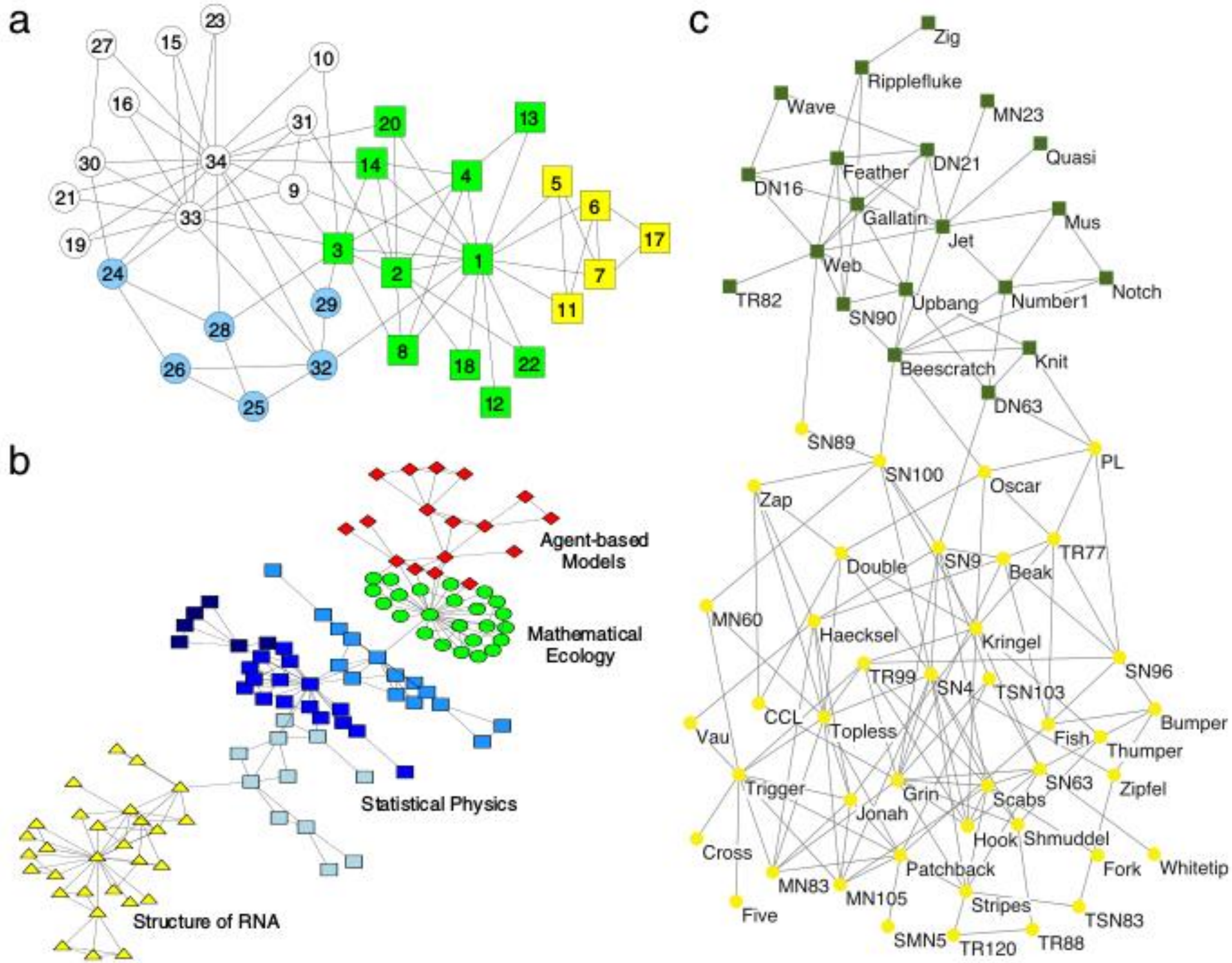
Lesson 5: Social Network Analysis

2. Community detection
2.1 Community detection

- Detect community in network
- Community members are similar in nature
- Communities can be related
- The number of communities depends on the algorithm



from Fortunato (2015)

