Community Analysis in Social Networks

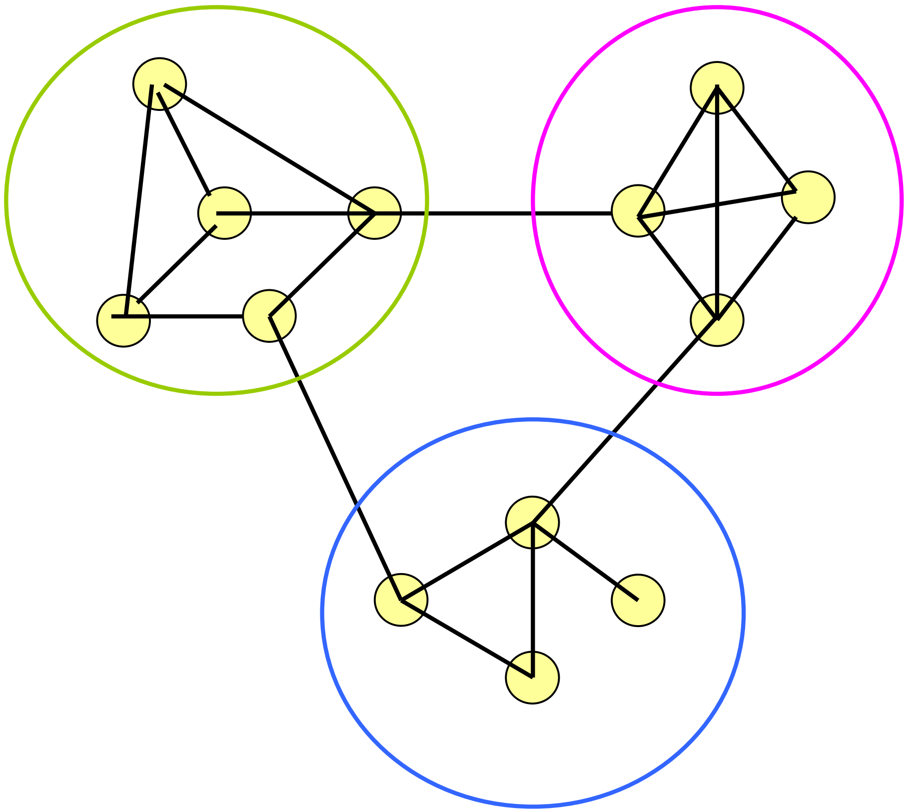
Outline

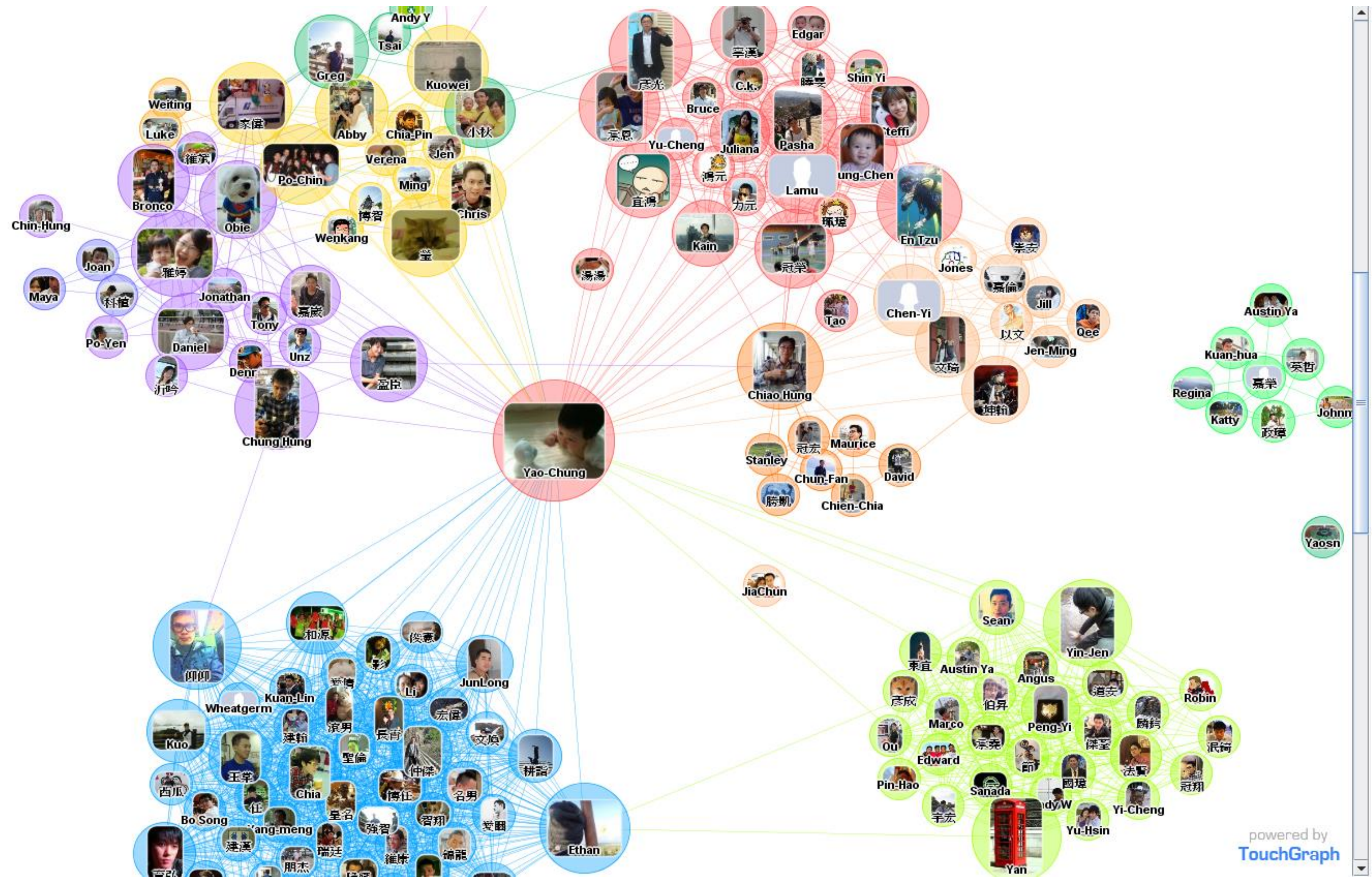
- What ?
- Why ?
- How ?
  - to obtain the structure
  - to evaluate the identified result

Community structure

- groups of vertices within which connections are dense but between which they are sparser.
  - a.k.a. group, community, cluster, module in different contexts
- Within-group(intra-group) edges.
  - High density
- Between-group(inter-group) edges.
  - Low density.

Community structure




Andy Y
Tsai
Greg
Kuowei
Weiting
Luke
Abby
Chia-Pin
Jen
Verena
Po-Chin
Ming
Chris
Bronco
Obie
Chin-Hung
Wenkang
Joan
Maya
Jonathan
Tony
Po-Yen
Daniel
Unz
Denr
Chung Hung
Edgar
Shin Yi
C.k.
Bruce
Yu-Cheng
Juliana
Pasha
Lamu
Kain
En Tzu
Jones
Jill
Qee
Chen-Yi
Jen-Ming
Tao
Chiao Hung
Maurice
Stanley
Chun-Fan
David
Chien-Chia
Yao-Chung
JiaChun
Austin Ya
Kuan-hua
Regina
Katty
Yaosn
Sean
Yin-Jen
Austin Ya
Angus
Robin
Marco
Peng-Yi
Ou
Edward
Pin-Hao
Sanada
Yi-Cheng
Yu-Hsin
Yan
JunLong
Li
Kuan-Lin
Wheatgerm
Kuo
Chia
Bo Song
Ethan
powered by
TouchGraph

Community

- Two types of groups in social media
  - Explicit Groups: formed by user **subscriptions**
  - Implicit Groups: implicitly formed by social interactions
- Network interaction provides rich information about the relationship between users
  - Can complement other kinds of information, e.g. **user profile**
  - Help network visualization and navigation
  - Provide basic information for other tasks, e.g. **recommendation**

The objective

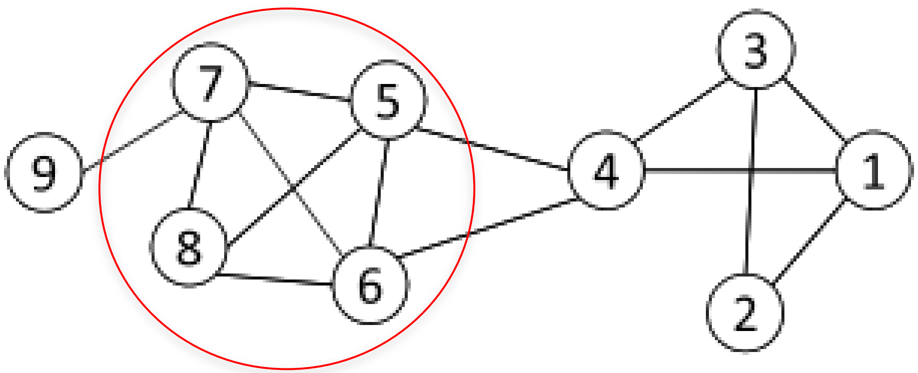
- Divide the network into non-empty groups (communities) in such a way that every vertex belongs to one of the communities.
- Many possible divisions could be done.
- We need a good division.
- Measurement of good division.

Community Detection Algorithms

- Graph partitioning Algorithm
  - The Clique Percolation Method
  - The Kernighan-Lin (KL) algorithm
- Structural Similarity Algorithm
- Edge Removal Algorithm
- Randomized Algorithm

Cliques

- Maximum Clique: a maximum complete subgraph in which all nodes are adjacent to each other



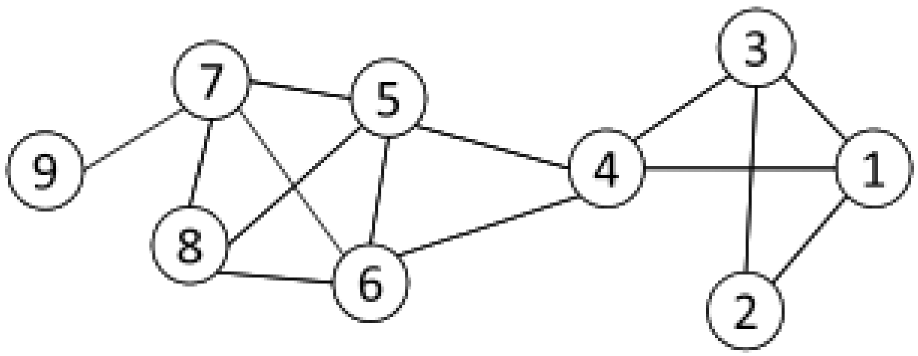
Nodes 5, 6, 7 and 8 form a clique

- Straightforward implementation to find cliques is very expensive in time complexity

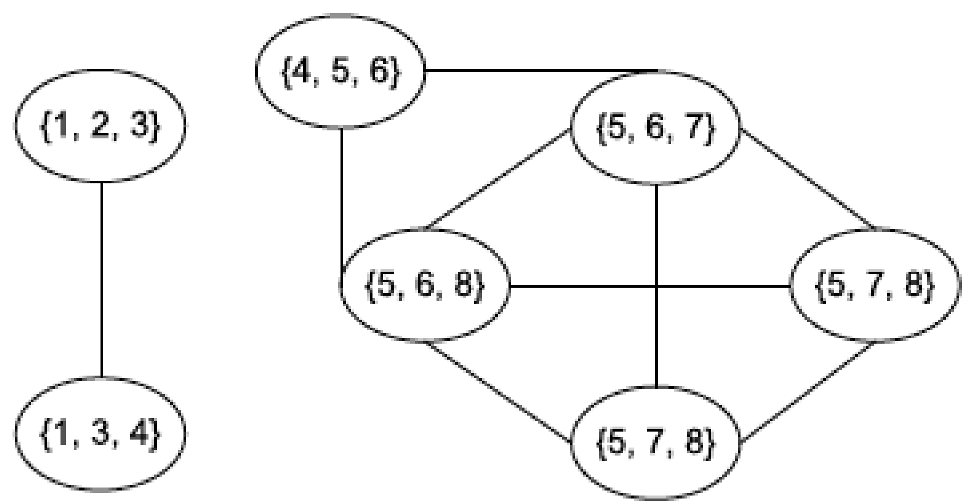
Clique Percolation Method (CPM)

- Clique is a **very strict** definition
- Normally use cliques as a core or a seed to find larger communities
- CPM is a method to find overlapping communities
  - **Input**
    - A parameter k, and a network
  - **Procedure**
    - Find out all **cliques of size k** in a given network
    - Construct a clique graph. Two cliques are adjacent if they share k-1 nodes
    - Each connected components in the clique graph form a community

CPM Example



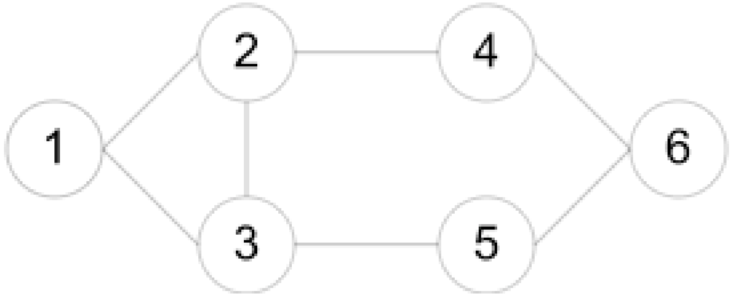
Cliques of size 3:
{1, 2, 3}, {1, 3, 4}, {4, 5, 6}, {5, 6, 7}, {5, 6, 8}, {5, 7, 8}, {6, 7, 8}



Communities:
{1, 2, 3, 4}
{4, 5, 6, 7, 8}

Many Variants of CPM method

- k-clique: a maximal subgraph in which the largest distance between any two nodes <= k in the original graph
- k-club: a substructure of diameter <= k



Cliques: {1, 2, 3}
2-cliques: {1, 2, 3, 4, 5}, {2, 3, 4, 5, 6}
2-clubs: {1,2,3,4}, {1, 2, 3, 5}, {2, 3, 4, 5, 6}

- Clique is a very strict definition
- A subgraph $G_s(V_s, E_s)$ is a $\gamma - dense$ quasi-clique if

$$\frac{2|E_s|}{|V_s|(|V_s|-1)} \geq \gamma$$

where the denominator is the maximum number of degrees.

Community Detection Algorithms

- Graph partitioning Algorithm
  - The Clique Percolation Method
  - The Kernighan-Lin (KL) algorithm
- Structural Similarity Algorithm
- Edge Removal Algorithm
- Randomized Algorithm

Community Detection Algorithms

- Graph partitioning Algorithm
  - The Clique Percolation Method
  - The Kernighan-Lin (KL) algorithm
- Structural Similarity Algorithm
- Edge Removal Algorithm
- Randomized Algorithm

KL ALGORITHM

Cut

- Most interactions are within group whereas interactions between groups are few
- community detection → minimum cut problem
- Cut: A partition of vertices of a graph into two disjoint sets
- Minimum cut problem: find a graph partition such that the number of edges between the two sets is minimized

How to find a cut ?

