Social Network Analysis

(SNA)

Dr. Rathachai Chawuthai

Department of Computer Engineering

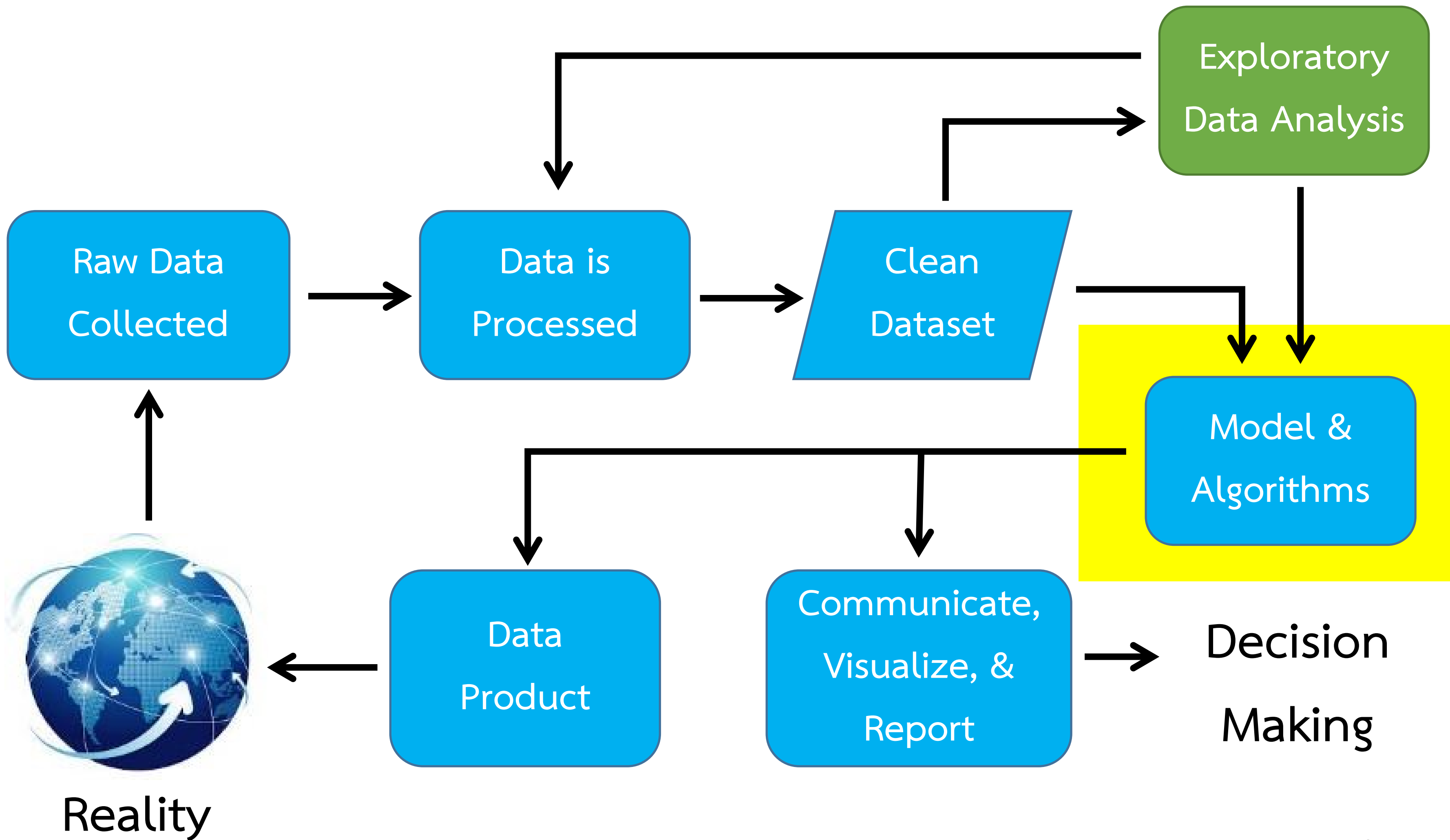
Faculty of Engineering

King Mongkut's Institute of Technology Ladkrabang

Agenda

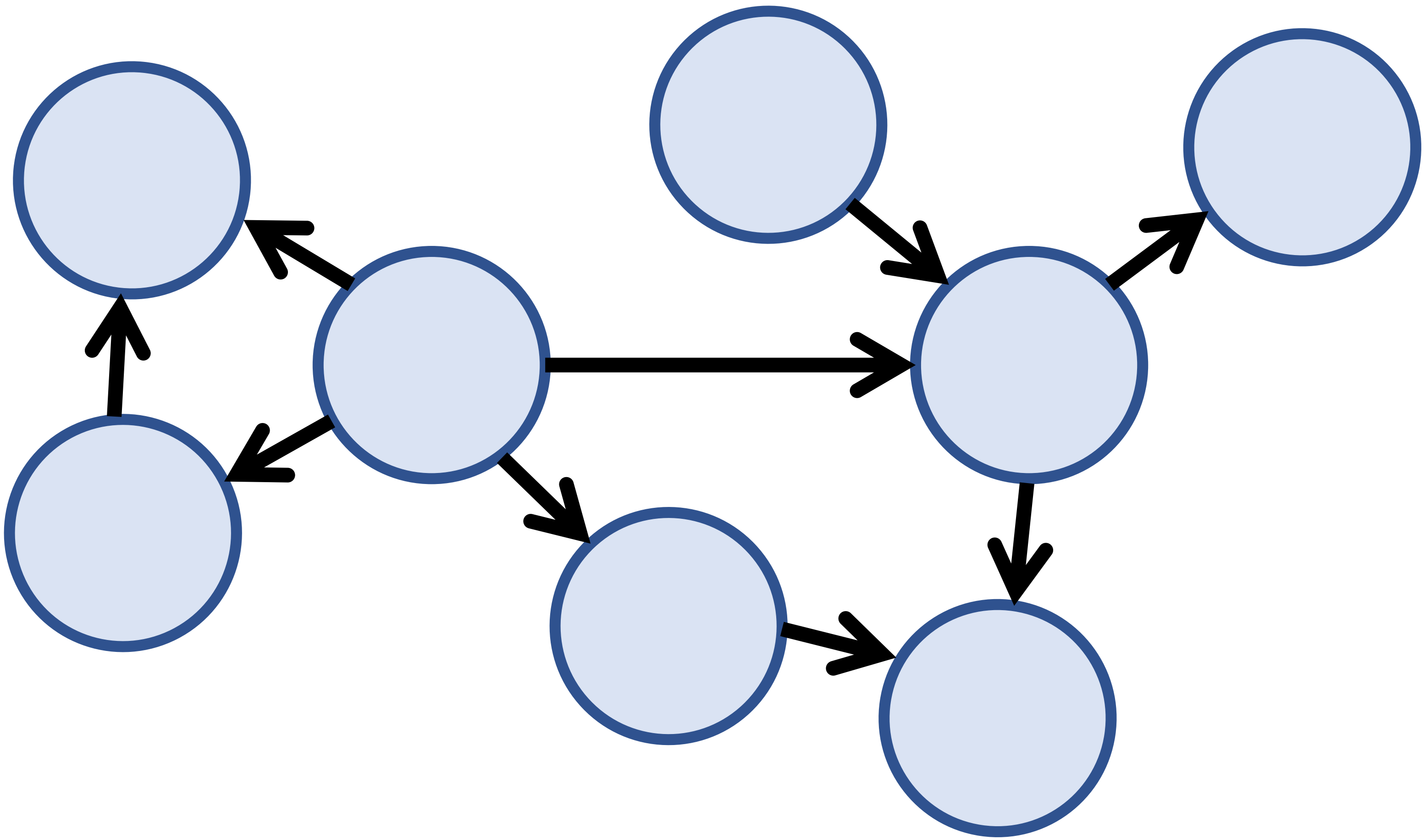
- Introduction
- SNA Properties
- Community Detection