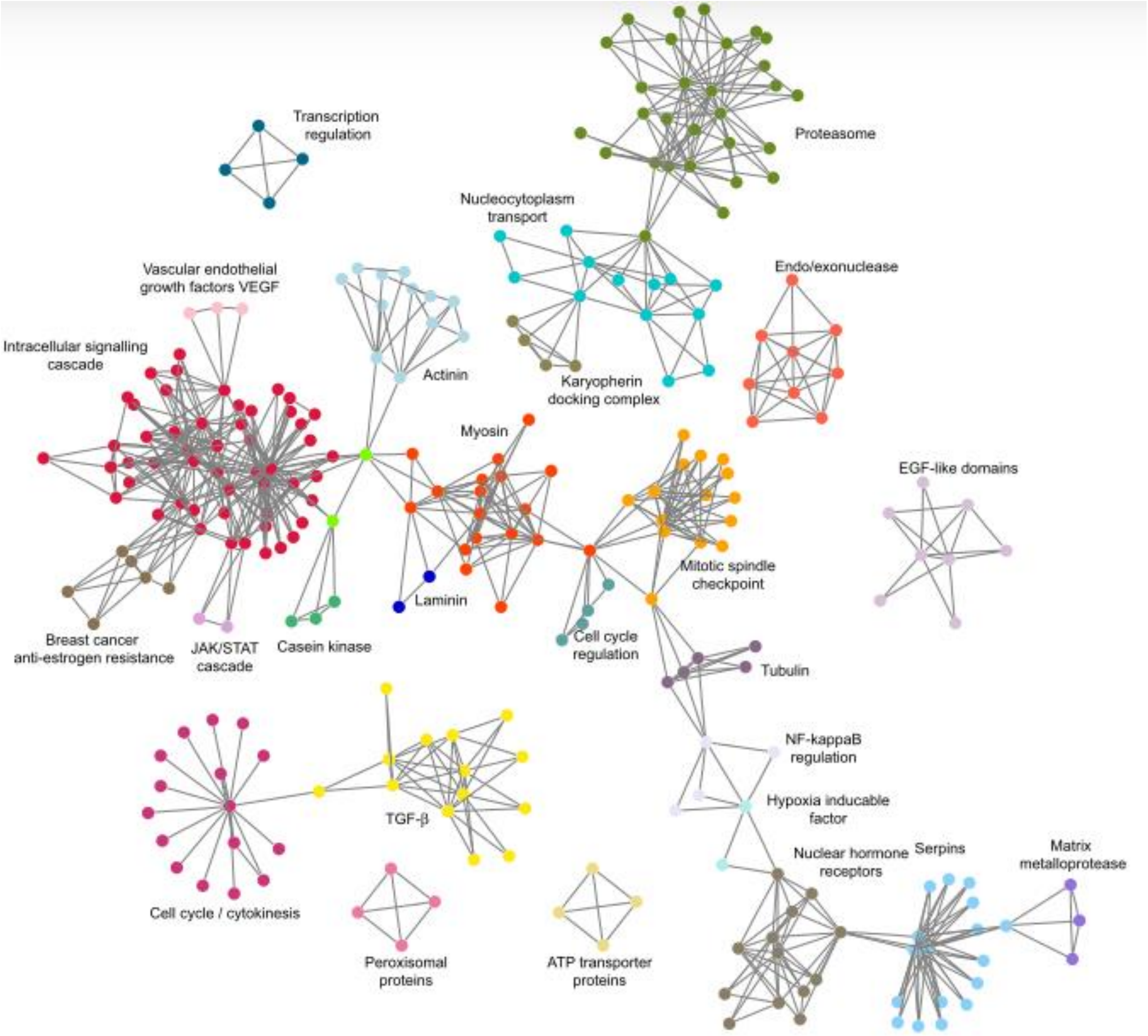


a) Zachary's karate club

b) Collaboration network between scientists working at the Santa Fe Institute

c) Lusseau's network of bottlenose dolphins

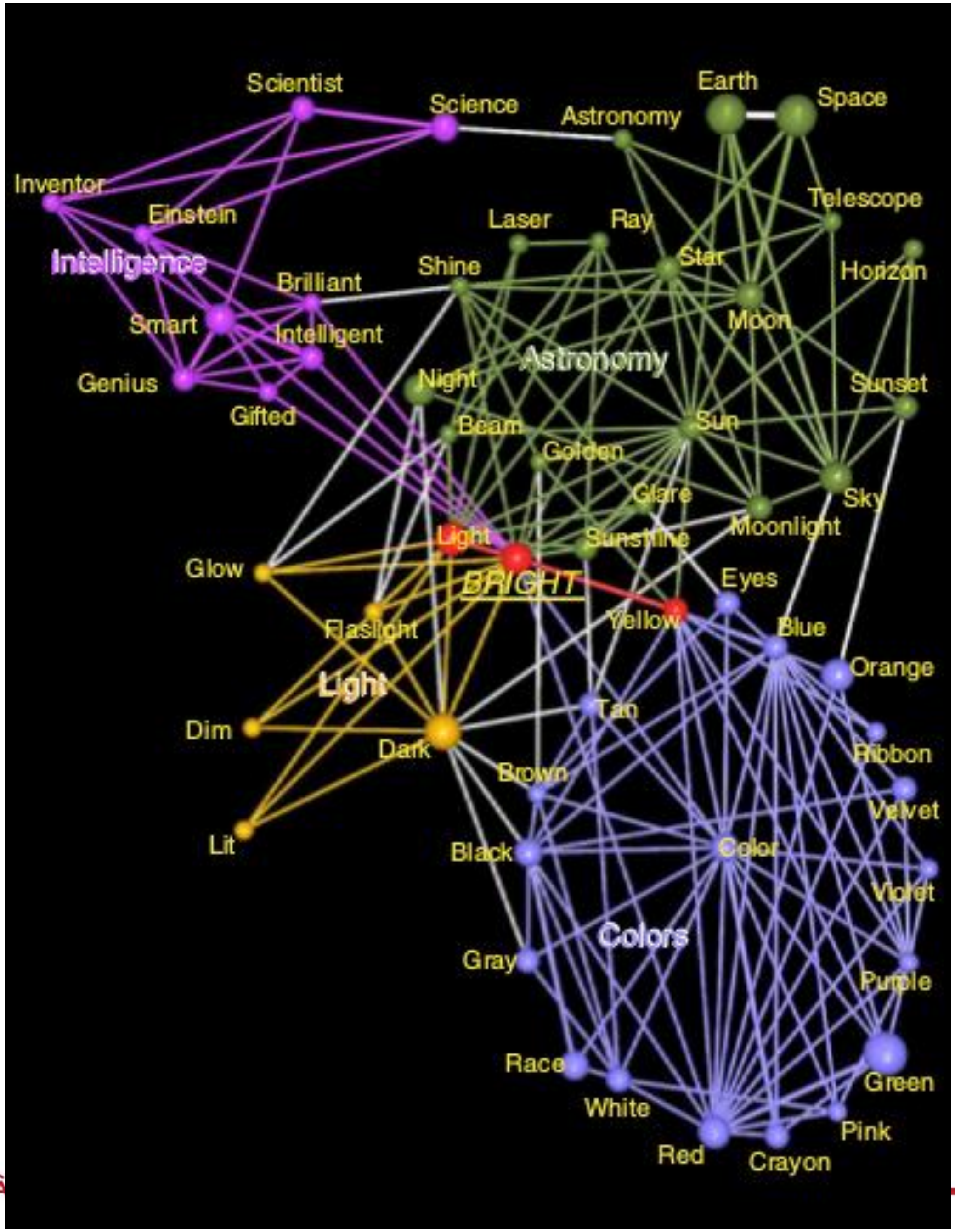
from Fortunato (2015)



Community structure in protein–protein interaction networks

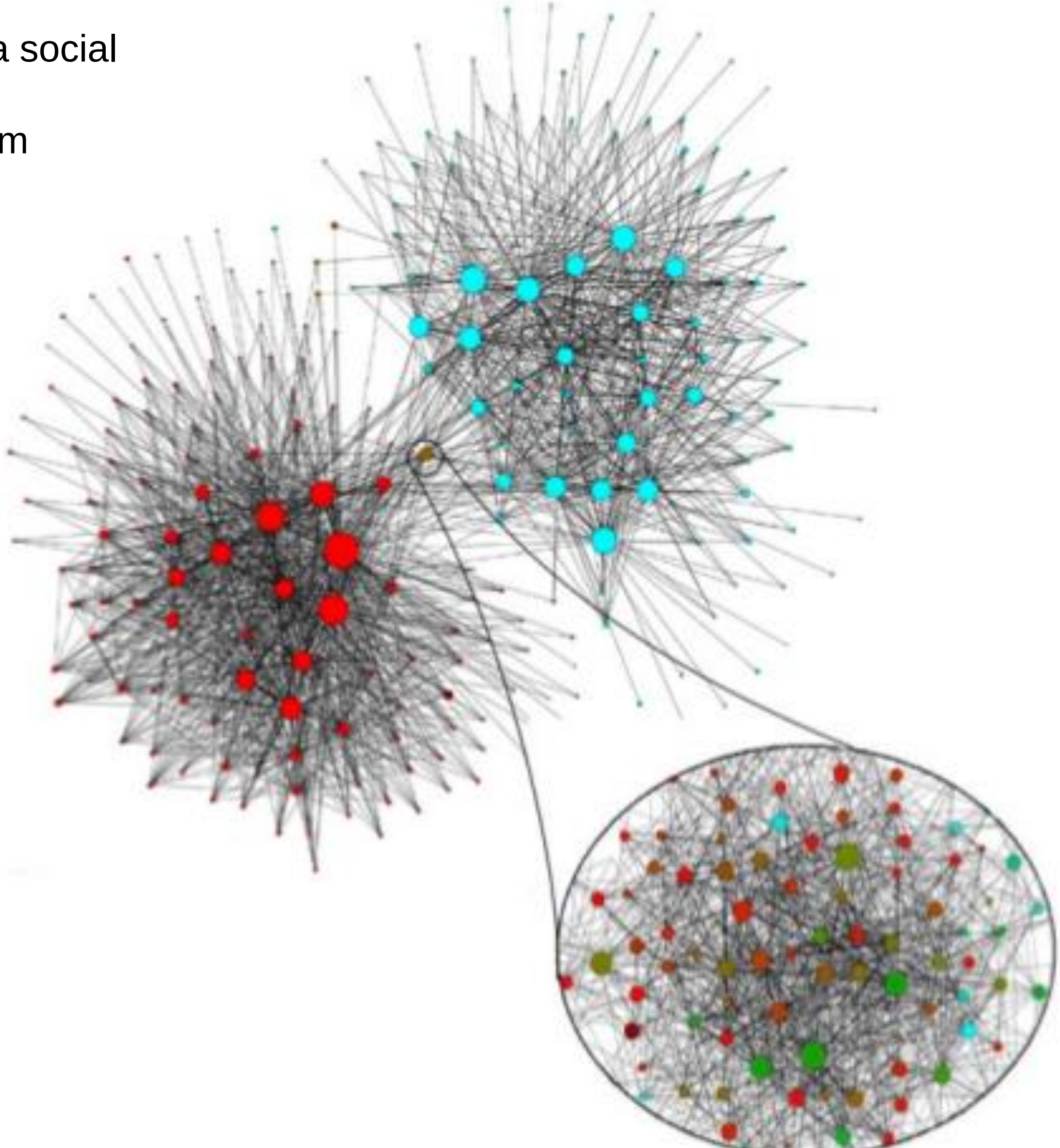
from Fortunato (2015)