Kernighan-Lin (KL) algorithm

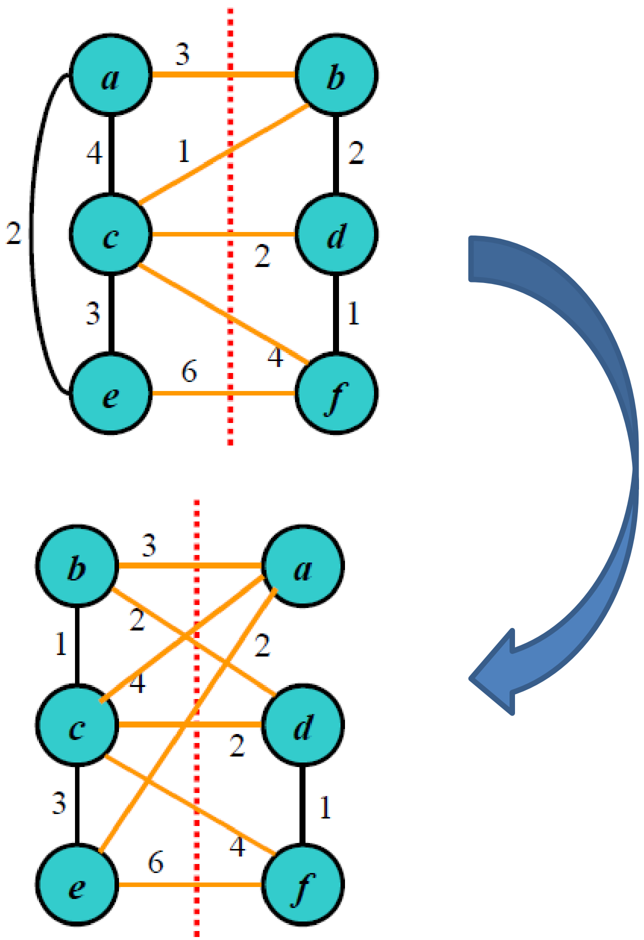
Kernighan-Lin (KL) algorithm

```
function Kernighan-Lin(G(V, E)) is
    determine a balanced initial partition of the nodes into sets A and B

    do
        compute D values for all a in A and b in B
        let gv, av, and bv be empty lists
        for n := 1 to |V| / 2 do
            find a from A and b from B, such that g = D[a] + D[b] − 2×c(a, b) is maximal
            remove a and b from further consideration in this pass
            add g to gv, a to av, and b to bv
            update D values for the elements of A = A \ a and B = B \ b
        end for
        find k which maximizes g_max, the sum of gv[1], ..., gv[k]
        if g_max > 0 then
            Exchange av[1], av[2], ..., av[k] with bv[1], bv[2], ..., bv[k]
    until (g_max ≤ 0)

    return G(V, E)
```

Kernighan-Lin (KL) algorithm



Swap a and b

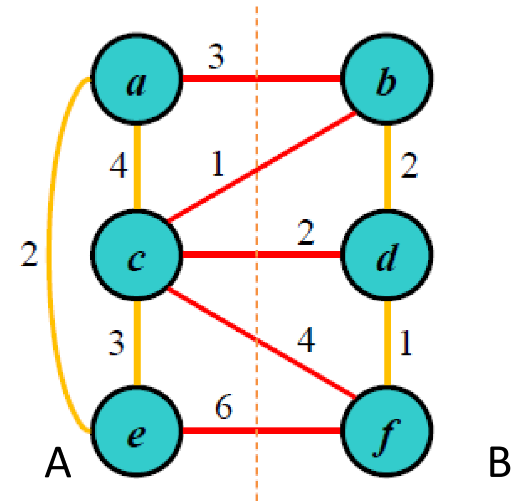
Better or Not?

Kernighan-Lin (KL) algorithm

Let us define for each $a \in A$,

– An **external cost** $E_a = \sum_{y \in B} w_{ay}$

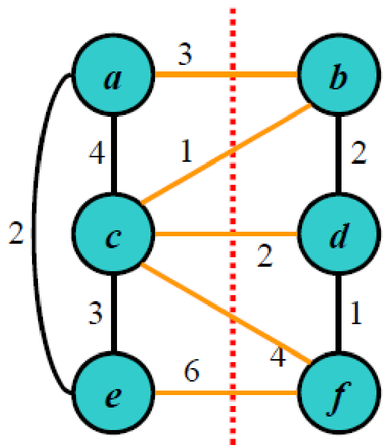
– An **internal cost** $I_a = \sum_{x \in A} w_{ax}$



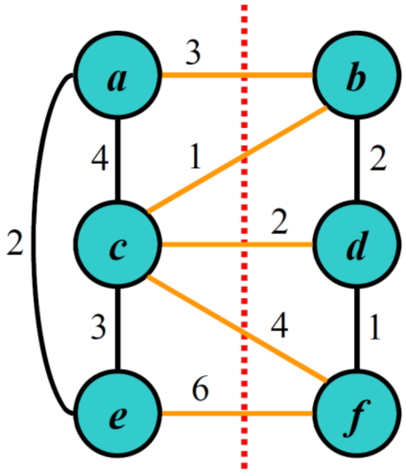
- Let $D_z = E_z - I_z$ for a node z;
- D_z is the difference between its external and internal costs for node z

$$D_a = E_a - I_a = -3 \quad (= 3 - 4 - 2)$$
$$D_c = E_c - I_c = \quad 0 \quad (= 1 + 2 + 4 - 4 - 3)$$
$$D_e = E_e - I_e = +1 \quad (= 6 - 2 - 3)$$

Kernighan-Lin (KL) algorithm



Kernighan-Lin (KL) algorithm



cut-size = 16

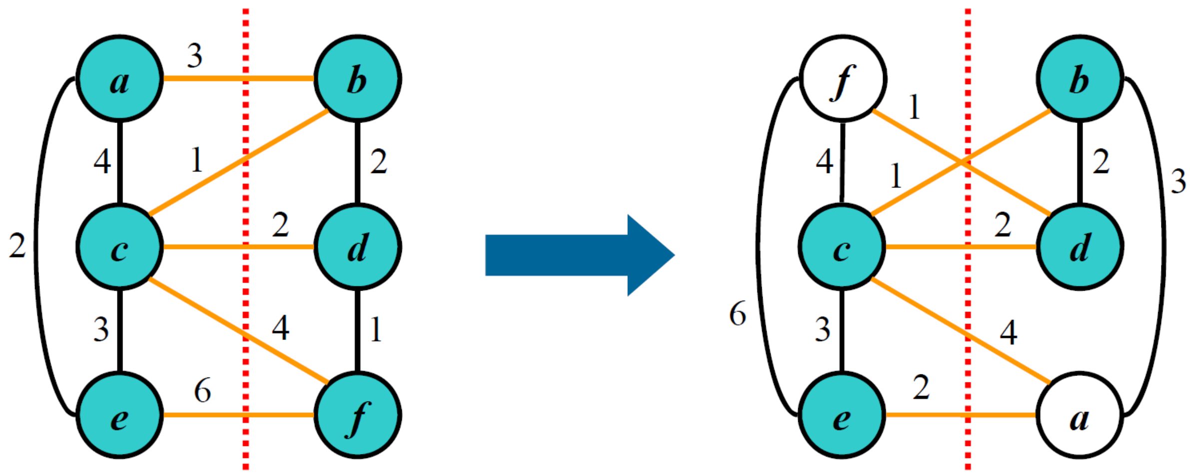
Compute the gains

$D_a = -3 \quad D_b = +2$
$D_c = \ \ 0 \quad D_d = -1$
$D_e = +1 \quad D_f = +9$

$$g_{ab} = D_a + D_b - 2w_{ab} = -3 + 2 - 2\cdot 3 = -7$$
$$g_{ad} = D_a + D_d - 2w_{ad} = -3 - 1 - 2\cdot 0 = -4$$
$$\boxed{g_{af} = D_a + D_f - 2w_{af} = -3 + 9 - 2\cdot 0 = +6}$$
$$g_{cb} = D_c + D_b - 2w_{cb} = 0 + 2 - 2\cdot 1 = 0$$
$$g_{cd} = D_c + D_d - 2w_{cd} = 0 - 1 - 2\cdot 2 = -5$$
$$g_{cf} = D_c + D_f - 2w_{cf} = 0 + 9 - 2\cdot 4 = +1$$
$$g_{eb} = D_e + D_b - 2w_{eb} = +1 + 2 - 2\cdot 0 = +1$$
$$g_{ed} = D_e + D_d - 2w_{ed} = +1 - 1 - 2\cdot 0 = 0$$
$$g_{ef} = D_e + D_f - 2w_{ef} = +1 + 9 - 2\cdot 6 = -2$$

Pair with maximum gain

Kernighan-Lin (KL) algorithm



cut-size = 16

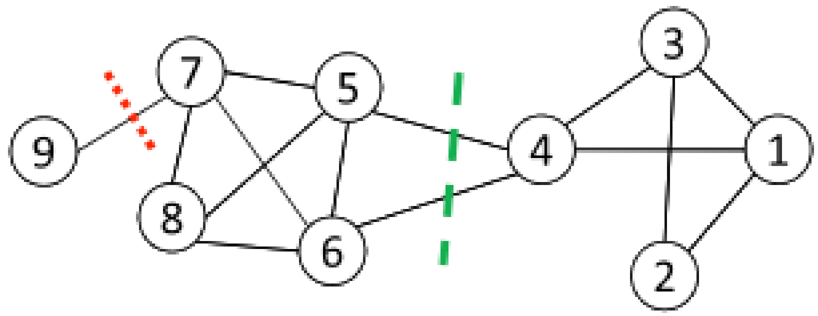
cut-size = 16 – 6 = 10

Exchange nodes *a* and *f*

Then lock up nodes *a* and *f*

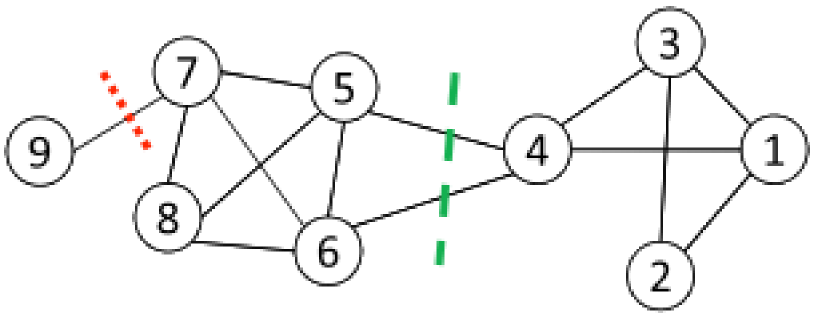
$$g_{af} = D_a + D_f - 2w_{af} = -3 + 9 - 2\cdot 0 = +6$$

Min Cut ?



- Minimum cut often returns an <u>imbalanced</u> partition, with one set being a singleton, e.g. node 9

Ratio Cut & Normalized Cut



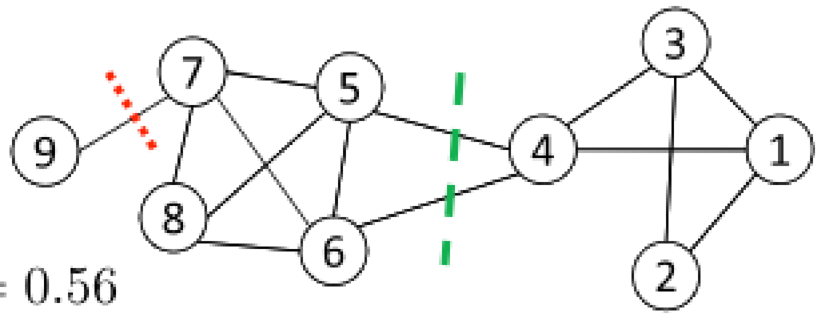
- Change the objective function to consider community size

$$\text{Ratio Cut}(\pi) = \frac{1}{k}\sum_{i=1}^{k}\frac{cut(C_i, \bar{C}_i)}{|C_i|},$$

$$\text{Normalized Cut}(\pi) = \frac{1}{k}\sum_{i=1}^{k}\frac{cut(C_i, \bar{C}_i)}{vol(C_i)}$$

C_i: a community
$|C_i|$: number of nodes in C_i
$vol(C_i)$: sum of degrees in C_i

Ratio Cut & Normalized Cut Example



For partition in red: π_1

$$\text{Ratio Cut}(\pi_1) = \frac{1}{2}\left(\frac{1}{1} + \frac{1}{8}\right) = 9/16 = 0.56$$

$$\text{Normalized Cut}(\pi_1) = \frac{1}{2}\left(\frac{1}{1} + \frac{1}{27}\right) = 14/27 = 0.52$$

For partition in green: π_2

$$\text{Ratio Cut}(\pi_2) = \frac{1}{2}\left(\frac{2}{4} + \frac{2}{5}\right) = 9/20 = 0.45 < \text{Ratio Cut}(\pi_1)$$

$$\text{Normalized Cut}(\pi_2) = \frac{1}{2}\left(\frac{2}{12} + \frac{2}{16}\right) = 7/48 = 0.15 < \text{Normalized Cut}(\pi_1)$$

Both ratio cut and normalized cut prefer a balanced partition

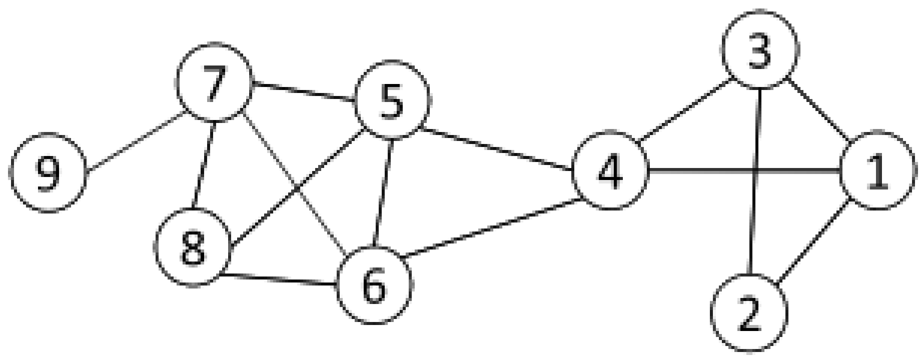
Community Detection Algorithms

- Graph partitioning Algorithm
  - The Clique Percolation Method
  - The Kernighan-Lin (KL) algorithm
- Structural Similarity Algorithm
- Edge Removal Algorithm
- Randomized Algorithm

Clustering based on **Vertex Similarity**

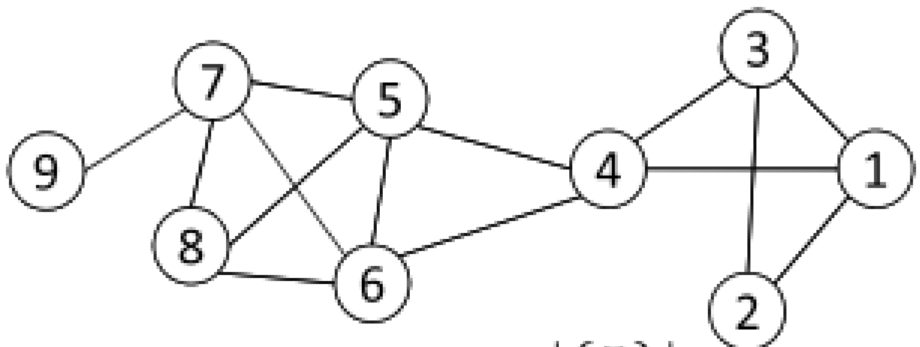
- Apply k-means or similarity-based clustering to nodes
- Vertex similarity is defined in terms of the similarity of their neighborhood
- Structural equivalence: two nodes are structurally equivalent iff they are connecting to the same set of actors

Nodes 1 and 3 are structurally equivalent; So are nodes 5 and 6.



