Motifs, Graphlets, Structural similarity

Natalia Semenova

Higher School of Economics

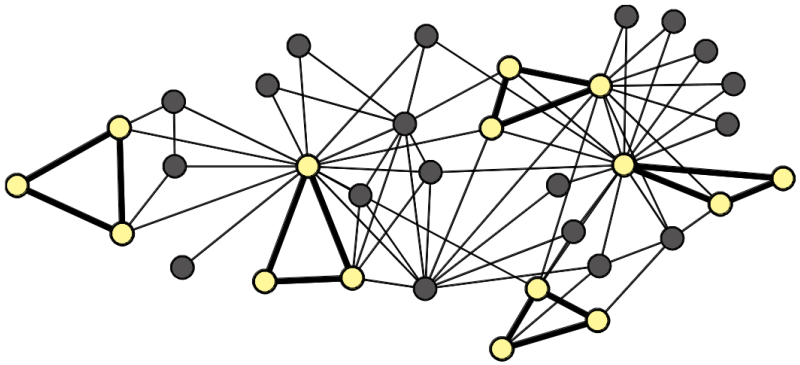
nasemenova_2@edu.hse.ru

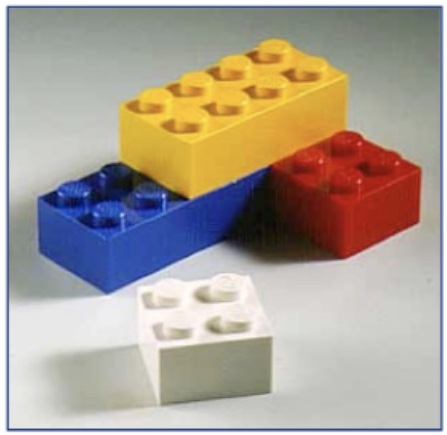
Structural Analysis and Visualization of Networks
13.02.2024

Presentation Overview

Subnetworks

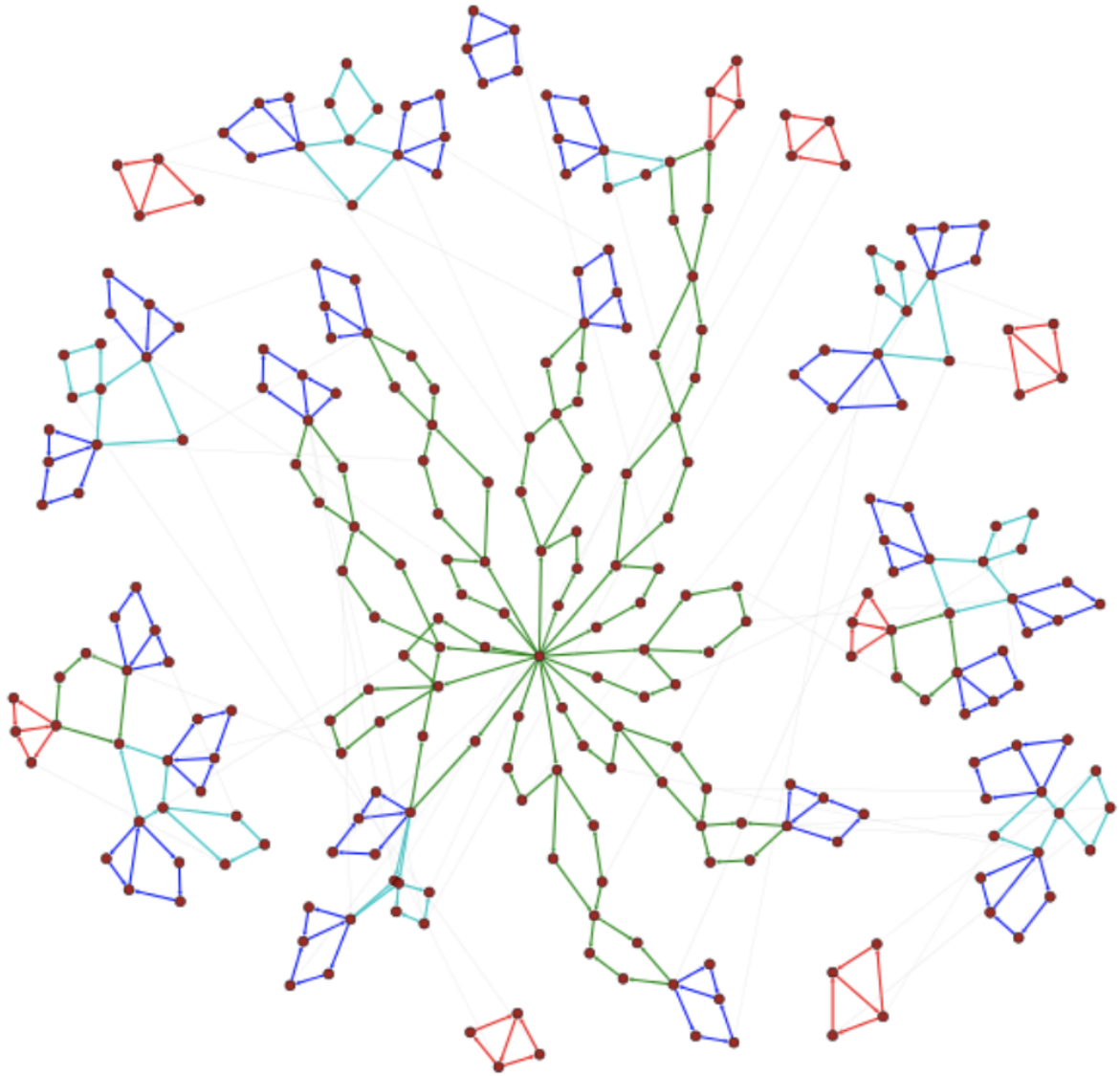
Subnetworks, or **subgraphs**, are the building blocks of networks