Overlapping communities in a network of word association



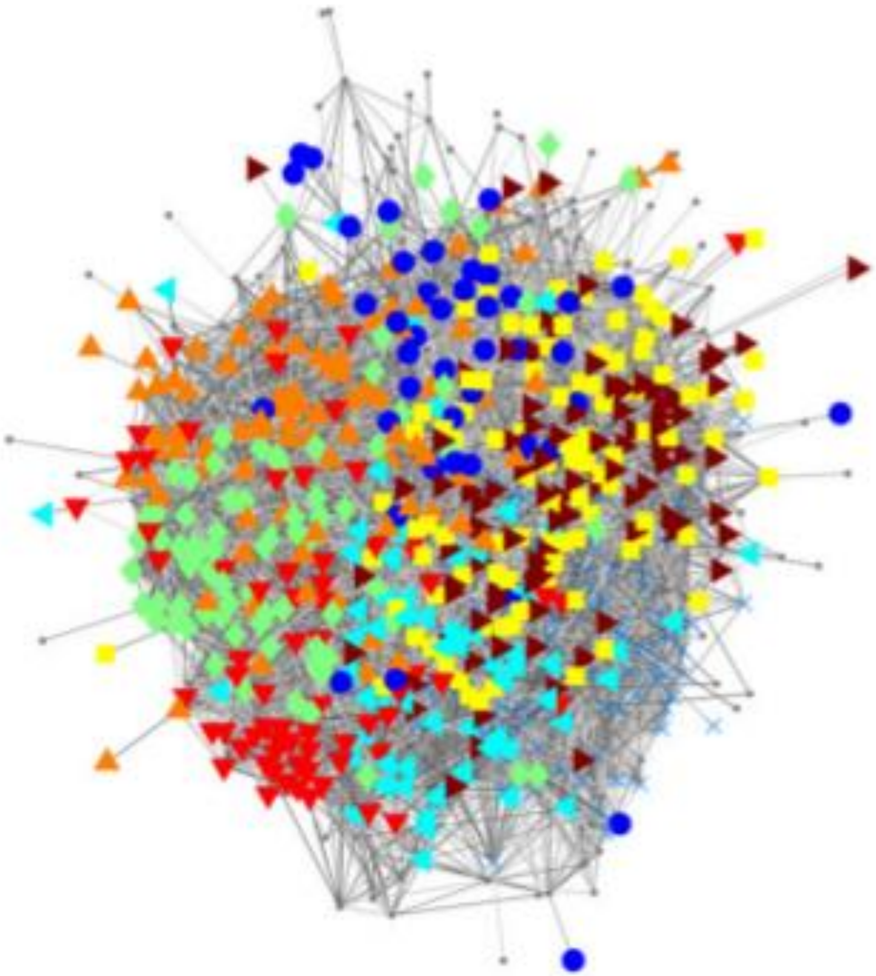
from Fortunato (2015)

Community structure of a social network of mobile phone communication in Belgium

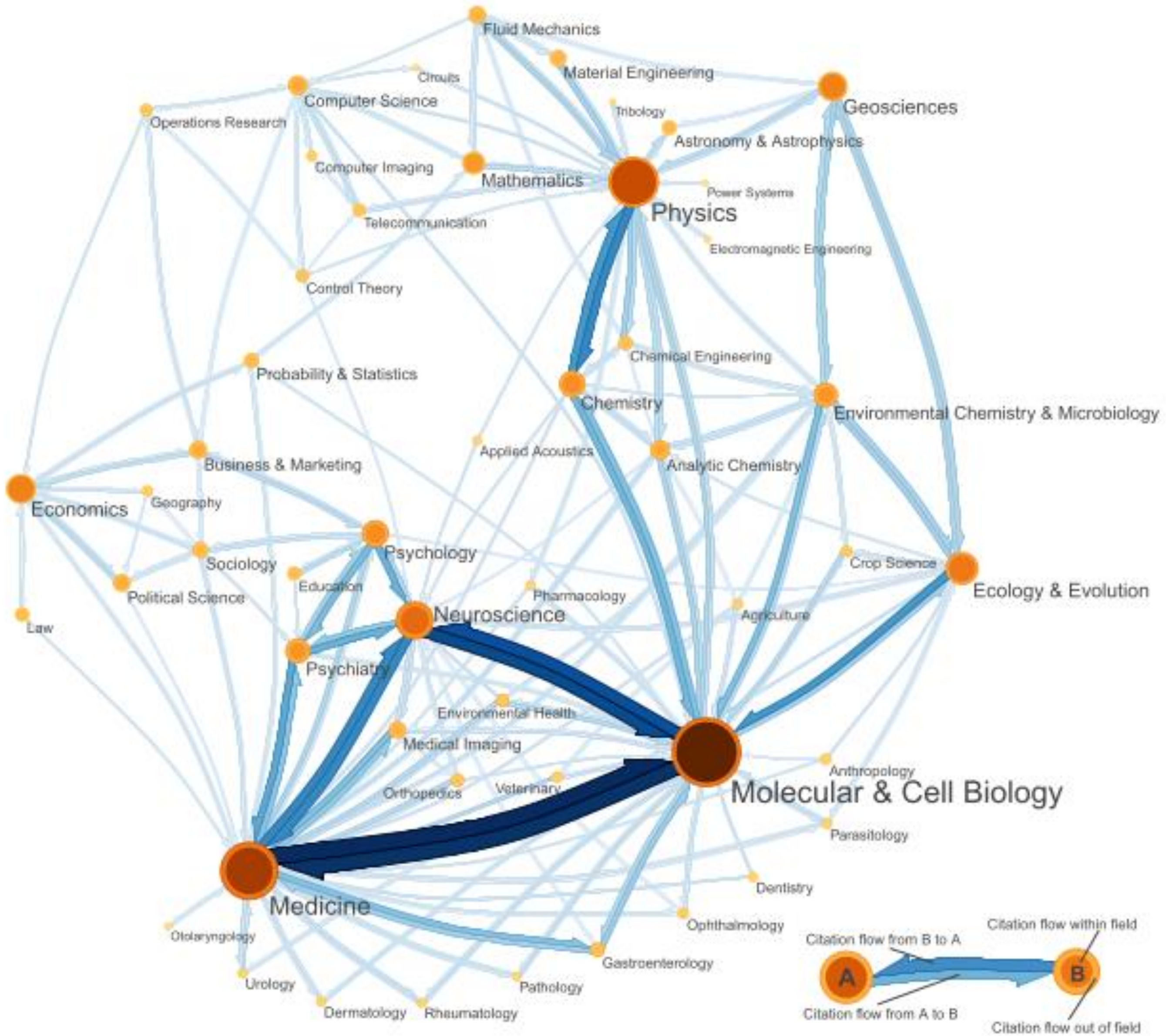


from Fortunato (2015)

Network of friendships
between students at Caltech



from Fortunato (2015)

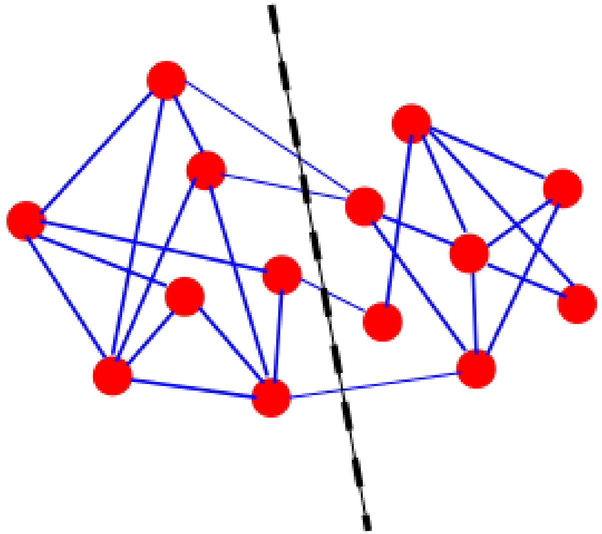


Map of science derived from a clustering analysis of a citation network

from Fortunato (2015)

2.2 Kernighan–Lin algorithm

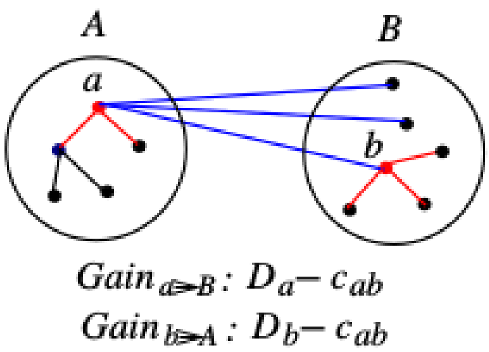
Minimum cut problem: Divide the domain of the undirected graph into two regions with the same number of vertices so that the sum of the weights of the edges connecting the two clusters is minimal.