Vertex Similarity

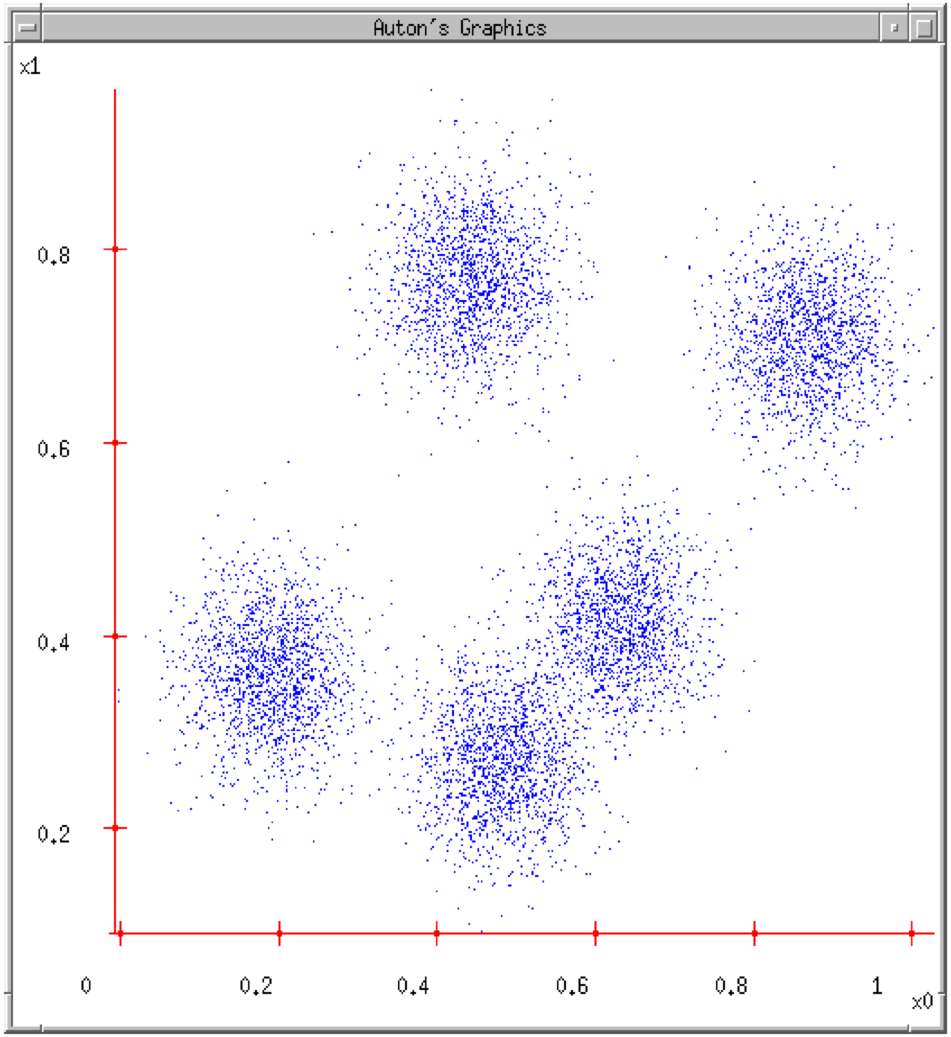
- Jaccard Similarity $Jaccard(v_i, v_j) = \frac{|N_i \cap N_j|}{|N_i \cup N_j|}$
- Cosine similarity $Cosine(v_i, v_j) = \frac{|N_i \cap N_j|}{\sqrt{|N_i| \cdot |N_j|}}$



$$Jaccard(4, 6) = \frac{|\{5\}|}{|\{1, 3, 4, 5, 6, 7, 8\}|} = \frac{1}{7}$$

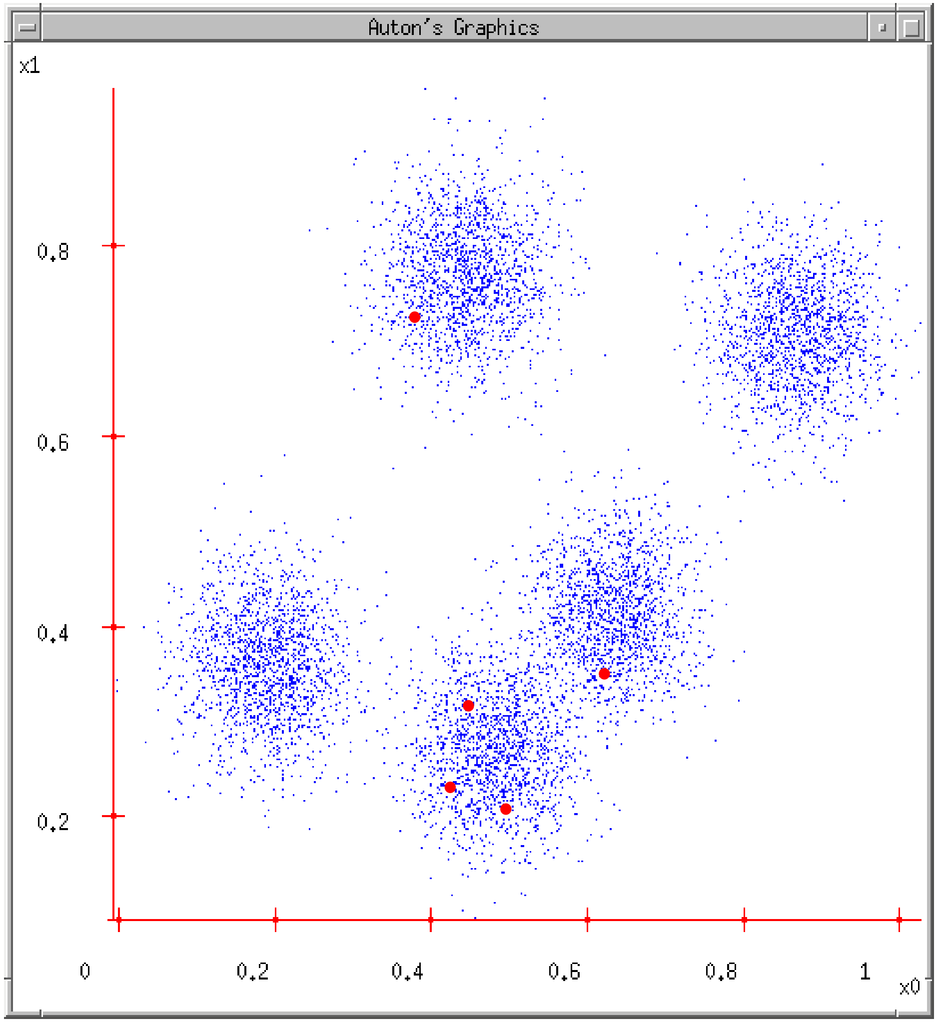
$$cosine(4, 6) = \frac{1}{\sqrt{4 \cdot 4}} = \frac{1}{4}$$

K-means illustration



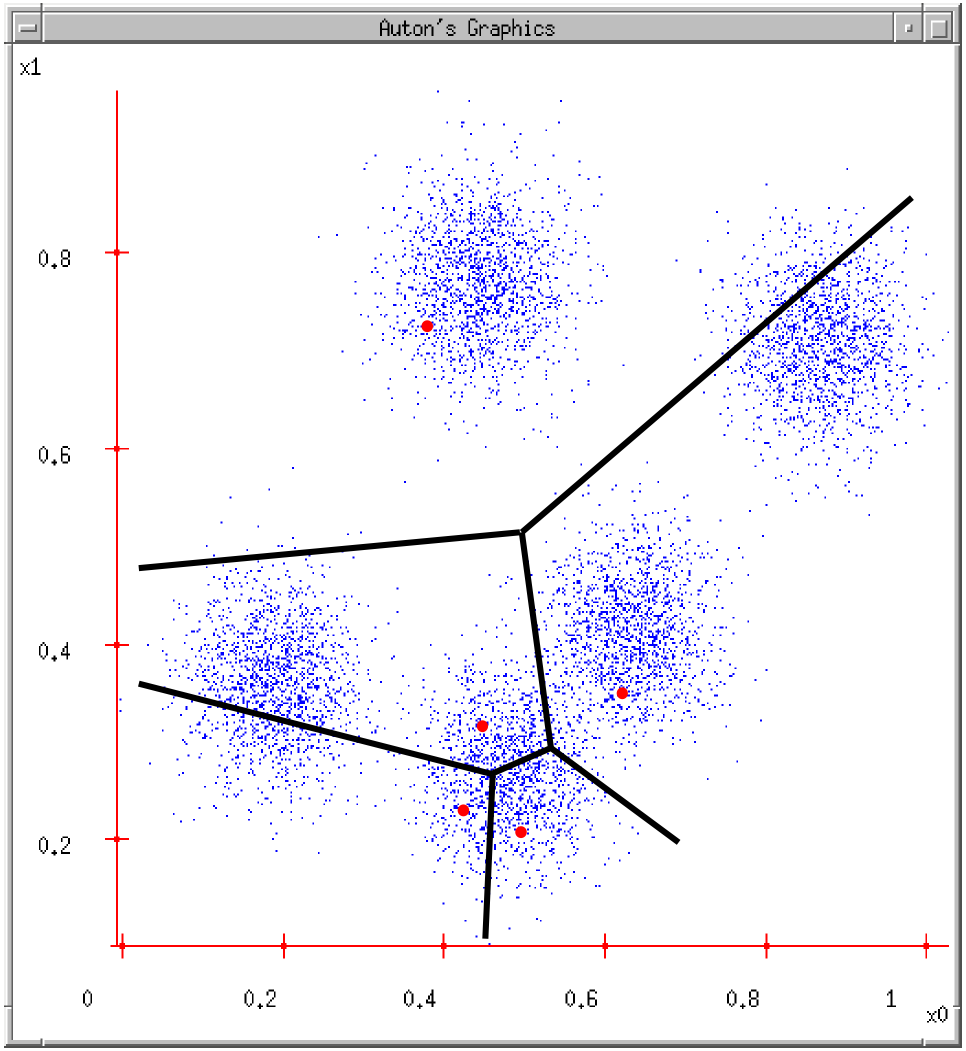
1. User set up the number of clusters they'd like. *(e.g. k=5)*

K-means illustration



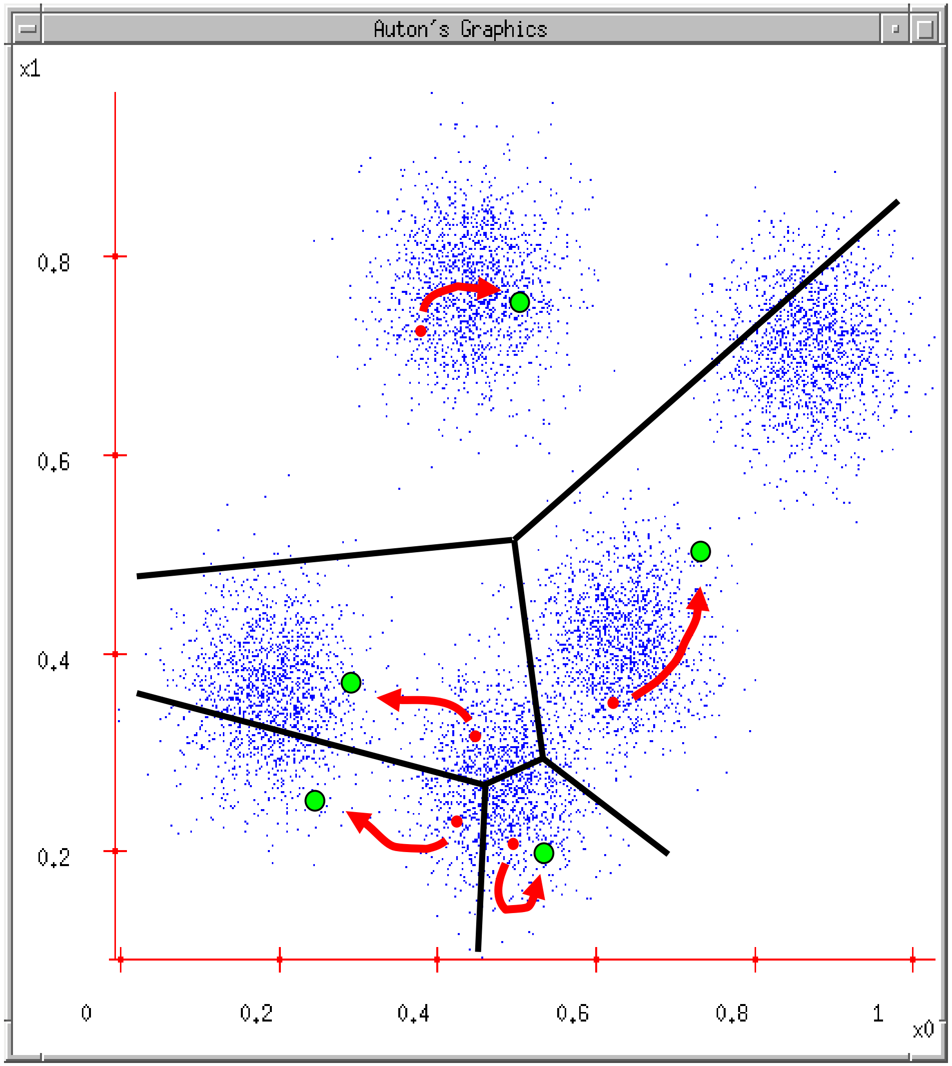
1. User set up the number of clusters they'd like. *(e.g. K=5)*
2. Randomly guess K cluster Center locations

K-means illustration



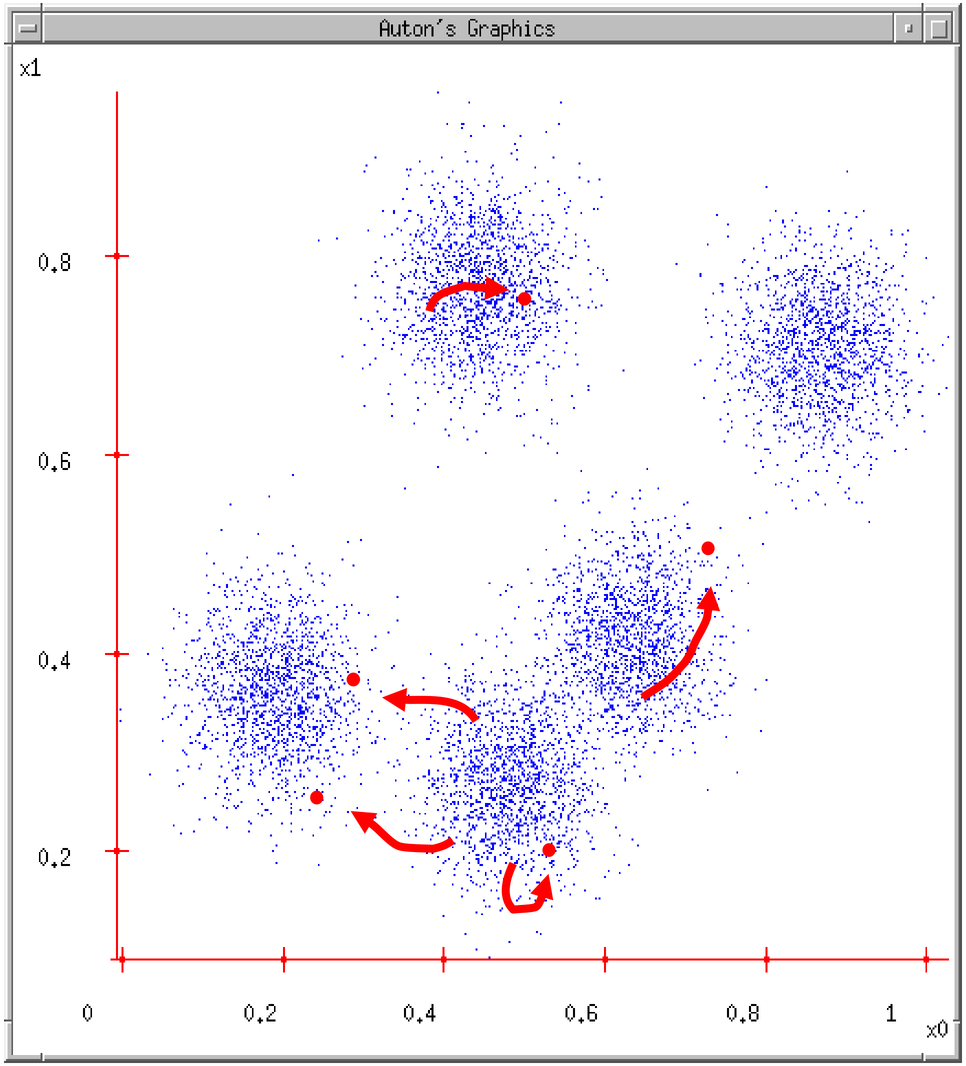
1. User set up the number of clusters they'd like. *(e.g. K=5)*
2. Randomly guess *K* cluster Center locations
3. Each data point finds out which Center it's closest to. (Thus each Center "owns" a set of data points)