Data Science Process

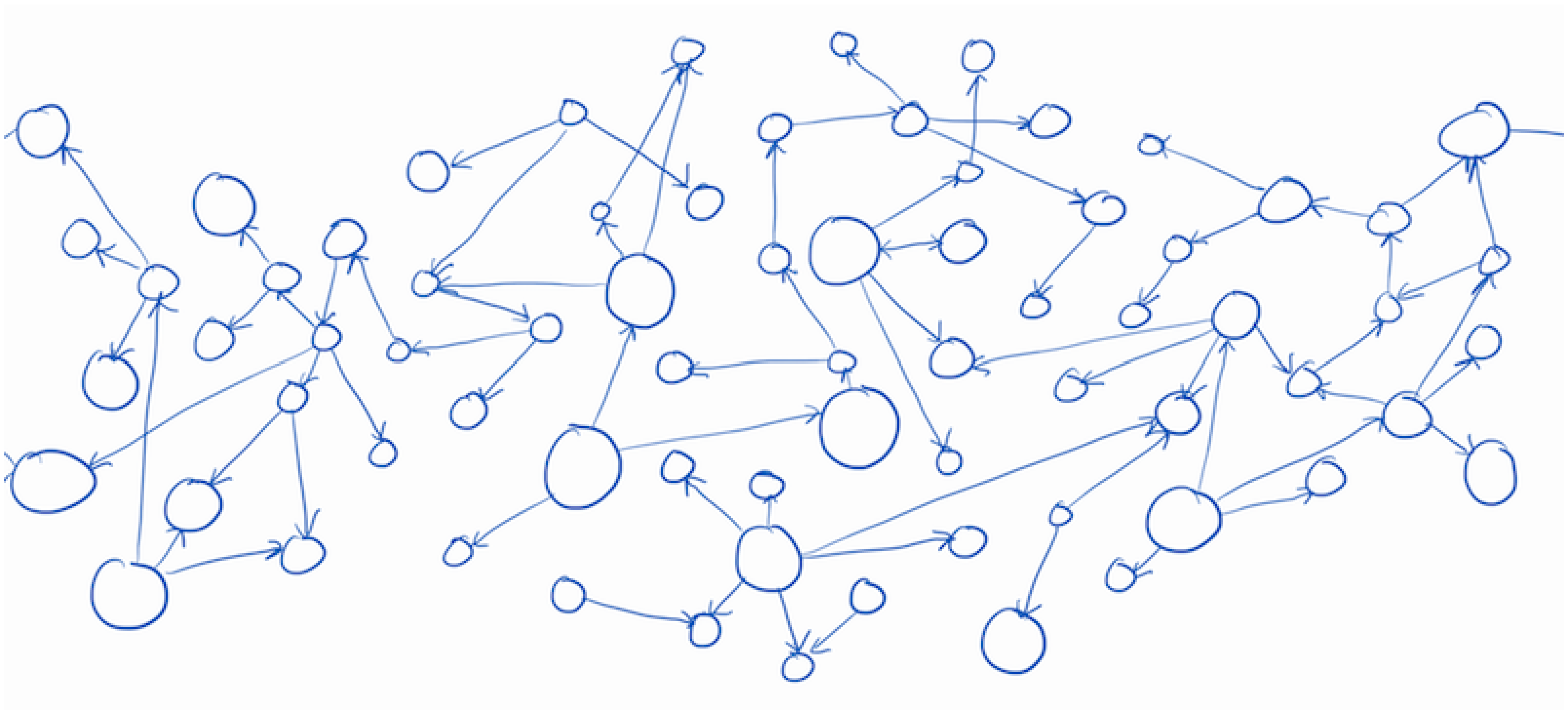


Introduction

Graph

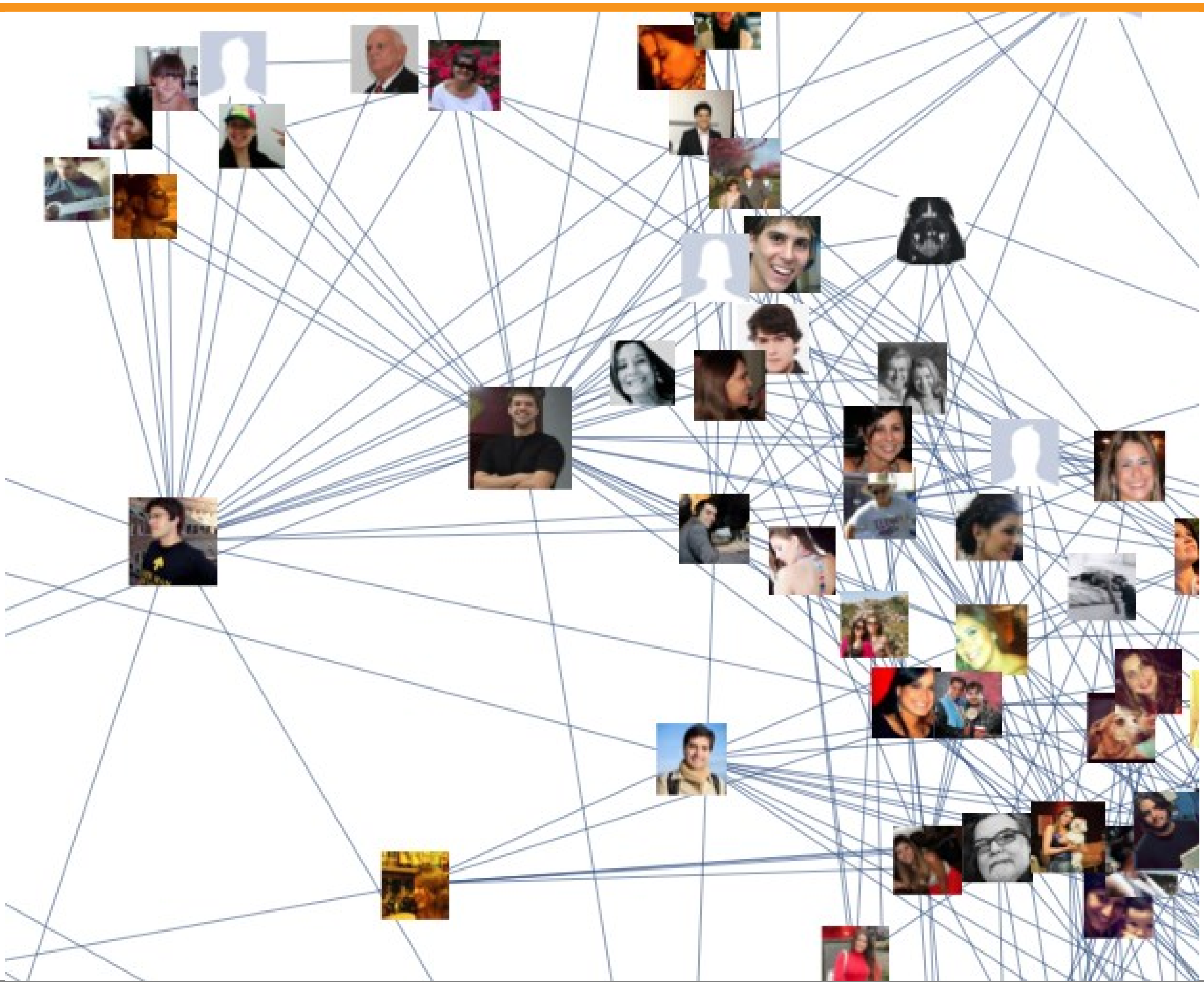


Graph



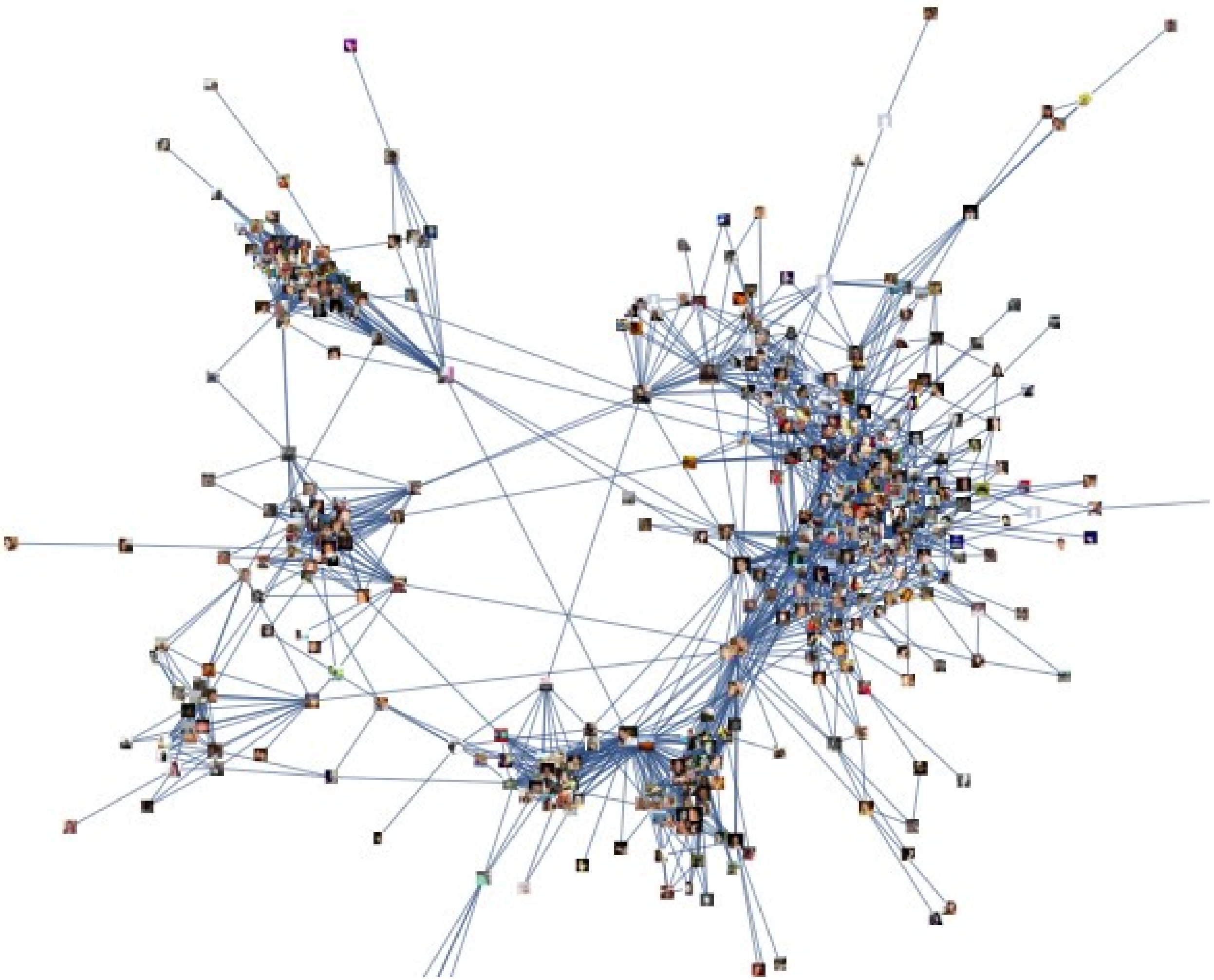
Ref: • (image) https://code-cartoons.com/a-cartoon-intro-to-facebook-s-relay-part-1-3ec1a127bca5

Graph



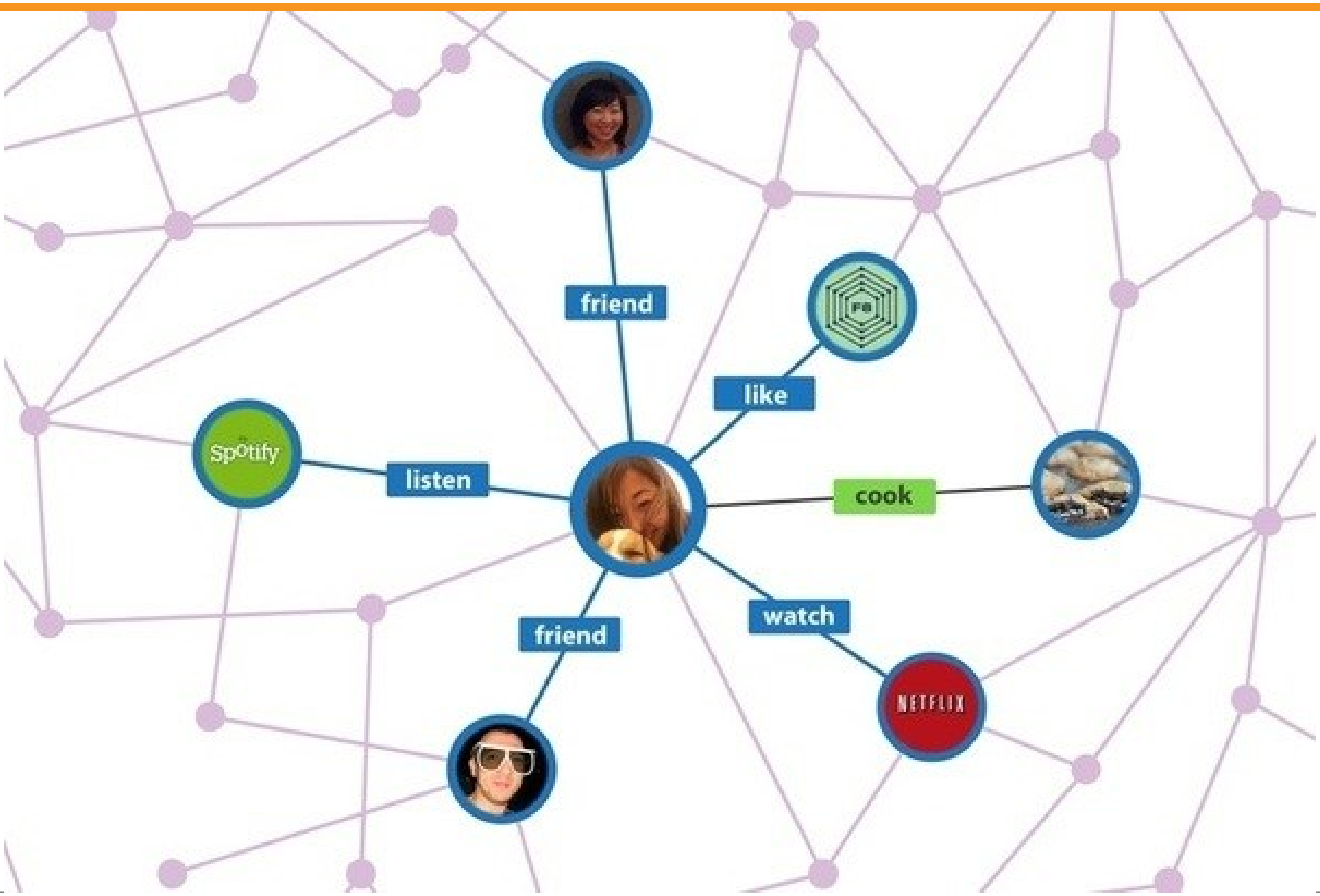
Ref: • (image) https://mathematica.stackexchange.com/questions/11673/how-to-play-with-facebook-data-inside-mathematica

Graph



Ref: • (image) https://mathematica.stackexchange.com/questions/11673/how-to-play-with-facebook-data-inside-mathematica

Graph



Ref: • (image) https://www.quora.com/How-does-graph-theory-apply-to-Facebook

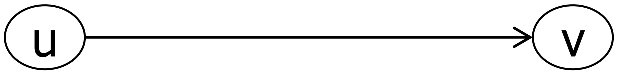
Definitions - Graph

A generalization of the simple concept of a set of dots, links, edges or arcs.

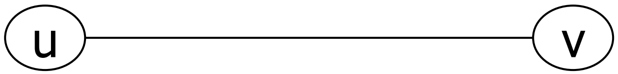
Representation: Graph G =(V, E) consists set of vertices denoted by V, or by V(G) and set of edges E, or E(G)

Edge Type

Directed: Ordered pair of vertices. Represented as (u, v) directed from vertex u to v.



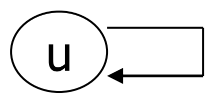
Undirected: Unordered pair of vertices. Represented as {u, v}. Disregards any sense of direction and treats both end vertices interchangeably.



Ref: • B. Prabhakaran: "Graph Theory"

Edge Type

- **Loop:** A loop is an edge whose endpoints are equal i.e., an edge joining a vertex to it self is called a loop. Represented as {u, u} = {u}

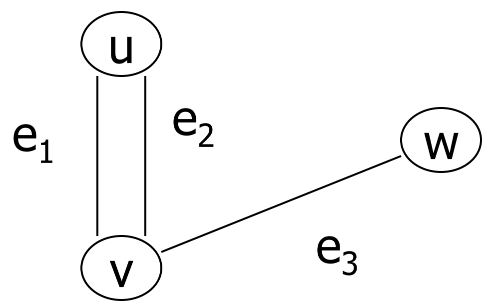


- **Multiple Edges:** Two or more edges joining the same pair of vertices.

Graph Type

Multigraph: G(V,E), consists of set of vertices V, set of Edges E and a function f from E to $\{\{u, v\}|\ u, v\ \ V, u \neq v\}$. The edges e1 and e2 are called multiple or parallel edges if f (e1) = f (e2).