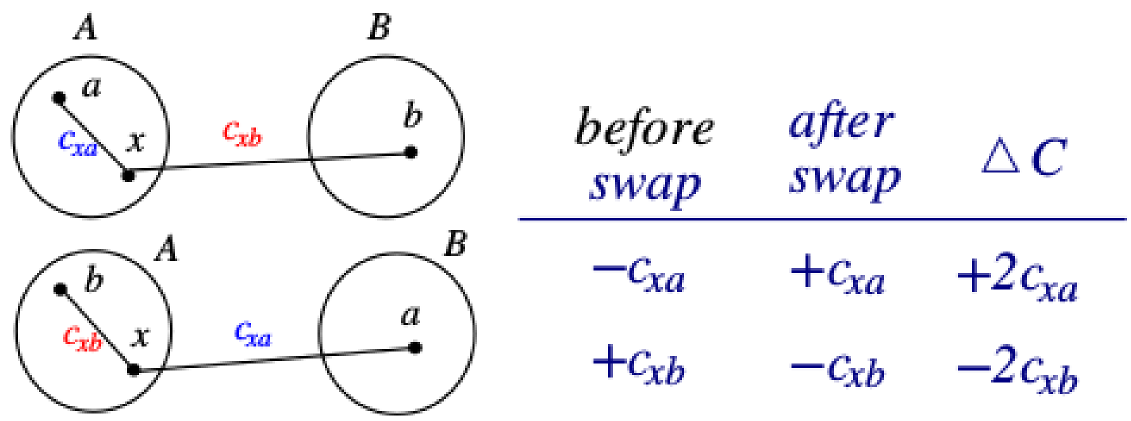
Algorithm

- $G = (V, E)$
- Separate nodes into to set A and B without overlap
- a ∈ A:
  - Intra-cost $I_a = \Sigma_{u \in A}\, c_{a,u}$
  - Inter-cost $E_a = \Sigma_{\in B}\, c_{a,v}$
  - $D_a = E_a - I_a$
- b ∈ B, cost decrease if swap a và b
  - $\mathrm{T}_{\mathrm{old}} - \mathrm{T}_{\mathrm{new}} = \mathrm{D}_\mathrm{a} + \mathrm{D}_\mathrm{b} - 2\mathrm{c}_{\mathrm{a,b}}$
- Repeat find feasible pair (a,b) to reduce cost while sum of cost (of the cut) decrease

Update cost

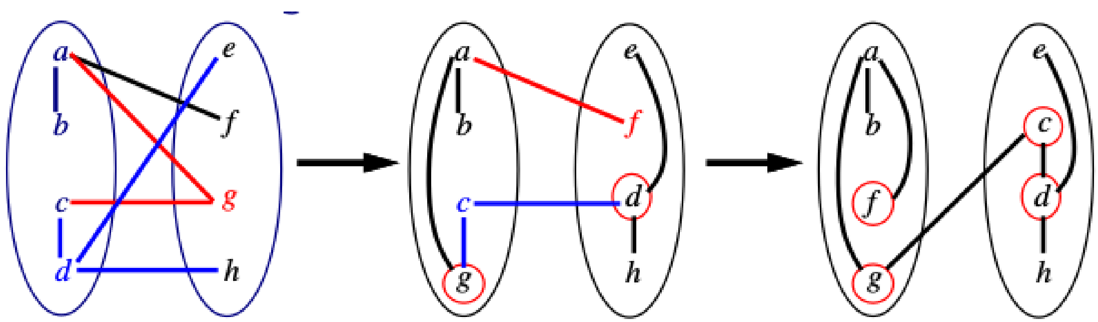


Internal cost vs. External cost



updating D–values

Example



Algorithm

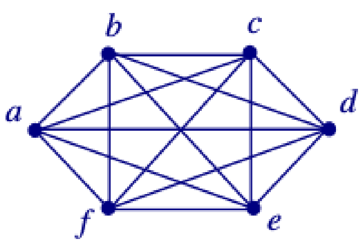
Algorithm: Kernighan-Lin(G)
Input: $G = (V, E), |V| = 2n$.
Output: Balanced bi-partition A and B with ''small'' cut cost.

begin
Bipartition G into A and B such that $|V_A| = |V_B|$, $V_A \cap V_B = \emptyset$, and $V_A \cup V_B = V$.
repeat
  Compute D_v, $\forall v \in V$.
  for $i = 1$ **to** n **do**
    Find a pair of unlocked vertices $v_{ai} \in V_A$ and $v_{bi} \in V_B$ whose exchange makes the largest decrease or smallest increase in cut cost;
    Mark v_{ai} and v_{bi} as locked, store the gain $\hat{g}_i$, and compute the new D_v, for all unlocked $v \in V$;
  Find k, such that $G_k = \sum_{i=1}^{k} \hat{g}_i$ is maximized;
  if $G_k > 0$ **then**
    Move $v_{a1}, \ldots, v_{ak}$ from V_A to V_B and $v_{b1}, \ldots, v_{bk}$ from V_B to V_A;
  Unlock v, $\forall v \in V$.
until $G_k \leq 0$;
end

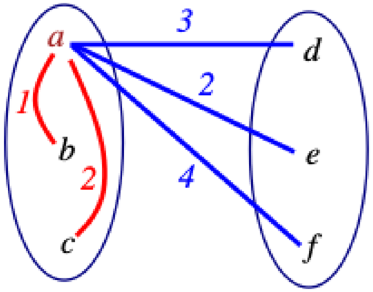
Complexity

- Initialize D: O(n^2) (line 4)
- Loop: O(n) (line 5)
- Loop body: O(n^2)
  - Step i need $(n – i + 1)^2$ time
- Each loop: O(n^3) (line 4-11)
- Assume that algorithm terminate after r loops
- Total time: O(rn^3)

Example



	a	b	c	d	e	f
a	0	1	2	3	2	4
b	1	0	1	4	2	1
c	2	1	0	3	2	1
d	3	4	3	0	4	3
e	2	2	2	4	0	2
f	4	1	1	3	2	0



costs associated with a

Initial cut cost = (3+2+4)+(4+2+1)+(3+2+1) = 22

- Iteration 1:

$$
\begin{array}{lll}
I_a = 1 + 2 = 3; & E_a = 3 + 2 + 4 = 9; & D_a = E_a - I_a = 9 - 3 = 6 \\
I_b = 1 + 1 = 2; & E_b = 4 + 2 + 1 = 7; & D_b = E_b - I_b = 7 - 2 = 5 \\
I_c = 2 + 1 = 3; & E_c = 3 + 2 + 1 = 6; & D_c = E_c - I_c = 6 - 3 = 3 \\
I_d = 4 + 3 = 7; & E_d = 3 + 4 + 3 = 10; & D_d = E_d - I_d = 10 - 7 = 3 \\
I_e = 4 + 2 = 6; & E_e = 2 + 2 + 2 = 6; & D_e = E_e - I_e = 6 - 6 = 0 \\
I_f = 3 + 2 = 5; & E_f = 4 + 1 + 1 = 6; & D_f = E_f - I_f = 6 - 5 = 1
\end{array}
$$

Example (cont.)