K-means illustration



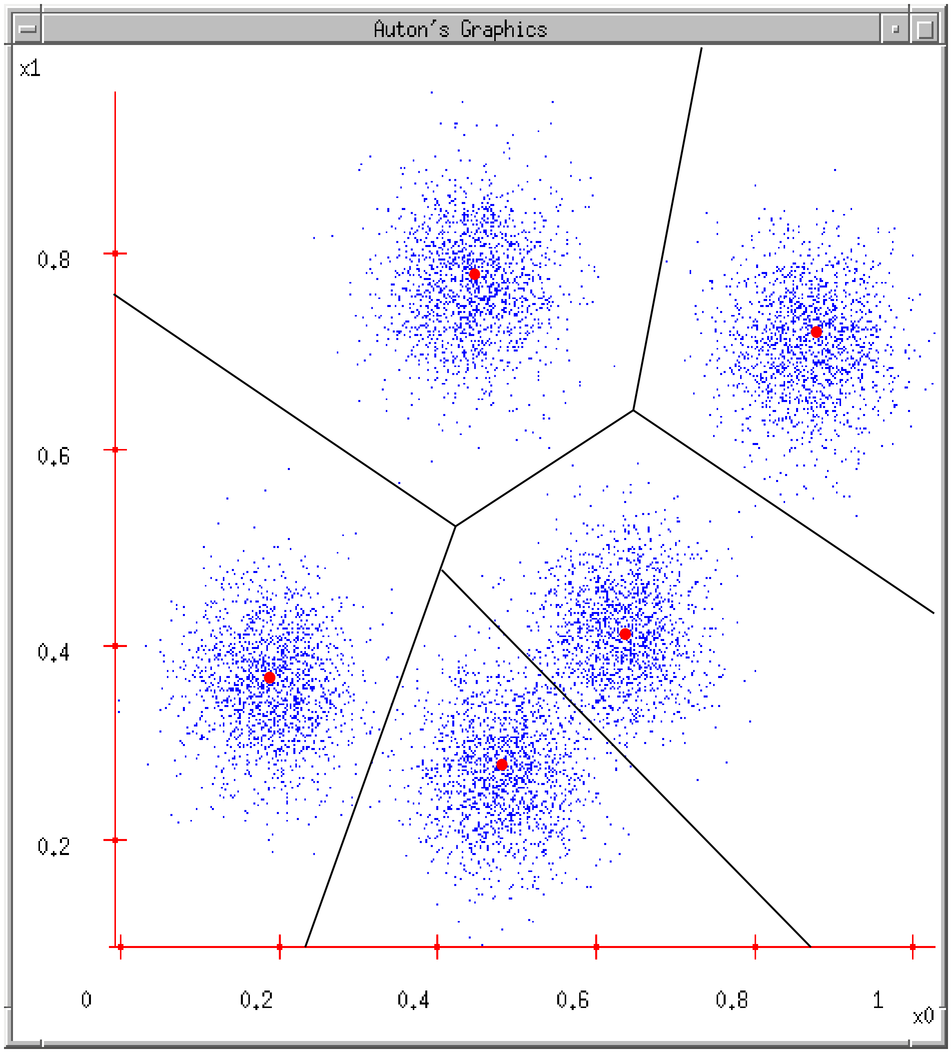
1. User set up the number of clusters they'd like. *(e.g. K=5)*
2. Randomly guess *K* cluster centre locations
3. Each data point finds out which centre it's closest to. (Thus each Center "owns" a set of data points)
4. Each centre finds the centroid of the points it owns

K-means illustration



1. User set up the number of clusters they'd like. (e.g. *K=5*)
2. Randomly guess *K* cluster centre locations
3. Each data point finds out which centre it's closest to. (Thus each centre "owns" a set of data points)
4. Each centre finds the centroid of the points it owns
5. …and jumps there

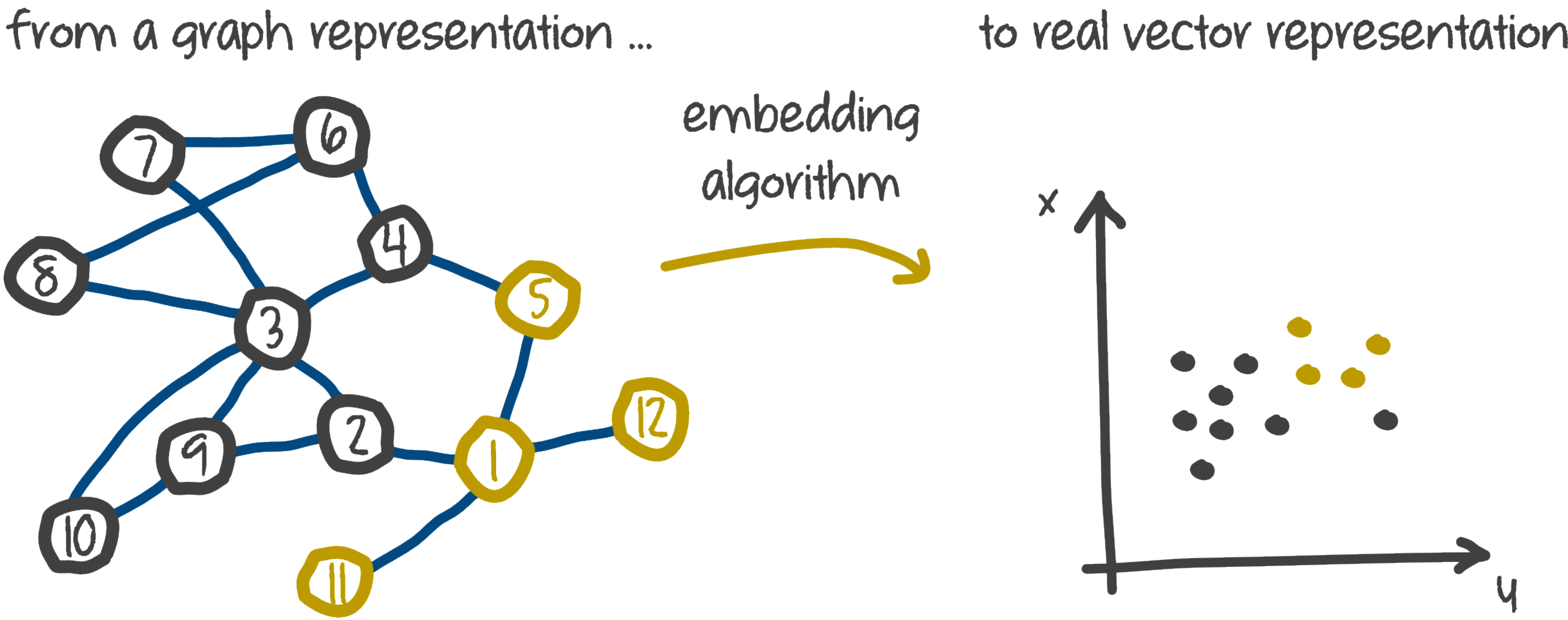
K-means illustration



1. User set up the number of clusters they'd like. (e.g. *K=5*)
2. Randomly guess *K* cluster centre locations
3. Each data point finds out which centre it's closest to. (Thus each centre "owns" a set of data points)
4. Each centre finds the centroid of the points it owns
5. …and jumps there
6. …Repeat until terminated!

Graph/Vertex Embedding

Vertex Embedding



Idea behind the Neural Embedding Representation

f(word) = 

Encode a word into a vector representation

- A possible way is one-hot-encoding form
  - That is, we represent the word w by

$$\overbrace{[0,0,\ldots,1,\ldots,0,0]}^{|V| \text{ elements}}$$

  - where the 1 is in a location unique to w. Any other word will have a 1 in some other location, and a 0 everywhere else.

Main Problem (1/2)

- The mathematician ran to the coffee store.
- The physicist ran to the coffee store.
- The mathematician and physicist ran into the open problem.

Let’s think one-hot-form of ”mathematician” and “physicist” ?

Physicist = [0, 0, 0, 0, 0, 1]
mathematician = [0, 0, 0, 0, 1, 0]
ran=[0, 0, 0, 1, 0, 0]
store=[0, 0, 1, 0, 0, 0]
coffee=[0, 1, 0, 0, 0, 0]
problem=[1, 0, 0, 0, 0, 0]

Semantic Similarity ?

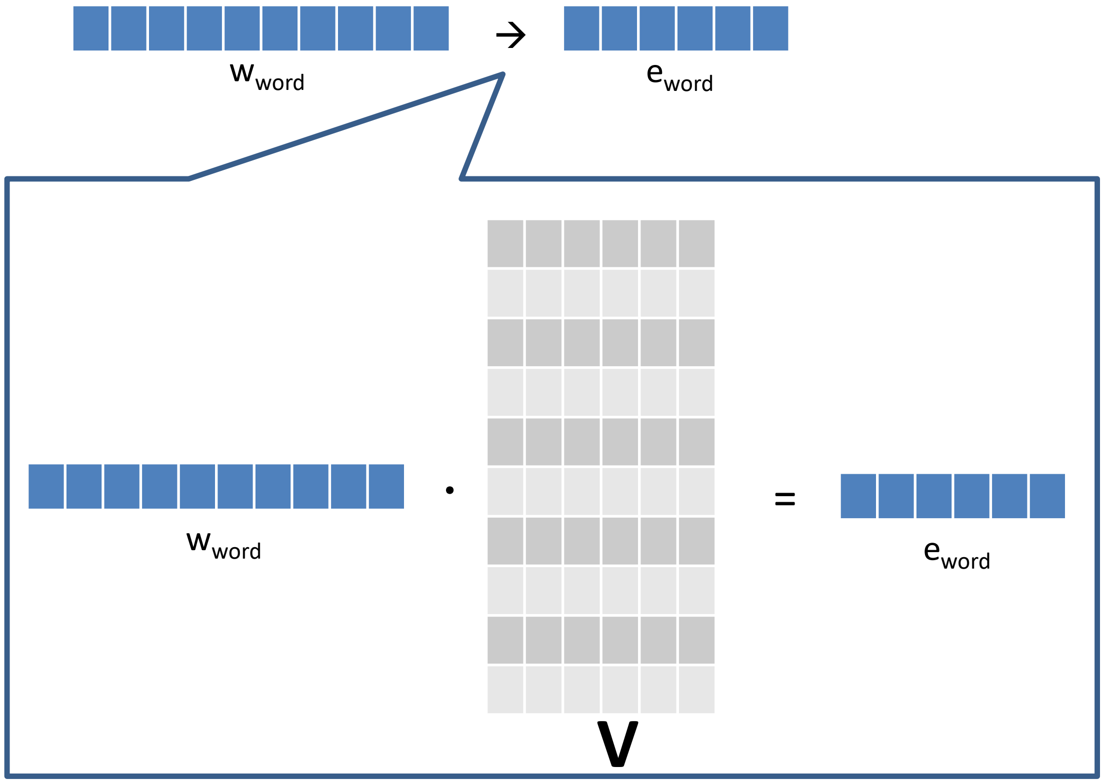
Main Problem (2/2)

- How can we solve this problem? That is, how could we actually encode semantic similarity in words? Maybe we think up some semantic attributes.
- For example, we see that both mathematicians and physicists can run, so maybe we give these words a high score for the "is able to run" semantic attribute
- If each attribute is a dimension, then we might give each word a vector, like this:

$$q_{\text{mathematician}} = \left[\overbrace{2.3}^{\text{can run}}, \overbrace{9.4}^{\text{likes coffee}}, \overbrace{-5.5}^{\text{majored in Physics}}, \dots \right]$$

$$q_{\text{physicist}} = \left[\overbrace{2.5}^{\text{can run}}, \overbrace{9.1}^{\text{likes coffee}}, \overbrace{6.4}^{\text{majored in Physics}}, \dots \right]$$

$$\text{Similarity}(\text{physicist}, \text{mathematician}) = \frac{q_{\text{physicist}} \cdot q_{\text{mathematician}}}{\|q_{\text{physicist}}\| \|q_{\text{mathematician}}\|} = \cos(\phi)$$

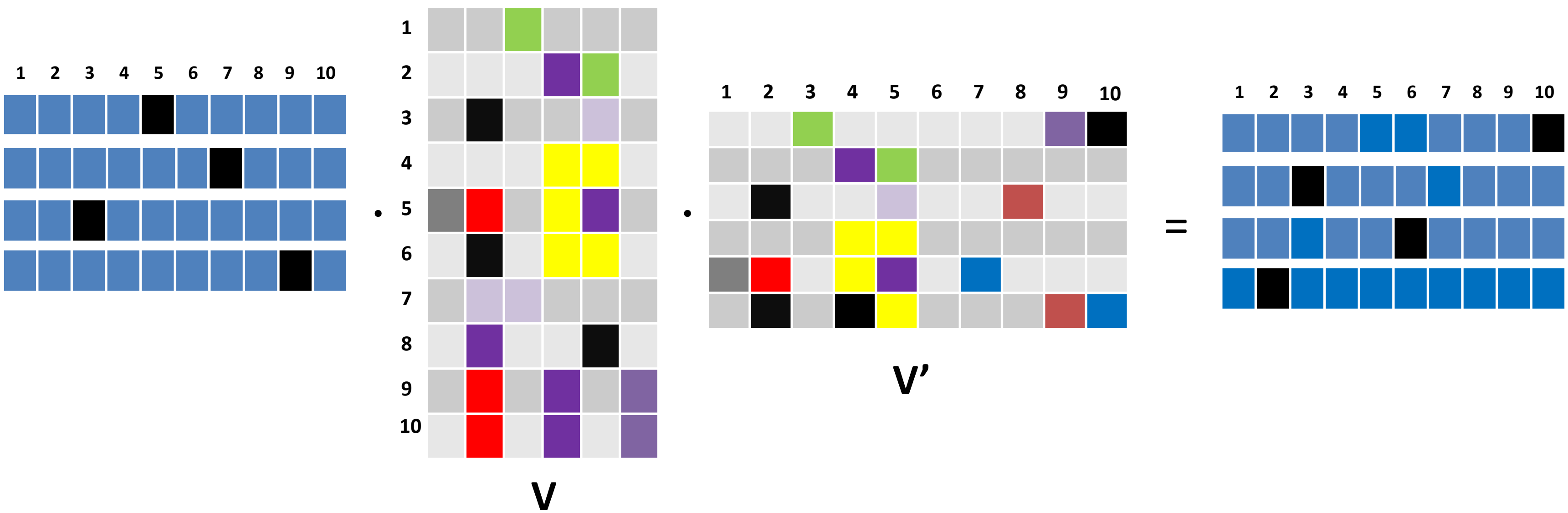


A Matrix can help !

$$\mathbf{w_{word}} \cdot \mathbf{V} = \mathbf{e_{word}}$$

Q: How to get **V** ?