They have the power to characterize and discriminate networks

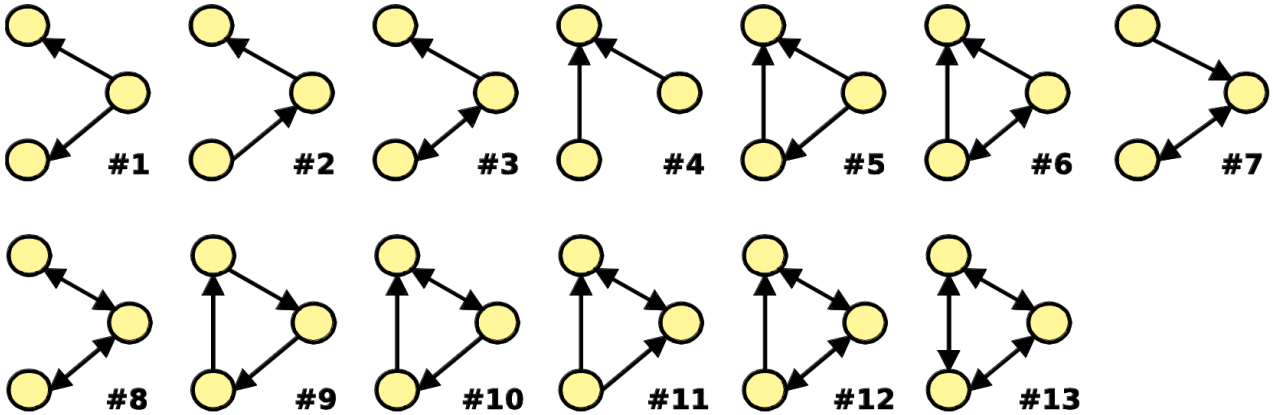
Building Blocks of Networks



Subgraph decomposition of an electronic circuit

Case example of Subgraphs

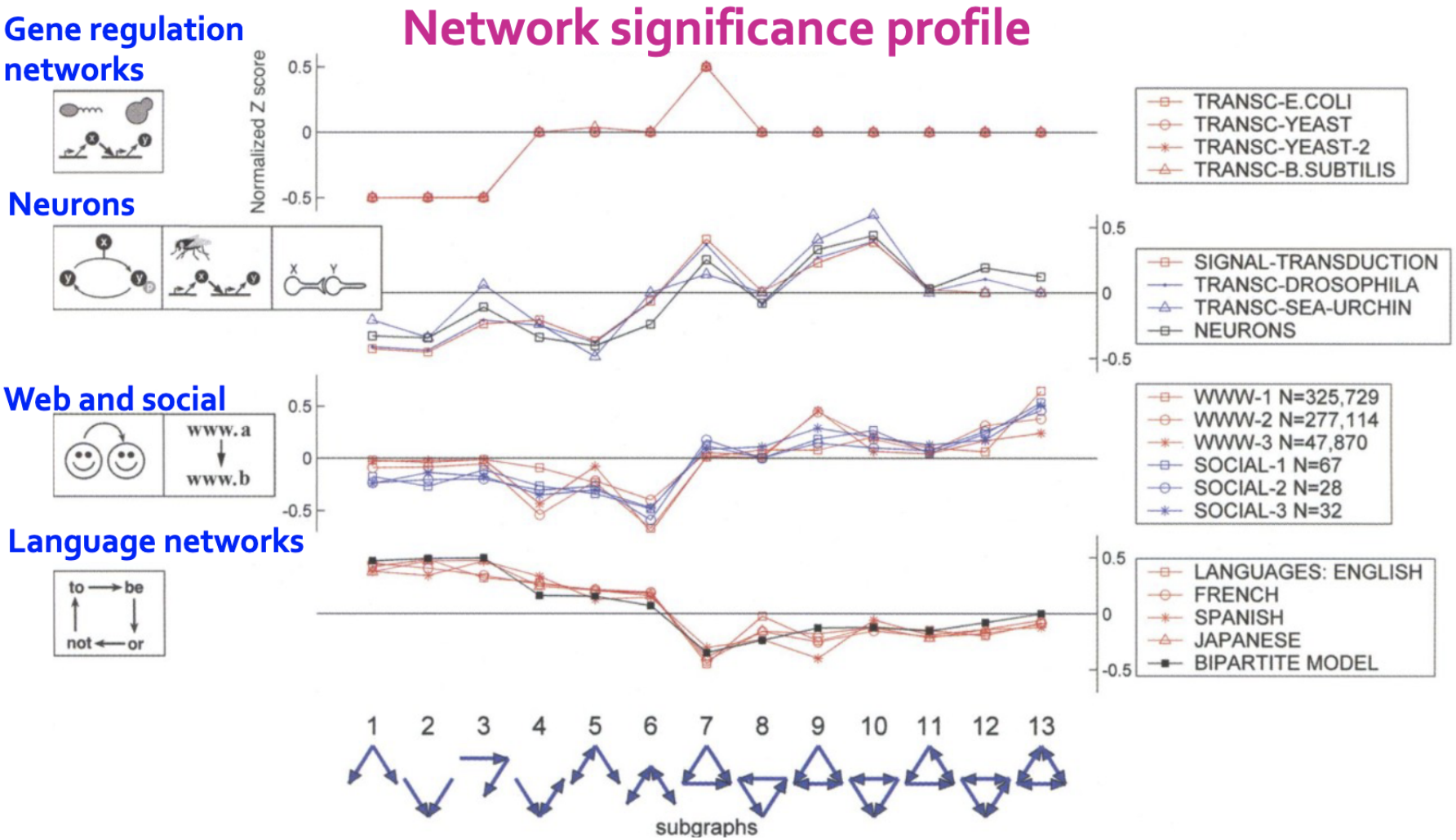
Let's consider all possible (non-isomorphoic) directed subgraphs of size 3