- Iteration 1:

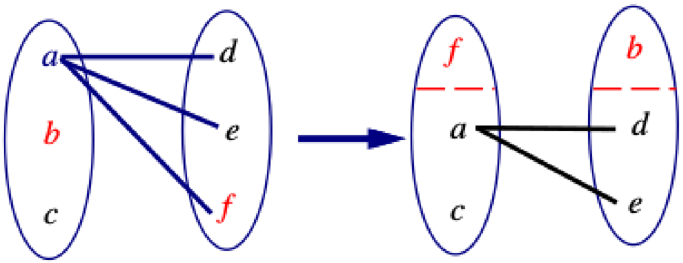
$$
\begin{array}{lll}
I_a = 1+2 = 3; & E_a = 3+2+4 = 9; & D_a = E_a - I_a = 9-3 = 6 \\
I_b = 1+1 = 2; & E_b = 4+2+1 = 7; & D_b = E_b - I_b = 7-2 = 5 \\
I_c = 2+1 = 3; & E_c = 3+2+1 = 6; & D_c = E_c - I_c = 6-3 = 3 \\
I_d = 4+3 = 7; & E_d = 3+4+3 = 10; & D_d = E_d - I_d = 10-7 = 3 \\
I_e = 4+2 = 6; & E_e = 2+2+2 = 6; & D_e = E_e - I_e = 6-6 = 0 \\
I_f = 3+2 = 5; & E_f = 4+1+1 = 6; & D_f = E_f - I_f = 6-5 = 1
\end{array}
$$

- $g_{xy} = D_x + D_y - 2c_{xy}$.

$$
\begin{aligned}
g_{ad} &= D_a + D_d - 2c_{ad} = 6+3-2\times 3 = 3 \\
g_{ae} &= 6+0-2\times 2 = 2 \\
g_{af} &= 6+1-2\times 4 = -1 \\
g_{bd} &= 5+3-2\times 4 = 0 \\
g_{be} &= 5+0-2\times 2 = 1 \\
g_{bf} &= 5+1-2\times 1 = 4 \ (maximum) \\
g_{cd} &= 3+3-2\times 3 = 0 \\
g_{ce} &= 3+0-2\times 2 = -1 \\
g_{cf} &= 3+1-2\times 1 = 2
\end{aligned}
$$

- Swap b and f! ($\widehat{g_1} = 4$)

Example (cont.)



- $D'_x = D_x + 2c_{xp} - 2c_{xq}, \forall x \in A - \{p\}$ (swap p and q, $p \in A$, $q \in B$)

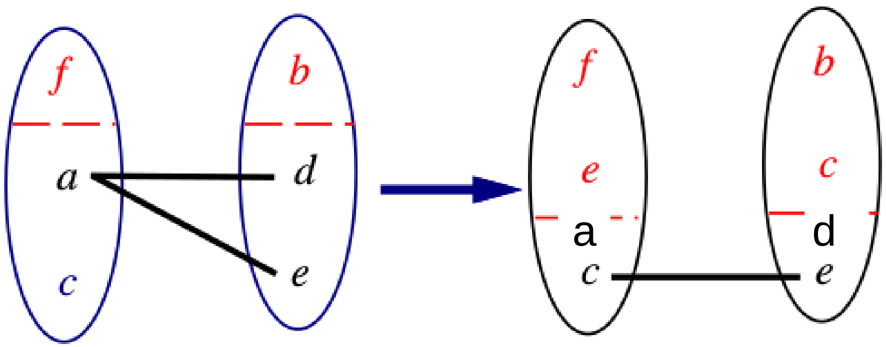
$$\begin{aligned} D'_a &= D_a + 2c_{ab} - 2c_{af} = 6 + 2 \times 1 - 2 \times 4 = 0 \\ D'_c &= D_c + 2c_{cb} - 2c_{cf} = 3 + 2 \times 1 - 2 \times 1 = 3 \\ D'_d &= D_d + 2c_{df} - 2c_{db} = 3 + 2 \times 3 - 2 \times 4 = 1 \\ D'_e &= D_e + 2c_{ef} - 2c_{eb} = 0 + 2 \times 2 - 2 \times 2 = 0 \end{aligned}$$

- $g_{xy} = D'_x + D'_y - 2c_{xy}$.

$$\begin{aligned} g_{ad} &= D'_a + D'_d - 2c_{ad} = 0 + 1 - 2 \times 3 = -5 \\ g_{ae} &= D'_a + D'_e - 2c_{ae} = 0 + 0 - 2 \times 2 = -4 \\ g_{cd} &= D'_c + D'_d - 2c_{cd} = 3 + 1 - 2 \times 3 = -2 \\ g_{ce} &= D'_c + D'_e - 2c_{ce} = 3 + 0 - 2 \times 2 = -1 \ (maximum) \end{aligned}$$

- Swap c and e! ($\hat{g_2} = -1$)

Example (cont.)



- $D''_x = D'_x + 2c_{xp} - 2c_{xq}, \forall x \in A - \{p\}$

$$\begin{aligned} D''_a &= D'_a + 2c_{ac} - 2c_{ae} = 0 + 2 \times 2 - 2 \times 2 = 0 \\ D''_d &= D'_d + 2c_{de} - 2c_{dc} = 1 + 2 \times 4 - 2 \times 3 = 3 \end{aligned}$$

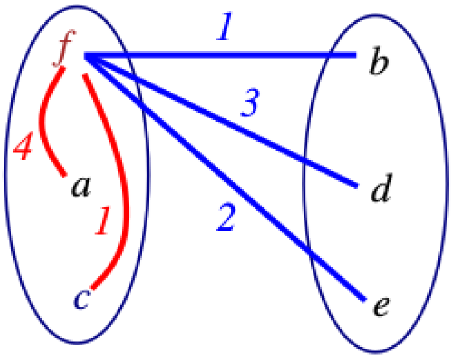
- $g_{xy} = D''_x + D''_y - 2c_{xy}$.

$$g_{ad} = D''_a + D''_d - 2c_{ad} = 0 + 3 - 2 \times 3 = -3(\hat{g}_3 = -3)$$

- Note that this step is redundant ($\sum_{i=1}^{n} \hat{g}_i = 0$).
- Summary: $\hat{g}_1 = g_{bf} = 4$, $\hat{g}_2 = g_{ce} = -1$, $\hat{g}_3 = g_{ad} = -3$.
- Largest partial sum $\max \sum_{i=1}^{k} \hat{g}_i = 4$ $(k = 1) \Rightarrow$ Swap b and f.

Example (cont.)

	a	b	c	d	e	f
a	0	1	2	3	2	4
b	1	0	1	4	2	1
c	2	1	0	3	2	1
d	3	4	3	0	4	3
e	2	2	2	4	0	2
f	4	1	1	3	2	0



Initial cut cost = (1+3+2)+(1+3+2)+(1+3+2) = 18 (22−4)

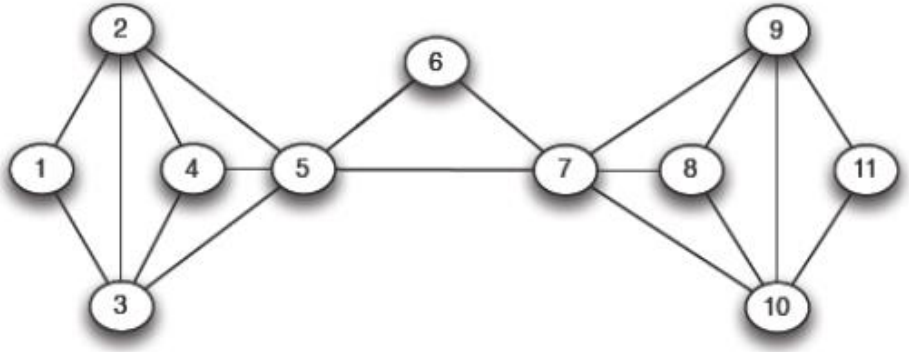
- Iteration 2: Repeat what we did at Iteration 1 (Initial cost$= 22-4 = 18$).
- Summary: $\hat{g_1} = g_{ce} = -1$, $\hat{g_2} = g_{ab} = -3$, $\hat{g_3} = g_{fd} = 4$.
- Largest partial sum $=$ max $\sum_{i=1}^{k} \hat{g_i} = 0$ $(k = 3) \Rightarrow$ Stop!

