Week 11: Graph Mining (Random Graph & Motif Analysis)

Instructor: Daejin Choi (djchoi@inu.ac.kr)

INCHEON NATIONAL UNIVERSITY

* Most slides are used from mmds.org (CS224W) with small updates

Plan for Today

- Generating random graph
  - Erdos-Renyi → Cannot mimic the degree distributions!
  - Small world

- Motif, subgraph, graphlet analysis

- Finding motif, graphlets in graph
  - Enumerating → ESU-Tree
  - Counting → Mckay's nauty algorithm

Generating Random Graph (Erdos-Renyi)

Simple Algorithm to Generate a Random Graph

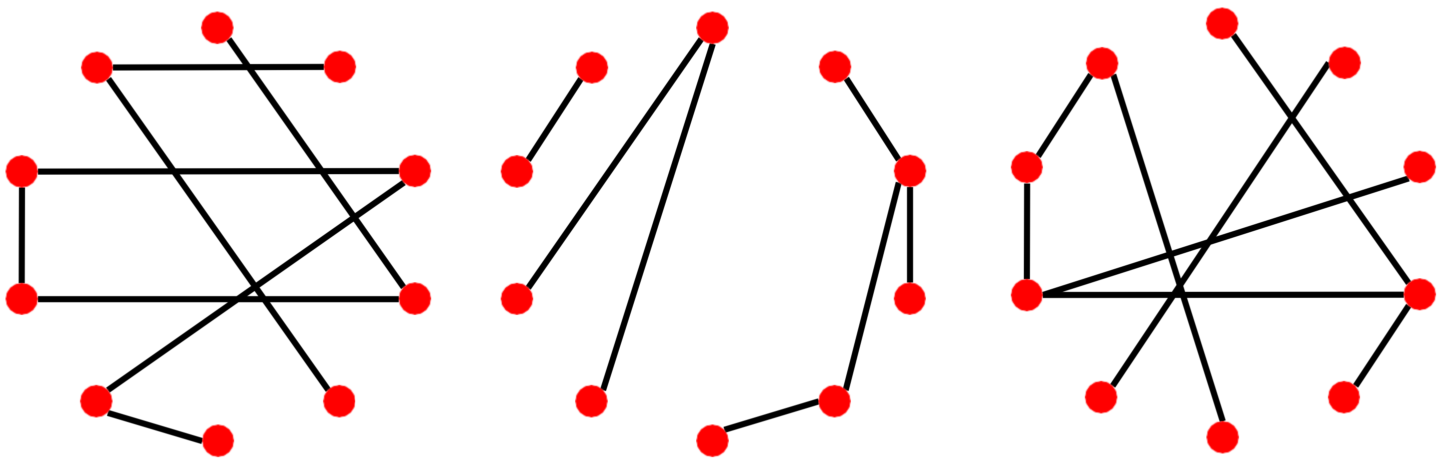
- A graph "uniformly" create or select links

- Two variants
  - G_{np} : undirected graph on n nodes where each edge (u, v) appears i.i.d. with probability p
  - G_{nm} : undirected graph with n nodes, and m edges picked uniformly at random

What kind of networks do such models produce?

Random Graph Model

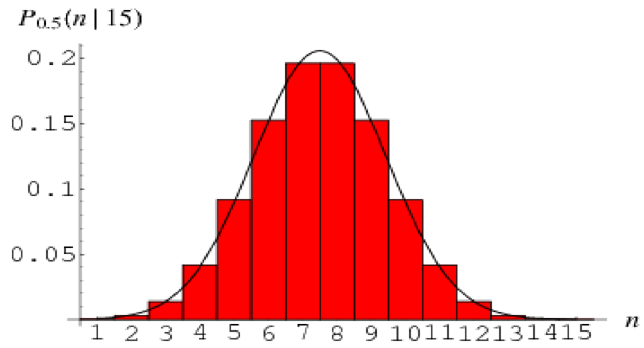
- *n* and *p* do not uniquely determine the graph!
  - The graph is a result of a random process
  - We can have many different realizations given the same ***n*** and ***p***



n = 10
p= 1/6

Properties of G_{np}

Degree distribution:



Avg. path length: $O(\log n)$

Avg. clustering coef.: $\bar{k}\,/\,n$

Largest Conn. Comp.: GCC exists when k>1.

Degree Distribution

- Degree distribution of G_{np} is binomial.

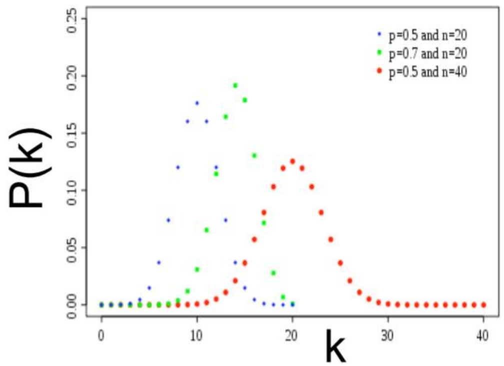
- Let $P(k)$ denote the fraction of nodes with degree k

$$P(k) = \binom{n-1}{k} p^k (1-p)^{n-1-k}$$

Select *k* nodes out of *n-1*

Probability of having *k* edges

Probability of missing the rest of the *n-1-k* edges



Mean, variance of a binomial distribution

$$\bar{k} = p(n-1)$$

$$\sigma^2 = p(1-p)(n-1)$$

$$\frac{\sigma}{\bar{k}} = \left[\frac{1-p}{p}\frac{1}{(n-1)}\right]^{1/2} \approx \frac{1}{(n-1)^{1/2}}$$

By the law of large numbers, as the network size increases, the distribution becomes increasingly narrow—we are increasingly confident that the degree of a node is in the vicinity of *k*.

Clustering Coefficient of G_{np}

- Remember: $$C_i = \frac{2e_i}{k_i(k_i-1)}$$

Where e_i is the number of edges between i's neighbors

- Edges in G_{np} appear i.i.d. with prob. p

- So, expected $E[e_i]$ is $$= p\frac{k_i(k_i-1)}{2}$$

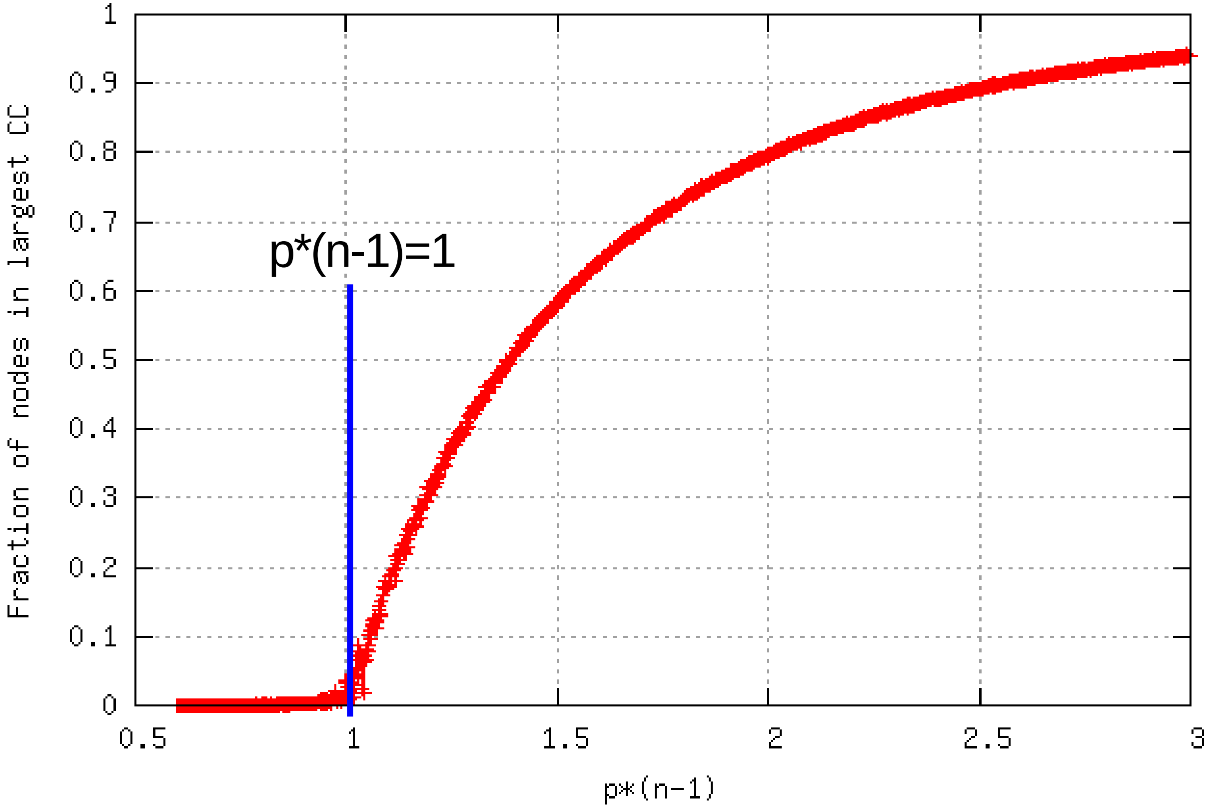
Each pair is connected with prob. p

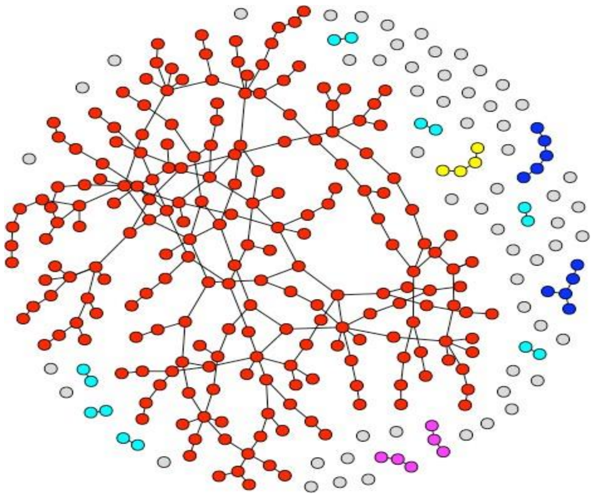
Number of distinct pairs of neighbors of node i of degree k_i

- Then $E[C_i]$: $$\frac{p \cdot k_i(k_i-1)}{k_i(k_i-1)} = p = \frac{\bar{k}}{n-1} \approx \frac{\bar{k}}{n}$$

Clustering coefficient of a random graph is small.
If we generate bigger and bigger graphs with fixed avg. degree k (that is we set $p = k \cdot 1/n$), then C decreases with the graph size n.