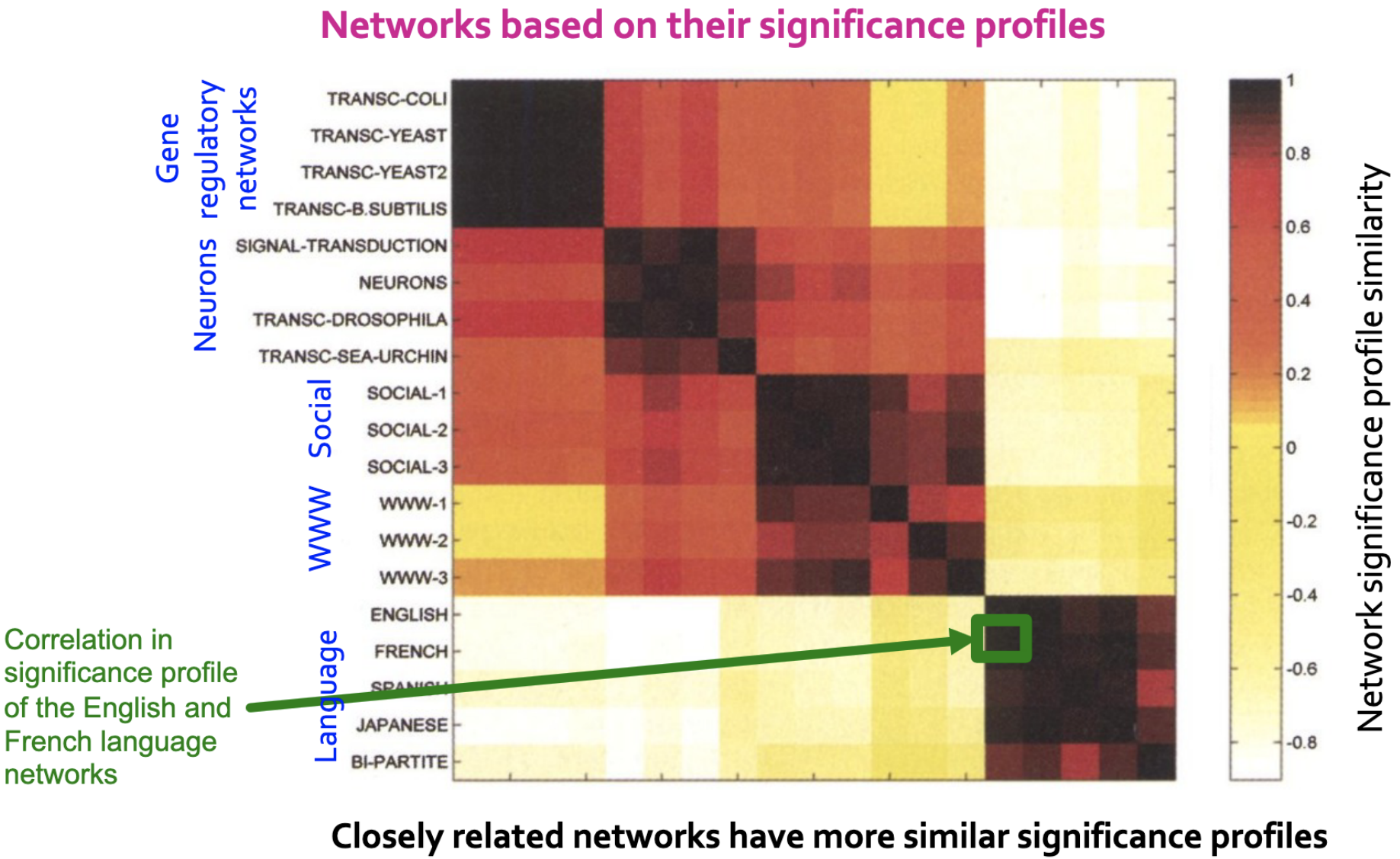
Case example of Subgraphs

- For each subgraph:
  - Imagine you have a metric capable of classifying the subgraph "significance"
    - 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



Case example of Subgraphs



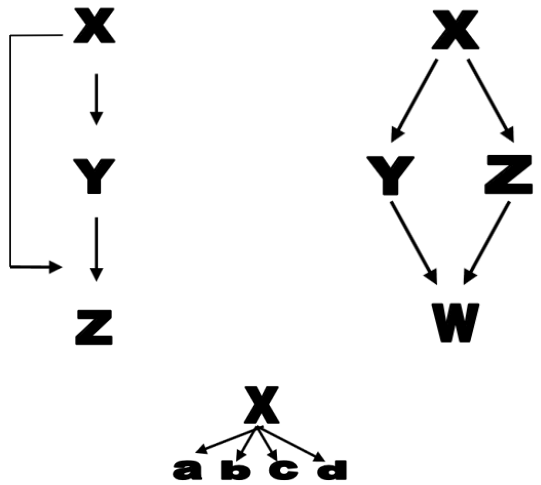
Network Motifs

Network motifs: “recurring, significant patterns of interconnections”
How to define a network motif:

- Pattern: Small induced subgraph
- Recurring: Found many times, i.e., with high frequency
- Significant: More frequent than expected, i.e., in randomly generated networks (Ex. Erdos-Renyi random graphs, scale-free networks)

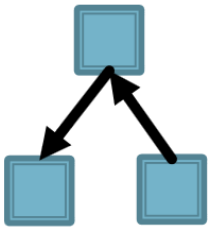
Why do we need Motifs?

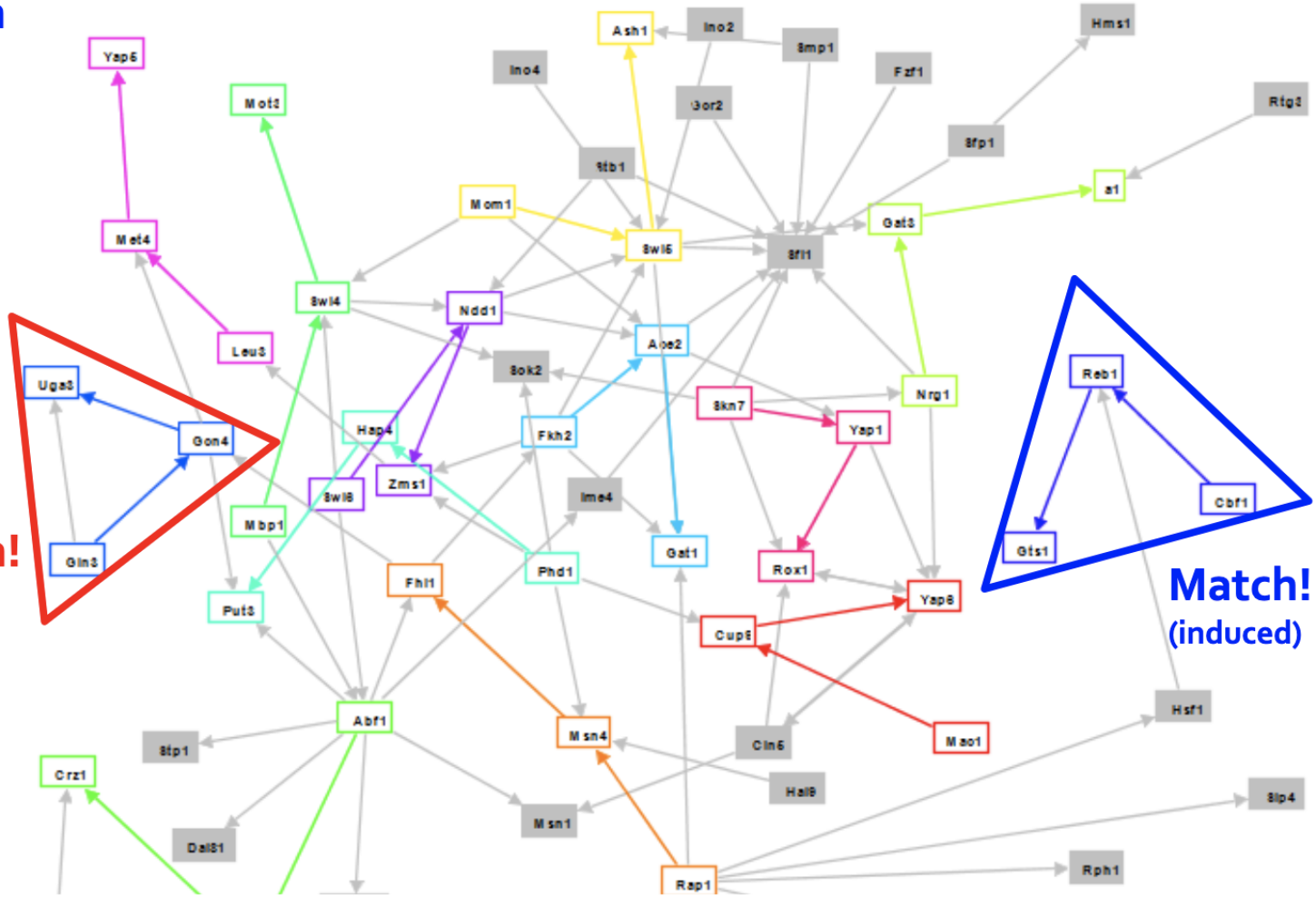
- Motifs:
  - Help us understand how networks work
  - Help us predict operation and reaction of the network in a given situation
- Examples:
  - Feed-forward loops: found in networks of neurons, where they neutralize “biological noise”
  - Parallel loops: found in food webs
  - Single-input modules: found in gene control networks