2.3 Girvan-Newman algorithm

- Find bridges between communities based on “edge betweenness”
- Repeat with each community to fin subcommunity
- Final results are hierarchical tree with root is the whole graph and leaves are nodes

Bridge and edge betweeness

- Bridge: connect communities
- Edge betweeness of an edge is the number of shortest paths between pairs of nodes that run along it. If there is k shortest path between a pair of nodes, each path is 1/k
- E.g: from node 1 to node 5 there 2 path, each has ½ flow unit



Bridge and edge betweeness

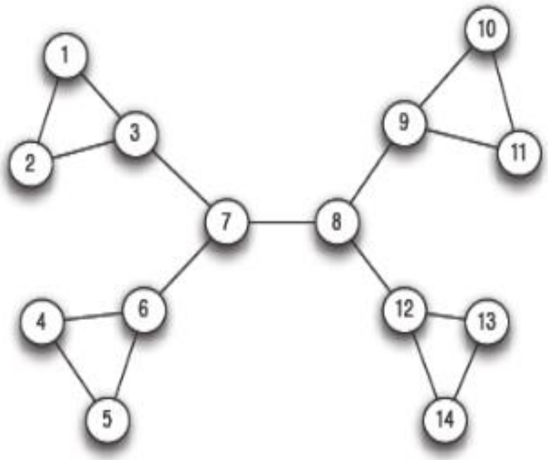
- E.g: edge betweeness

  7-8: 49

  3-7: 33

  1-3: 12

  1-2: 1



Algorithm

ALgorithm:

1) Compute betweeness of every edges

2) Remove edges with highest betweeness

3) Recompute betweeness

4) Go back to step 2, repeat until there no edge left

Compute betweeness

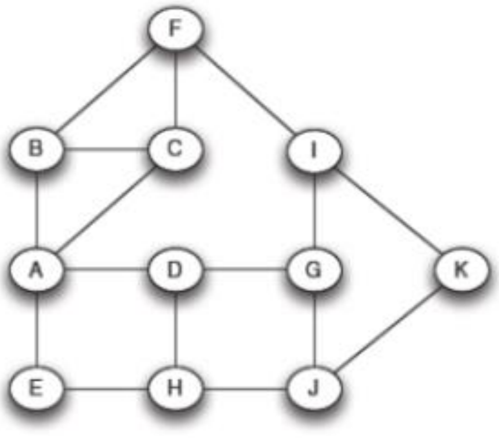
- Compute throughput using BFS
- For each node *u*
    - 1) BFS from node u
    - 2) Find number of shortest path from *u* to other nodes
    - 3) Find number of shoertest path from u to all nodes in graph

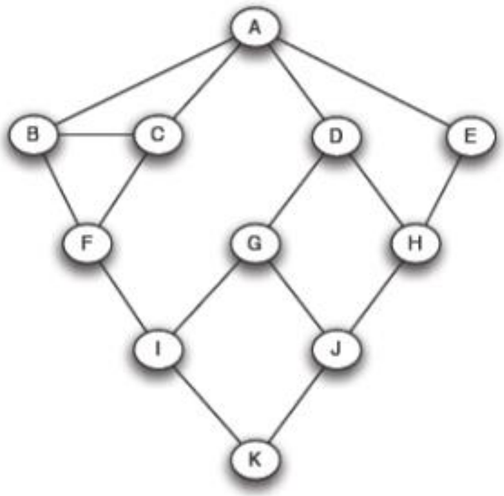
Comput betweeness

- BFS for every node
- Compute betweeness
- Devide by 2 (each shortest path is counted twice)

Example

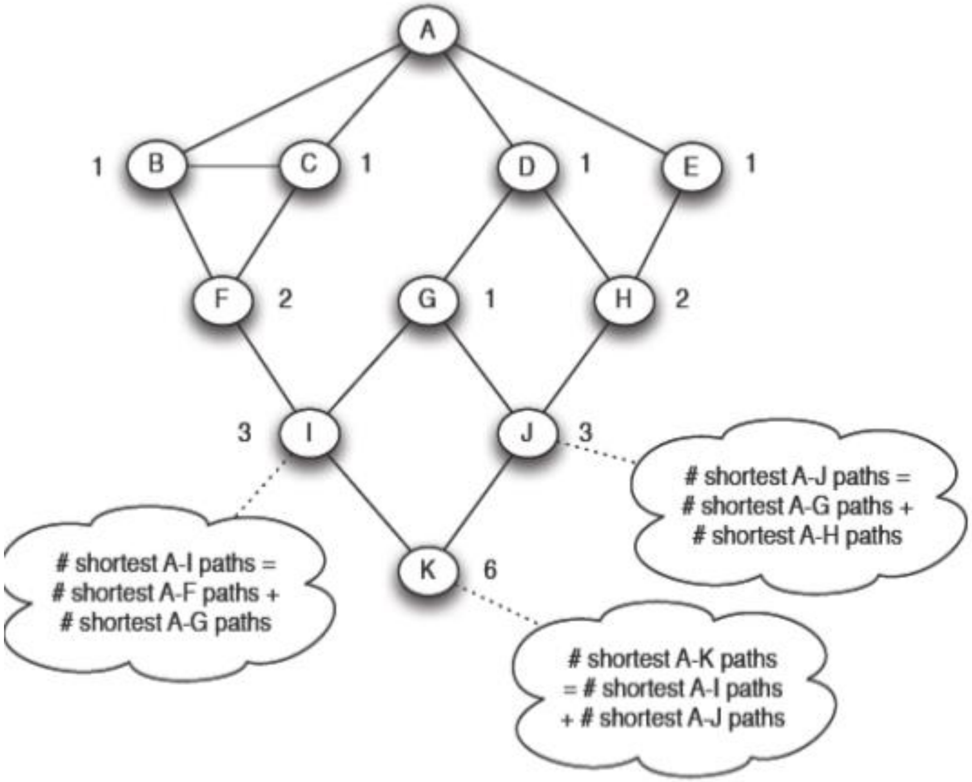
- Step 1:





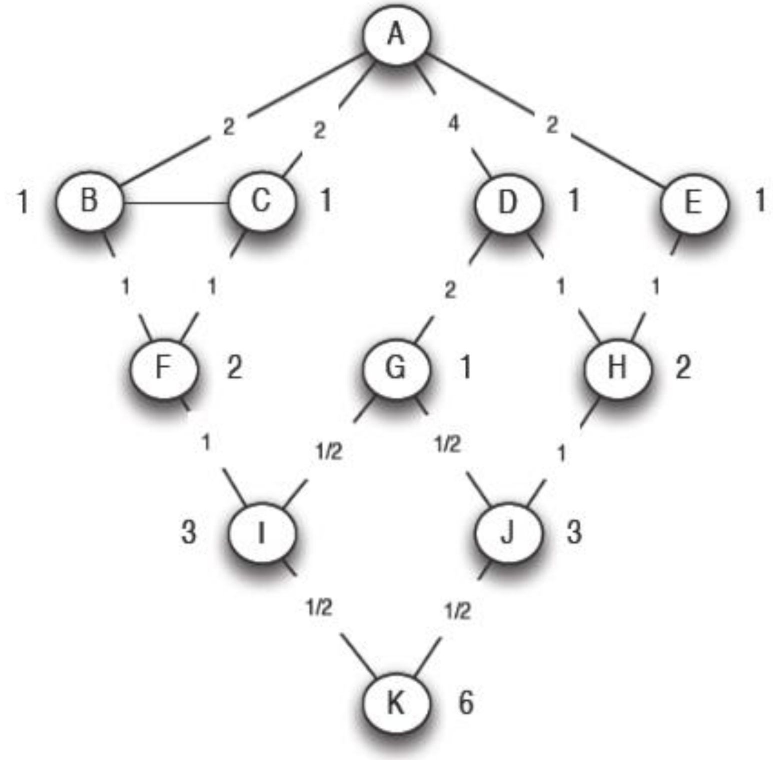
Example (cont.)