Simulation to Find GCC

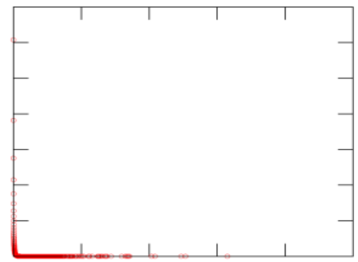
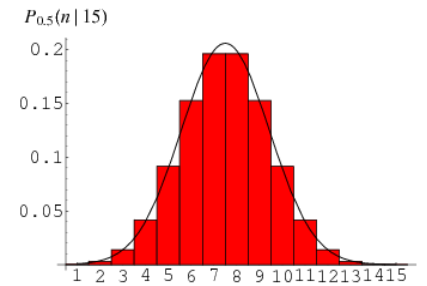




Fraction of nodes in the largest component

- G_{np}, n=100,000, $k=p(n-1)$ = 0.5 … 3

Let's back! MSN vs. Random Graph

	MSN	G_{np}	
Degree distribution:			✗
Avg. path length:	6.6	$O(\log n)$	✓
Avg. clustering coef.:	0.11	$\bar{k} / n$	✗
Largest Conn. Comp.:	99%	GCC exists when k>1.	✓

Real Network vs. G_{np}

- Are real networks like random graphs?
  - Giant connected component: ☺
  - Average path length: ☺
  - Clustering Coefficient: ☹
  - Degree Distribution: ☹

- Problems with the random networks model:
  - Degree distribution differs from that of real networks
  - No local structure – clustering coefficient is too low

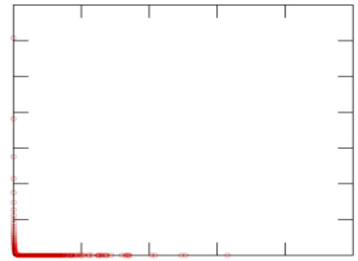
Real Network vs. G_{np}

- If G_{np} is wrong, why did we spend time on it?
  - It will help us calculate many quantities, that can then be compared to the real data
  - It will help us understand to what degree a particular property is the result of some random process

So, while G_{np} is **WRONG**, it will turn out to be extremely **USEFUL**!

The Small-World Model

Random Graph G_{np} Does NOT Reflect Real-World

	MSN	G_{np}
Degree distribution:		$\bar{k} / n$
Avg. clustering coef.:	0.11	$\bar{k} / n$

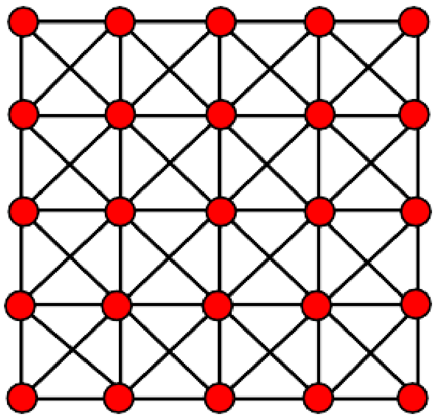
- Other examples

Network	h_{actual}	h_{random}	C_{actual}	C_{random}
Film actors	3.65	2.99	0.79	0.00027
Power Grid	18.70	12.40	0.080	0.005
C. elegans	2.65	2.25	0.28	0.05

h ... Average shortest path length
C ... Average clustering coefficient
“actual” … real network
“random” … random graph with same avg. degree

The Problem Comes from "Edge Locality"

- The major difference is that real-world network has "local structure"
  - **Triadic closure**: Friend of a friend is my friend



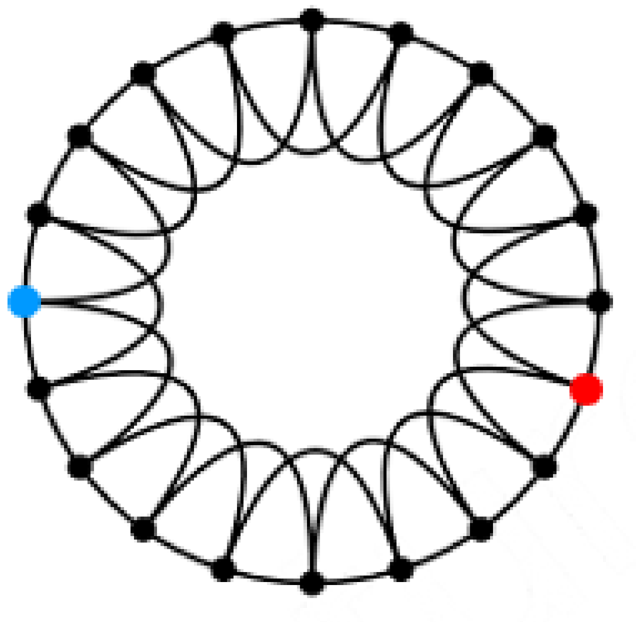
High clustering coefficient

BUT, High diameter (Not Log(n))

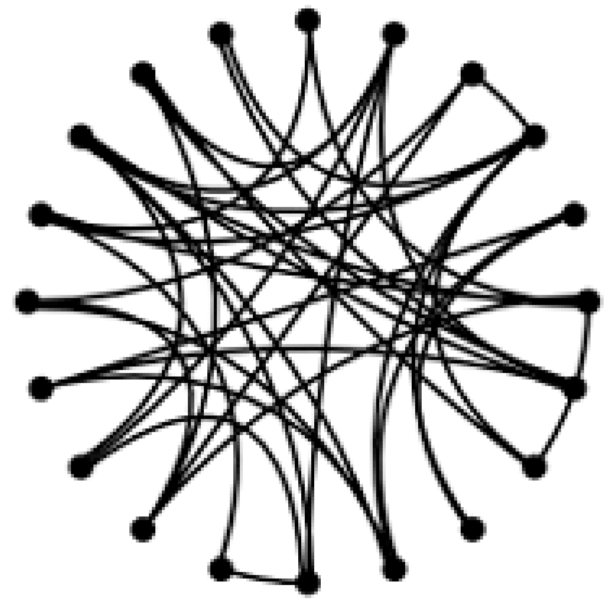
  - "Simply adding edges" (i.e., uniform triad) does not work

- How can we create a graph with high CC & low Diameter?

Creating the Graph w/ High CC and Low Dia.



High clustering
High diameter



Low clustering
Low diameter

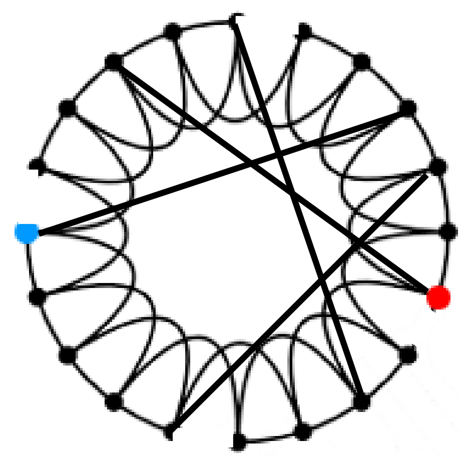
- The point is "how to add edges"
  - Clustering implies edge “locality”
  - Randomness enables “shortcuts”

Solution: The Small World Model

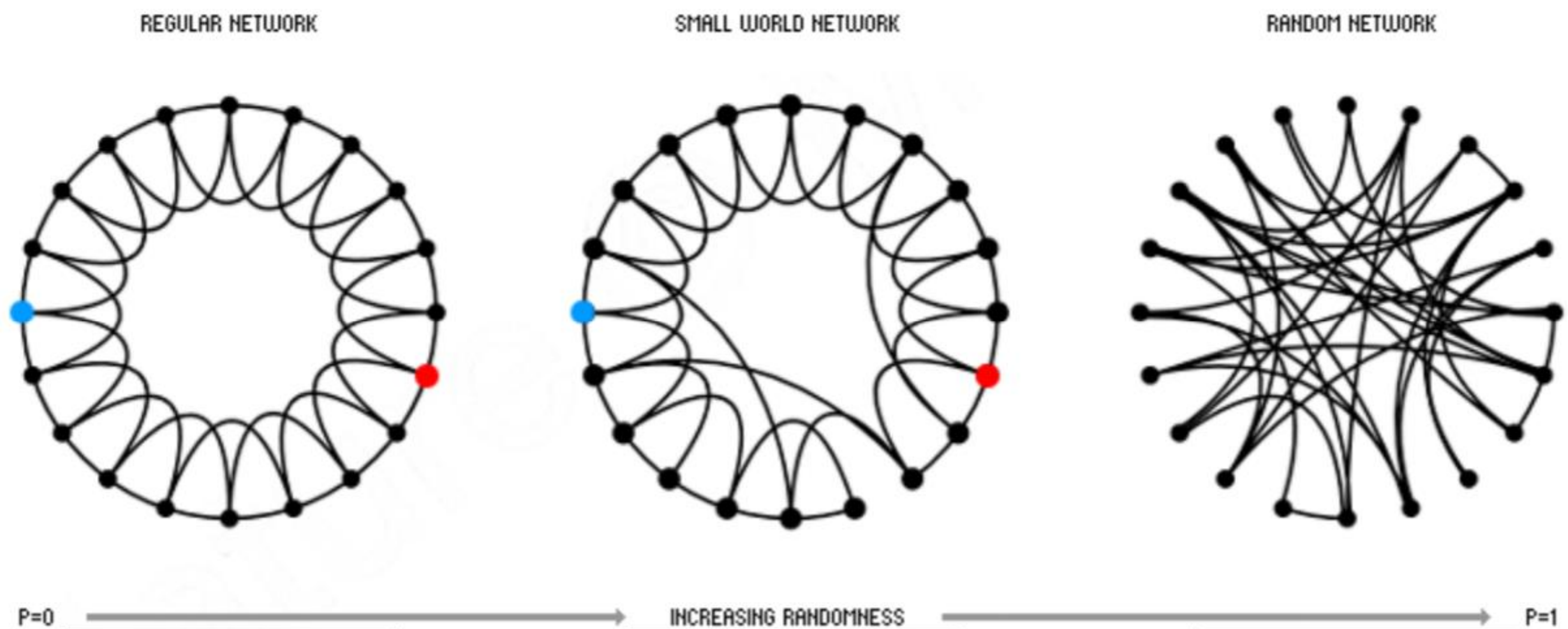
[Watts-Strogatz, '98]

- Start with a **low-dimensional regular lattice**
  - (In our case we are using a ring as a lattice)
  - Has high clustering coefficient

- **Rewire**: **Introduce randomness** ("shortcuts")
  - Add/remove edges to create shortcuts to join remote parts of the lattice
  - For each edge, with prob. p, move the other endpoint to a random node