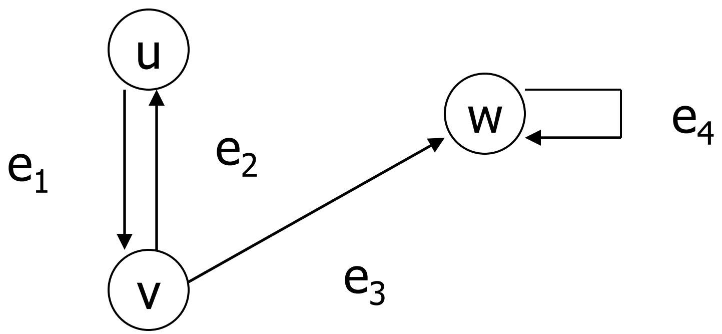
Representation Example: $V = \{u, v, w\}$, $E = \{e_1, e_2, e_3\}$



Graph Type

Directed Multigraph: G(V,E), consists of set of vertices V, set of Edges E and a function f from E to {{u, v}| u, v V}. The edges e1 and e2 are multiple edges if f(e1) = f(e2)

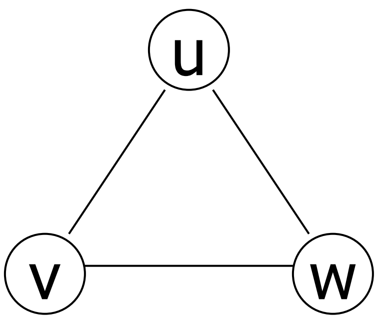
Representation Example: V = {u, v, w}, E = {e_1, e_2, e_3, e_4}



Ref: • B. Prabhakaran: “Graph Theory”

Representation- Adjacency Matrix

- Example: Undirected Graph G (V, E)

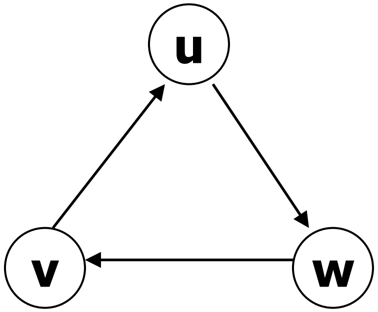


	v	u	w
v	0	1	1
u	1	0	1
w	1	1	0

Ref: • B. Prabhakaran: "Graph Theory"

Representation- Adjacency Matrix

- Example: Directed Graph G (V, E)

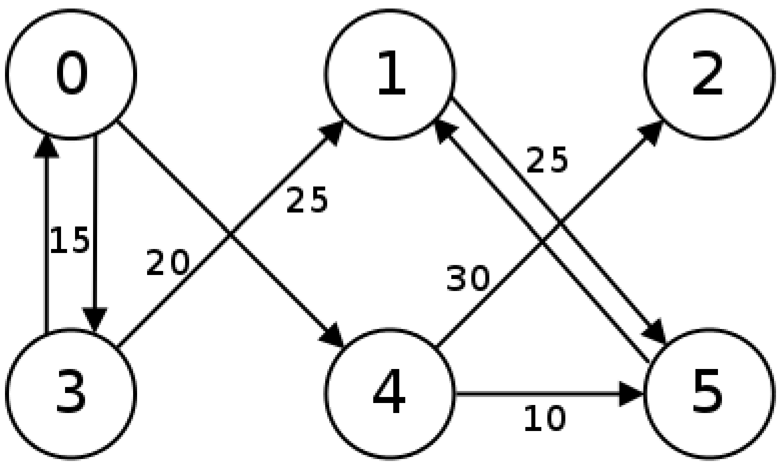


	v	u	w
v	0	1	0
u	0	0	1
w	1	0	0

Ref: • B. Prabhakaran: "Graph Theory"

Representation- Adjacency Matrix

- Example: Weighted Graph G (V, E, w)

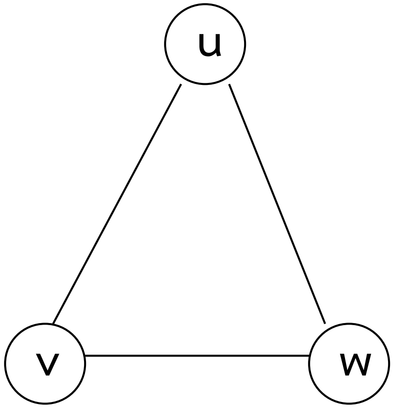


	0	1	2	3	4	5
0				15	20	
1						25
2						
3	15	25				
4			30			10
5		25				

Representation- Adjacency List

- Each node (vertex) has a list of which nodes (vertex) it is adjacent

Example: undirectd graph G (V, E)

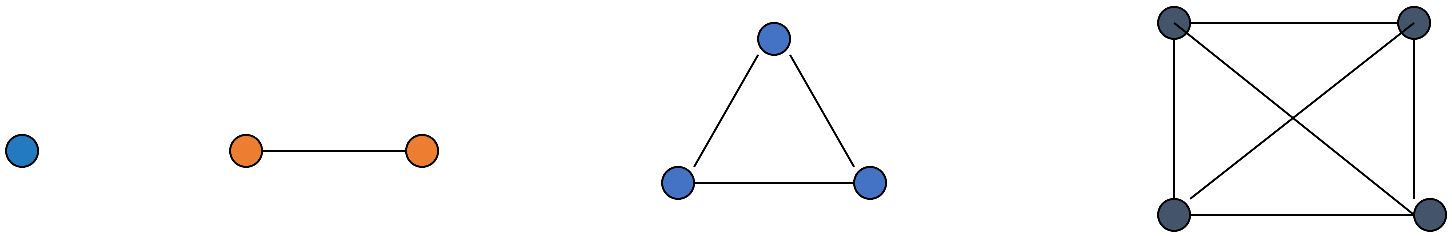


node	Adjacency List
u	v , w
v	w, u
w	u , v

Complete Graph

- **Complete graph:** K_n, is the simple graph that contains exactly one edge between each pair of distinct vertices.

Representation Example: K_1, K_2, K_3, K_4

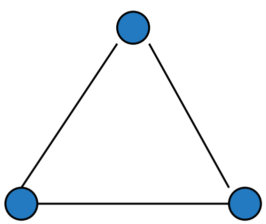


K_1 K_2 K_3 K_4

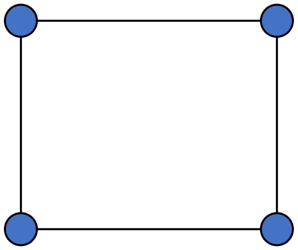
Title

- **Cycle:** C_n, $n \geq 3$ consists of n vertices $v_1, v_2, v_3 \ldots v_n$ and edges $\{v_1, v_2\}$, $\{v_2, v_3\}, \{v_3, v_4\} \ldots \{v_{n-1}, v_n\}, \{v_n, v_1\}$

  Representation Example: C_3, C_4



C_3



C_4

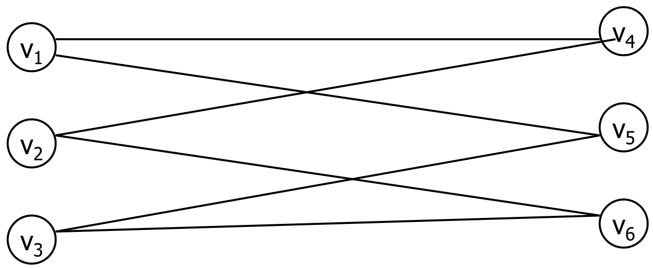
Ref: • B. Prabhakaran: “Graph Theory”

Bipartite Graphs

- In a simple graph G, if V can be partitioned into two disjoint sets V_1 and V_2 such that every edge in the graph connects a vertex in V_1 and a vertex V_2 (so that no edge in G connects either two vertices in V_1 or two vertices in V_2)

  Application example: Representing Relations

  Representation example: $V_1 = \{v_1, v_2, v_3\}$ and $V_2 = \{v_4, v_5, v_6\}$,

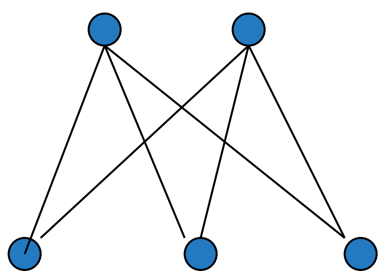


Ref: • B. Prabhakaran: "Graph Theory"

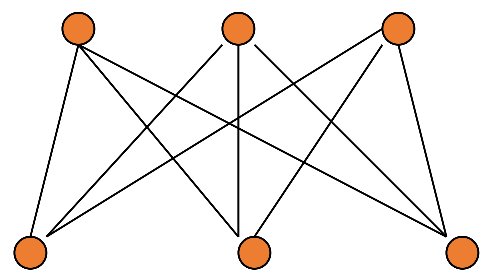
Complete Bipartite Graph

- $K_{m,n}$ is the graph that has its vertex set portioned into two subsets of m and n vertices, respectively There is an edge between two vertices if and only if one vertex is in the first subset and the other vertex is in the second subset.

  Representation example: $K_{2,3,}$ $K_{3,3}$



$K_{2,3}$



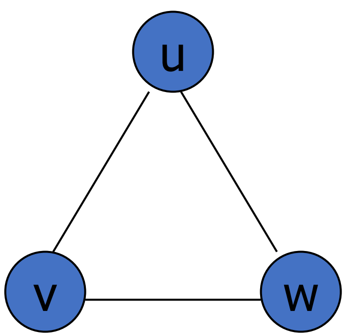
$K_{3,3}$

Subgraphs

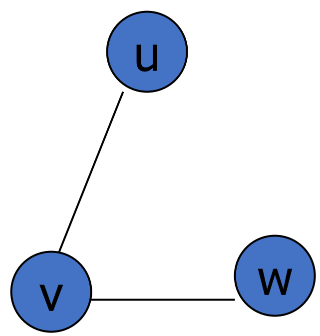
- A subgraph of a graph G = (V, E) is a graph H =(V’, E’) where V’ is a subset of V and E’ is a subset of E

  Application example: solving sub-problems within a graph

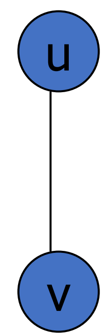
  Representation example: V = {u, v, w}, E = ({u, v}, {v, w}, {w, u}}, H_1 , H_2



G



H_1



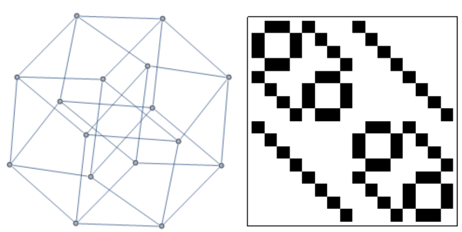
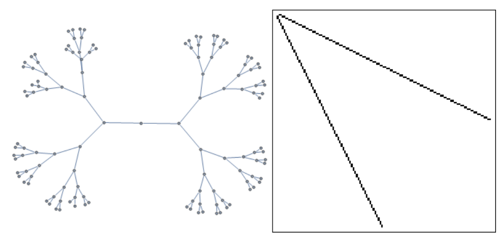
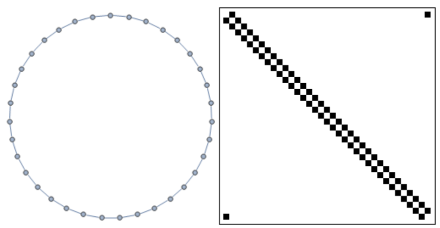
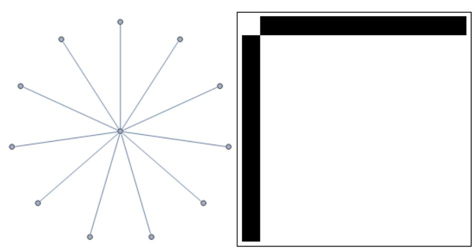
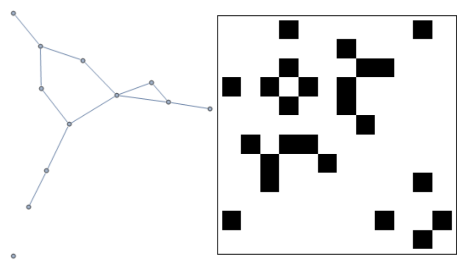
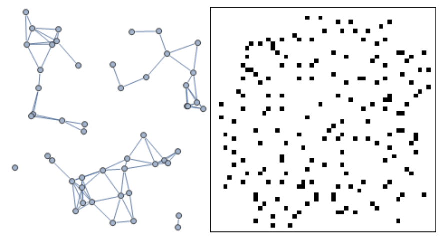
H_2

Ref: • B. Prabhakaran: “Graph Theory”

More Graphs

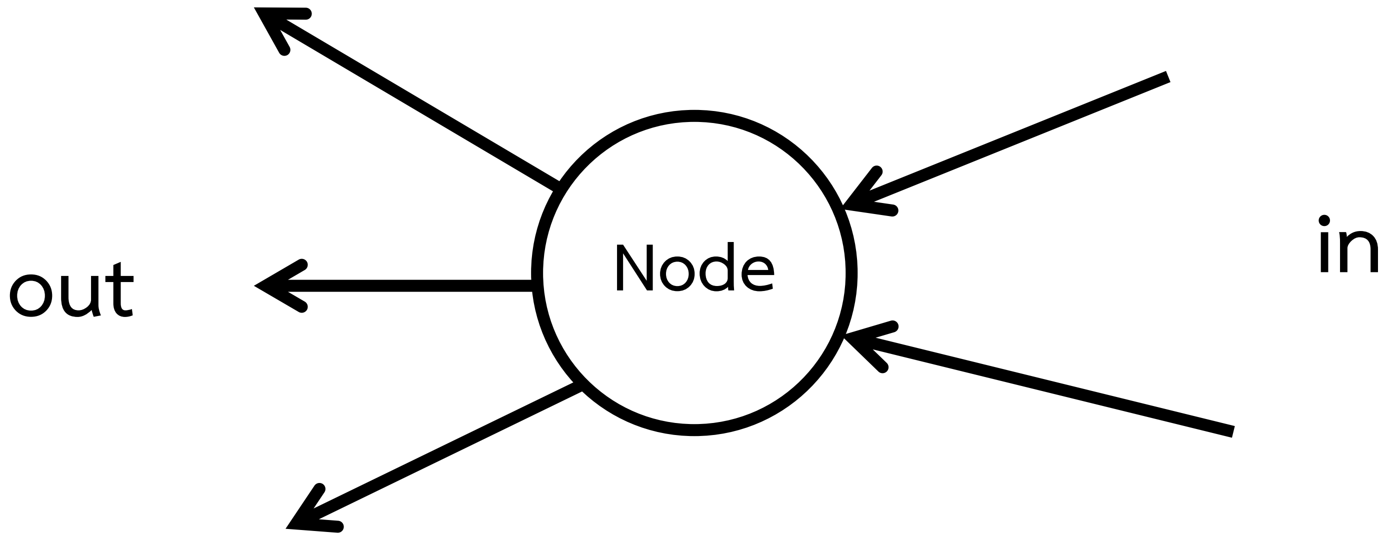
- Go to

http://www.orbifold.net/Mathematica/Networks/

Properties

Degree



degree = 5

indegree = 2

outdegree = 3

Degree

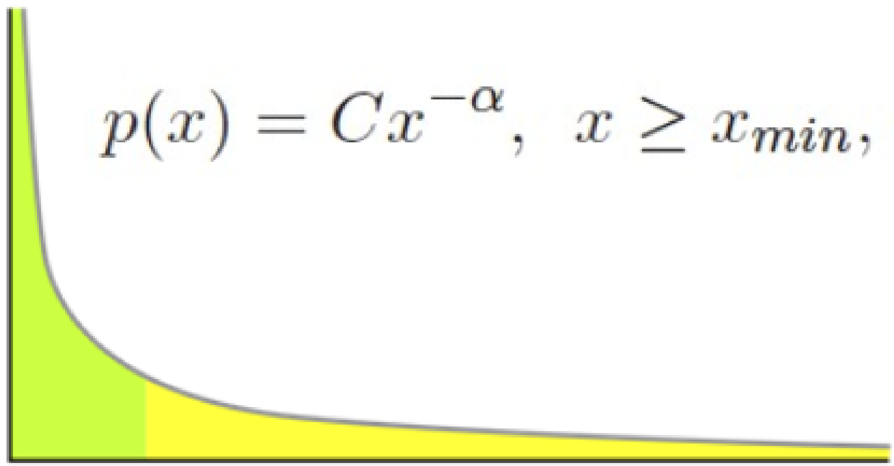
- Min Degree
- Max Degree
- Average Degree