Motifs: Induced Subgraphs

Induced subgraph of interest (aka Motif):





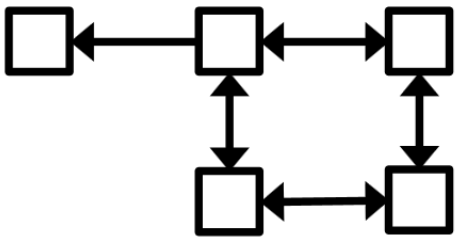
No match!
(not induced)

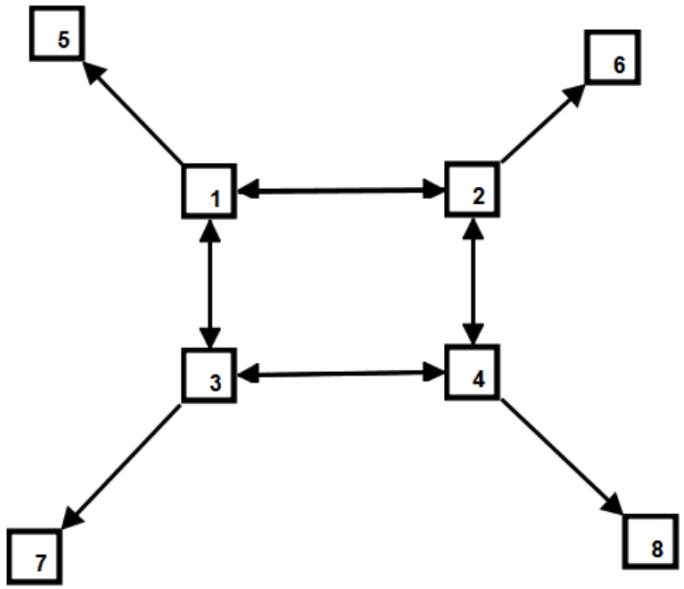
Match!
(induced)

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.

Motifs: Recurrence

- 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

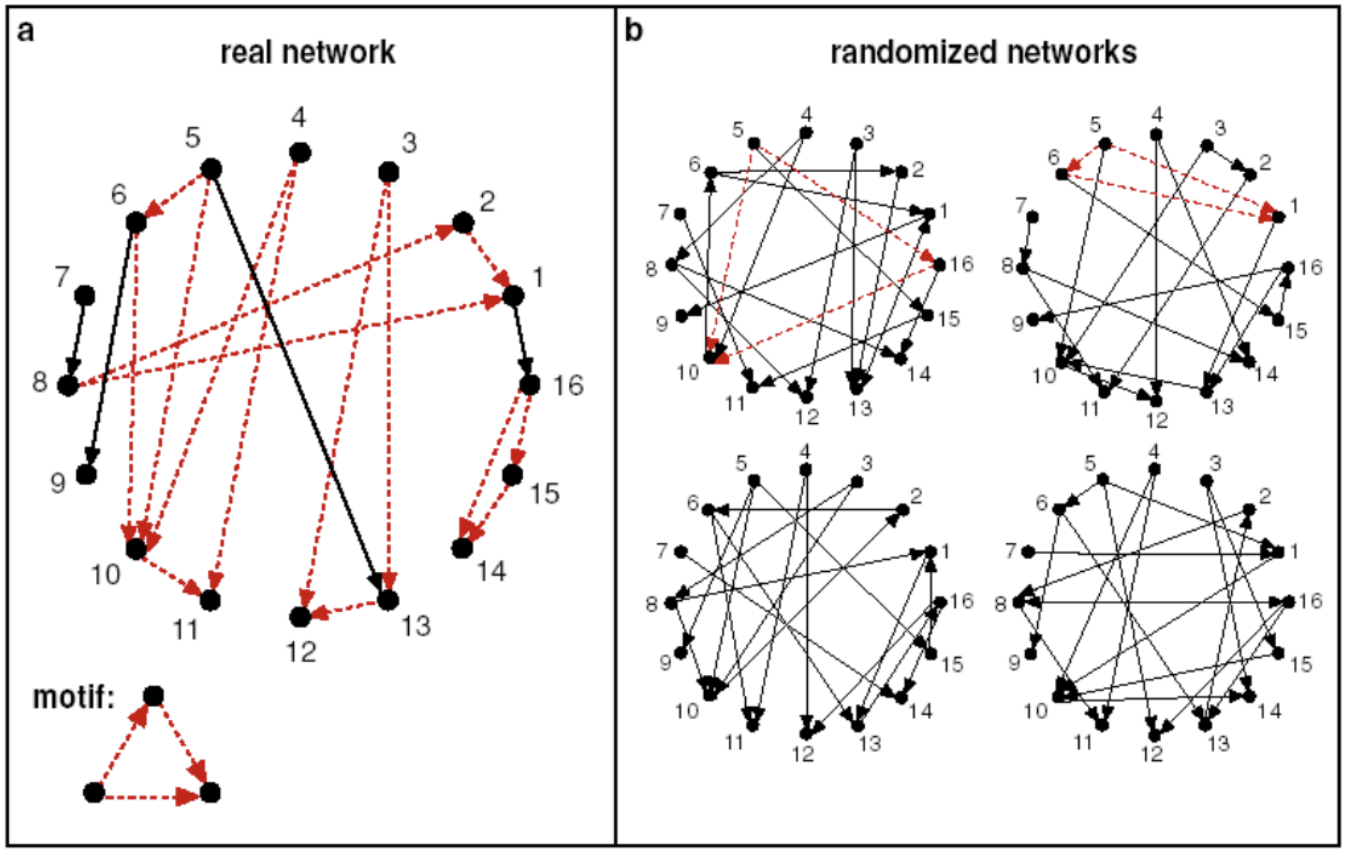




Significance of Motifs

Key Idea

Subgraphs that occur in a real network much more often than in a random network have functional significance