The Small World Model



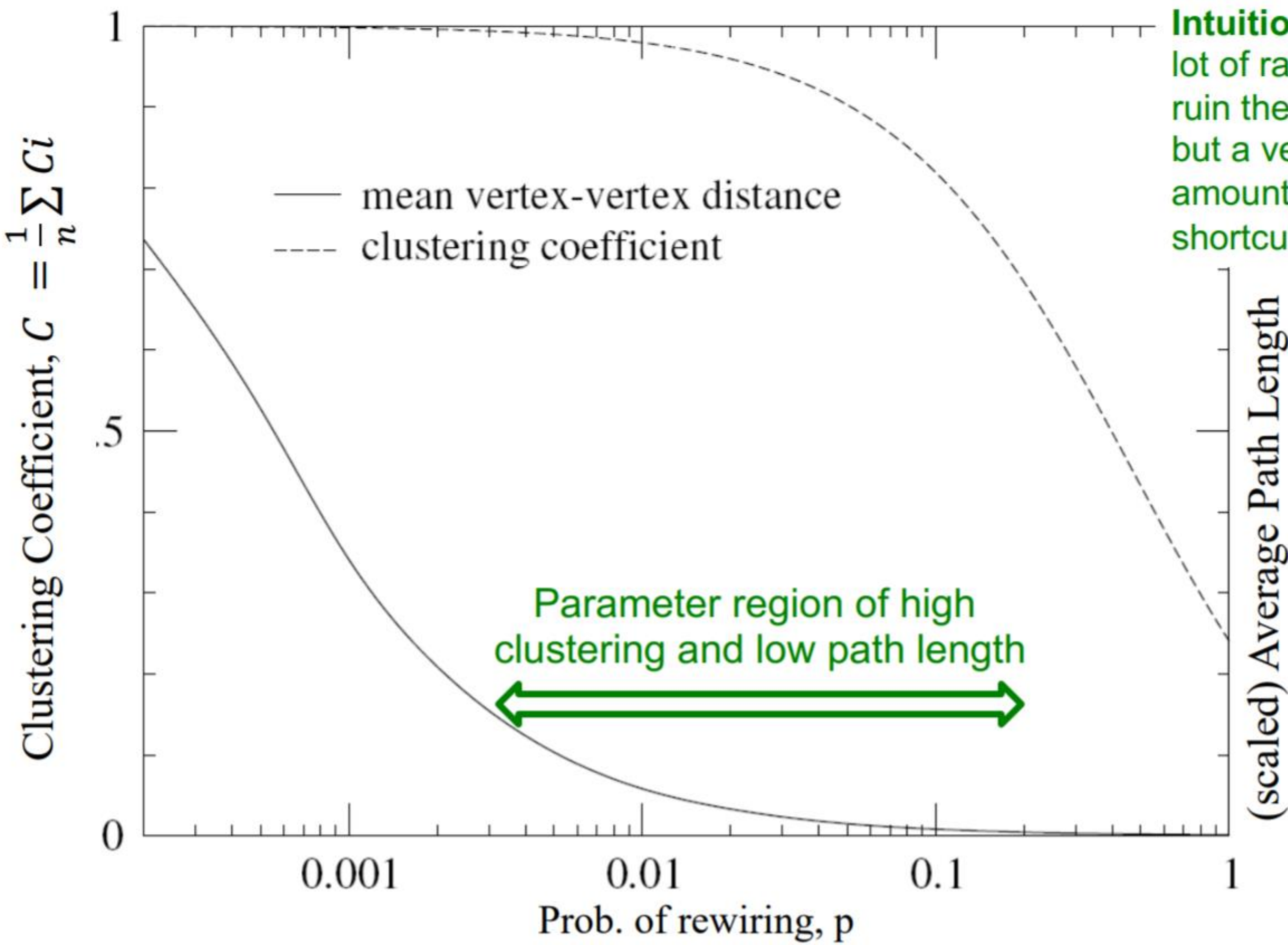
High clustering High diameter	High clustering Low diameter	Low clustering Low diameter

$h = \frac{N}{2\bar{k}}$ $\quad C = \frac{1}{2}$

$h = \frac{\log N}{\log \alpha}$ $\quad C = \frac{\bar{k}}{N}$

Rewiring allows us to “interpolate” between a regular lattice and a random graph

The Small World Model



Intuition: It takes a lot of randomness to ruin the clustering, but a very small amount to create shortcuts.

The Small World: Summary

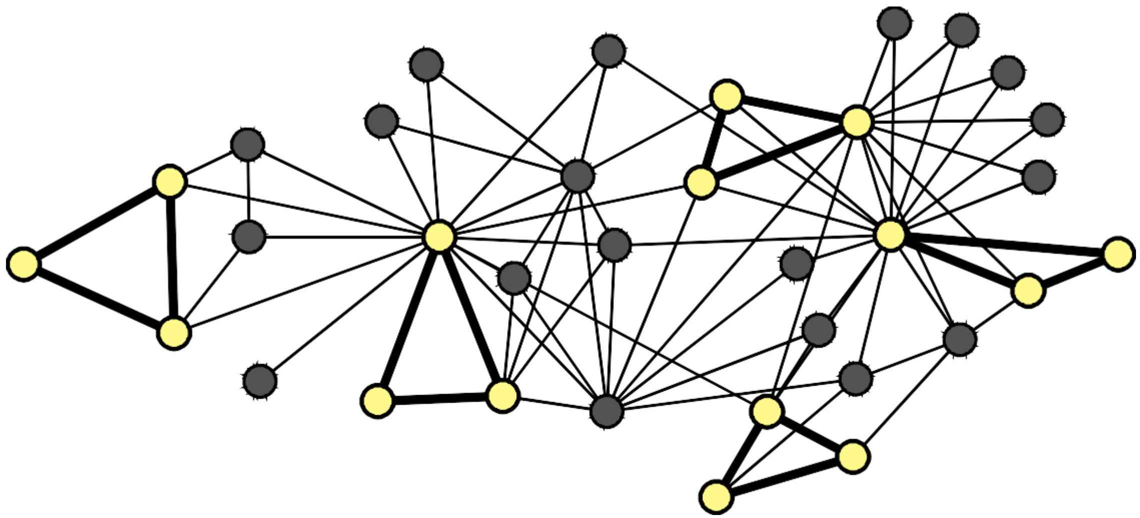
- Could a network with high clustering be at the same time a small world?
  - Yes! You don't need more than a few random links

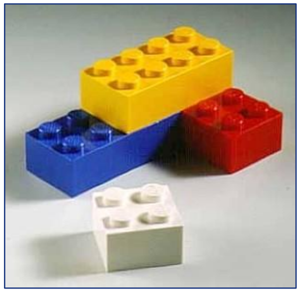
- The Watts Strogatz Model:
  - Provides insight on the interplay between clustering and the small-world
  - Captures the structure of many realistic networks
  - Accounts for the high clustering of real networks
  - Does not lead to the correct degree distribution

Subgraphs, Motifs, and Graphlets

Subnetworks: A Property of Graph

- Subnetworks, or subgraphs, are the building blocks of networks

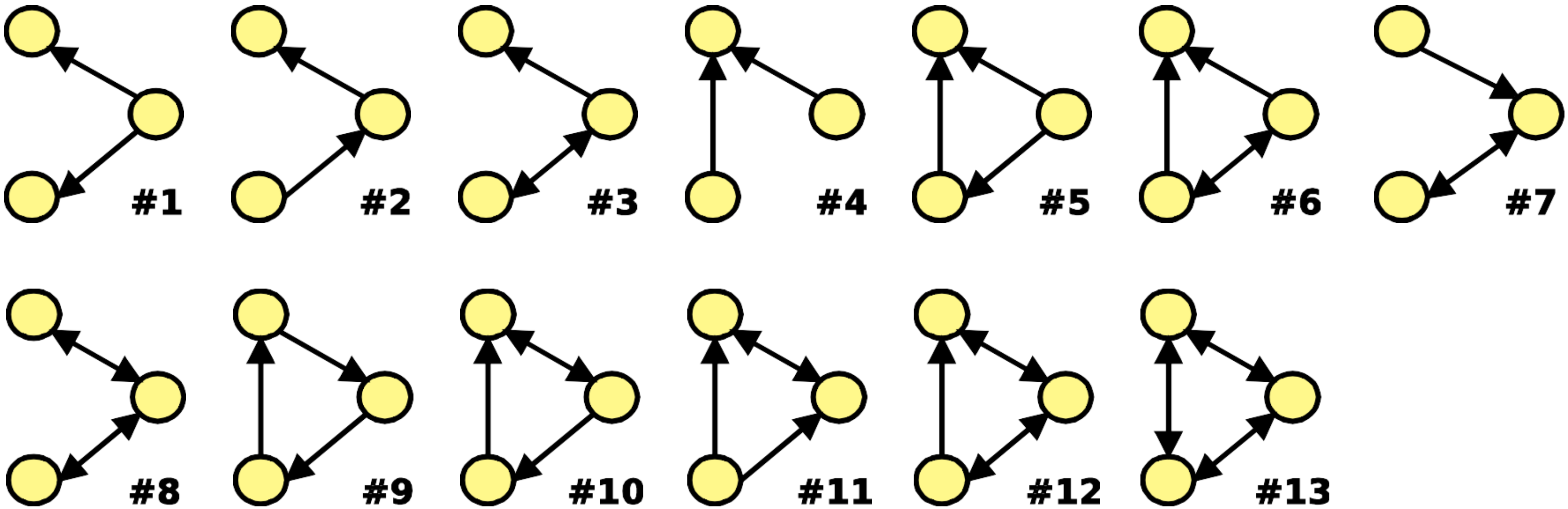




- They have the power to **characterize** and **discriminate** networks

How to Characterize a Graph from Subgraphs?