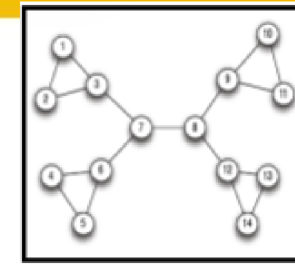
- Step 2:



Example (cont.)

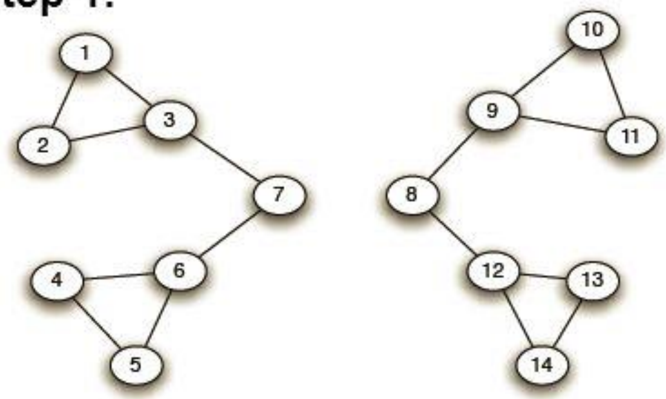
- Step 3:



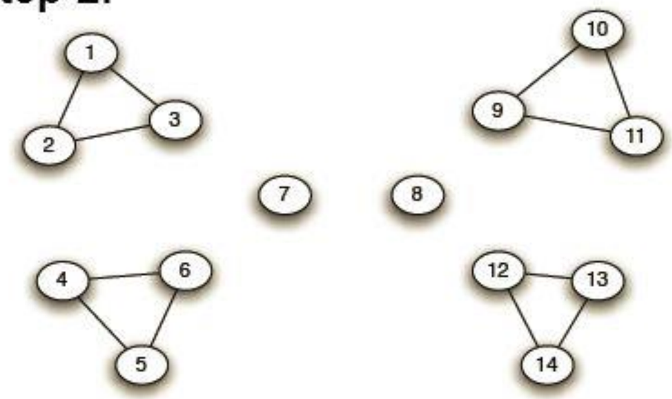


Girvan-Newman: Example

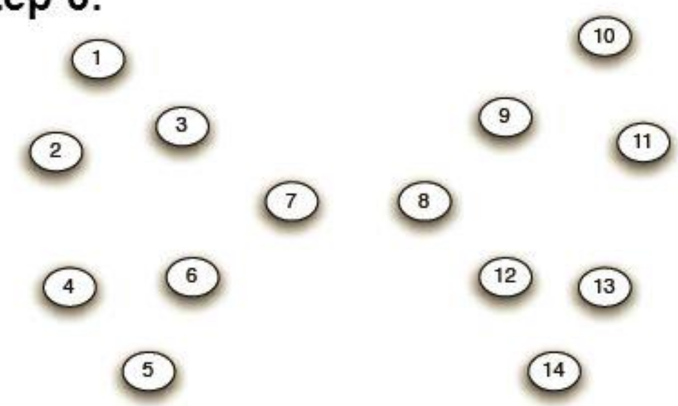
Step 1:



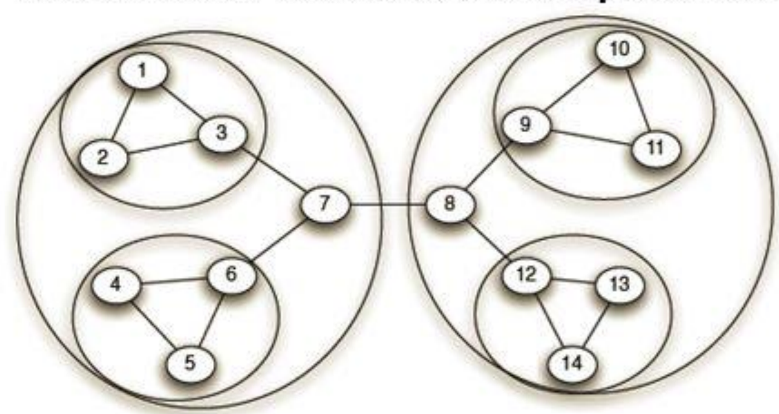
Step 2:



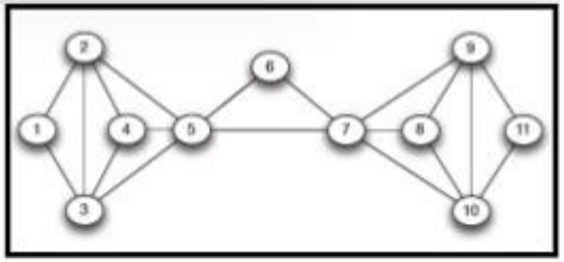
Step 3:

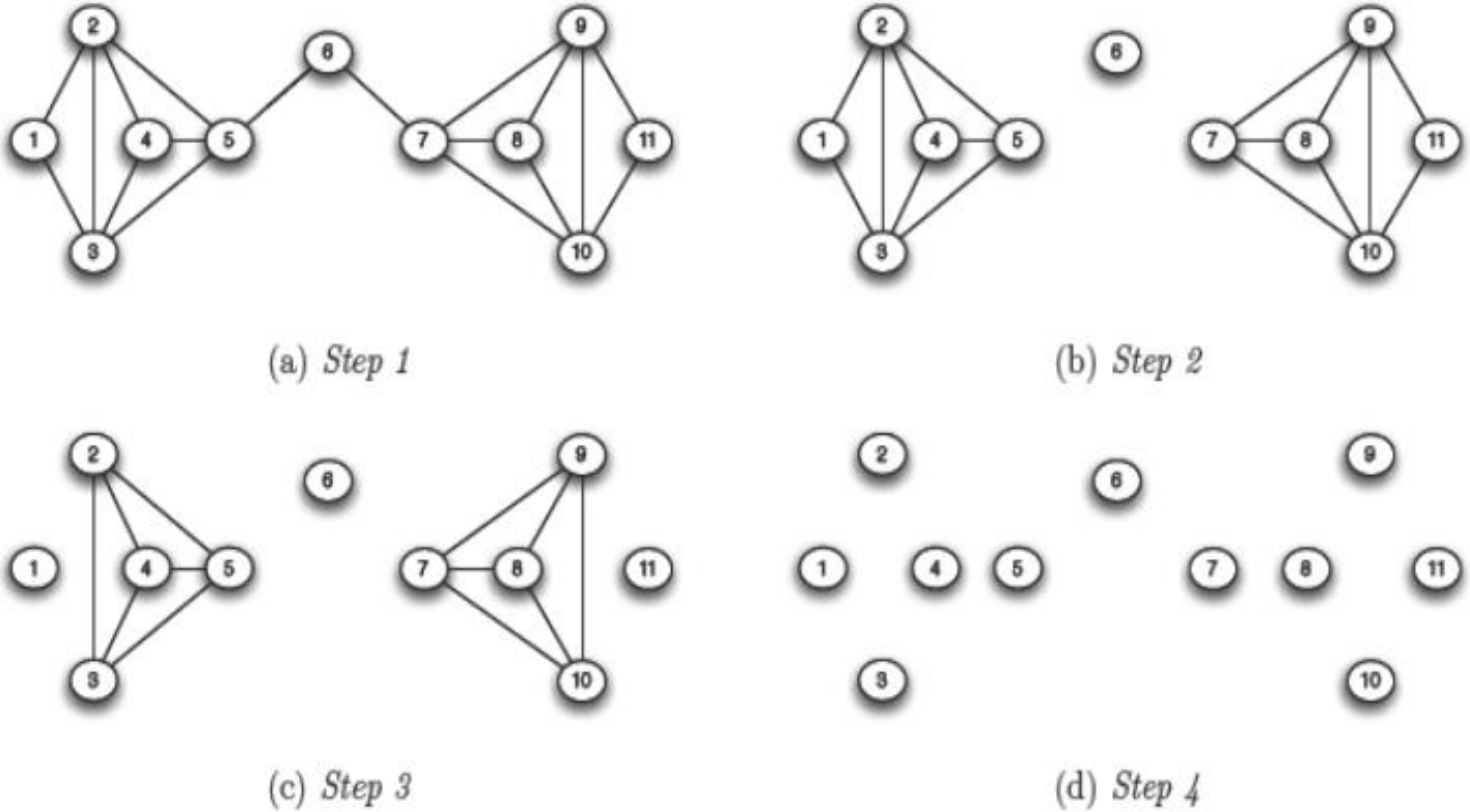


Hierarchical network decomposition:



Example 2

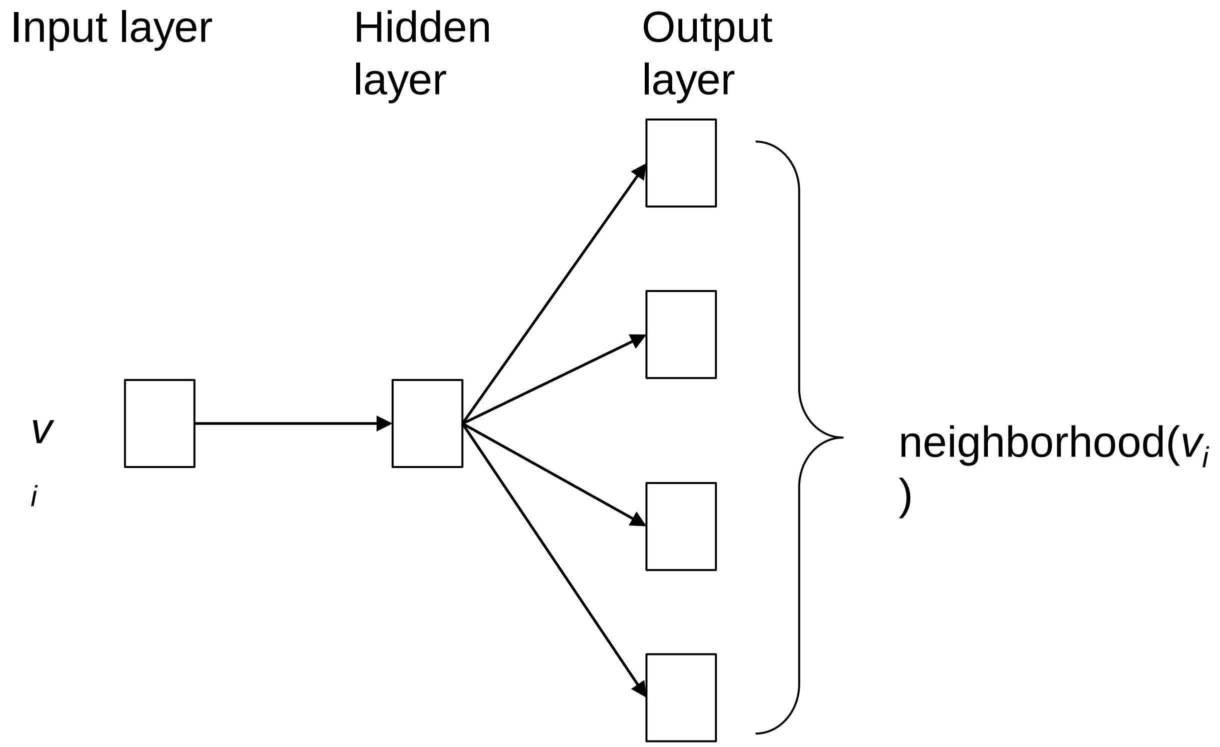




3. Graph Representation

- Adjacent matrix is sparse, high dimension
- Need representation of graph with low dimension
- Application in graph analysis

node2vec



SKIP-GRAM MODEL

Input layer

- One-hot encoding for nodes
  - 1 for current node, 0 for the other
  - V dimension, V is number of nodes

Hidden layer

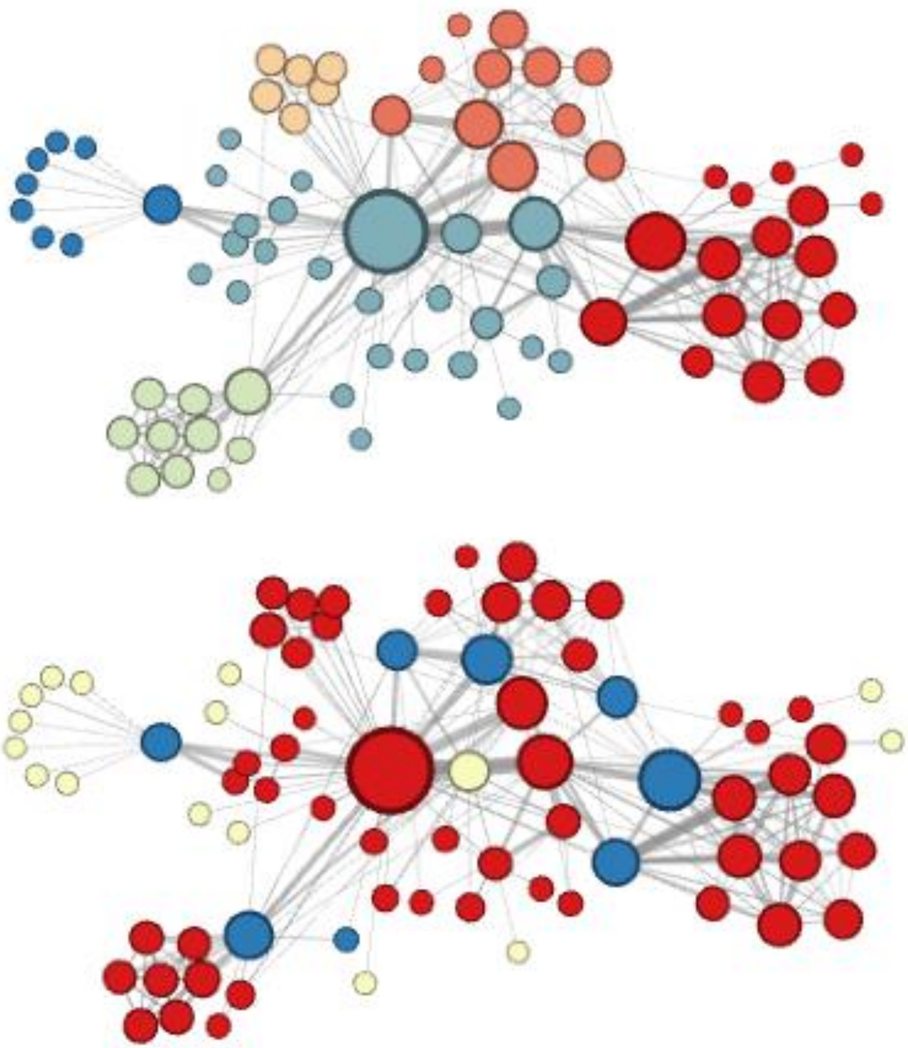
- *K dimension*
- Number of connection between input layer and hidden layer *V x K*
- Connection weight between input layer and hidden layer is used as representation for nodes

Output layer

- V dimension – number of nodes
- Skip-gram model use current nodes to predict adjacent nodes neighborhood(v_i)
- *Softmax* activation function
- *log-likelihood* loss function

Neighborhood(v_i)

- BFS:
  - Sample using adjacent nodes of v_i
  - Nodes in the same community have the same representation
- DFS:
  - Sample using nodes in DFS order
  - Nodes in the same roles have the same representation (leaves, central, bridge)
- Random walk: Balance between BFS and DFS
- Sample size k (k = 3)



from Aditya Grover and Jure Leskovec. "*node2vec: Scalable Feature Learning for Networks*". KDD2016

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

Thank you for your attentions!