Significance of Motifs

- Motifs are overrepresented in a network when compared to randomized networks:
  - Z_i captures statistical significance of motif i:

    $$Z_i = \frac{(N_i^{real} - N_i^{rand})}{\sigma(N_i^{rand})}$$

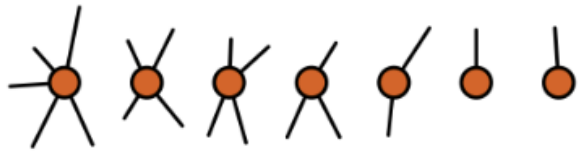
    - N_i^{real} - #(subgraphs of type i) in network G^{real}
    - N_i^{rand} is #(subgraphs of type i) in randomized network G^{rand}
- Network significance profile (SP):

$$SP_i = \frac{Z_i}{\sqrt{\sum_j Z_i^2}}$$

  - 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

- Random graph with n nodes with a given degree sequence: $D = \{k_1, k_2, k_3..k_n\}$ and $m = 1/2 \sum_i k_i$ edges.
- Construct by randomly matching two stubs and connecting them by an edge.

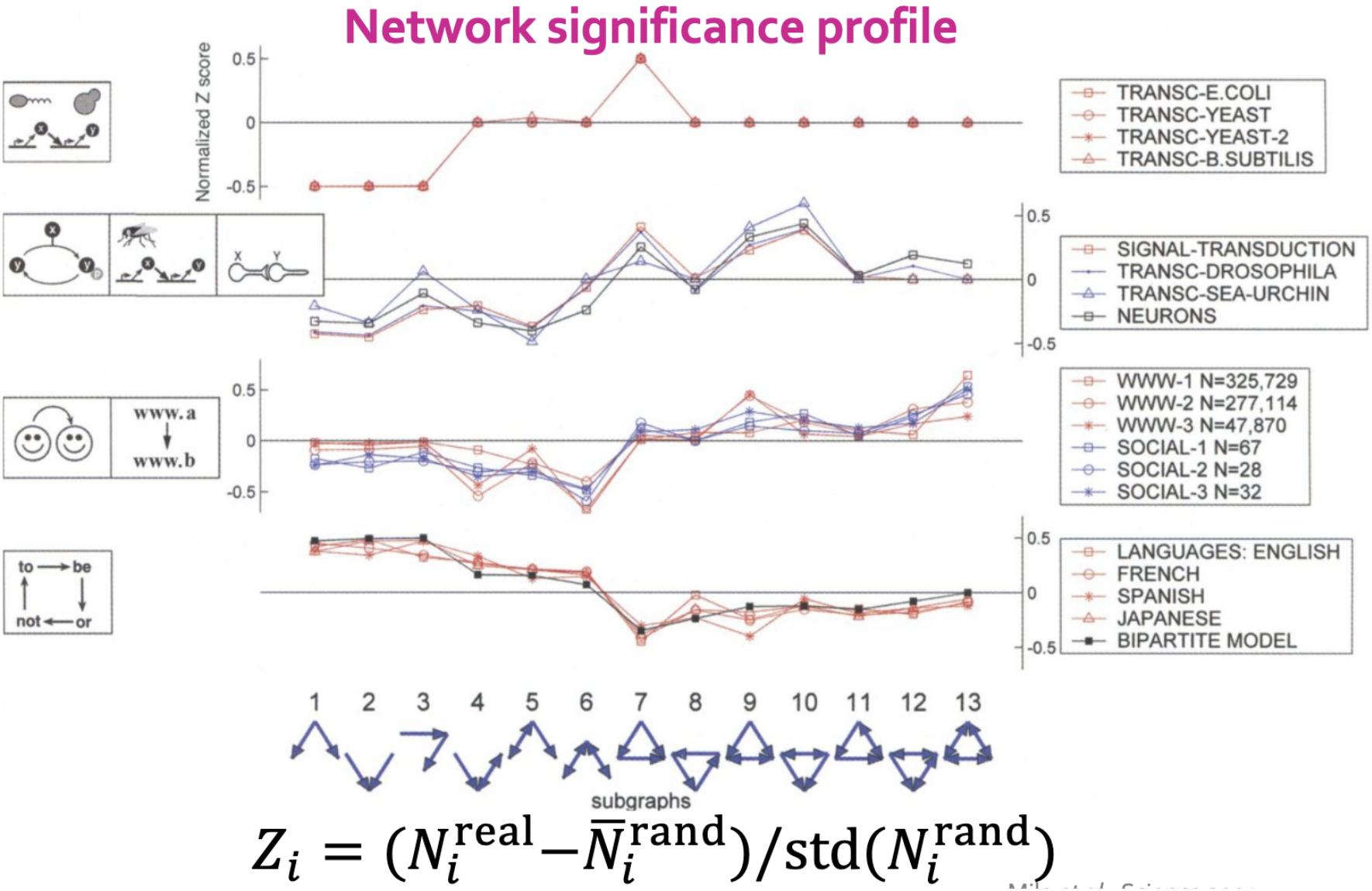


- Can contain self loops and multiple edges
- Probability that two nodes i and j are connected

$$p_{ij} = \frac{k_i k_j}{2m - 1}$$

- Will be a simple graph for special "graphical degree sequence"