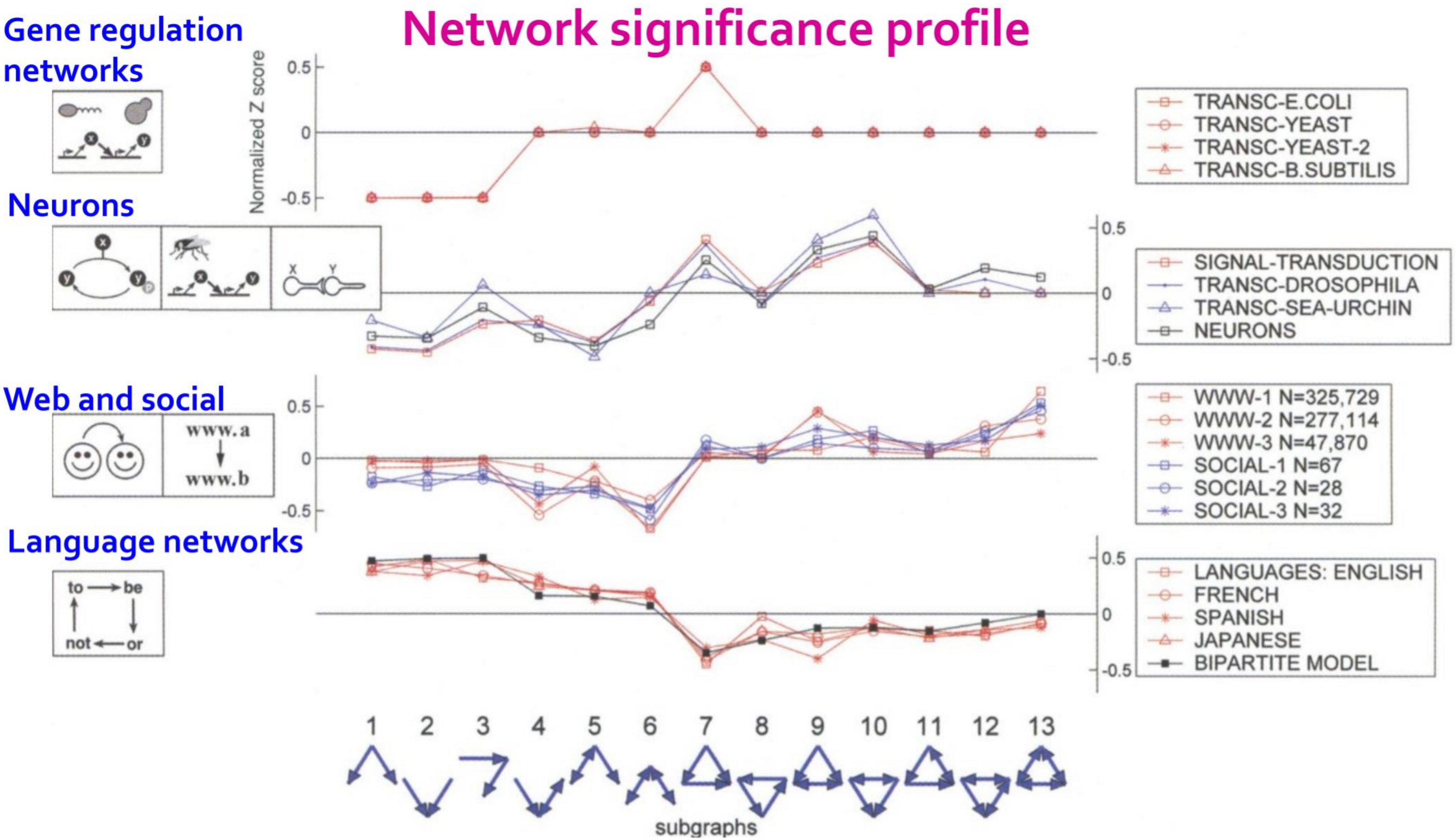
Let's consider all possible (non-isomorphoic) directed subgraphs of size 3



How to Characterize a Graph from Subgraphs?

- For each subgraph:
  - Imagine you have a metric capable of classifying the subgraph "significance" [more on that later]
    - **Negative values** indicate **under-representation**
    - **Positive values** indicate **over-representation**
- We create a **network significance profile**:
  - A feature vector with values for all subgraph types
- Next: Compare profiles of different networks:
  - Regulatory network (gene regulation)
  - Neuronal network (synaptic connections)
  - World Wide Web (hyperlinks between pages)
  - Social network (friendships)
  - Language networks (word adjacency)

Case Example of Subgraphs



Networks from the same domain have similar significance profiles

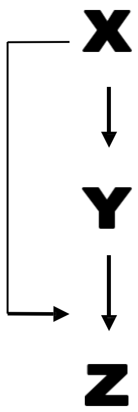
Network Motifs

- Network motifs: "recurring, significant patterns of interconnections"

- How to define a network motif:
  - **Pattern**: Small partial **subgraph**
  - **Recurring**: Found many times, i.e., with high frequency
  - **Significant**: More frequent than expected, i.e., in randomly generated networks
    - Erdos-Renyi random graphs, scale-free networks

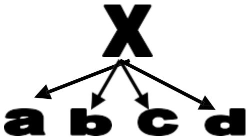
Why Do We Need Motifs?

- Motifs:
  - Help us understand how networks are structured
  - Help us predict operation and reaction of the network in a given situation

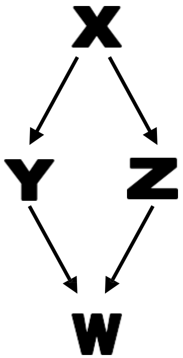


Feed-forward loop

- Examples:
  - Feed-forward loops: found in networks of neurons, where they neutralize "biological noise"
  - Parallel loops: found in **food webs**
  - Single-input modules: found in **gene control** networks



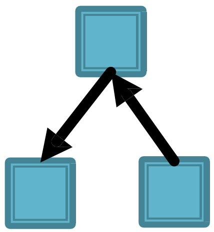
Single-input module

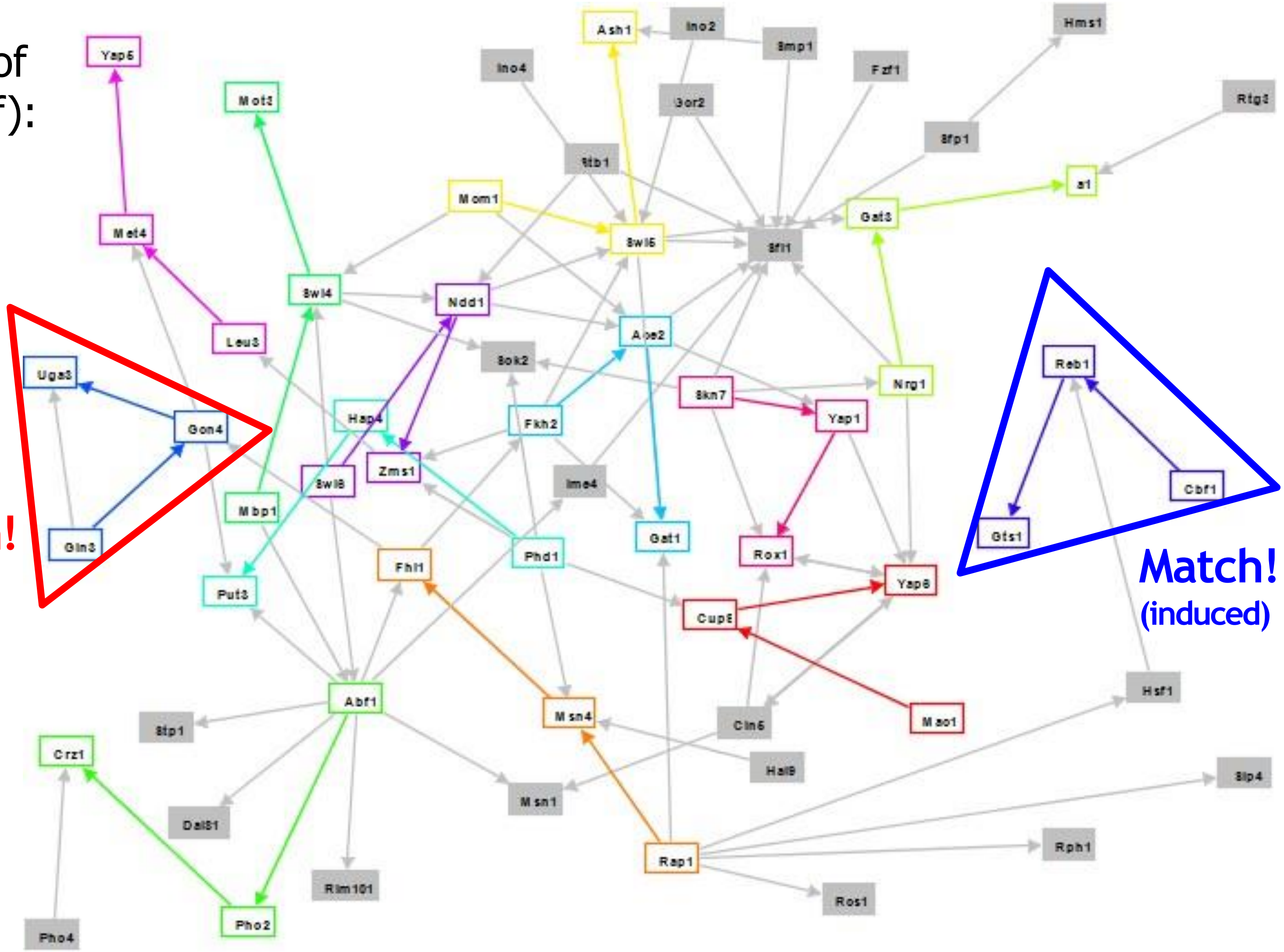


Parallel loop

Motifs: Partial Subgraphs

Partial subgraph of interest (aka Motif):

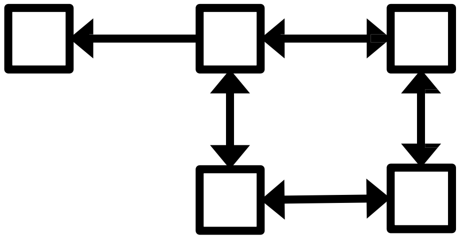




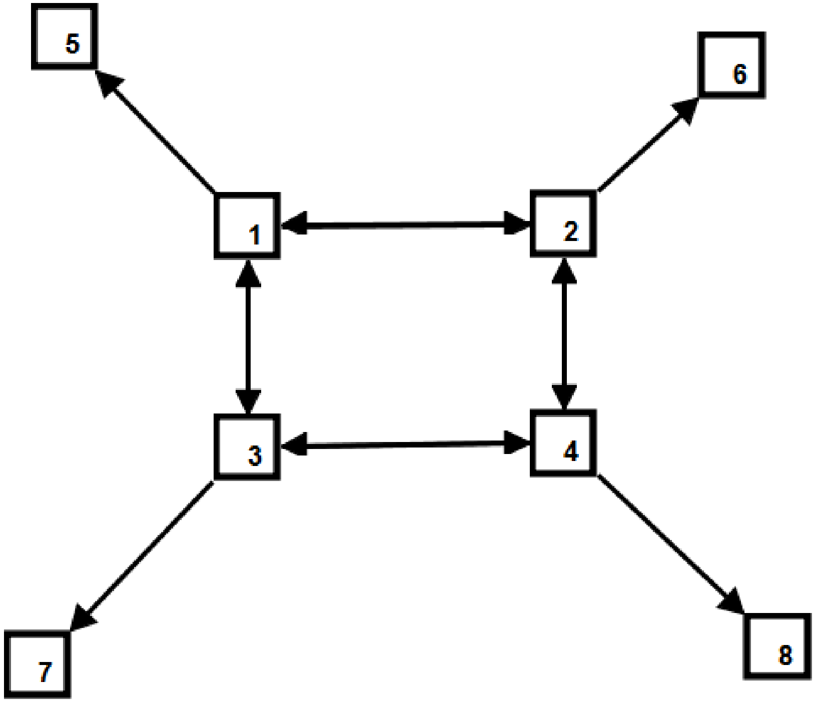
No match!
(not induced)

Induced subgraph of graph G is a graph, formed from a subset X of the vertices of graph G and all of the edges connecting pairs of vertices in subset X.