Ref:
- (content) URL
- (image) URL

Degree Distribution

- Degree distribution in large-scale networks often follows a power law, that is, the fraction $p(x)$ of nodes in the network having x connections to other nodes goes for large values of x as:

$$p(x) = Cx^{-\alpha}, \quad x \geq x_{min}, \quad \alpha > 1$$

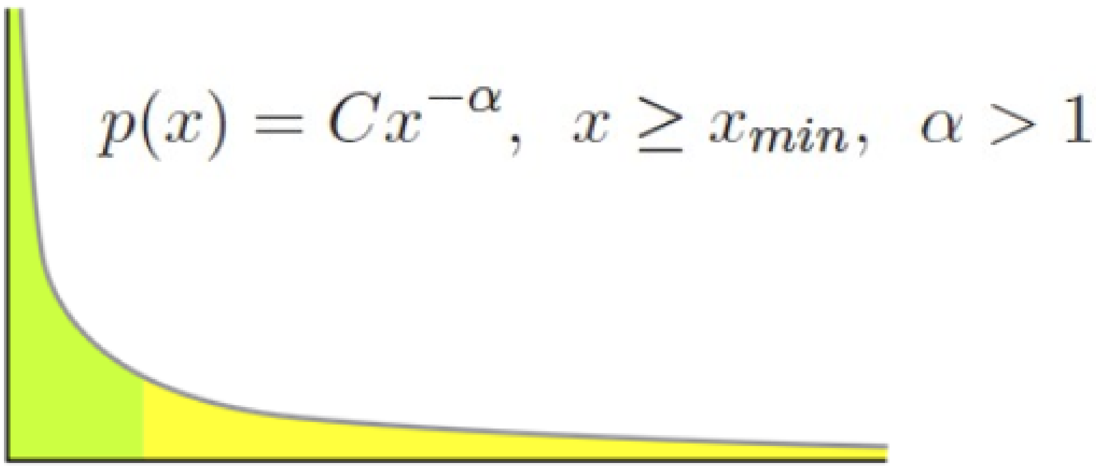


- A.k.a. long tail distribution, scale-free distribution

Ref: • http://slideplayer.com/slide/8897150/

Degree Distribution

- Degree distribution in large-scale networks often follows a power law, that is, the fraction $p(x)$ of nodes in the network having x connections to other nodes goes for large values of x as:

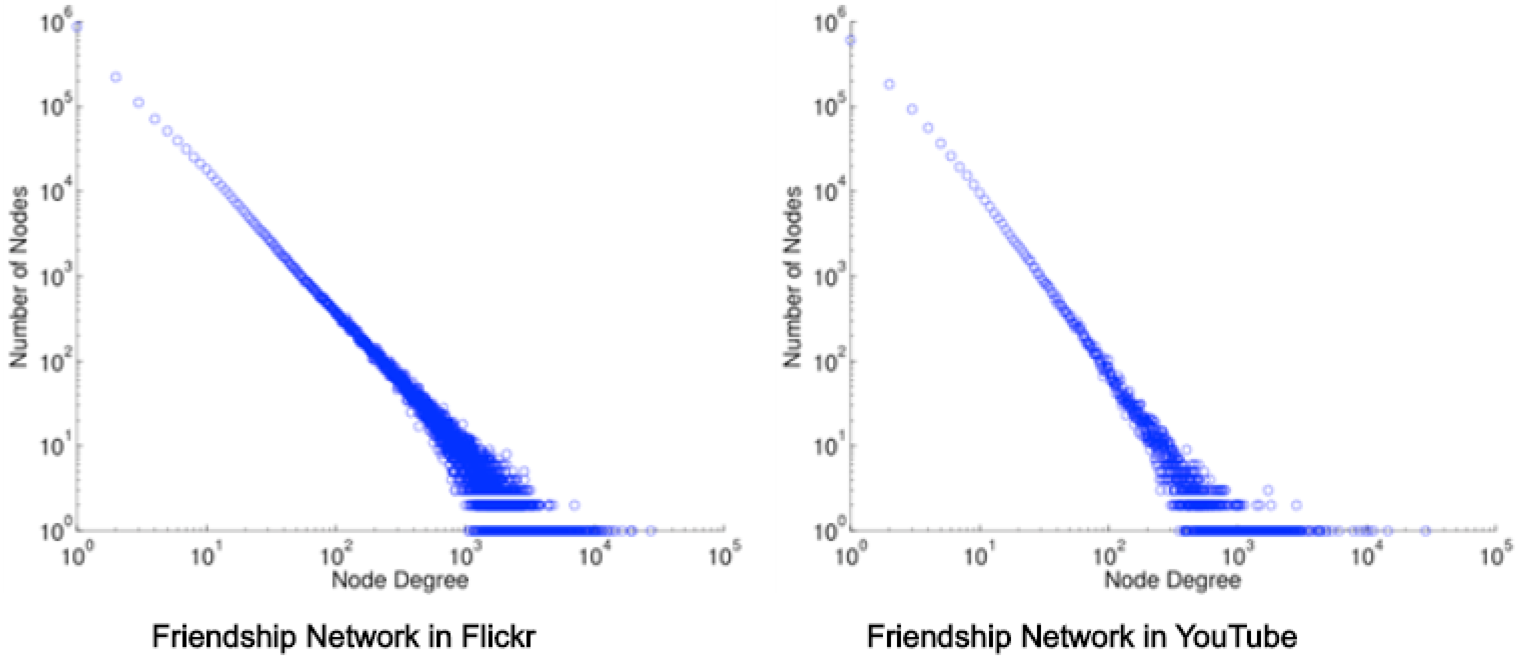


- A.k.a. long tail distribution, scale-free distribution

Ref: • http://slideplayer.com/slide/8897150/

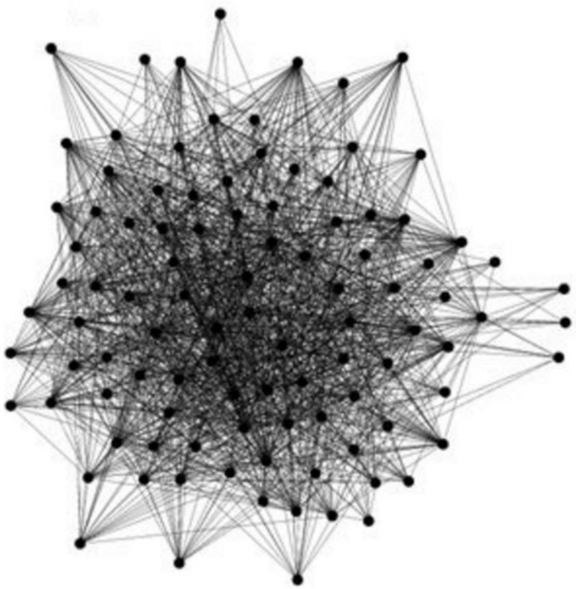
Log-Plot

- Power law distribution becomes a straight line if plotted in a log-log scale



Ref: • http://slideplayer.com/slide/8897150/

Dense/Sparse



Dense Graph

Sparse Graph

Ref: • (image) https://blog.metaflow.fr/sparse-coding-a-simple-exploration-152a3c900a7c?gi=b8cb243083b1

Density

- In mathematics, a dense graph is a graph in which the number of edges is close to the maximal number of edges. The opposite, a graph with only a few edges, is a sparse graph. The distinction between sparse and dense graphs is rather vague, and depends on the context.

Undirected Graph

$$D = \frac{2|E|}{|V|\,(\,|V| - 1\,)}$$

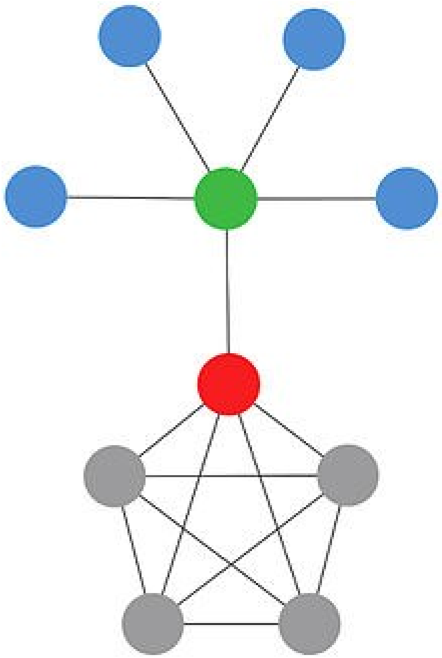
Directed Graph

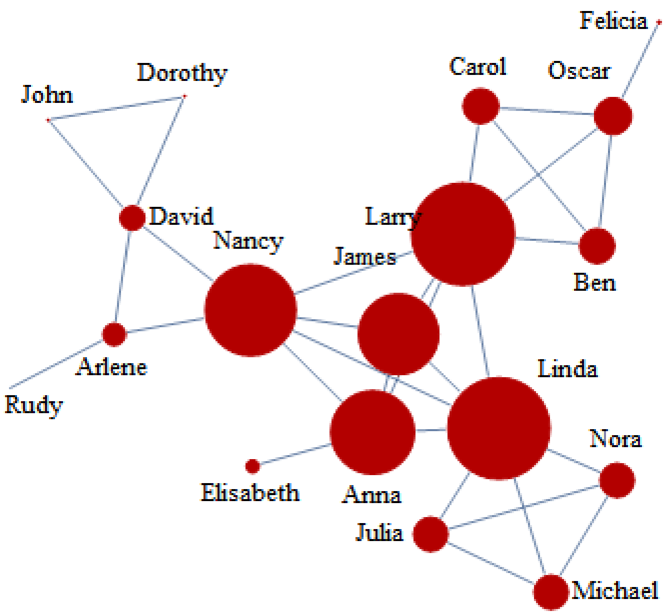
$$D = \frac{|E|}{|V|\,(\,|V| - 1\,)}$$

Ref: • https://en.wikipedia.org/wiki/Dense_graph

Centrality

- Content





Ref:
- (image) https://en.wikipedia.org/wiki/Centrality
- (image) http://www.orbifold.net/Mathematica/Networks/

Closeness Centrality

- In a connected graph, the normalized closeness centrality (or closeness) of a node is the average length of the shortest path between the node and all other nodes in the graph. Thus the more central a node is, the closer it is to all other nodes. Closeness was defined by Bavelas (1950) as the reciprocal of the farness, that is:

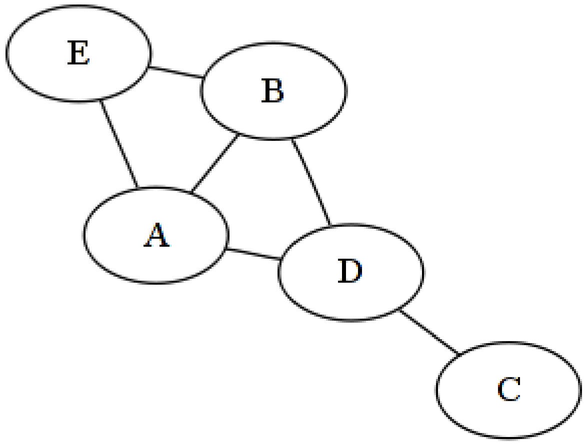
$$C(x) = \frac{N-1}{\sum_y d(y,x)}$$

- where d(x,y) is the distance between vertices x and y. However, when speaking of closeness centrality, people usually refer to its normalized form, generally given by the previous formula multiplied by N-1, where N is the number of nodes in the graph. This adjustment allows comparisons between nodes of graphs of different sizes.
- Taking distances from or to all other nodes is irrelevant in undirected graphs, whereas it can produce totally different results in directed graphs (e.g. a website can have a high closeness centrality from outgoing link, but low closeness centrality from incoming links).