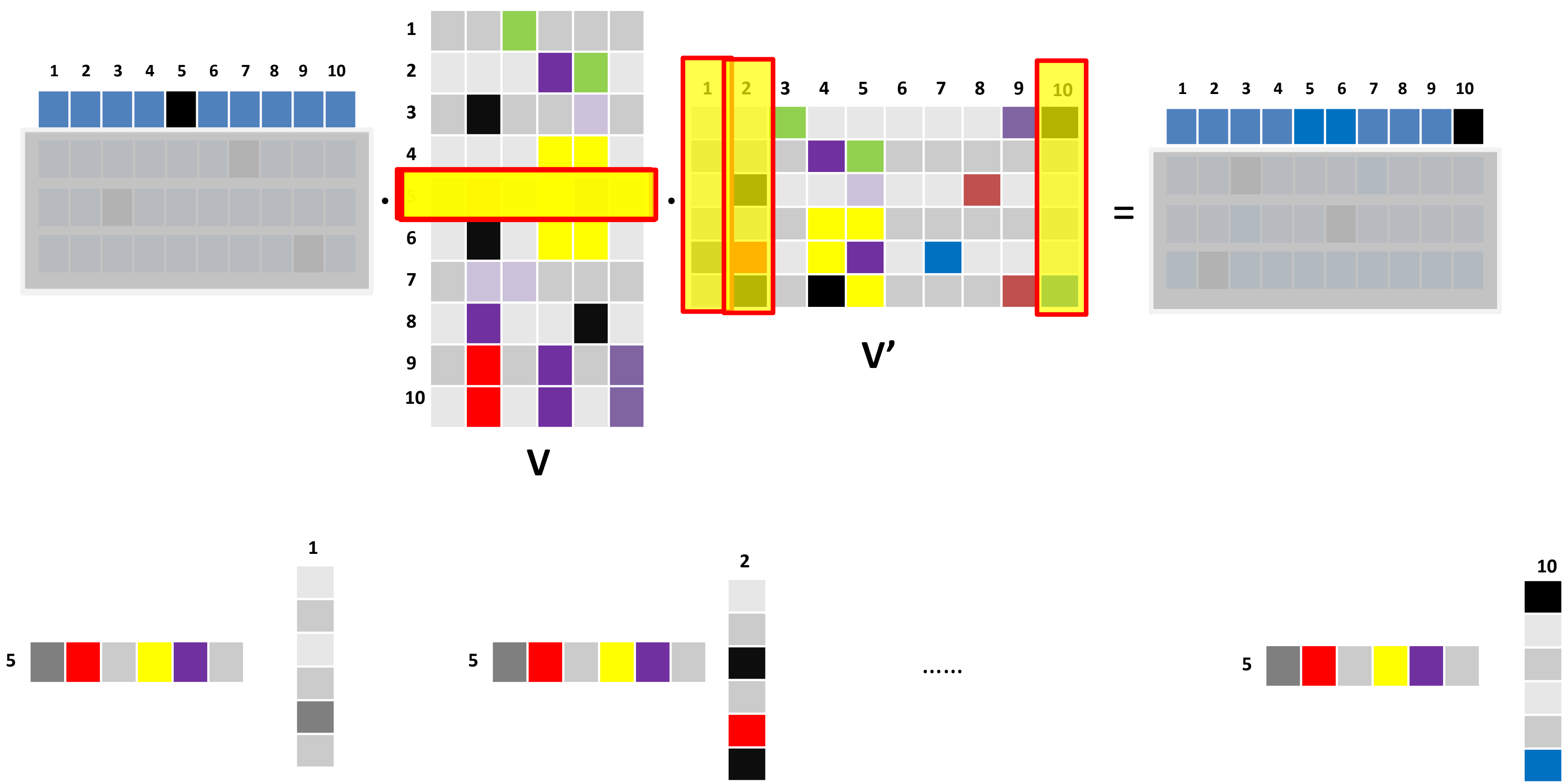
A: Learning from Data



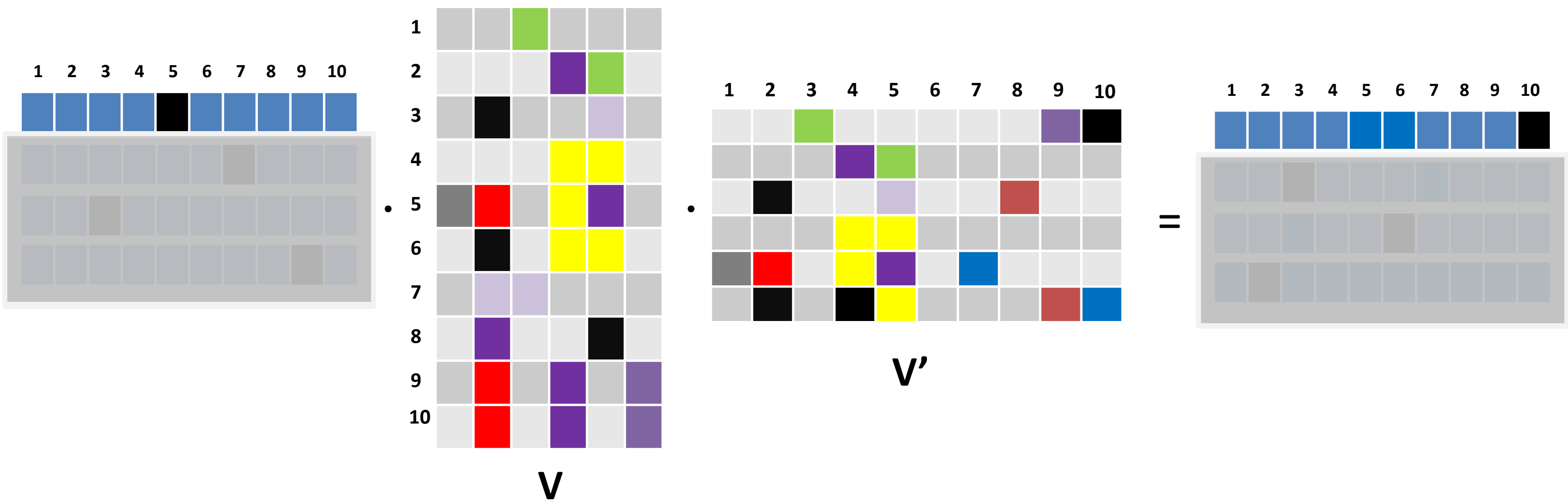
Objective Function

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\le j\le c, j\ne 0}\log p(w_{t+j}|w_t)$$

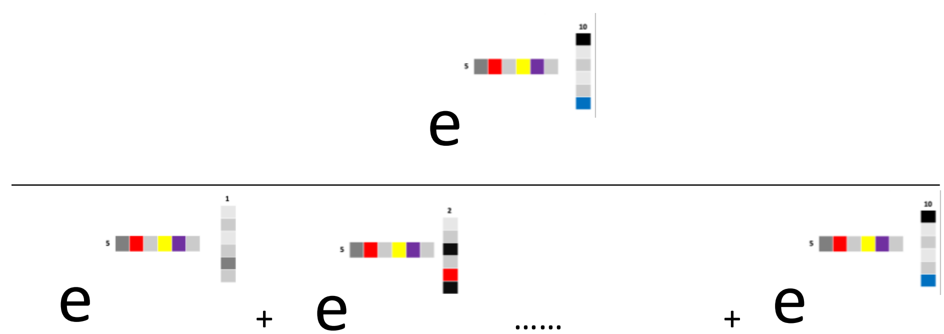
$$p(w_O|w_I)=\frac{\exp\left({v'_{w_O}}^{\top} v_{w_I}\right)}{\sum_{w=1}^{W}\exp\left({v'_{w}}^{\top} v_{w_I}\right)}$$



1 2 3 4 5 6 7 8 9 10
V
V'
=
1
2
10
5
......



$$p(w_O|w_I) = \frac{\exp\left({v'_{w_O}}^\top v_{w_I}\right)}{\sum_{w=1}^{W} \exp\left({v'_w}^\top v_{w_I}\right)} =$$



DeepWalk: Online Learning of Social Representations, KDD 2014

GRAPH EMBEDDING

Algorithm 1 DeepWalk(G, w, d, γ, t)

Input: graph $G(V, E)$
window size w
embedding size d
walks per vertex γ
walk length t
Output: matrix of vertex representations $\Phi \in \mathbb{R}^{|V| \times d}$

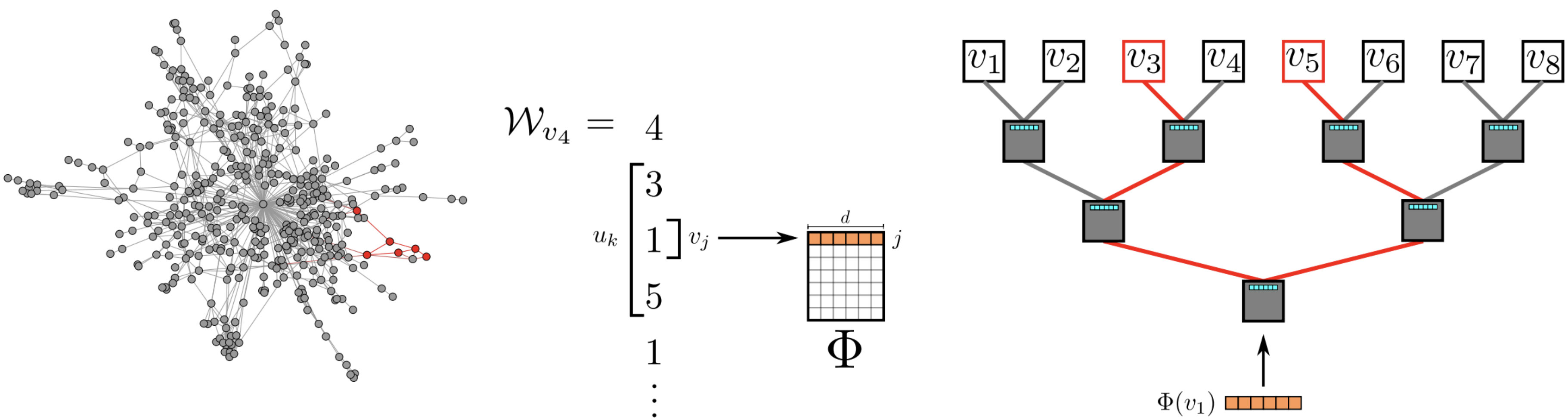
Initialization: Sample Φ from $\mathcal{U}^{|V| \times d}$
Build a binary Tree T from V
for $i = 0$ to γ **do**
    $\mathcal{O} = \text{Shuffle}(V)$
    for each $v_i \in \mathcal{O}$ **do**
        $\mathcal{W}_{v_i} = RandomWalk(G, v_i, \text{t})$
        SkipGram(Φ, $\mathcal{W}_{v_i}$, w)
    end for
end for

Algorithm 2 SkipGram(Φ, $\mathcal{W}_{v_i}$, w)

for each $v_j \in \mathcal{W}_{v_i}$ **do**
    for each $u_k \in \mathcal{W}_{v_i}[j - w : j + w]$ **do**
        $J(\Phi) = -\log \Pr(u_k \mid \Phi(v_j))$
        $\Phi = \Phi - \alpha * \frac{\partial J}{\partial \Phi}$
    end for
end for

$$\Pr(u_k \mid \Phi(v_j)) = \prod_{l=1}^{\lceil \log |V| \rceil} \Pr(b_l \mid \Phi(v_j))$$

$$\Pr(b_l \mid \Phi(v_j) = 1/(1 + e^{-\Phi(v_j) \cdot \Psi(b_l)})$$

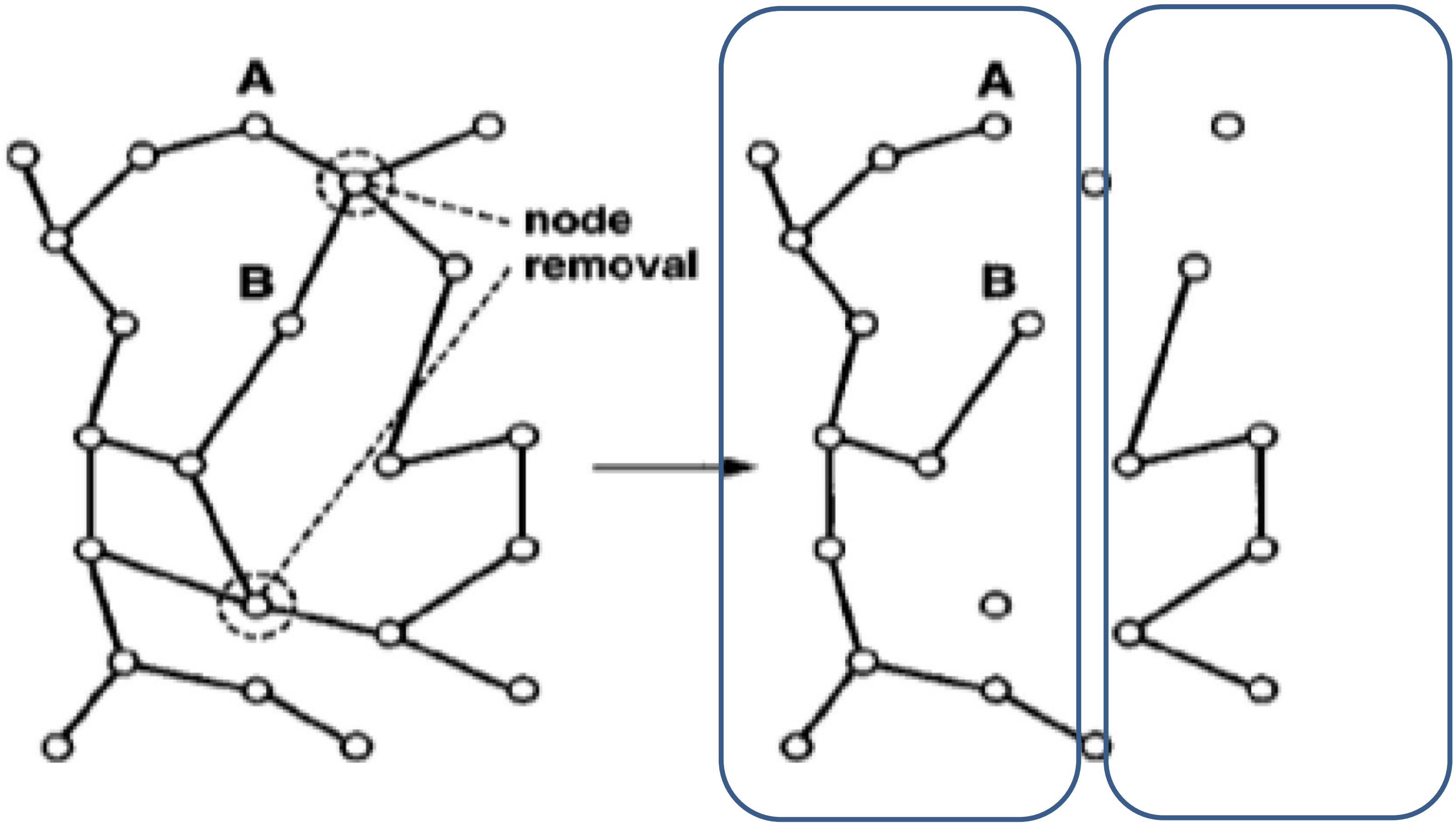


(a) Random walk generation. (b) Representation mapping. (c) Hierarchical Softmax.