Motif of interest:

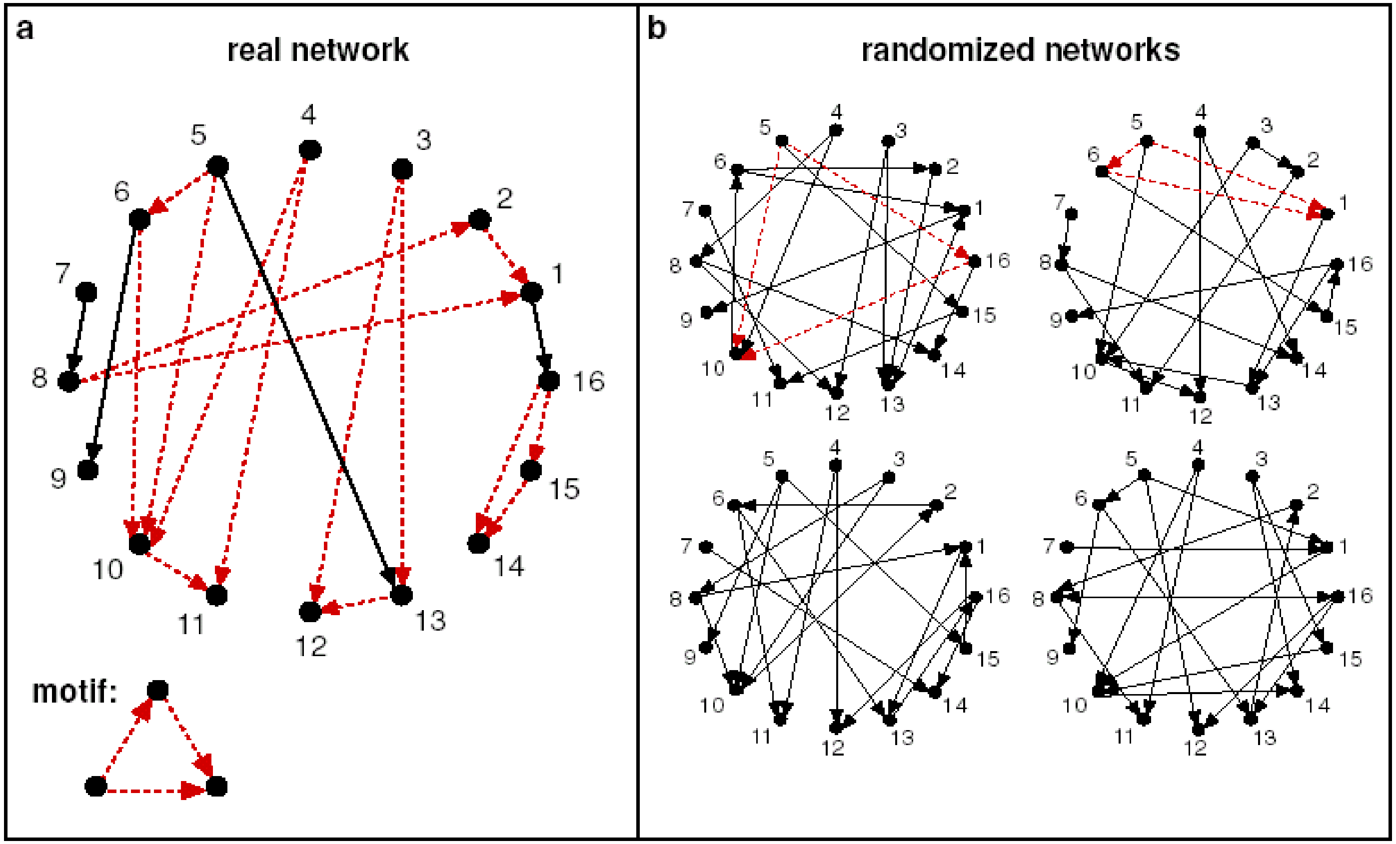


- Allow **overlapping of motifs**
- Network on the right has 4 occurrences of the motif:
  - {1,2,3,4,5}
  - {1,2,3,4,6}
  - {1,2,3,4,7}
  - {1,2,3,4,8}



Significance of Motif

- Key idea: Subgraphs that occur in a real network **much more frequently** than in a random network have **functional significance**



Milo *et. al.*, Science 2002

Significance of Motif

- Motifs are overrepresented in a network when compared to randomized networks:
  - $\boldsymbol{Z_i}$ captures statistical significance of motif $\boldsymbol{i}$:
  - $\boldsymbol{Z_i = (N_i^{real} - \bar{N}_i^{rand}) / std(N_i^{rand})}$
    - N_i^{real} is #(subgraphs of type $\boldsymbol{i}$) in network G^{real}
    - N_i^{rand} is #(subgraphs of type $\boldsymbol{i}$) in network G^{rand}

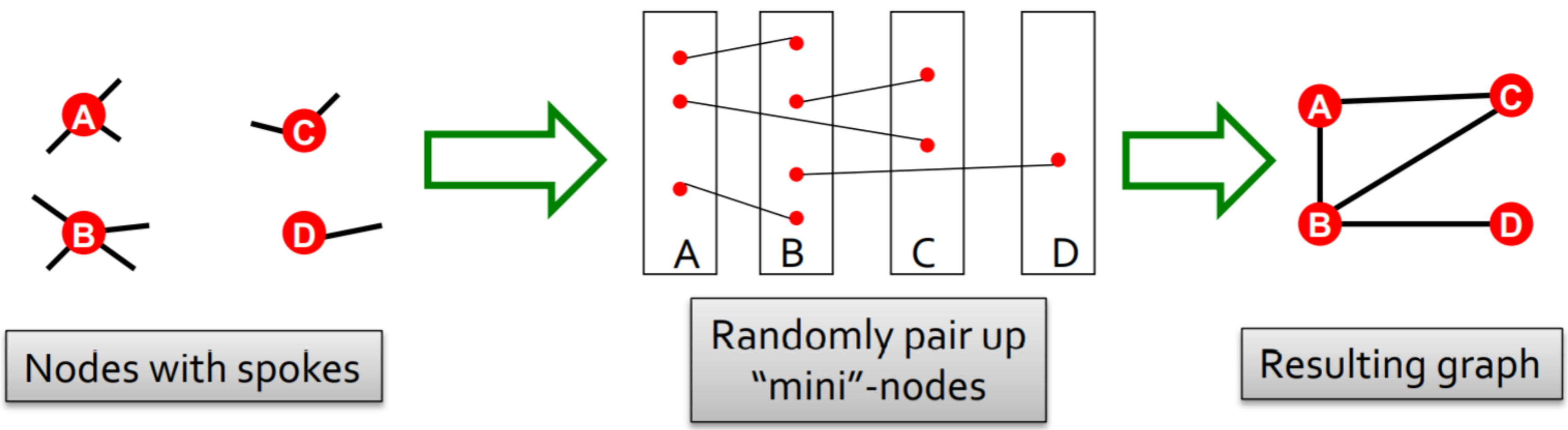
- Network significance profile (SP):

$$SP_i = Z_i / \sqrt{\sum_j Z_j^2}$$

  - SP is a vector of **normalized Z-scores**
  - SP emphasizes relative significance of subgraphs:
    - Important for comparison of networks of different sizes
    - Generally, larger networks display higher Z-scores

Configuration Model

- **Goal**: Generate a random graph with a given degree sequence $k_1, k_2, \ldots, k_n$
- Useful as a "null" model of networks:
  - We can compare the real network G^{real} and a "random" G^{rand} which has the same degree sequence as G^{real}
- Configuration model:



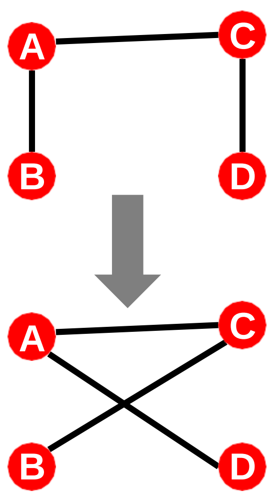
We ignore double edges and self-loops when creating the final graph

Alternative for Spokes: Switching

- Start from a given graph $\boldsymbol{G}$

#experiments

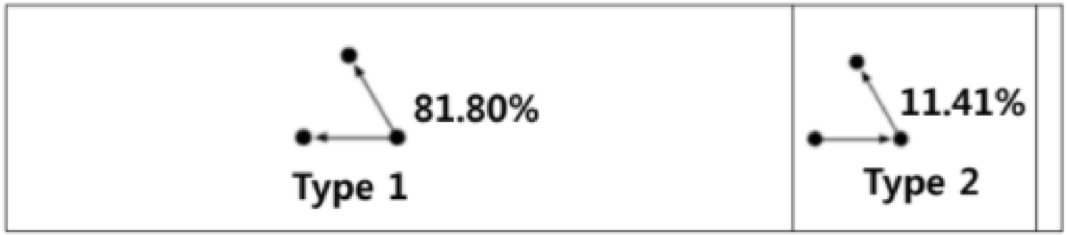
- Repeat the switching step $Q \cdot |E|$ times:
  - Select **a pair of edges** A → B, C → D at random
  - Exchange the endpoints to give A → D, C → B
  - Exchange edges only if no multiple edges or self-edges are generated



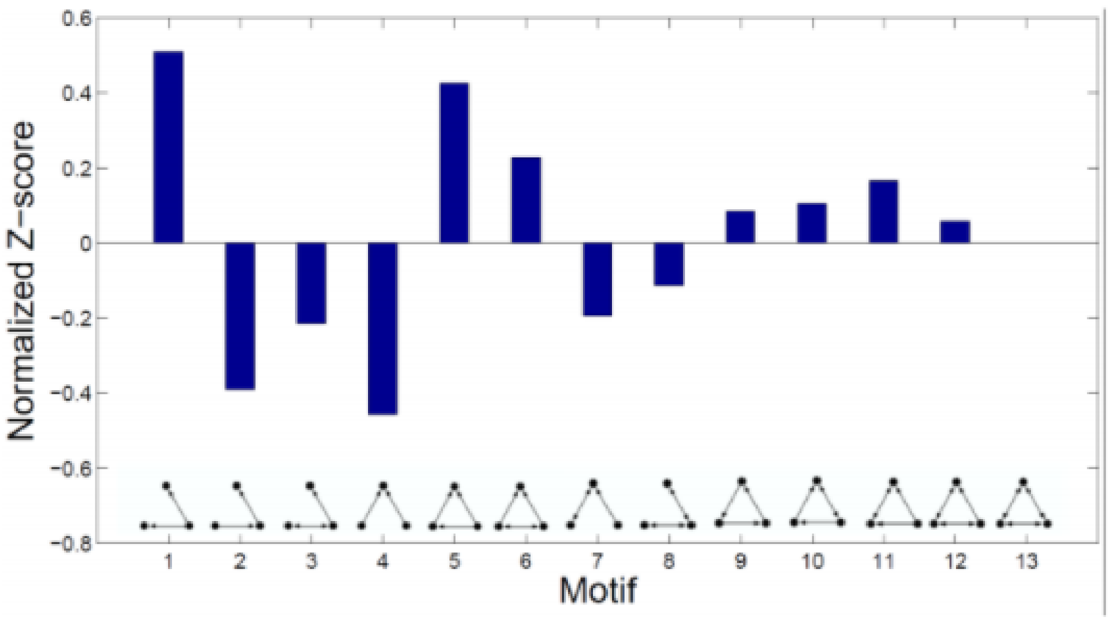
- Result: A randomly **rewired** graph:
  - Same node degrees, randomly rewired edges