Ref: • (content) https://en.wikipedia.org/wiki/Centrality

Closeness Centrality

$$C(x) = \frac{N-1}{\sum_y d(y,x)}$$



$$C(A) = \frac{5-1}{d(B,A)+d(C,A)+d(D,A)+d(E,A)}$$

$$C(A) = \frac{4}{1+2+1+1} = \frac{4}{5} = 0.8$$

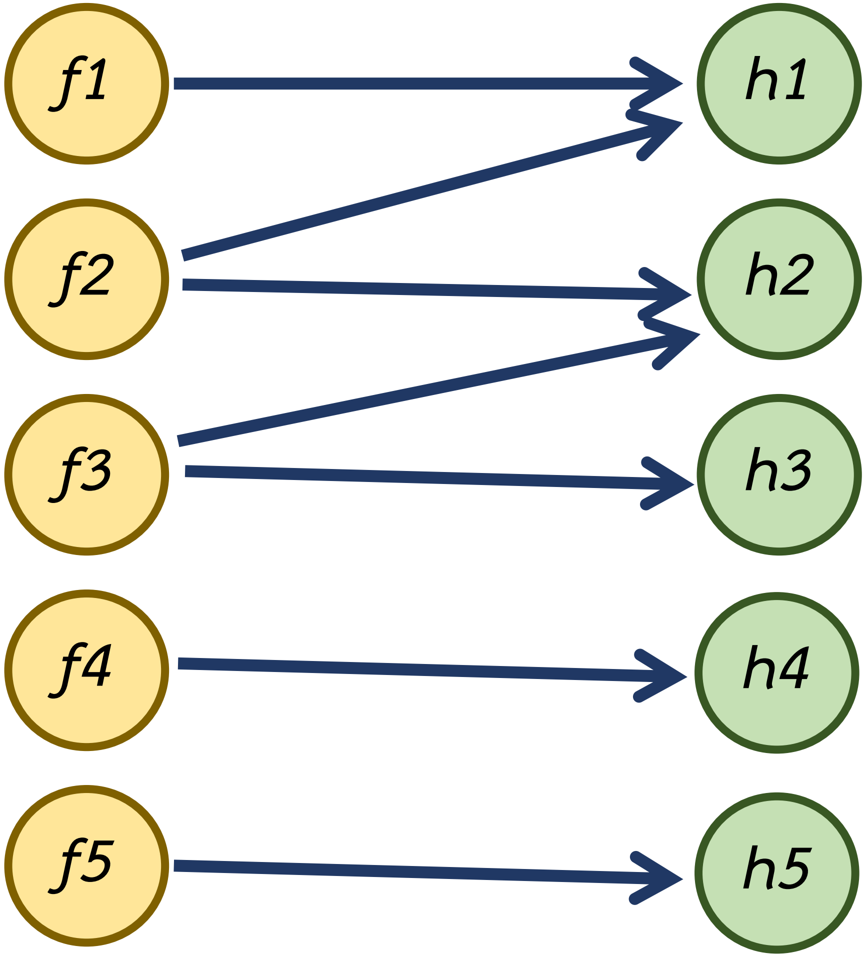
Ref:

Other Centrality

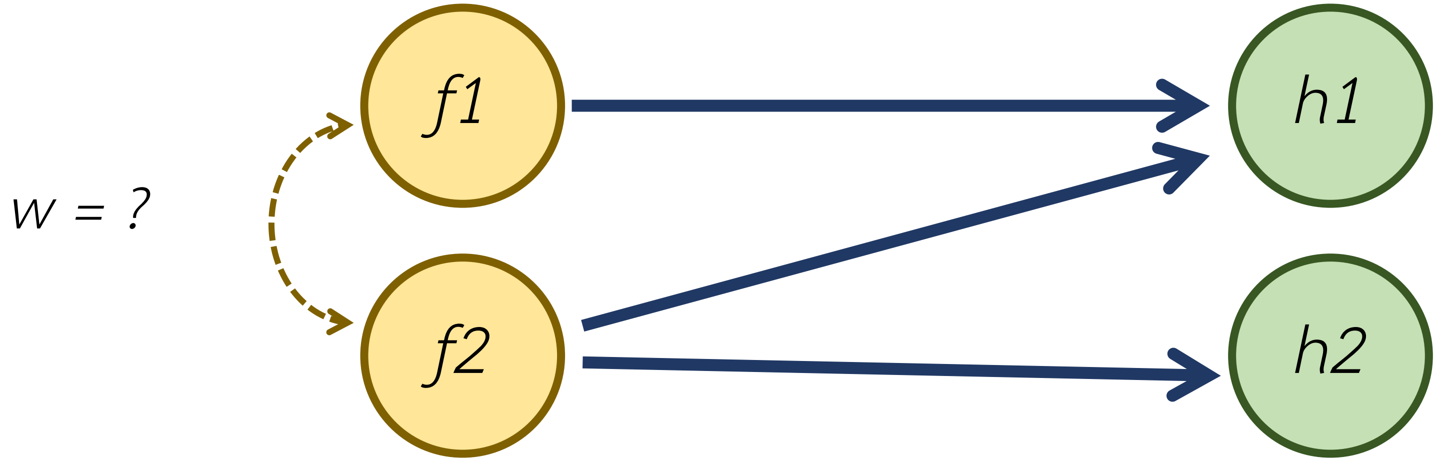
- Degree centrality
- Closeness Centrality
- Harmonic centrality
- Betweenness centrality
- Eigenvector centrality
- Katz centrality
- PageRank centrality
- Percolation centrality
- Cross-clique centrality
- Freeman Centralization

Ref: • Graph Theory

Similarity



Similarity



Jaccard Index

$$w(f1, f2) = \frac{|\Gamma(f1) \cap \Gamma(f2)|}{|\Gamma(f1) \cup \Gamma(f2)|} = \frac{|\{h1, h2\} \cap \{h2\}|}{|\{h1, h2\} \cup \{h2\}|} = \frac{|\{h2\}|}{|\{h1, h2\}|}$$

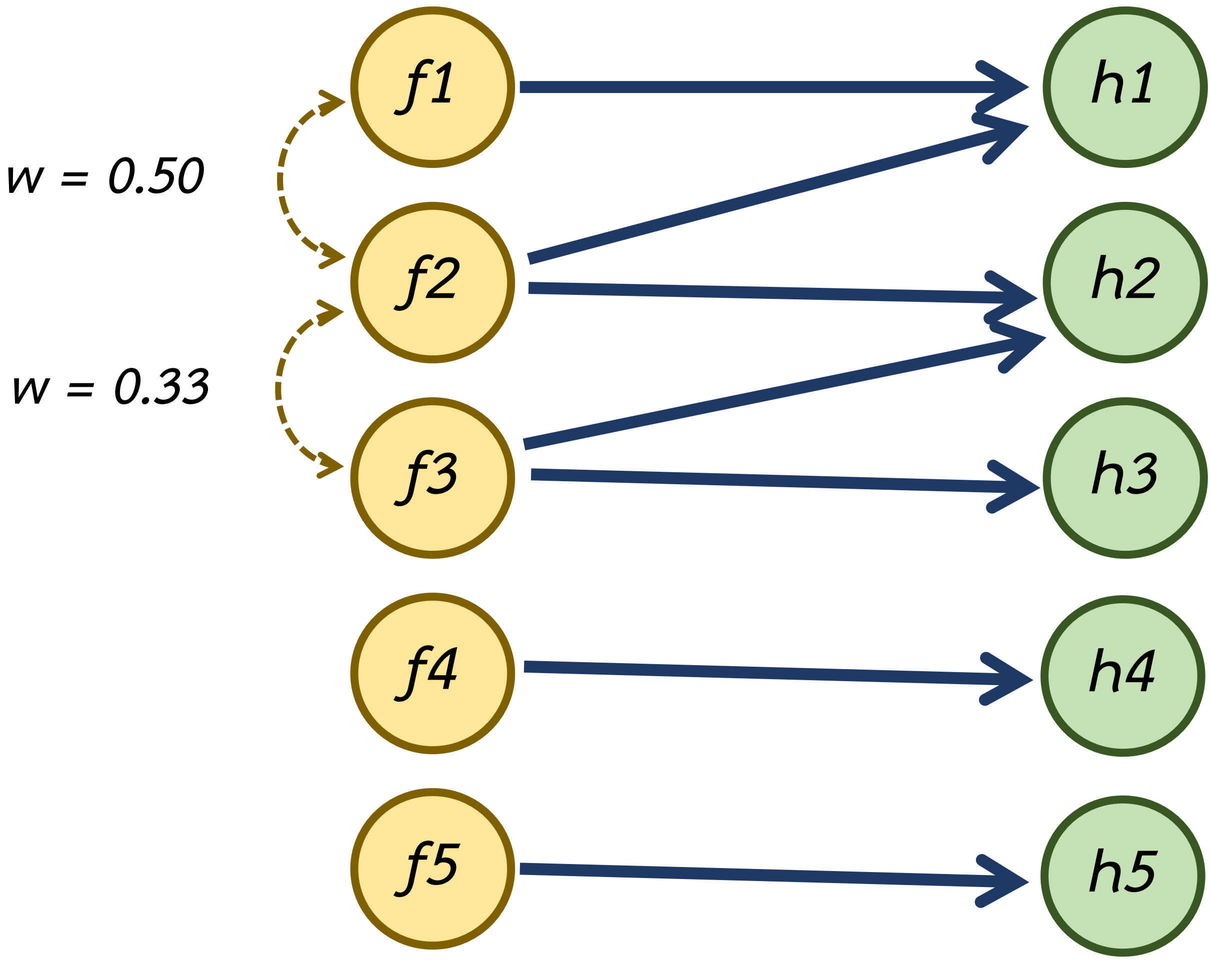
$$= \frac{1}{2} \quad = 0.50$$

Similarity Indices

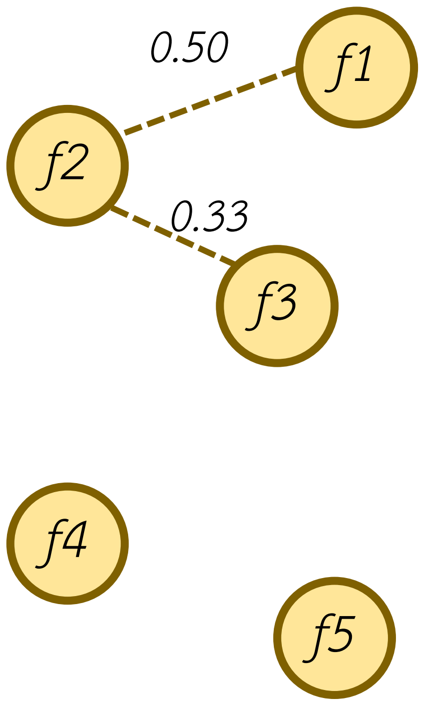
Γ(n) is a function that returns a set of nodes that interact with the node *n*.
Example: $\Gamma(f2) = \{ h1, h2 \}$

- Common Neighbors (CN) : $|\Gamma(x) \cap \Gamma(y)|$
- Jaccard Index: $\frac{|\Gamma(x)\cap\Gamma(y)|}{|\Gamma(x)\cup\Gamma(y)|}$
- Sørensen index: $\frac{|\Gamma(x)\cap\Gamma(y)|}{|\Gamma(x)|+|\Gamma(y)|}$
- Hub Depressed Index (HDI): $\frac{|\Gamma(x)\cap\Gamma(y)|}{\max(\Gamma(x),\Gamma(y))}$
- Resource Allocation Index (RA) : $\sum_{Z\in\Gamma(x)\cap\Gamma(y)} \frac{1}{|z|}$