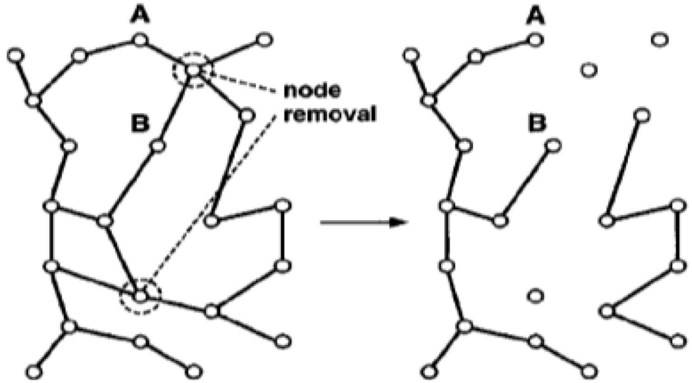
Community Detection Algorithms

- Graph partitioning Algorithm
  - The Clique Percolation Method
  - The Kernighan-Lin (KL) algorithm
- Structural Similarity Algorithm
- Edge Removal Algorithm
- Randomized Algorithm

Edge Removal Algorithm

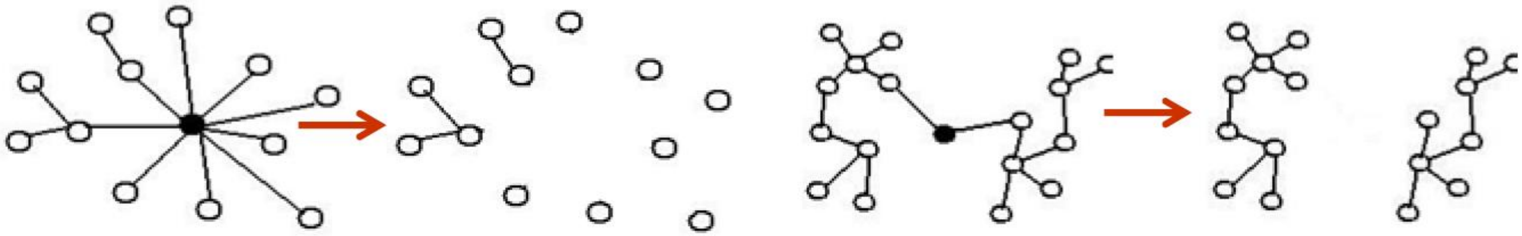


Edge Removal Algorithm



By **defining a measure** to rank the nodes

- By degree ?
- By clustering coefficient ?
- By betweeness ?



By Detweeness

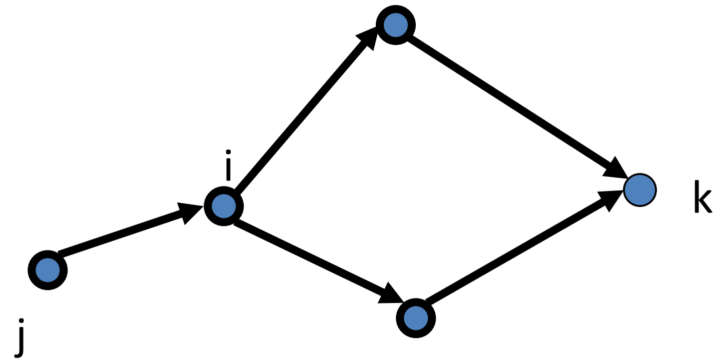
ratio of the number of shortest paths passing through a node *v* out of all shortest paths between all node pairs in a network

betweenness of vertex i

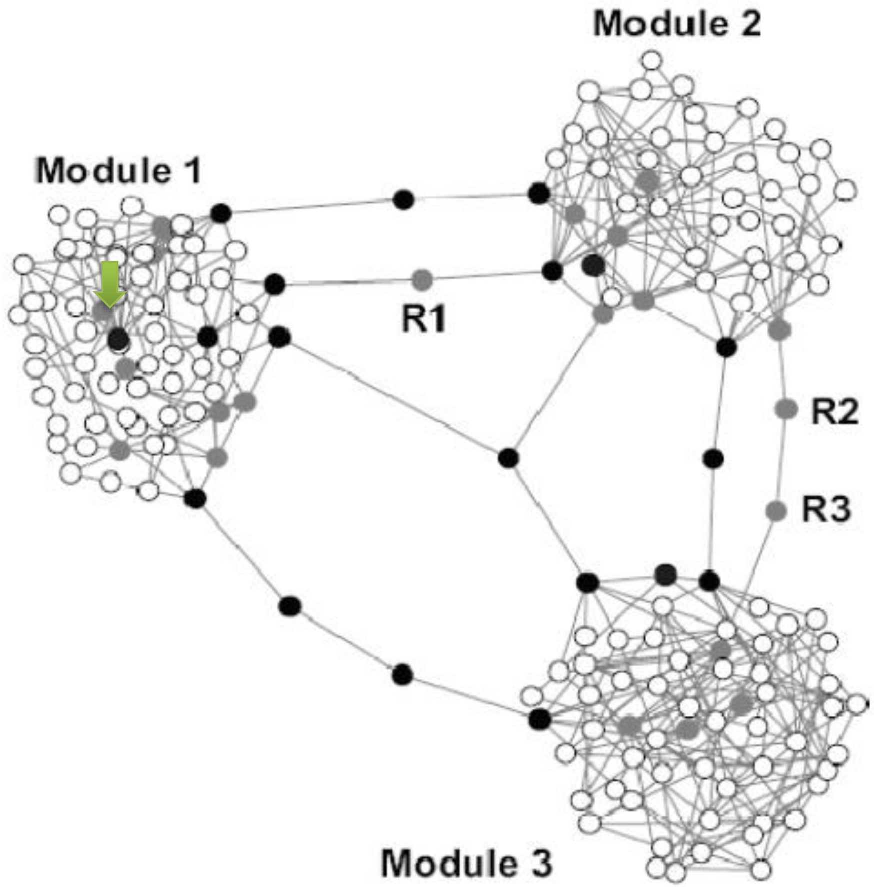
shortest paths between j and k that pass through i

$$C_B(n_i) = \sum_{j,k} g_{jk}(i) / g_{jk}$$

all shortest paths between j and k



By Detweeness ?



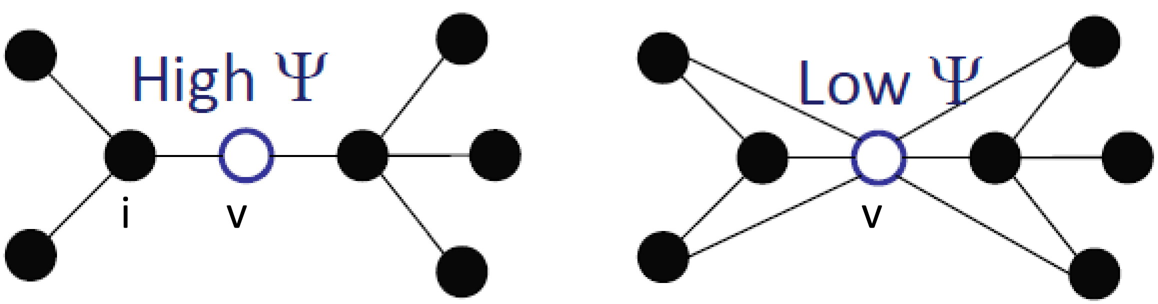
Bridging coefficient

δ(i) = |N(i)-N(v)-v|

– For node v:

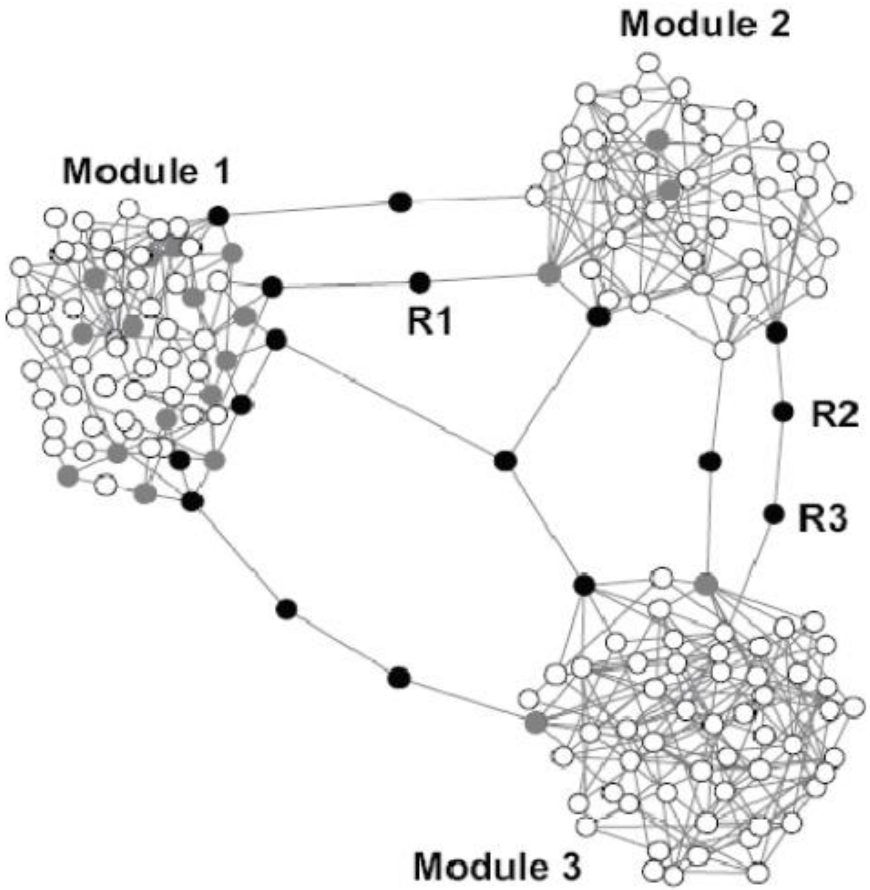
The probability of leaving the direct neighbors of a node v.

$$\Psi(v) = \frac{1}{d(v)} \sum_{i \in N(v)} \frac{\delta(i)}{d(i) - 1}$$

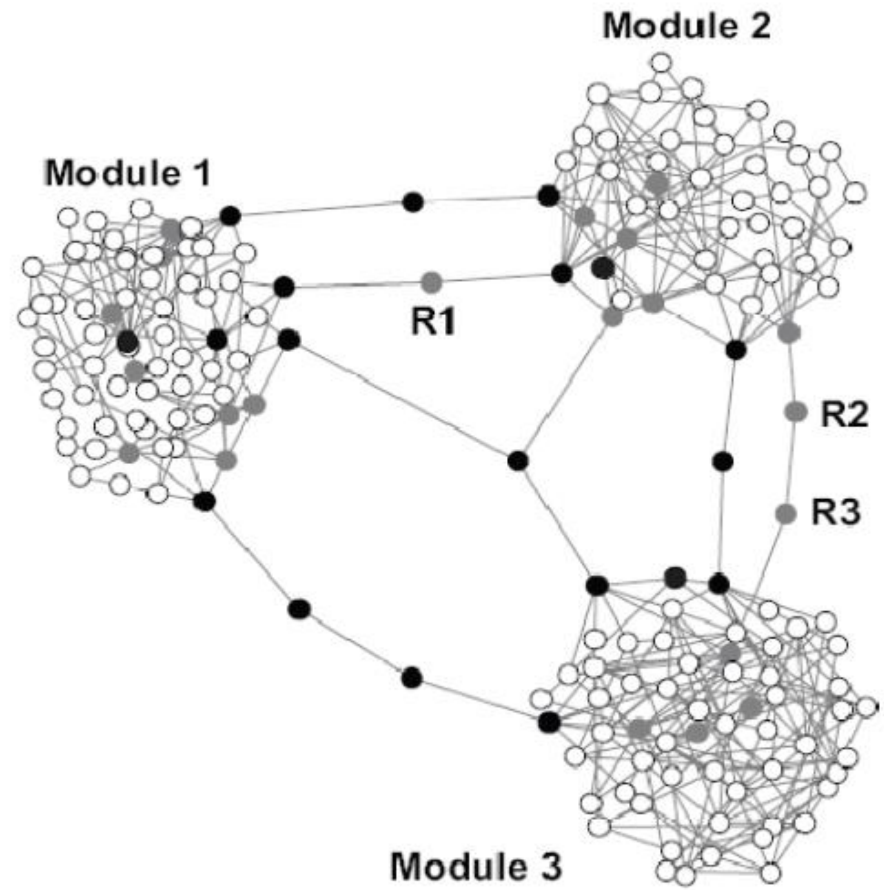


Bridging Coefficient: a measurement that measuring the extent how well a node or edge is located between well connected regions.

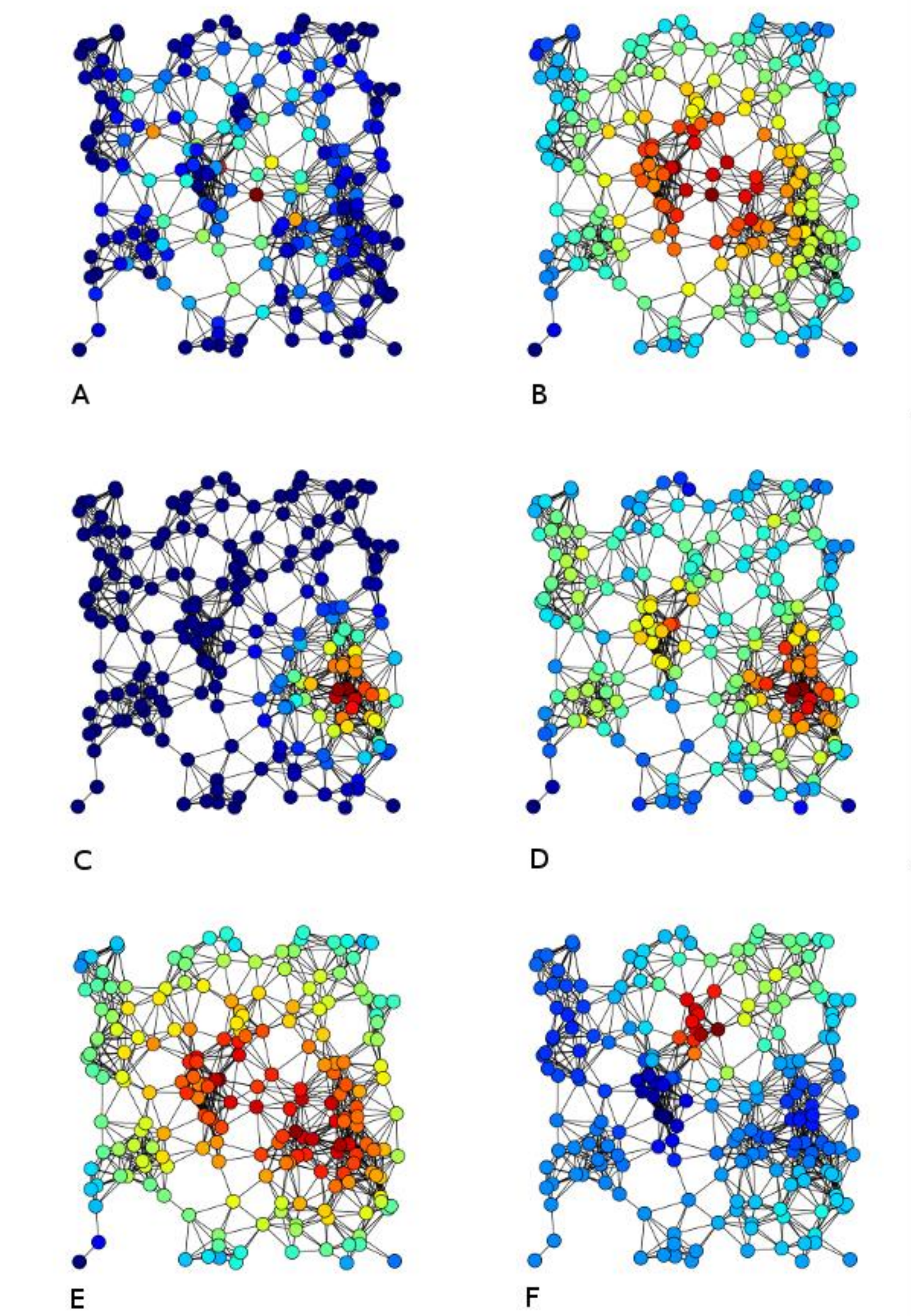
Experiment



Bridging centrality



Betweenness centrality



Examples of

A) [Betweenness centrality](),

B) [Closeness centrality](),

C) [Eigenvector centrality](),

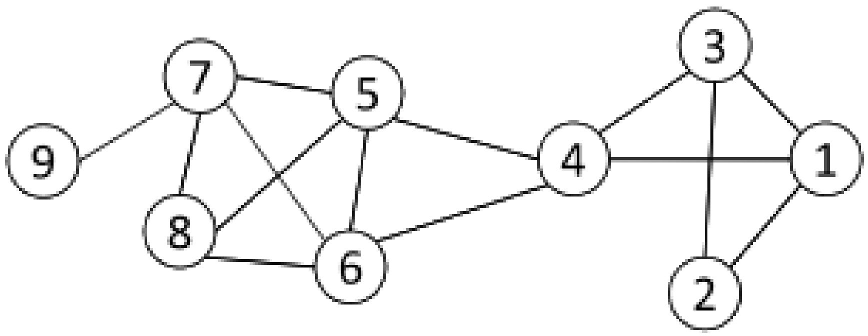
D)[Degree centrality](),

E) [Harmonic centrality]()