- Q is chosen large enough (e.g., $Q = 100$) for the process to converge

Case Example: Invitation Network in Aion



(a) Motifs Distributions

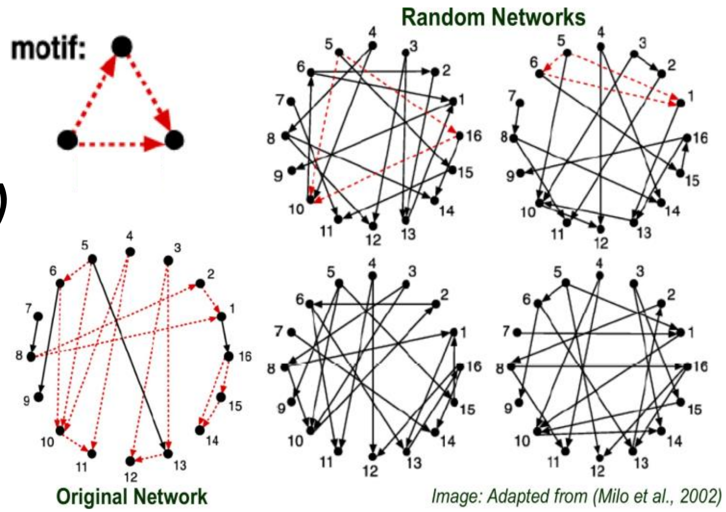


(b) Normalized Z-scores of Invitation networks.

RECAP: Detecting Motifs

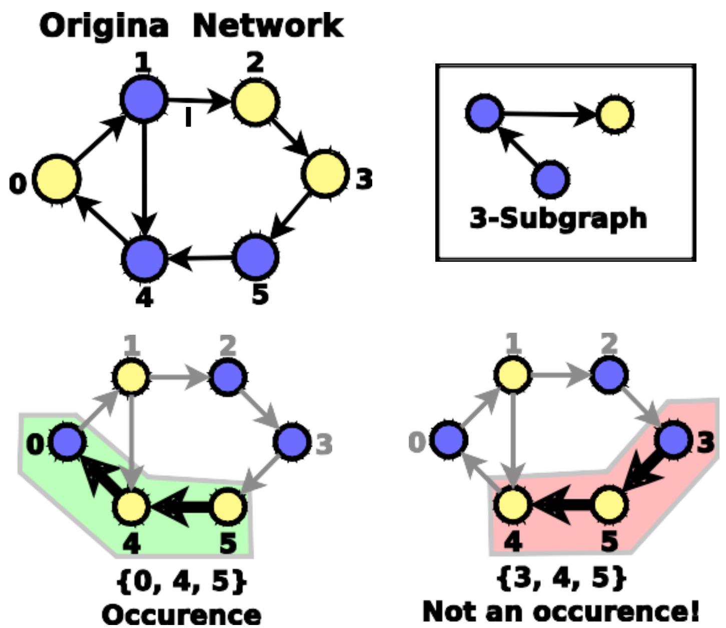
- Count **<u>subgraphs i</u>** in G^{real}
- Count subgraphs i in random networks G^{rand}
- Configuration model: Each G^{rand} has the same #(nodes), #(edges) and #(degree distribution) as G^{real}

- Assign Z-score to i:
  - $Z_i = (N_i^{real} - \bar{N}_i^{rand}) / std(N_i^{rand})$
- High Z-score: Subgraph i is a network motif of $\boldsymbol{G}$

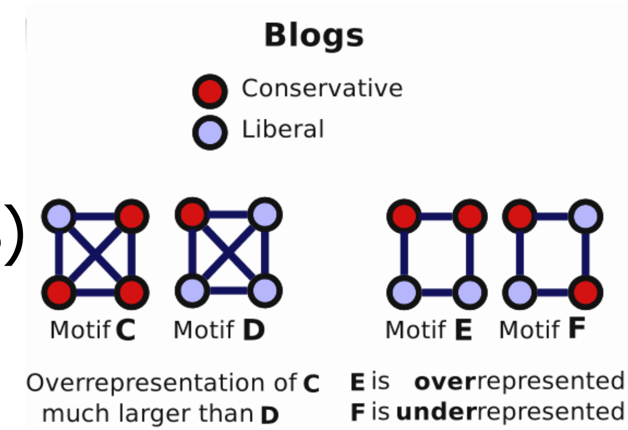


Variations of Motif Concept

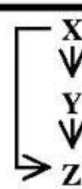
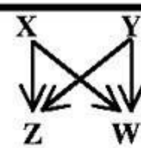
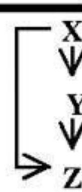
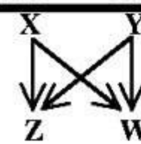
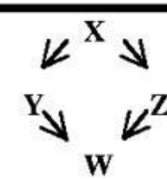
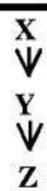
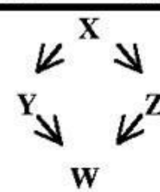
- Canonical definition:
  - Directed and undirected
  - Colored and uncolored
  - Temporal and static motifs



- Variations on the concept
  - Different **frequency concepts**
  - Different **significance metrics**
  - **Under-Representation** (anti-motifs)
  - Different constraints for **null model**



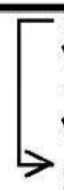
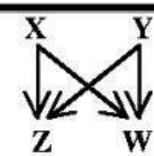
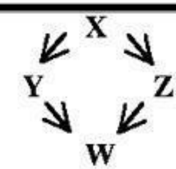
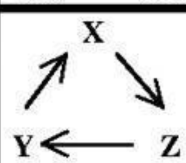
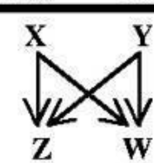
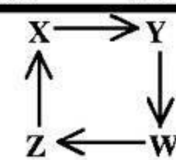
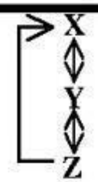
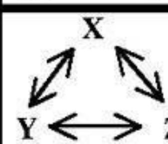
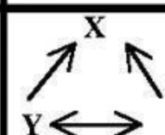
Experiments: Detecting Motifs

Network	Nodes	Edges	N_{real}	$N_{rand} \pm SD$	Z score	N_{real}	$N_{rand} \pm SD$	Z score	N_{real}	$N_{rand} \pm SD$	Z score
Gene regulation (transcription)					**Feed-forward loop**			**Bi-fan**			
E. coli	424	519	40	7 ± 3	10	203	47 ± 12	13			
*S. cerevisiae**	685	1,052	70	11 ± 4	14	1812	300 ± 40	41			
Neurons					**Feed-forward loop**			**Bi-fan**			**Bi-parallel**
C. elegans†	252	509	125	90 ± 10	3.7	127	55 ± 13	5.3	227	35 ± 10	20
Food webs					**Three chain**			**Bi-parallel**			
Little Rock	92	984	3219	3120 ± 50	2.1	7295	2220 ± 210	25			
Ythan	83	391	1182	1020 ± 20	7.2	1357	230 ± 50	23			
St. Martin	42	205	469	450 ± 10	NS	382	130 ± 20	12			
Chesapeake	31	67	80	82 ± 4	NS	26	5 ± 2	8			
Coachella	29	243	279	235 ± 12	3.6	181	80 ± 20	5			
Skipwith	25	189	184	150 ± 7	5.5	397	80 ± 25	13			
B. Brook	25	104	181	130 ± 7	7.4	267	30 ± 7	32			

Milo *et al.*, Science 2004

Z-scores of individual motifs for different networks

Experiments: Detecting Motifs

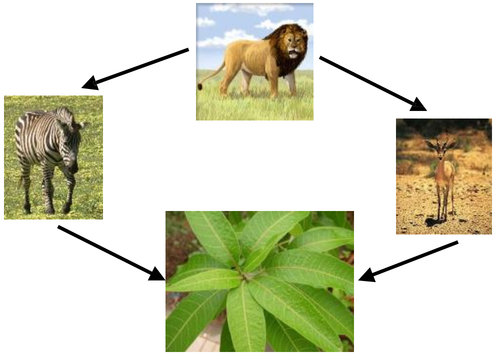
Network	Nodes	Edges	N_{real}	$N_{rand} \pm SD$	Z score	N_{real}	$N_{rand} \pm SD$	Z score	N_{real}	$N_{rand} \pm SD$	Z score
Electronic circuits (forward logic chips)					**Feed-forward loop**			**Bi-fan**			**Bi-parallel**
s15850	10,383	14,240	424	2 ± 2	285	1040	1 ± 1	1200	480	2 ± 1	335
s38584	20,717	34,204	413	10 ± 3	120	1739	6 ± 2	800	711	9 ± 2	320
s38417	23,843	33,661	612	3 ± 2	400	2404	1 ± 1	2550	531	2 ± 2	340
s9234	5,844	8,197	211	2 ± 1	140	754	1 ± 1	1050	209	1 ± 1	200
s13207	8,651	11,831	403	2 ± 1	225	4445	1 ± 1	4950	264	2 ± 1	200
Electronic circuits (digital fractional multipliers)					**Three-node feedback loop**			**Bi-fan**			**Four-node feedback loop**
s208	122	189	10	1 ± 1	9	4	1 ± 1	3.8	5	1 ± 1	5
s420	252	399	20	1 ± 1	18	10	1 ± 1	10	11	1 ± 1	11
s838‡	512	819	40	1 ± 1	38	22	1 ± 1	20	23	1 ± 1	25
World Wide Web					**Feedback with two mutual dyads**			**Fully connected triad**			**Uplinked mutual dyad**
nd.edu§	325,729	1.46e6	1.1e5	2e3 ± 1e2	800	6.8e6	5e4±4e2	15,000	1.2e6	1e4 ± 2e2	5000

Milo *et al.*, Science 2004

Z-scores of individual motifs for different networks

What Do We Learn from Prior 2 Slides?