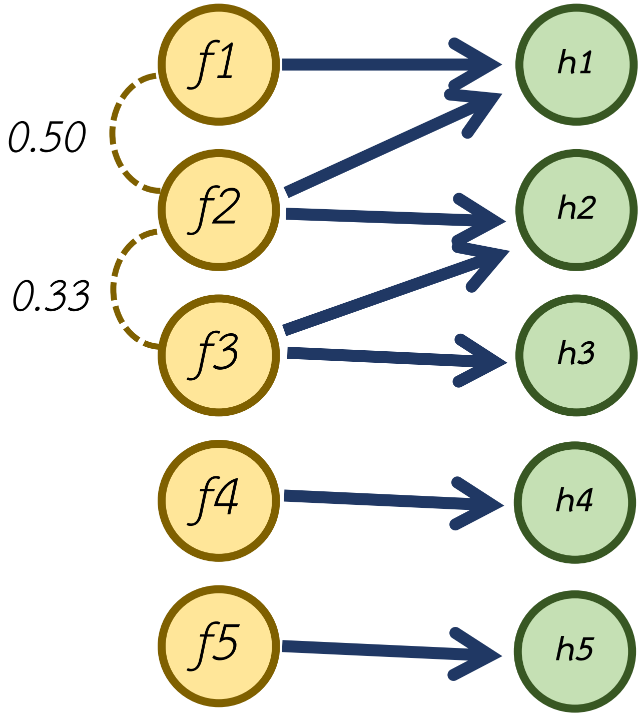
Similarity



Projection



Projection of F



Bipartite Graph

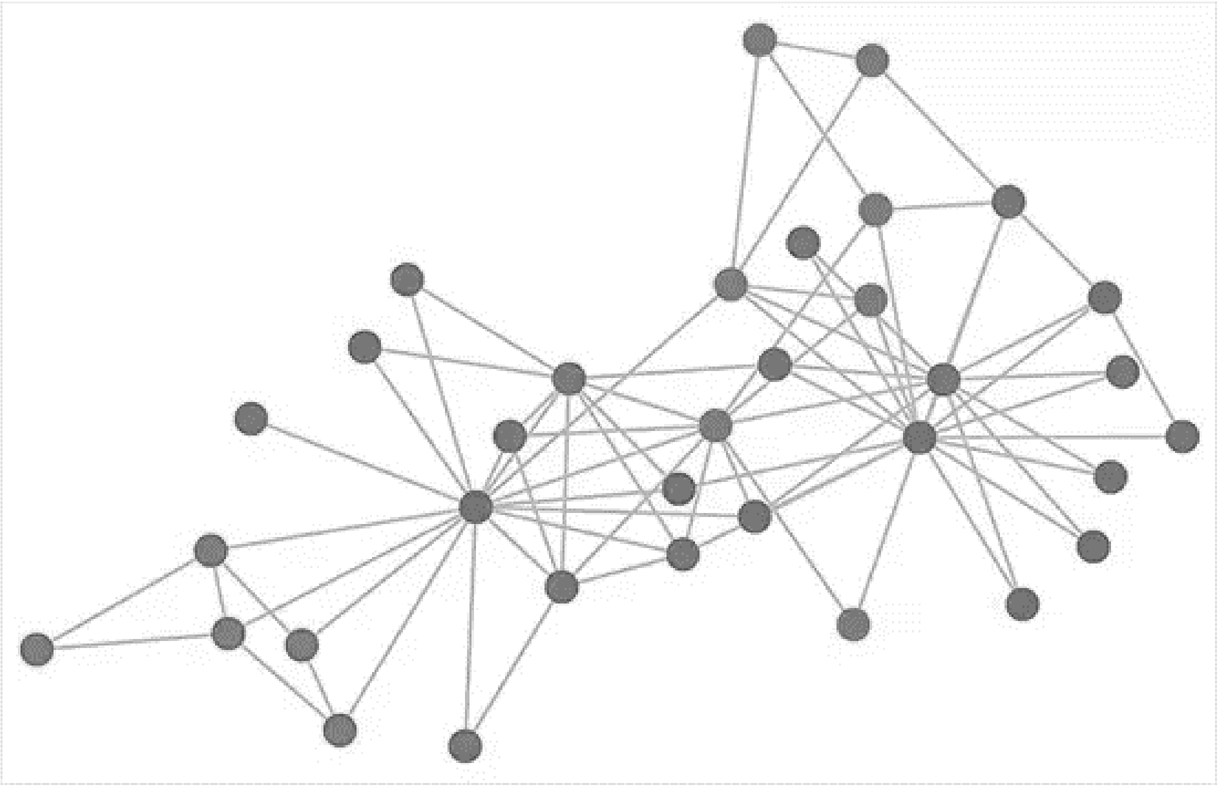
Community Detection

Modularity Maximization

- Modularity is a **measure** of the network structure.
- It was designed to **evaluate the strength of division of a network** into modules (also called groups, clusters or communities).
- Networks with high modularity have **dense connection**s between the nodes within modules but sparse connections between nodes in different modules.
- Modularity is often used in **optimization methods** for detecting community structure in networks
- A simple calculation is good for unweighted and undirected graphs

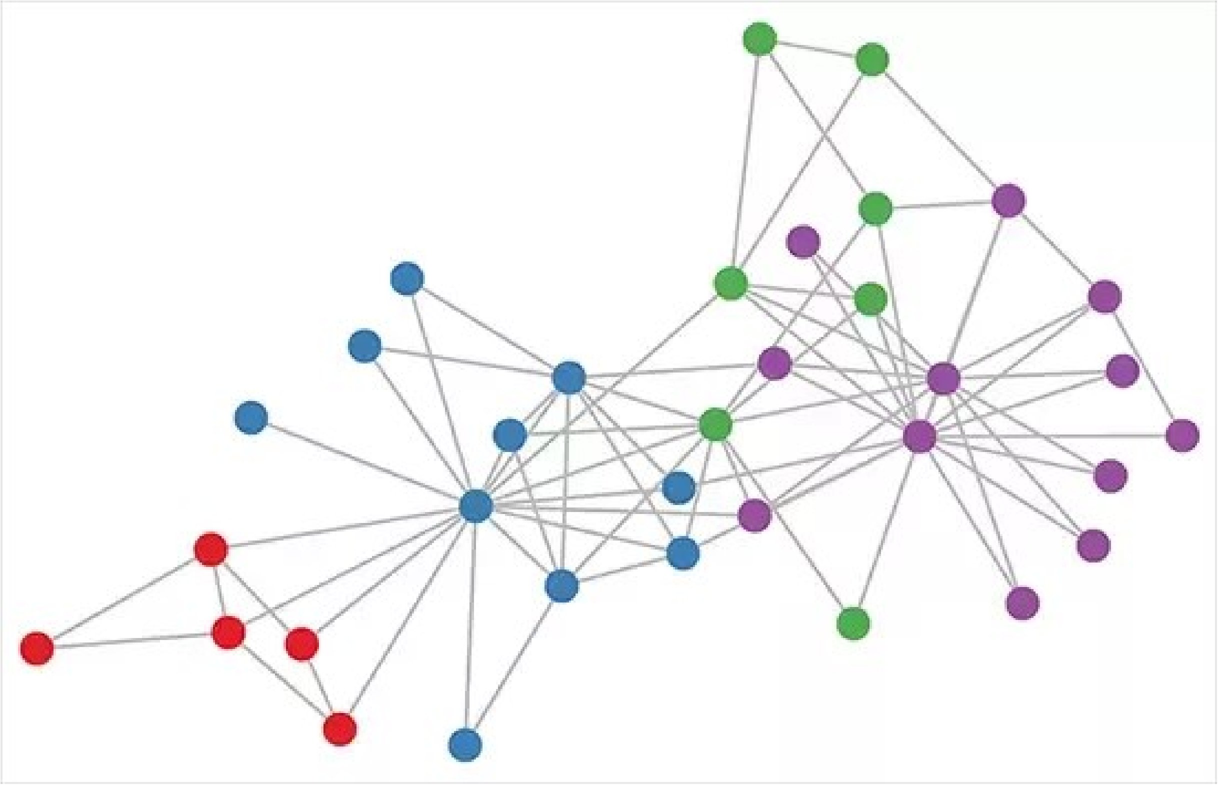
Ref:
- (content) URL
- (image) URL

Community Structure in SNA



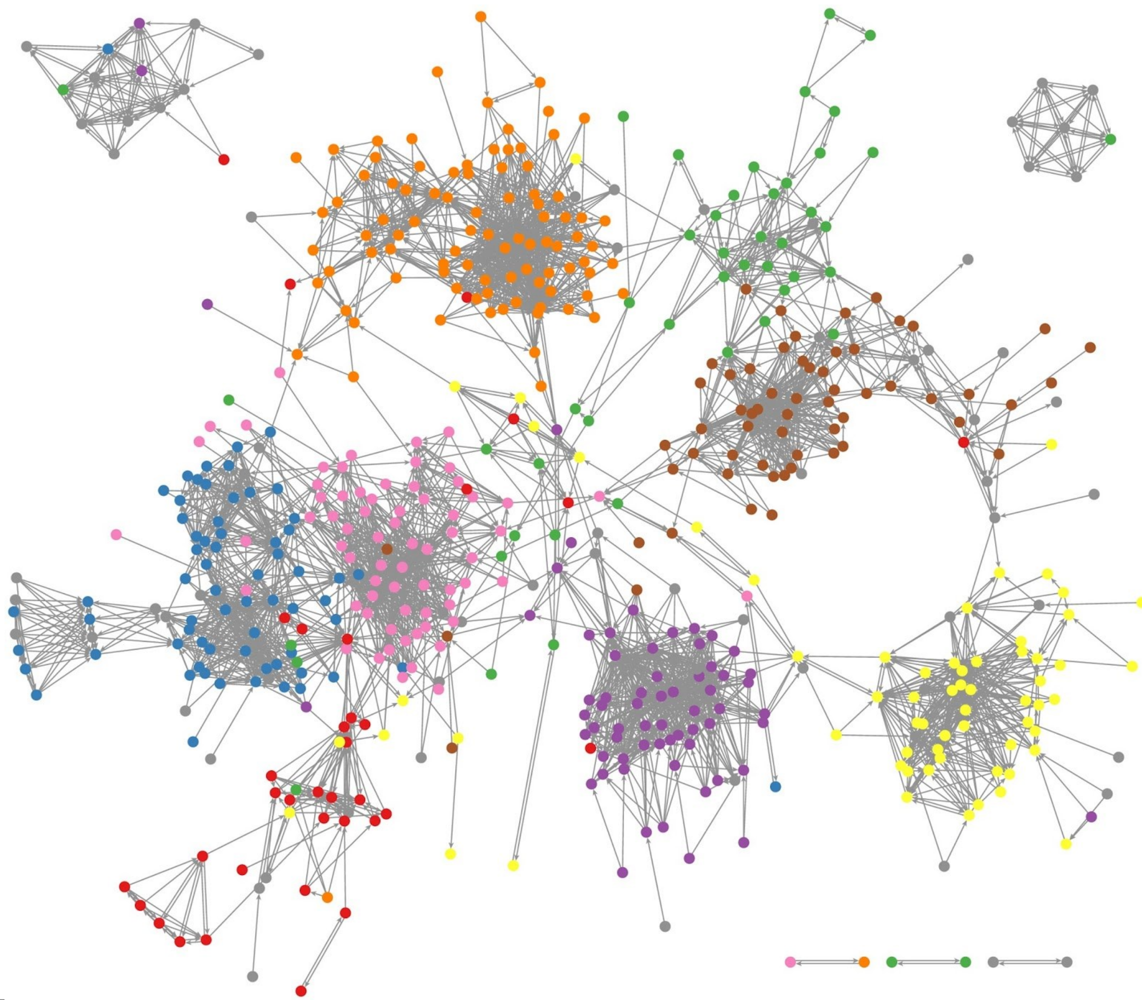
Ref: • (image) https://www.quora.com/What-are-message-passing-neural-networks

Community Structure in SNA



Ref: • (image) https://www.quora.com/What-are-message-passing-neural-networks

Community Structure in SNA



Ref: • (image) http://sociograph.blogspot.com/2013/05/revealing-community-structure-with.html

Equation

$$Q = \frac{1}{2m} \sum_{i,j \in C} \left(A_{i,j} - \frac{d_i d_j}{2m} \right)$$

Equation

$$Q = \frac{1}{2m} \sum_{i,j \in C} \left(A_{i,j} - \frac{d_i d_j}{2m} \right)$$

Every pairs of nodes i and j being in the same community

Equation

$$Q = \frac{1}{2m} \sum_{i,j \in C} \left(A_{i,j} - \frac{d_i d_j}{2m} \right)$$

m is the number of links

Equation

$$Q = \frac{1}{2m} \sum_{i,j \in C} \left(A_{i,j} - \frac{d_i d_j}{2m} \right)$$

Adjacency matrix

If having link, $A_{i,j} = 1$

If having no link, $A_{i,j} = 0$

Equation

$$Q = \frac{1}{2m} \sum_{i,j \in C} \left(A_{i,j} - \frac{d_i d_j}{2m} \right)$$

probability a random edge would go between i and j

Equation

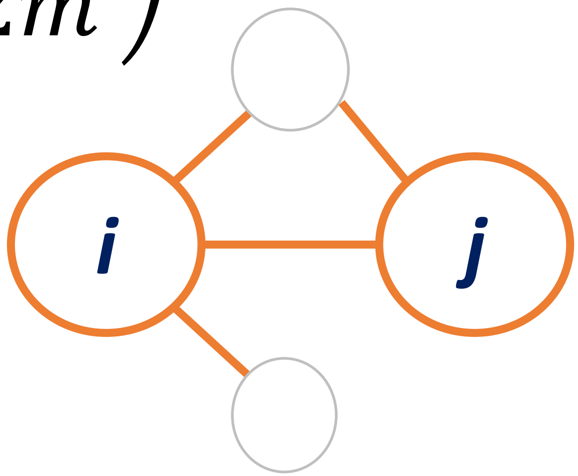
d is a degree of a node === number of edges connected