F) [Katz centrality]() of the same graph

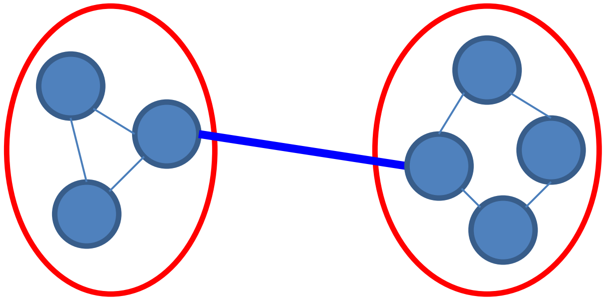
Edge Betweenness

- The strength of a tie can be measured by edge betweenness
- Edge betweenness: the number of shortest paths that pass along with the edge

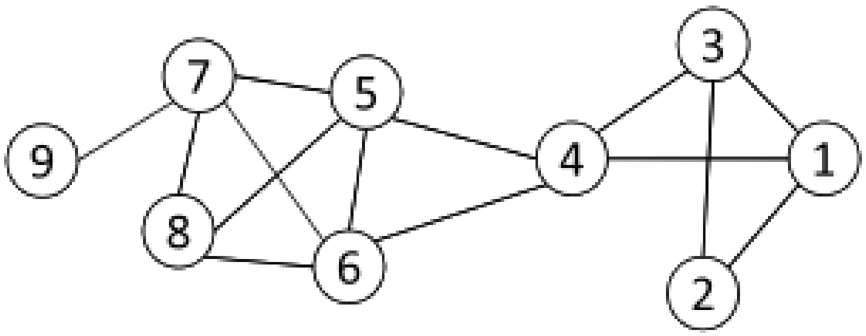


The edge betweenness of e(1, 2) is 4 (=6/2 + 1), as all the shortest paths from 2 to {4, 5, 6, 7, 8, 9} have to either pass e(1, 2) or e(2, 3), and e(1,2) is the shortest path between 1 and 2

- The edge with higher betweenness tends to be the <u>bridge</u> between two communities.



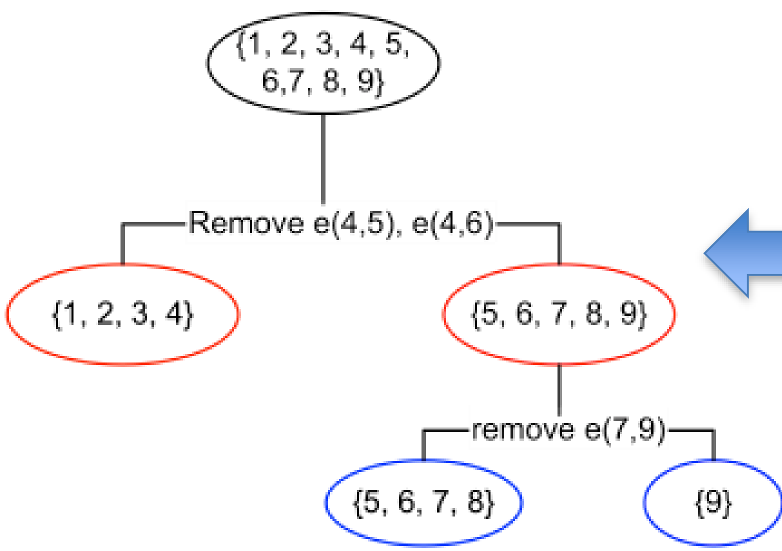
Divisive clustering based on edge betweenness



Initial betweenness value

Table 3.3: Edge Betweenness

	1	2	3	4	5	6	7	8	9
1	0	4	1	9	0	0	0	0	0
2	4	0	4	0	0	0	0	0	0
3	1	4	0	9	0	0	0	0	0
4	9	0	9	0	10	10	0	0	0
5	0	0	0	10	0	1	6	3	0
6	0	0	0	10	1	0	6	3	0
7	0	0	0	0	6	6	0	2	8
8	0	0	0	0	3	3	2	0	0
9	0	0	0	0	0	0	8	0	0



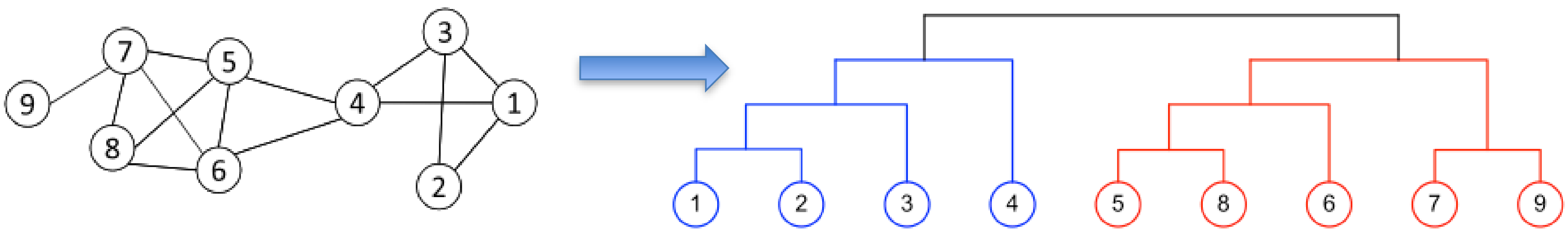
After remove e(4,5), the betweenness of e(4, 6) becomes 20, which is the highest;

After remove e(4,6), the edge e(7,9) has the highest betweenness value 4, and should be removed.

Idea: progressively removing edges with the highest betweenness

Agglomerative Hierarchical Clustering

- Initialize each node as a community
- Merge communities successively into larger communities following a certain criterion



Community Detection Algorithms

- Graph partitioning Algorithm
  - The Clique Percolation Method
  - The Kernighan-Lin (KL) algorithm
- Structural Similarity Algorithm
- Edge Removal Algorithm
- Randomized Algorithm

What is a Random Walk?

- Given a graph and a starting point (node), we select a neighbor of it at random, and move to this neighbor;
- Then we select a neighbor of this node and move to it, and so on;
- The (random) sequence of nodes selected this way is a random walk on the graph

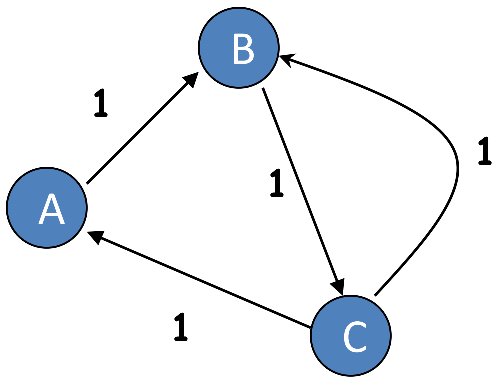
An example

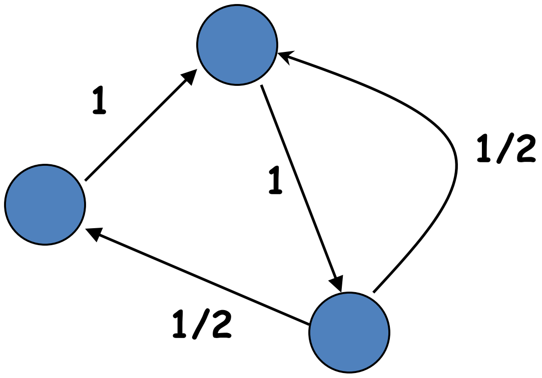
0	1	0
0	0	1
1	1	0

Adjacency matrix A

0	1	0
0	0	1
1/2	1/2	0

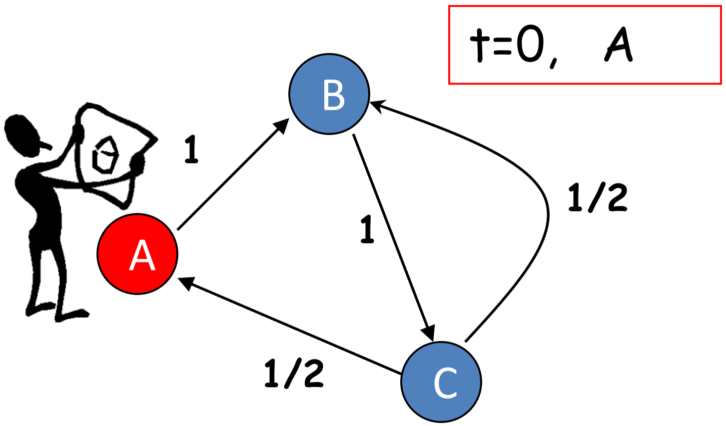
Transition matrix P





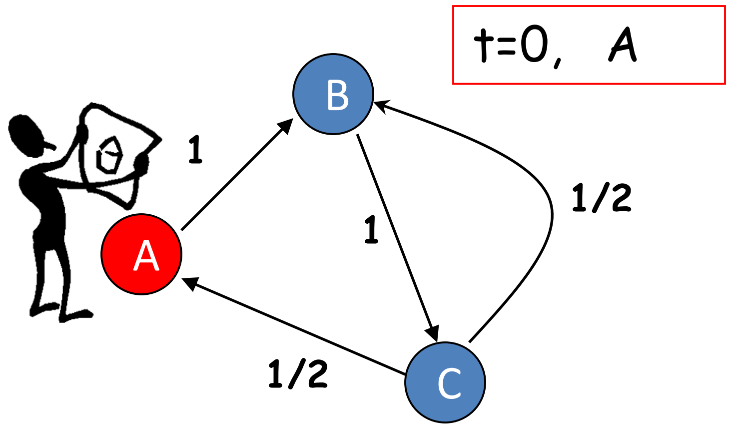
Slide from Purnamitra Sarkar, *Random Walks on Graphs: An Overview*