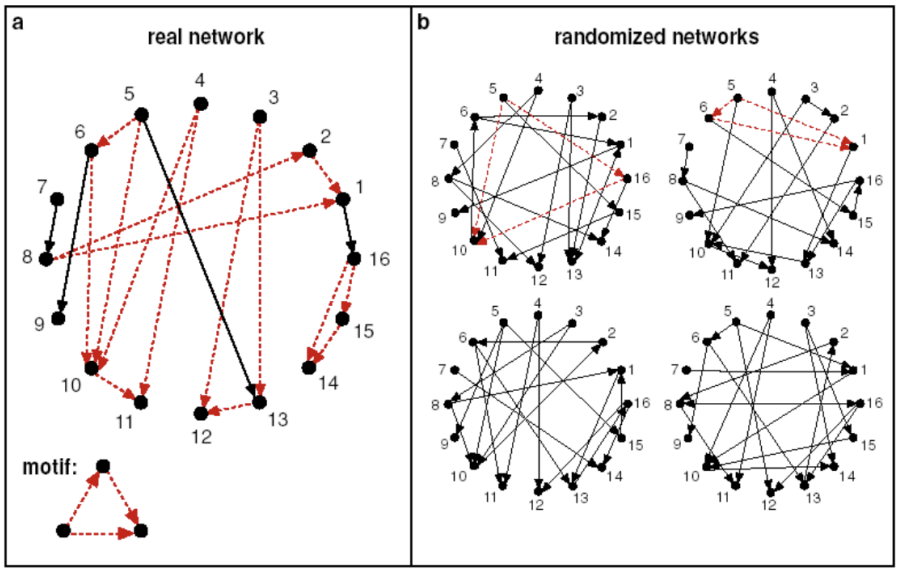
Motifs: Significance Example



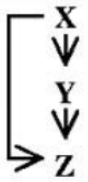
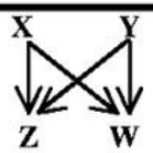
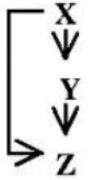
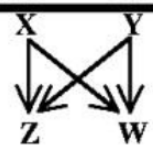
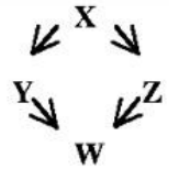
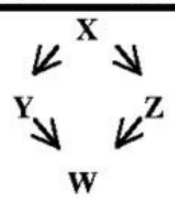
$$Z_i = (N_i^{\text{real}} - \overline{N}_i^{\text{rand}})/\text{std}(N_i^{\text{rand}})$$

Detecting Motifs

- Count subgraphs i 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: $Z_i = \frac{(N_i^{real} - N_i^{rand})}{\sigma(N_i^{rand})}$
- **High Z-score**: Subgraph i is a network motif of G

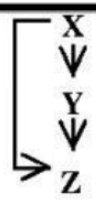
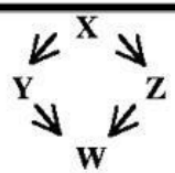
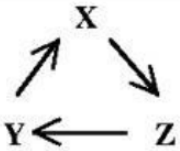
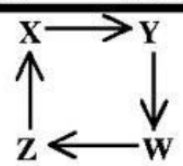
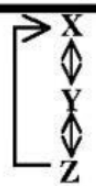
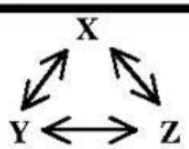
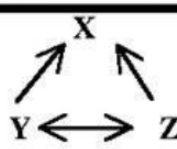


Experiments: Detecting Motifs

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			

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

Z-scores of individual motifs for different networks

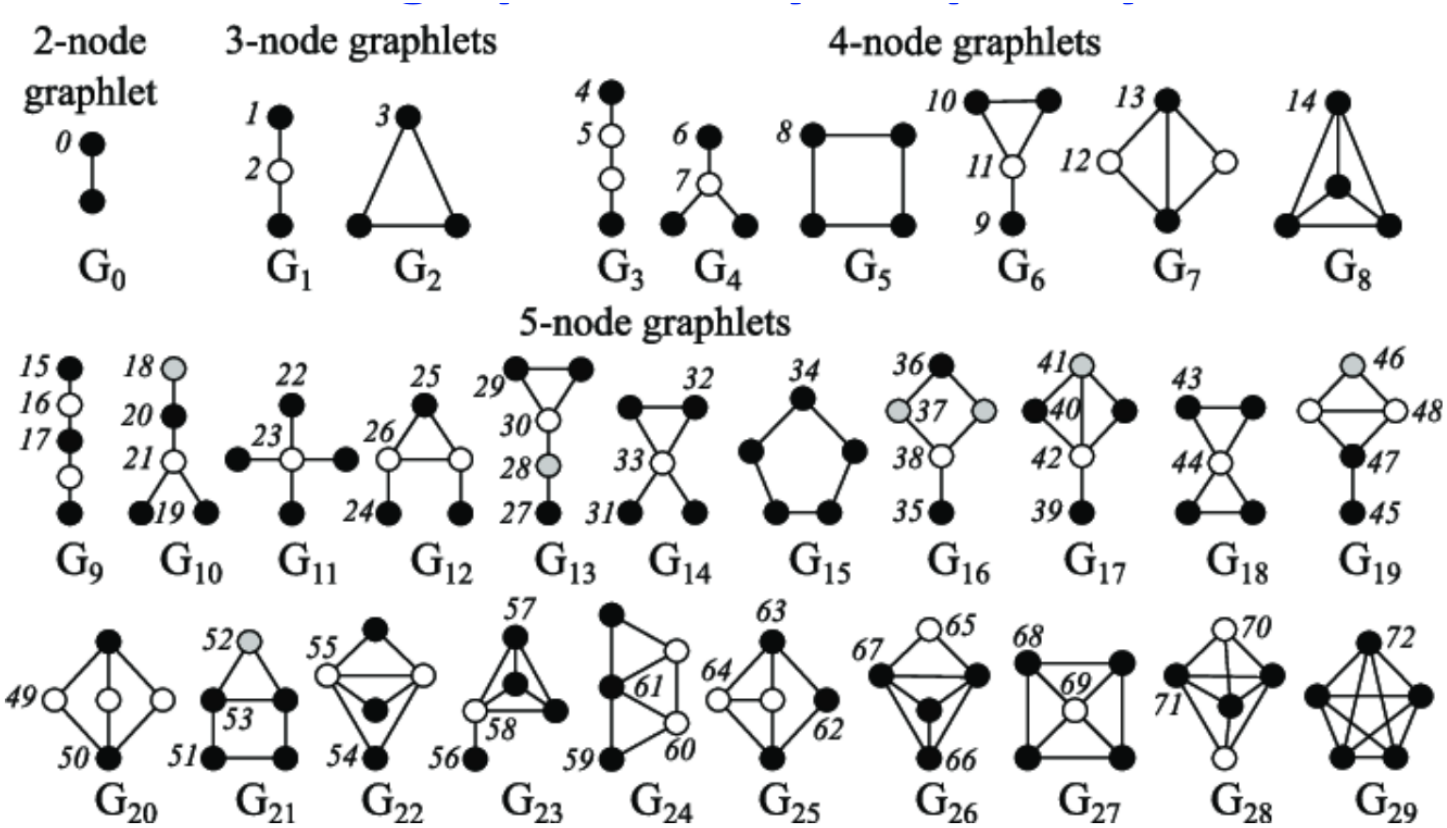
What do we learn from previous tables?