$$Q = \frac{1}{2m} \sum_{i,j \in C} \left(A_{i,j} - \frac{d_i d_j}{2m} \right)$$

In this case,

$d_i = 3$

$d_j = 2$

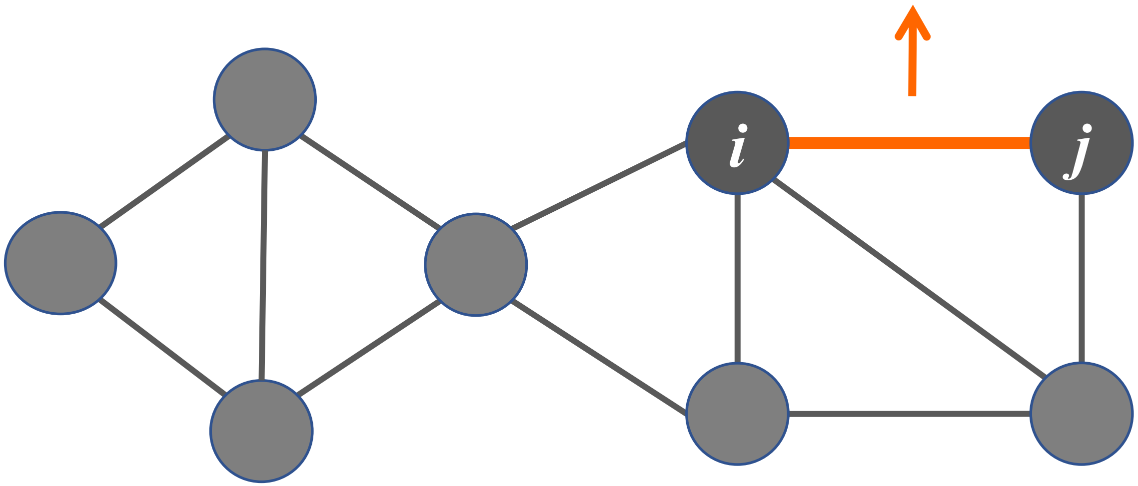


How To Calculate

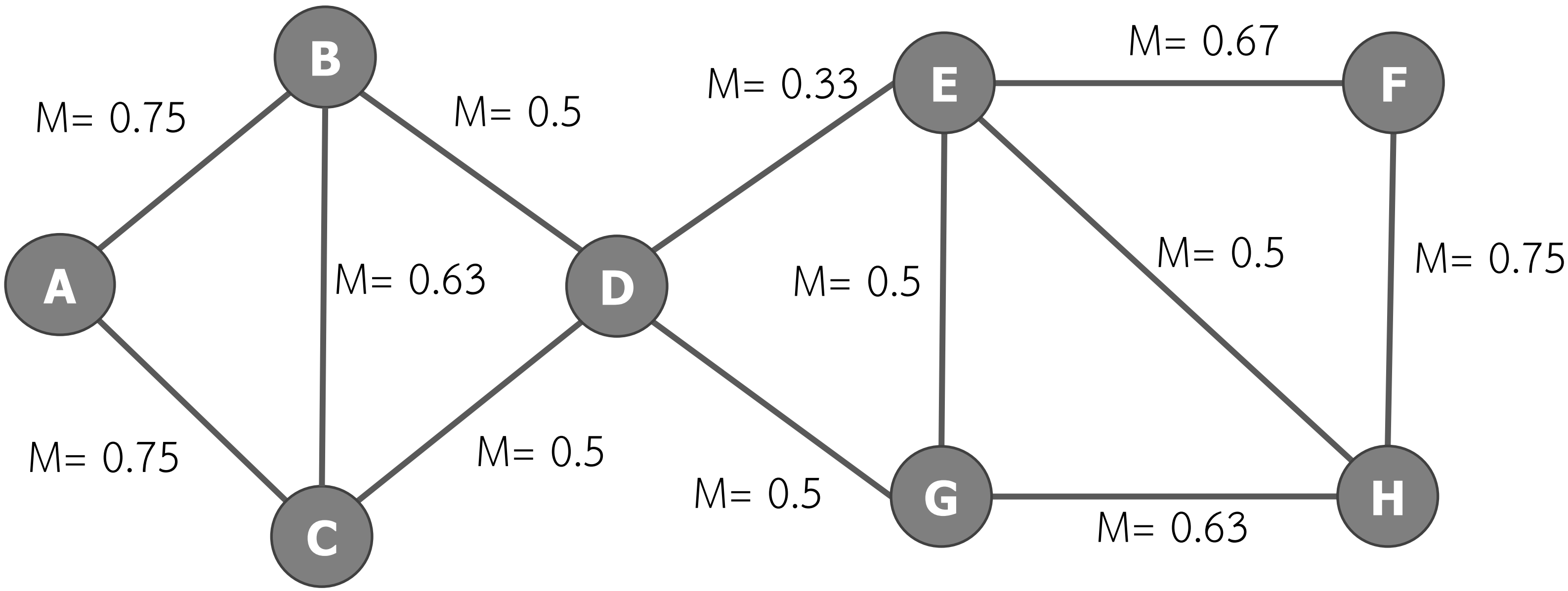
$$A_{i,j} - \frac{d_i d_j}{2m} = \boxed{1 - \frac{4 \times 2}{2 \times 12} = 0.67}$$

12 edges

m = 12

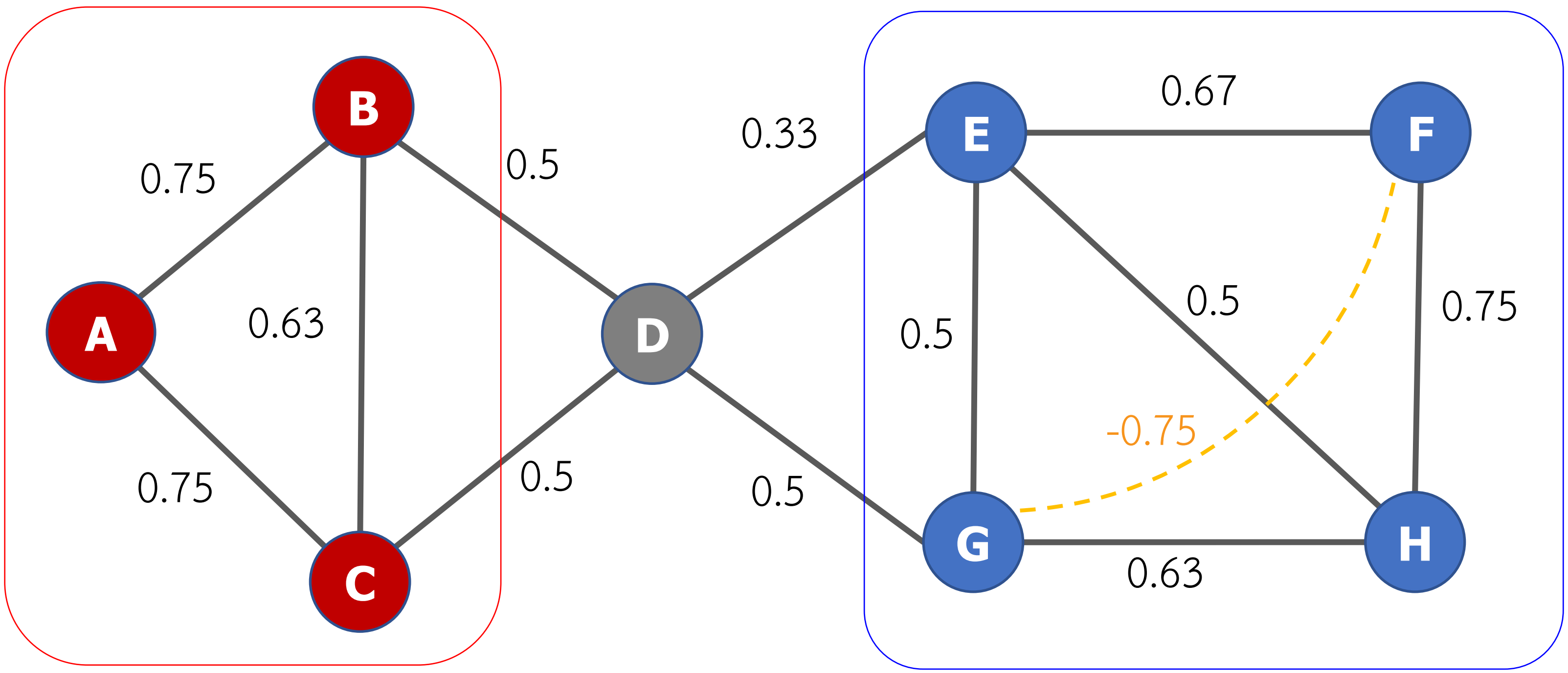


After Calculate



All edges have own Modularity Score

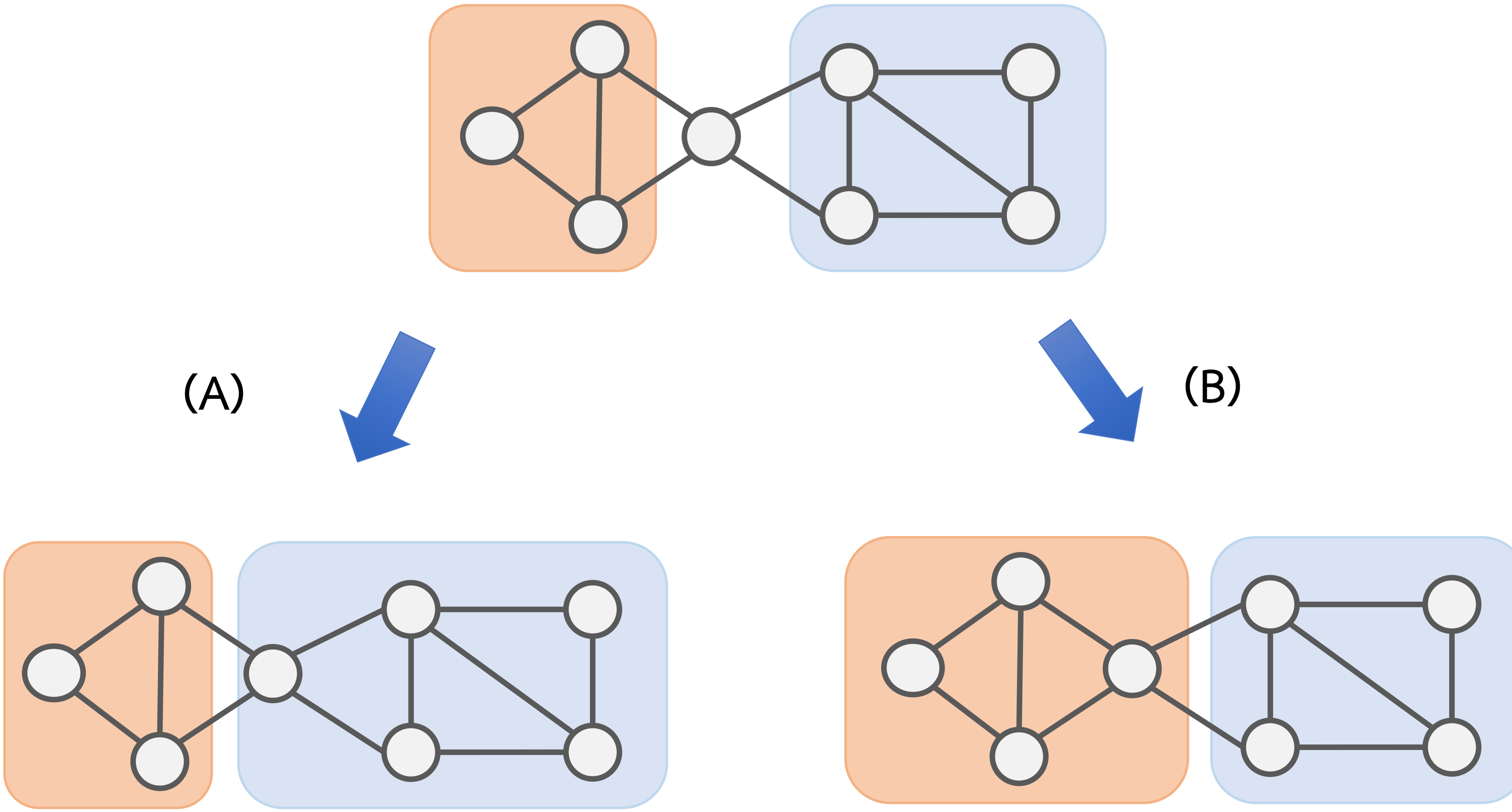
After Calculate



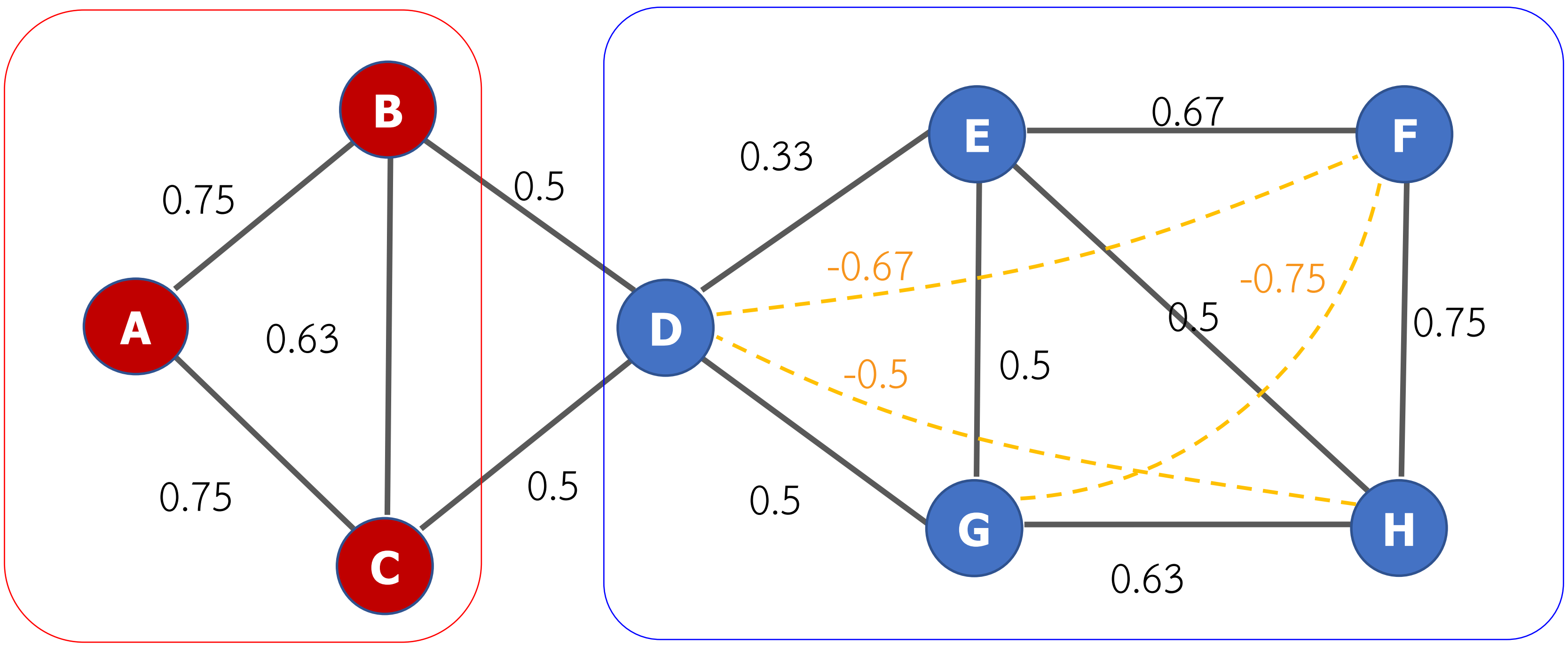
$$Q = ((\,0.75 + 0.75 + 0.63) + (0.5 + 0.5 + 0.63 + 0.67 + 0.75 - 0.75))/2m$$