- Network of neurons and a gene network contain **similar motifs**:
  - **Feed-forward loops** and **bi-fan structures**
  - Both are information processing networks with sensory and acting components
- Food webs have parallel loops:
  - Prey of a particular predator share prey



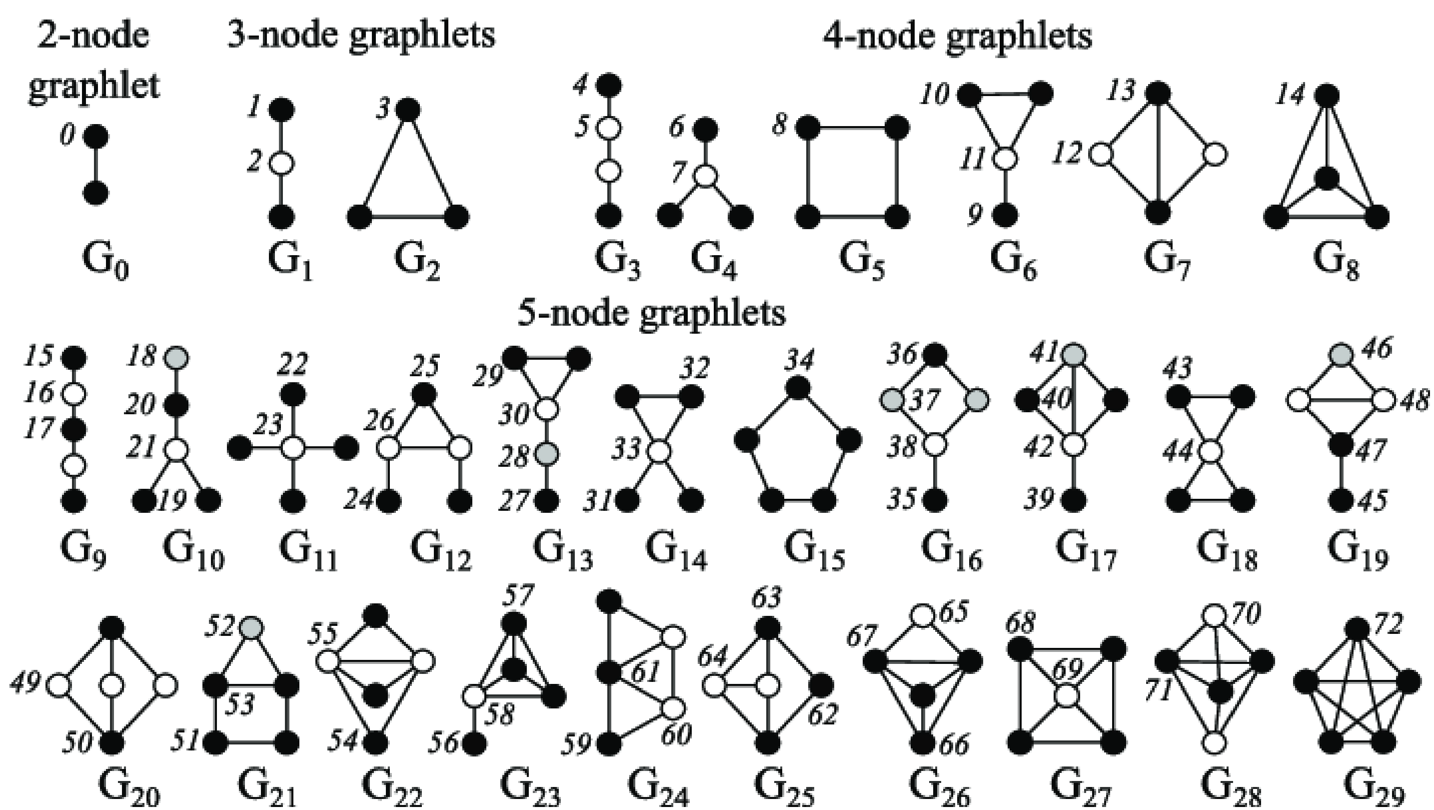
- WWW network has bidirectional links
  - Design that allows the shortest path between sets of related pages

WE UNDERSTAND HOW NETWORK ARE STRUTURED!

Graphlets: Node Feature Vectors

New Concept: Graphlets

- **Graphlets**: connected non-isomorphic subgraphs
  - Induced subgraphs of any frequency

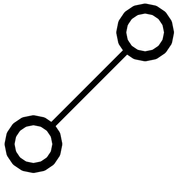


For n = $3, 4, 5, \ldots 10$ there are $2, 6, 21, \ldots 11716571$ graphlets!

Graphlet Degree Vector (GDV)

- Next: Use graphlets to obtain a **node-level subgraph** metric

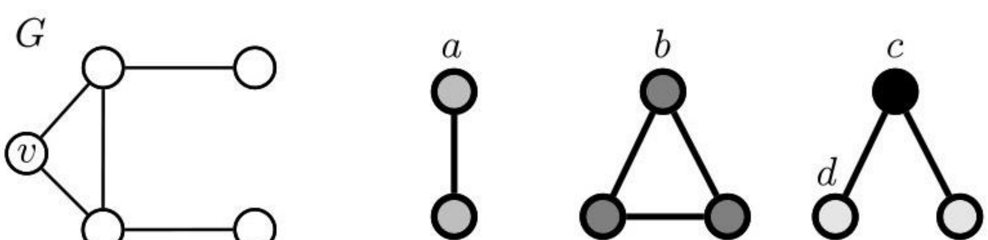
- **Degree** counts **#(edges)** that a node touches:
  - Can we generalize this notion for graphlets? – Yes!



- **Graphlet degree vector** counts **#(graphlets)** that a node touches

Automorphism Orbit

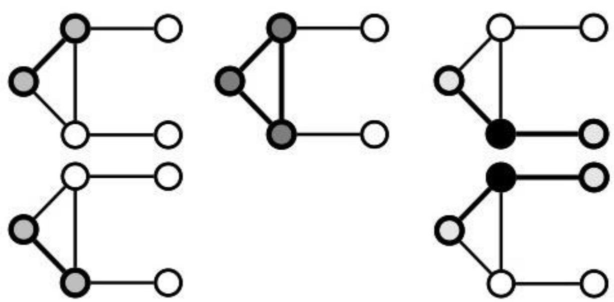
- An automorphism orbit takes into account the **symmetries of a subgraph**
- Graphlet Degree Vector (GDV): a vector with the frequency of the node in each orbit position
- Example: Graphlet degree vector of node v



For a node u of graph G, the automorphism orbit of u is $Orb(u) = \{v \in V(G); v = f(u) \text{ for some } f = \mathrm{Aut}(G)\}$.

The Aut denotes an automorphism group of G, *i.e.*, an isomorphism from G to itself.

$orbit$	a	b	c	d
$GDV(v)$	2	1	0	2



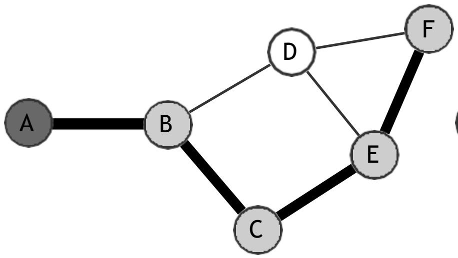
Graphlet Degree Vector (GDV)

- **Graphlet degree vector** counts **#(graphlets)** that a node touches at a particular orbit

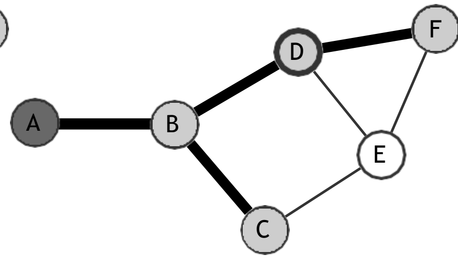
- Considering graphlets on 2 to 5 nodes we get:
  - **Vector of 73 coordinates** is a signature of a node that describes the topology of node's neighborhood
  - Captures its interconnectivities out to **a distance of 4 hops**

- Graphlet degree vector provides a measure of a **node's local network topology**:
  - Comparing vectors of two nodes provides a highly constraining measure of local topological similarity between them

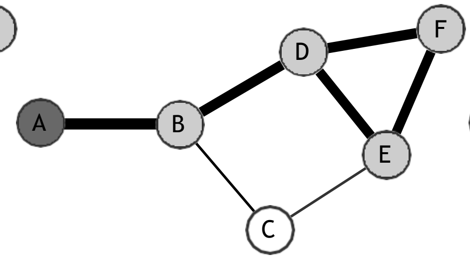
Graphlet Degree Vector: Example



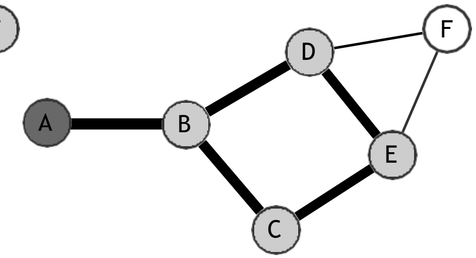
Orbit 15



Orbit 19



Orbit 27



Orbit 35

Orbit	0	1	2…3	4	5	6	7…14	15	16…18	19	20…26	27	28…34	35	36…72
GDV(A)	1	2	0…0	3	0	1	0…0	1	0…0	1	0…0	1	0…0	1	0…0

- **Graphlet Degree Vector (GDV) of node A**:
  - i-th element of GDV(A): #(graphlets) that touch A at orbit i
  - Highlighted are graphlets that touch node A at orbits 15, 19, 27, and 35 from left to right

Yaveroglu et al., Journal of Statistical Software 2015

Finding Motifs and Graphlets

Finding Motifs and Graphlets

- Finding size-k motifs/graphlets requires solving two challenges:
  1) **Enumerating** all size-k connected subgraphs
  2) **Counting** #(occurrences of each subgraph type)

- Just knowing if a certain subgraph exists in a graph is a hard computational problem!
  - Subgraph isomorphism is **NP-complete**

- Computation time grows exponentially as the size of the motif/graphlet increases
  - **Feasible motif size is usually small (3 to 8)**

Computing Subgraphs

- Network-centric approaches:

  1) Enumerating all size-k connected subgraphs

  2) Counting #(occurrences of each subgraph type) via graph isomorphisms test

- Algorithms:
  - **Exact subgraph enumeration (ESU) [Wernicke 2006]**
  - Kavosh [Kashani et al. 2009]
  - Subgraph sampling [Kashtan et al. 2004]

Exact Subgraph EnUmeration (ESU)

- Two sets
  - $V_{subgraph}$: currently constructed subgraph (motif)
  - $V_{extension}$: set of candidate nodes to extend the motif

- **Idea**: Starting with a node v, add those nodes u to $V_{extension}$ set that have two properties:
  - u's node_id must be larger than that of v
  - u may only be neighbored to some newly added node w but **not** of any node already in $V_{subgraph}$

- ESU is implemented as a recursive function:
  - The running of this function can be displayed as a **tree-like structure** of depth $\boldsymbol{k}$, called the **ESU-Tree**

인천대학교 INCHEON NATIONAL UNIVERSITY

Exact Subgraph Enumeration (ESU)

Algorithm: $\textsc{EnumerateSubgraphs}(G, k)$ (ESU)
Input: A graph $G = (V, E)$ and an integer $1 \leq k \leq |V|$.
Output: All size-k subgraphs in G.