- 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

Graphlets

Graphlets

- Connected non-isomorphic subgraphs
- Induced subgraphs of any frequency



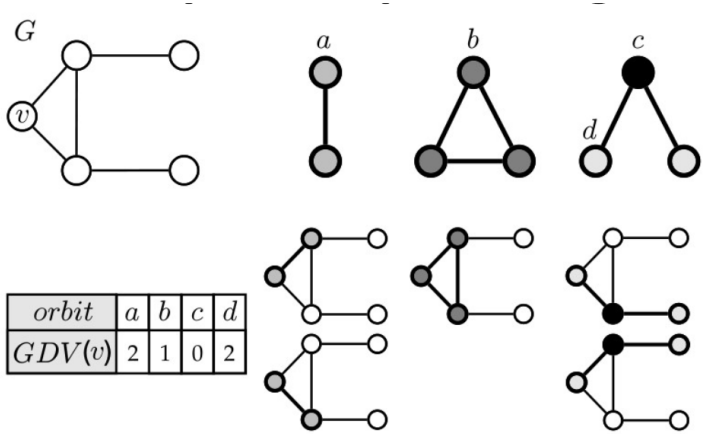
For $n = 3, 4, 5, \ldots 10$ there are 2, 6, 21, ... 11716571 graphlets!

Graphlet Degree Vector

- 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)

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*



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

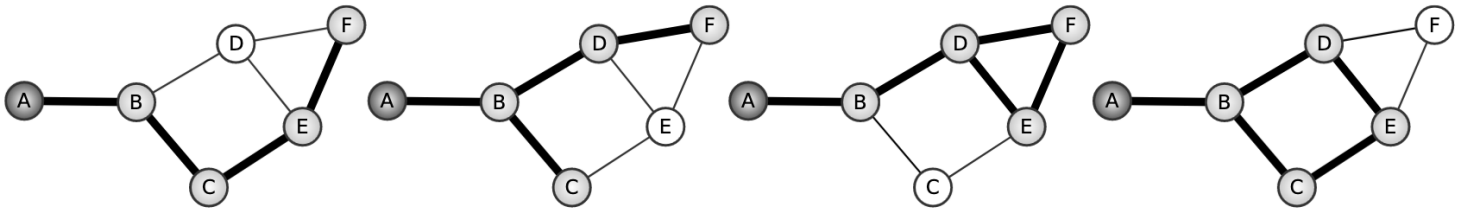
For a node u of graph G, the automorphism orbit of u is $Orb(u) = \{v \in V(G); v = f(u)$ for some $f = \mathrm{Aut}(G)\}$.

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

Graphlet Degree Vector

- 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): #(graphets) that touch A at orbit *i*
- Highlighted are graphlets that touch node A at orbits 15,19, 27, 35 from left to right
  - Can we generalize this notion for graphlets? – Yes!
- Graphlet degree vector counts #(graphlets)

Finding motifs and graphlets

- Finding size-k motifs/graphlets requires solving two challenges:
  - Enumerating all size-k connected subgraphs
  - 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)

Counting Subgraphs

Network-centric approaches:

- Enumerating all size-k connected subgraphs
- 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]

Today: ESU algorithm

Exact subgraph enumeration

Two sets:

- $V_{subgraph}$: currently constructed subgraph (motif)
- $V_{extension}$ 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 beneighbored 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 k, called **ESU-tree**

Exact subgraph enumeration (ESU)

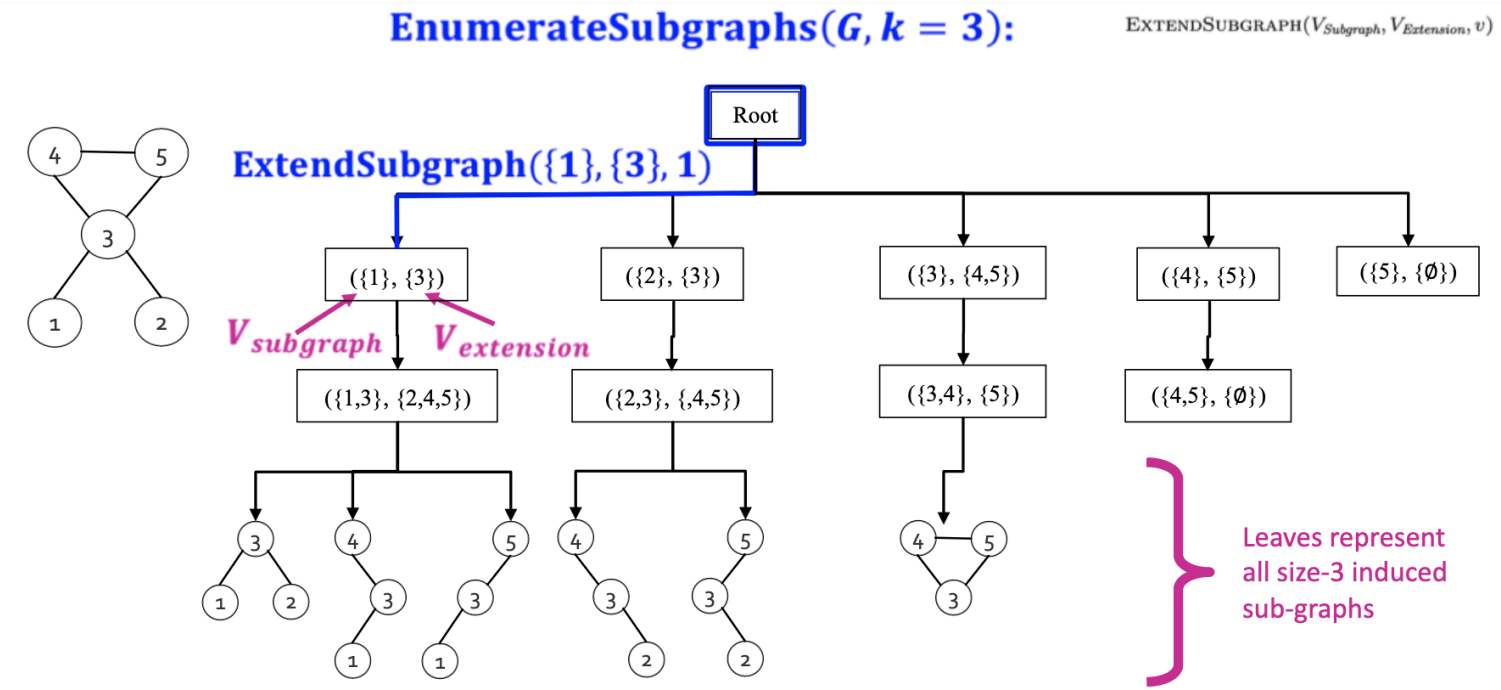
Algorithm: EnumerateSubgraphs(G, k) (ESU)
Input: A graph $G = (V, E)$ and an integer $1 \leq k \leq |V|$.
Output: All size-k subgraphs in G.