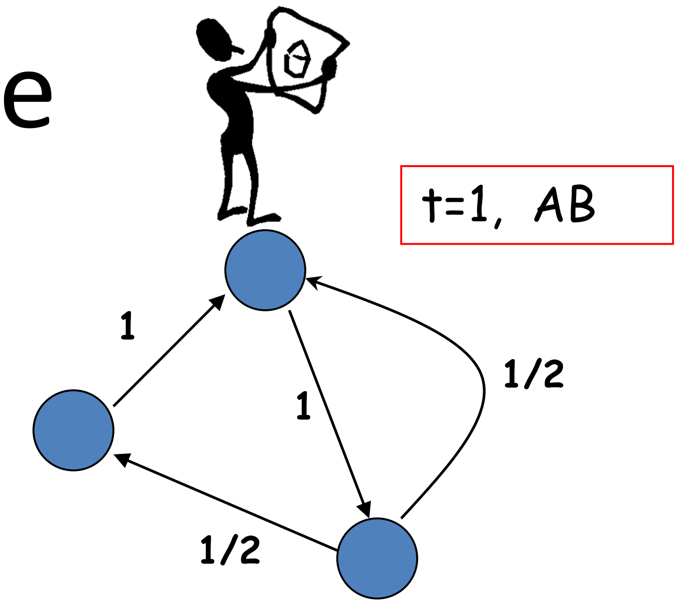
An example



Slide from Purnamitra Sarkar, *Random Walks on Graphs: An Overview*

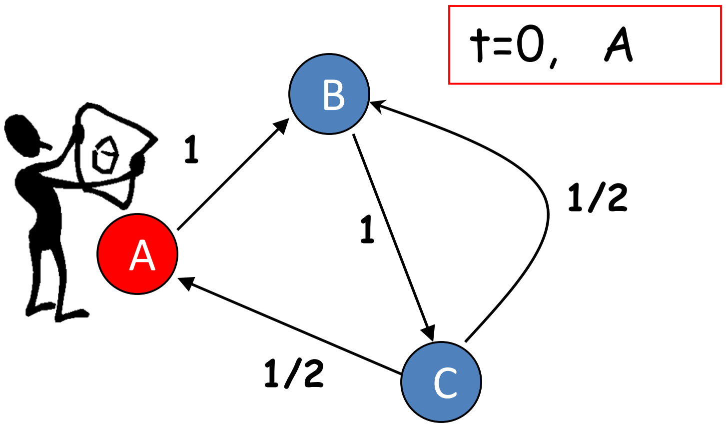
An example

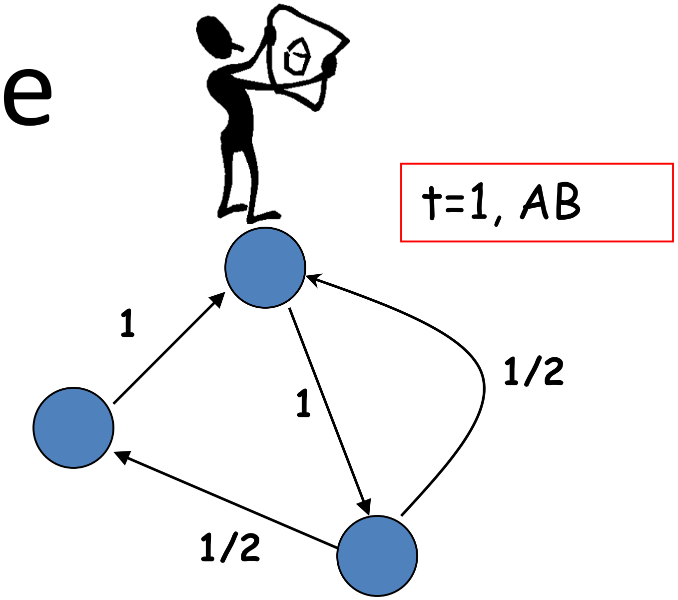


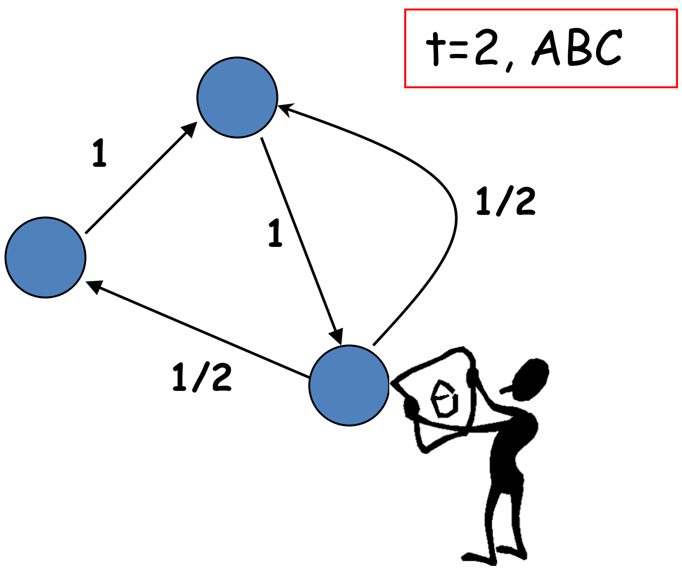


Slide from Purnamitra Sarkar, *Random Walks on Graphs: An Overview*

An example

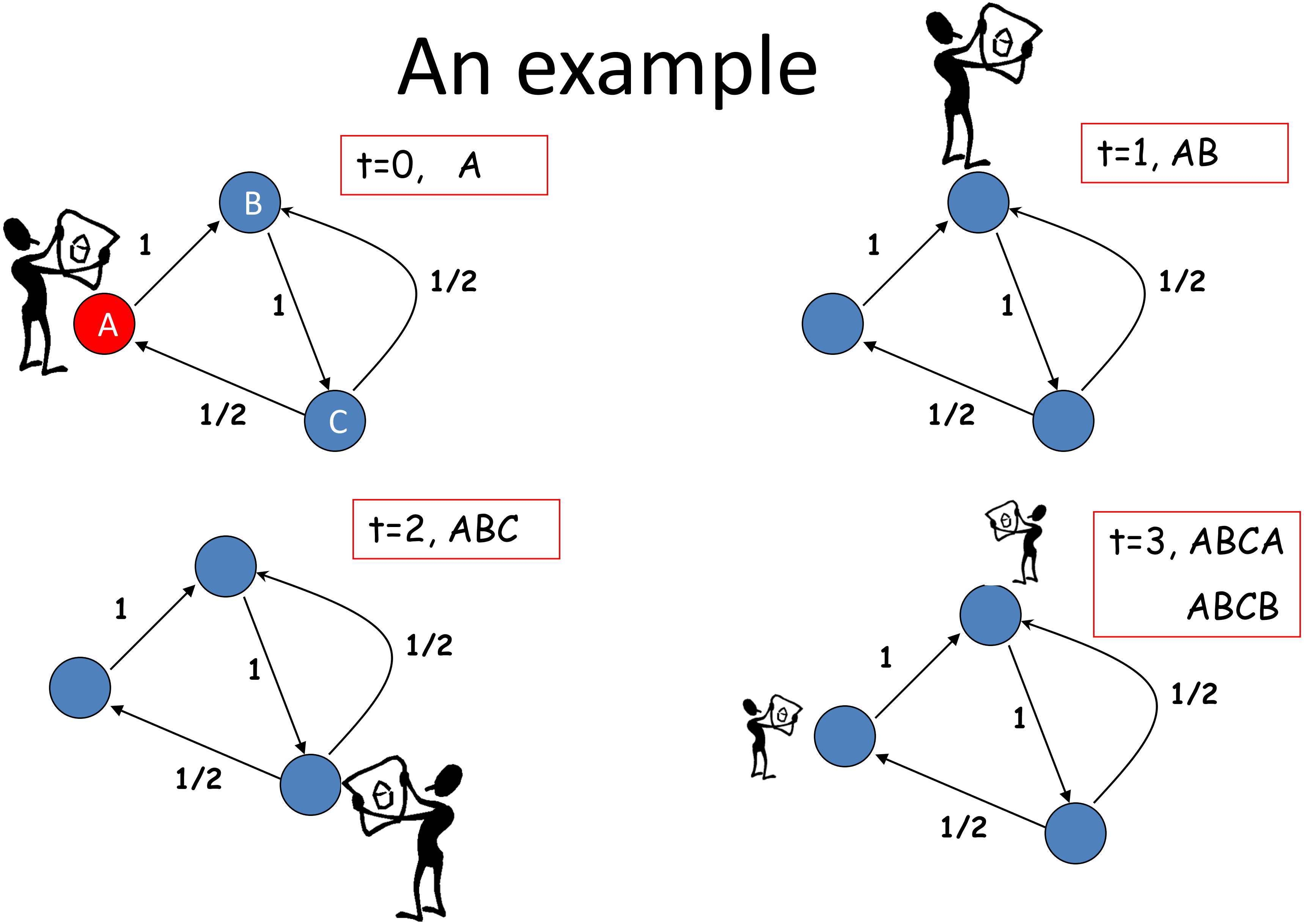






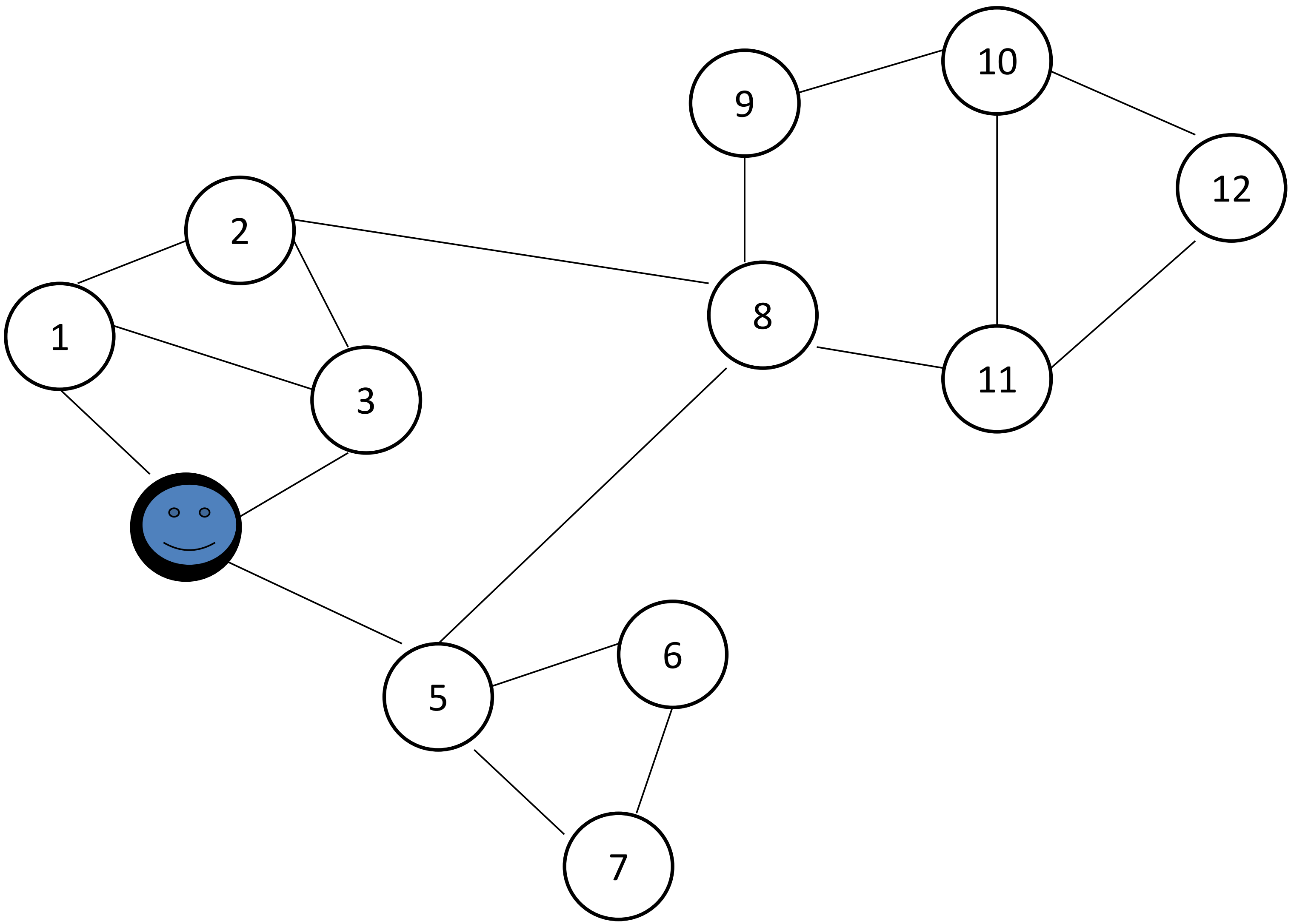
Slide from Purnamitra Sarkar, *Random Walks on Graphs: An Overview*

An example

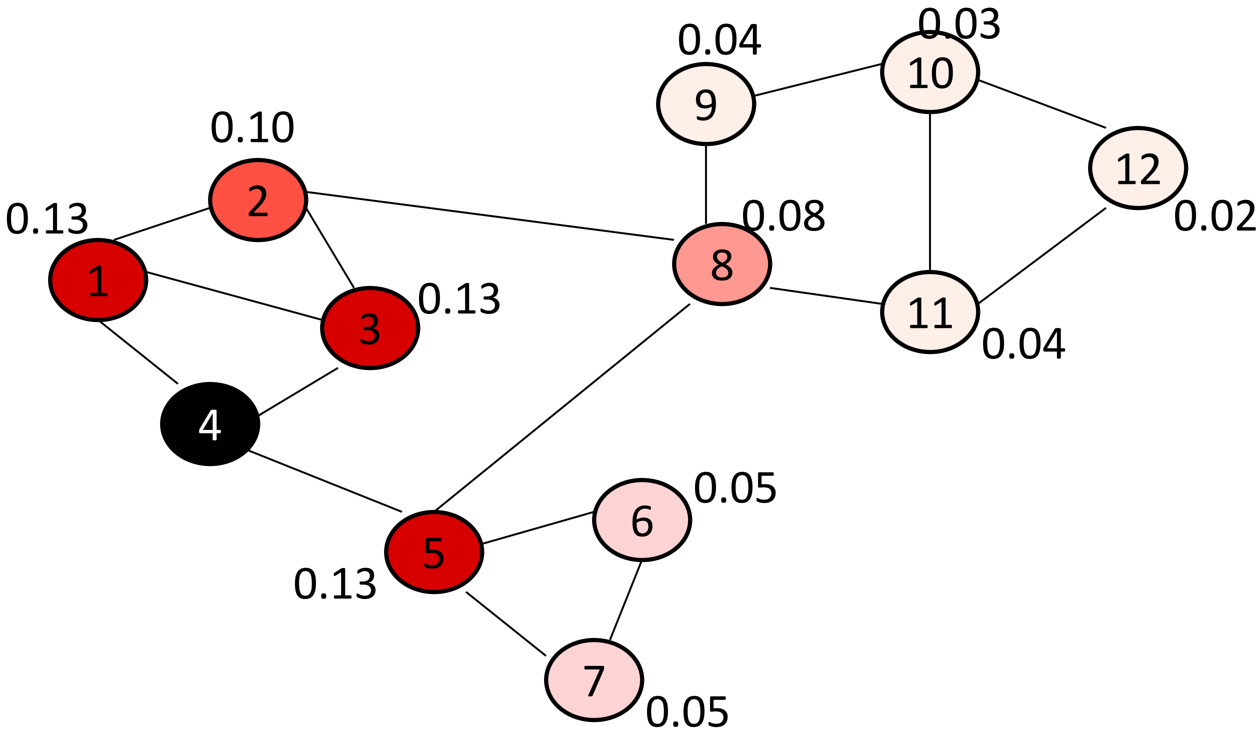


Slide from Purnamitra Sarkar, *Random Walks on Graphs: An Overview*

Random walk with restart



Random walk with restart



Nearby nodes, higher scores

More red, more relevant

	Node 4
Node 1	0.13
Node 2	0.10
Node 3	0.13
Node 4	0.22
Node 5	0.13
Node 6	0.05
Node 7	0.05
Node 8	0.08
Node 9	0.04
Node 10	0.03
Node 11	0.04
Node 12	0.02

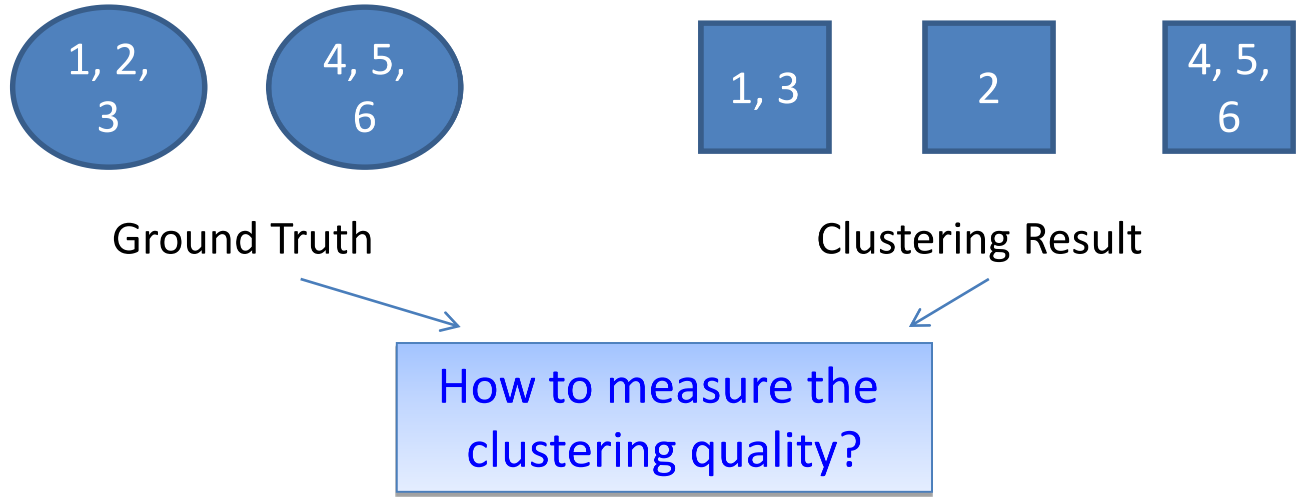
Ranking vector

$\vec{r}_4$

Community Detection Algorithms

- Graph partitioning Algorithm
  - The Clique Percolation Method
  - The Kernighan-Lin (KL) algorithm
- Structural Similarity Algorithm
- Edge Removal Algorithm
- Randomized Algorithm

Measuring a clustering/community Result



- The number of communities after grouping can be different from the ground truth
- No clear community <u>correspondence</u> between clustering result and the ground truth

Accuracy of Pairwise Community Memberships

- Consider all the possible pairs of nodes and check whether they reside in the same community
- An error occurs *if*
  - Two nodes belonging to the same community are assigned to different communities after clustering
  - Two nodes belonging to different communities are assigned to the same community
- Construct a contingency table or confusion matrix

		Ground Truth	
		$C(v_i) = C(v_j)$	$C(v_i) \neq C(v_j)$
Clustering Result	$C(v_i) = C(v_j)$	a	b
	$C(v_i) \neq C(v_j)$	c	d

$$accuracy = \frac{a + d}{a + b + c + d}$$