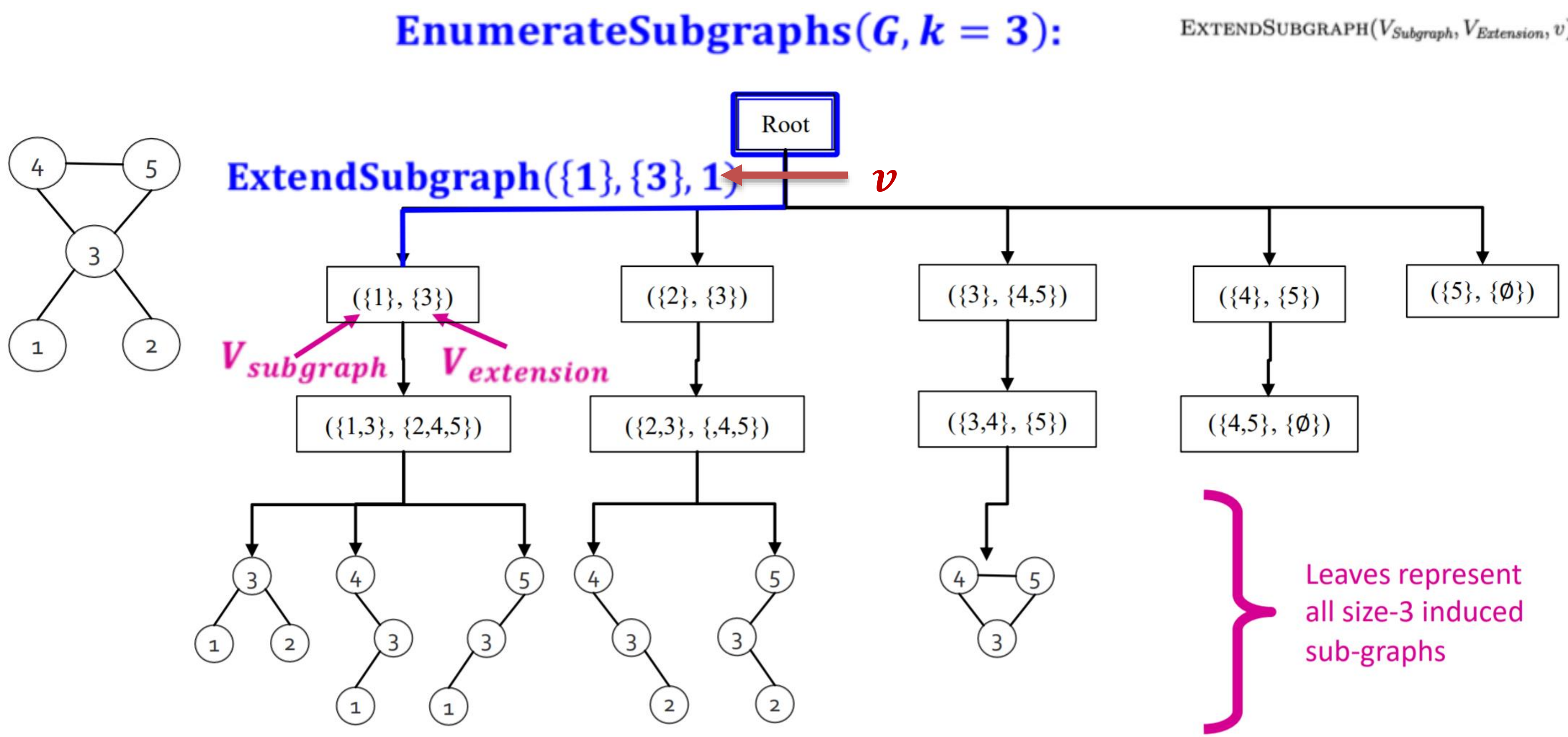
```
for each vertex v ∈ V do
    V_Extension ← {u ∈ N({v}) : u > v}
    call ExtendSubgraph({v}, V_Extension, v)
return
```

$\textsc{ExtendSubgraph}(V_{Subgraph}, V_{Extension}, v)$

```
if |V_Subgraph| = k then output G[V_Subgraph] and return
while V_Extension ≠ ∅ do
    Remove an arbitrarily chosen vertex w from V_Extension
    V'_Extension ← V_Extension ∪ {u ∈ N_excl(w, V_Subgraph) : u > v}
    call ExtendSubgraph(V_Subgraph ∪ {w}, V'_Extension, v)
return
```

$N_{excl}(w, V_{Subgraph}) = N(w) \backslash (V_{Subgraph} \cup N(V_{Subgraph}))$ is exclusive neighborhood: All nodes neighboring w but not of $V_{Subgraph}$ or $N(V_{Subgraph})$

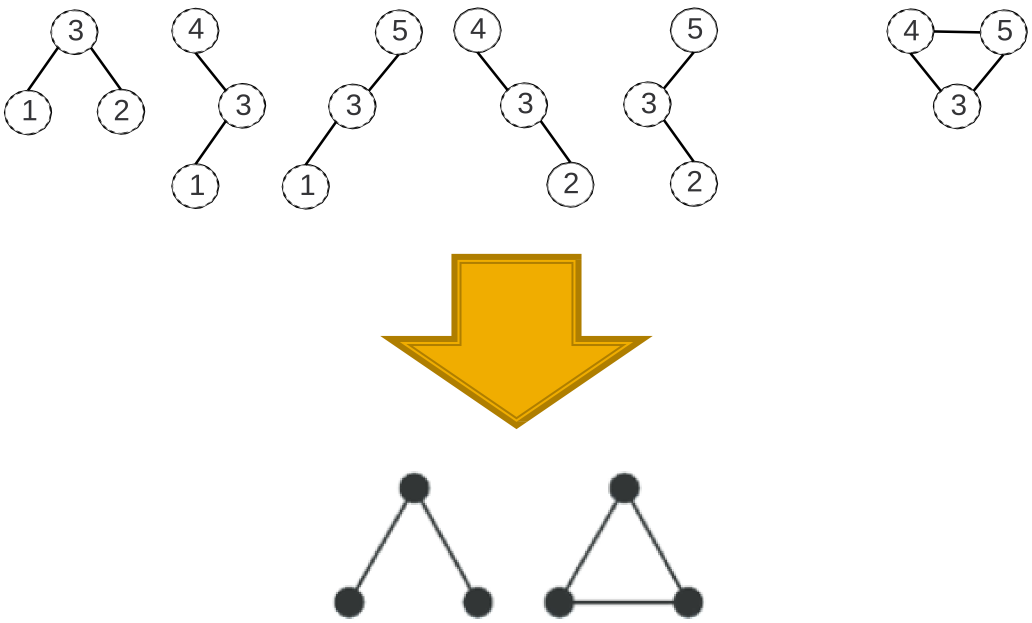
ESU-Tree Example



- Nodes in the ESU-tree include two adjoining sets:
  - $\mathrm{V}_{\mathrm{subgraph}}$: currently constructed subgraph (motif)
  - $\mathrm{V}_{\mathrm{extension}}$: Nodes adjacent to $\mathrm{V}_{\mathrm{subgraph}}$ whose node_ids are larger than starting node v

Use ESU-Tree to Count Subgraphs

- So far, we enumerated all size-k subgraphs in the input graph
- Next step: Count the graphs



Count: 5 1

Use ESU-Tree to Count Subgraphs

- So far, we enumerated all size-k subgraphs in the input graph
- Next step: Count the graphs

Classify subgraphs placed in the ESU-Tree leaves **into non-isomorphic size-k classes**:

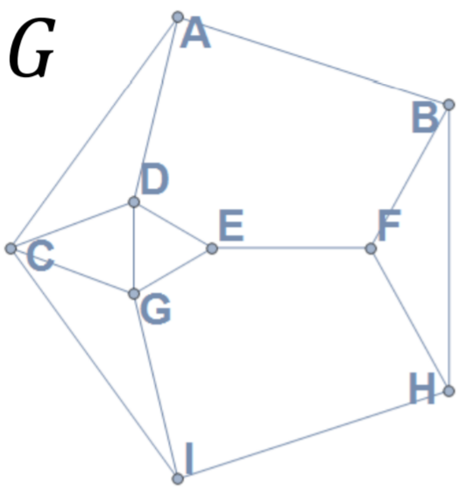
- Determine which subgraphs in ESU-Tree leaves are **topologically equivalent (isomorphic)** and group them into subgraph classes accordingly
- Use **McKay's nauty algorithm** [McKay 1981]

Graph Isomorphism

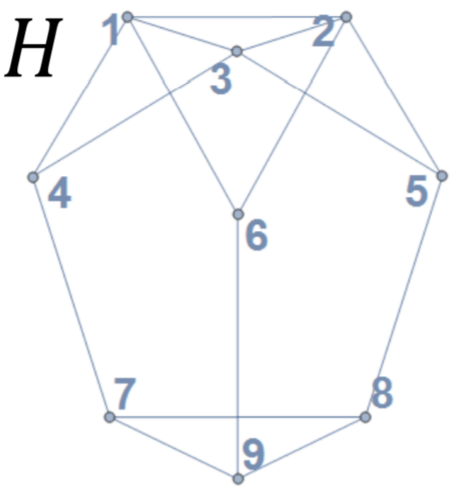
- Graphs G and H are isomorphic if there exists a bijection $f: V(G) \rightarrow V(H)$ such that:
  - Any two nodes $\boldsymbol{u}$ and $\boldsymbol{v}$ of $\boldsymbol{G}$ are adjacent in $\boldsymbol{G}$ iff $\boldsymbol{f}(\boldsymbol{u})$ and $\boldsymbol{f}(\boldsymbol{v})$ are adjacent in $\boldsymbol{H}$

- Example: Are G and H topologically equivalent?

G



H



f:

A	4
B	7
C	1
D	3
E	5
F	8
G	2
H	9
I	6

Need to check 9! possible bijections between node sets

Hard computational problem!

G and H are isomorphic!

How to Compute Motifs & Graphlets?

- Dyad (motifs between 2 nodes) and Triad (motifs among 3 nodes) are already implemented in igraph
  - igraph_dyad_census
  - igraph_triad_census

- RAND-ESU algorithms are also implemented
  - motifs_randesu(size=3, cut_prob=None, callback=None)
  - motifs_randesu_estimate(size=3, cut_prob=None, sample)
  - motifs_randesu_no(size=3, cut_prob=None)

- See document https://igraph.org/python/doc/igraph.GraphBase-class.html in more detail

Summary

- Generating random graph
  - Erdos-Renyi → Cannot mimic the degree distributions!
  - Small world

- Motif, subgraph, graphlet analysis

- Finding motif, graphlets in graph
  - ESU-Tree

Understanding the algorithms is enough
Focus more on what analysis you will do &
what the results imply

Thank you!

Instructor: Daejin Choi (djchoi@inu.ac.kr)