$$= 0.184$$

Next Iteration



Candidate - A

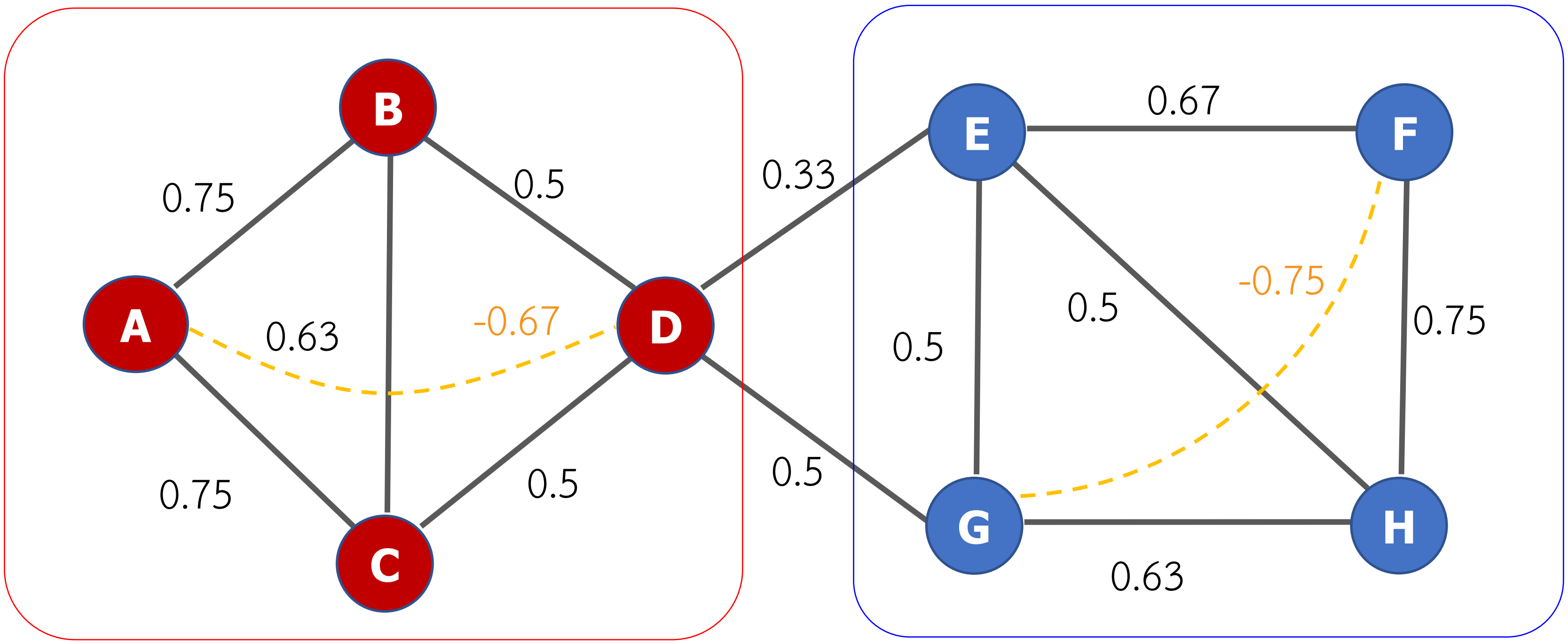


Q = ((0.75 + 0.75 + 0.63) + (0.5 + 0.5 + 0.63 + 0.67 + 0.75 –0.5 -0.67 -0.75))/2m

= 0.135

(A)

Candidate - B



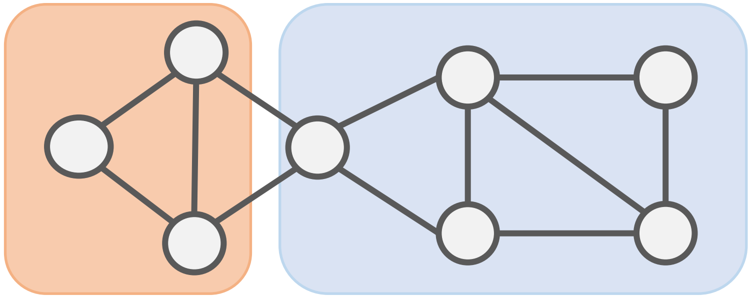
$$Q = ((0.75 + 0.75 + 0.63 - 0.67) + (0.5 + 0.5 + 0.63 + 0.67 + 0.75 - 0.75))/2m$$

$$= 0.156$$

(B)

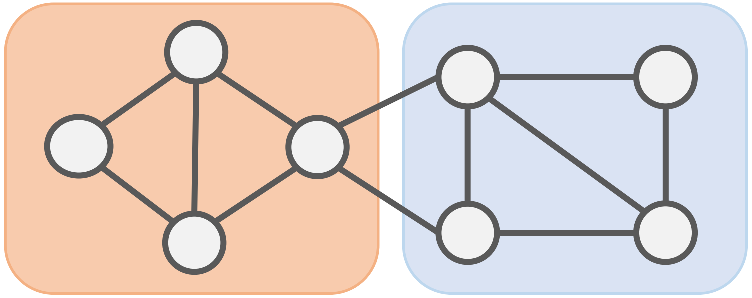
We Choose

(A)



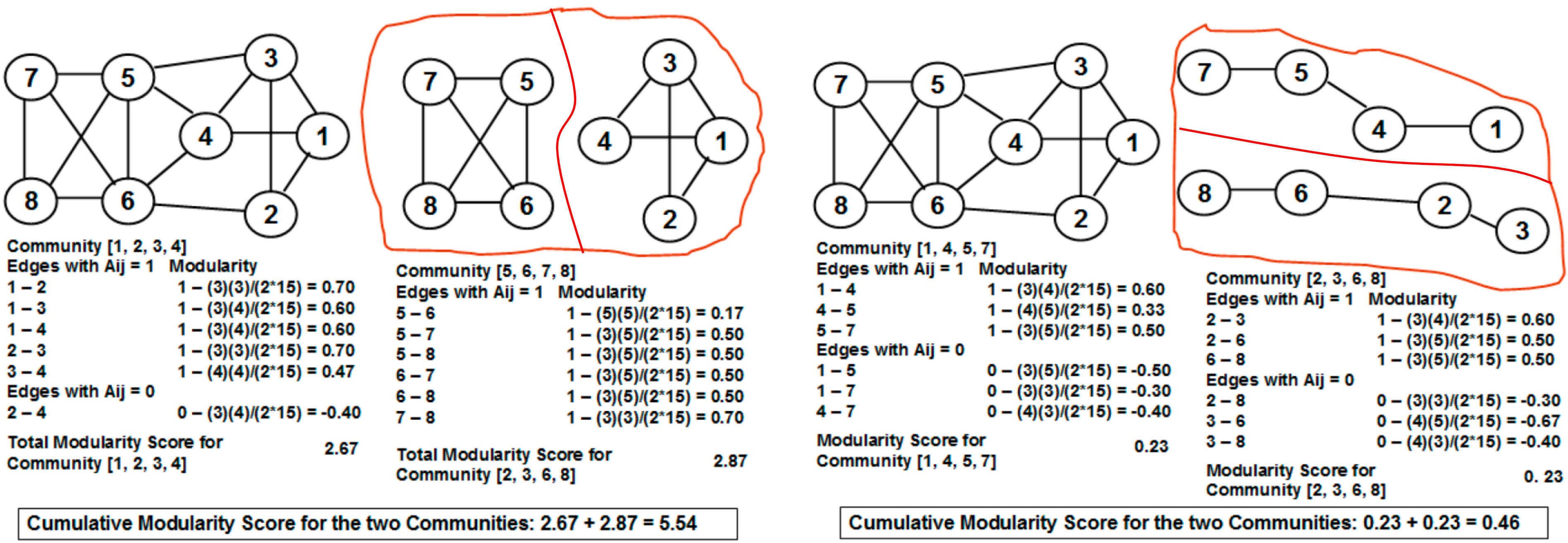
$Q = 0.135$

(B)



$Q = 0.156$

Other Example



(a) (b)

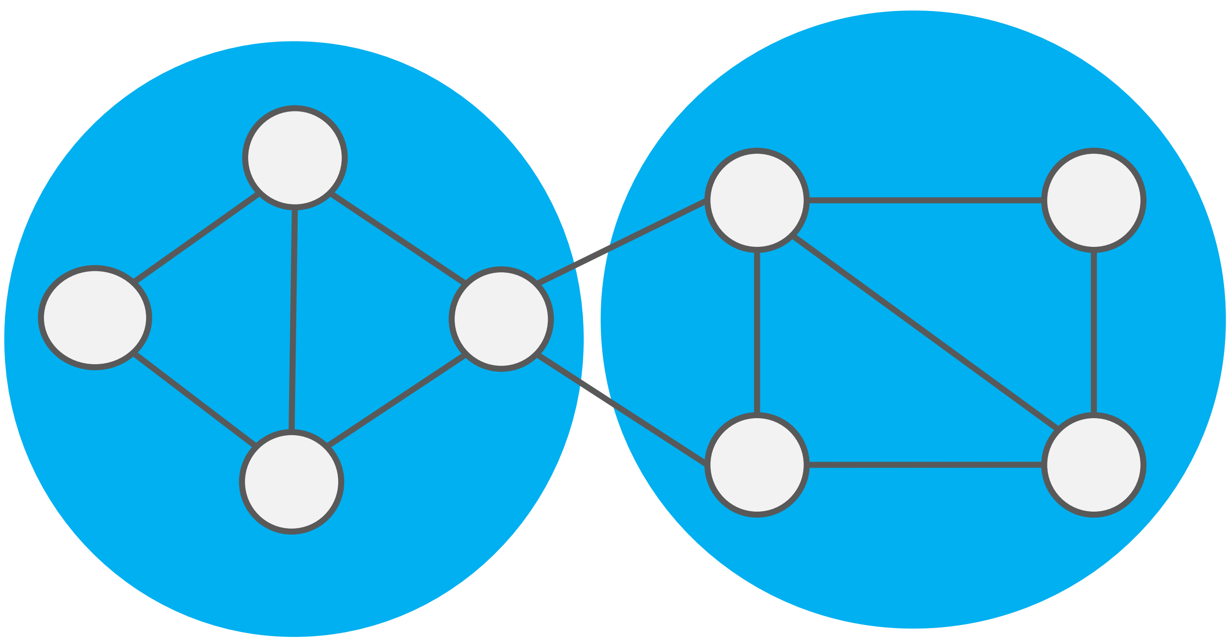
Q= 5.54/2m Q= 0.44/2m

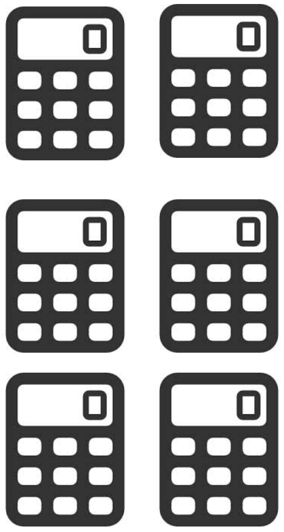
Ref: • N. Meghanathan: “A Greedy Algorithm for Neighborhood Overlap-Based Community Detection”

Walk Trap

Random Walk

- Calculate **Whole Modularity** in every iteration





Every once in a while, a new technology, an old problem, and a big idea turn into an innovation.

Dean Kamen

つづく