01 **for** each vertex $v \in V$ **do**
02 $\quad V_{Extension} \leftarrow \{u \in N(\{v\}) : u > v\}$
03 $\quad$ **call** ExtendSubgraph$(\{v\}, V_{Extension}, v)$
04 **return**

ExtendSubgraph$(V_{Subgraph}, V_{Extension}, v)$
E1 **if** $|V_{Subgraph}| = k$ **then output** $G[V_{Subgraph}]$ and **return**
E2 **while** $V_{Extension} \neq \emptyset$ **do**
E3 $\quad$ Remove an arbitrarily chosen vertex w from $V_{Extension}$
E4 $\quad V'_{Extension} \leftarrow V_{Extension} \cup \{u \in N_{excl}(w, V_{Subgraph}) : u > v\}$
E5 $\quad$ **call** ExtendSubgraph$(V_{Subgraph} \cup \{w\}, V'_{Extension}, v)$
E6 **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 $N(V_{Subgraph})$

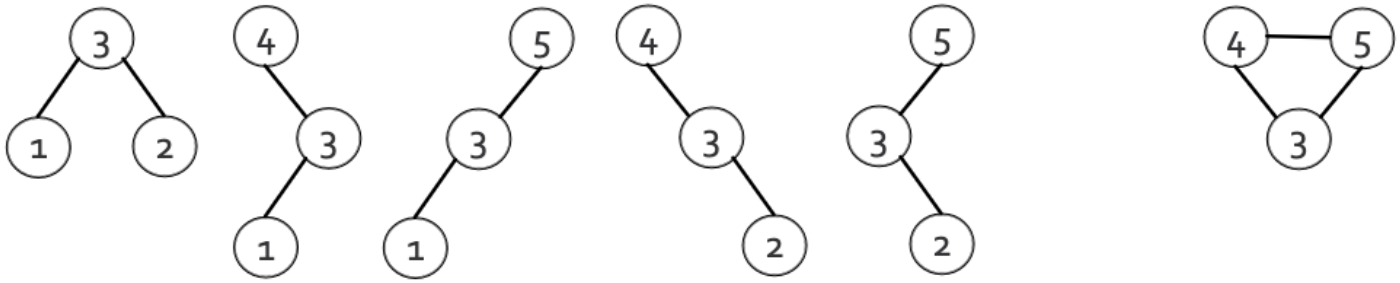
ESU-tree example



- Nodes in the ESU-tree include two adjoining sets:
  - $V_{subgraph}$: Current subgraph (a set of adjacent nodes)
  - $V_{extension}$: Nodes adjacent to $V_{subgraph}$ whose node_ids are larger than starting node v

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

ESU-tree to Count Subgraphs

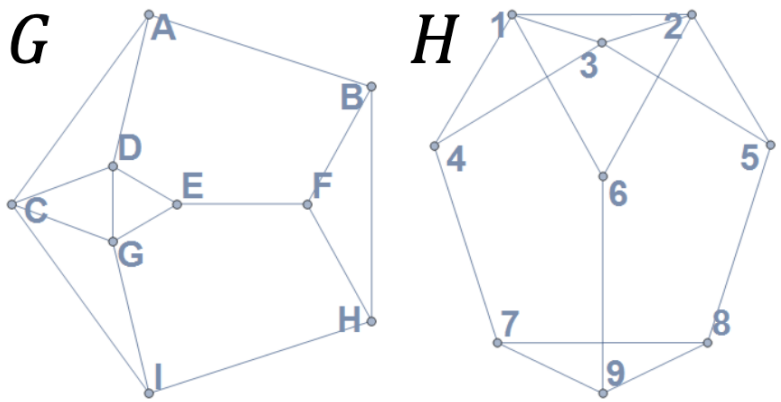
- So far, we enumerated all size-k subgraphs in the input graph
- Next step: Count the graphs

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 $f : V(G) \to V(H)$ such that:
  - Any two nodes u and v of G are adjacent in H if $f(u)$ and $f(v)$ are adjacent in H
- Example: Are G and V topologically equivalent?



Need to check 9! possible bijections between node sets

Hard computational problem!

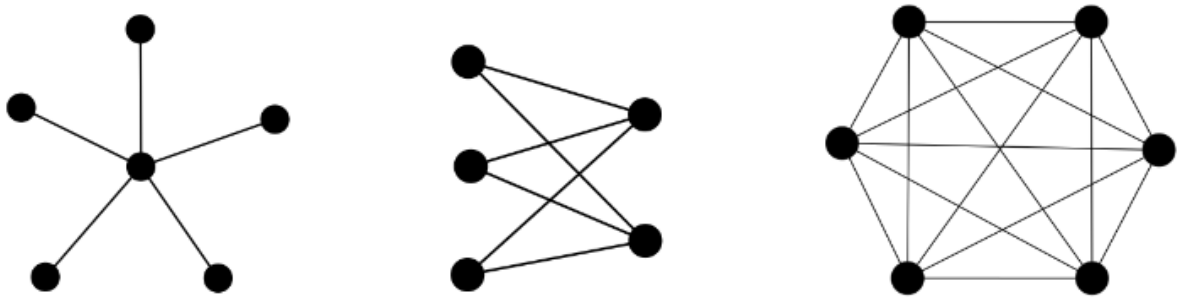
f:

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

***G* and *H* are isomorphic!**

Graph Isomorphism

- In order for adjacent vertices to be structurally equivalent, they should have self loops.
- Sometimes called "strong structural equivalence"
- Sometimes relax requirements for self loops for adjacent nodes



Similarity measures

- Jaccard similarity

$$J(v_i, v_j) = \frac{|\mathcal{N}(v_i) \cap \mathcal{N}(v_j)|}{|\mathcal{N}(v_i) \cup \mathcal{N}(v_j)|}$$

- Cosine similarity (vectors in n-dim space)

$$\sigma(v_i, v_j) = cos(\theta_{ij}) = \frac{\mathbf{v}_i^T \mathbf{v}_j}{|\mathbf{v}_i||\mathbf{v}_j|} = \frac{\sum_k A_{ik} A_{kj}}{\sqrt{\sum A_{ik}^2}\sqrt{\sum A_{jk}^2}}$$

- Pearson correlation coefficient:

$$r_{ij} = \frac{\sum_k (A_{ik} - \langle A_i \rangle)(A_{jk} - \langle A_j \rangle)}{\sqrt{\sum_k (A_{ik} - \langle A_i \rangle)^2}\sqrt{\sum_k (A_{jk} - \langle A_j \rangle)^2}}$$

Similarity measures

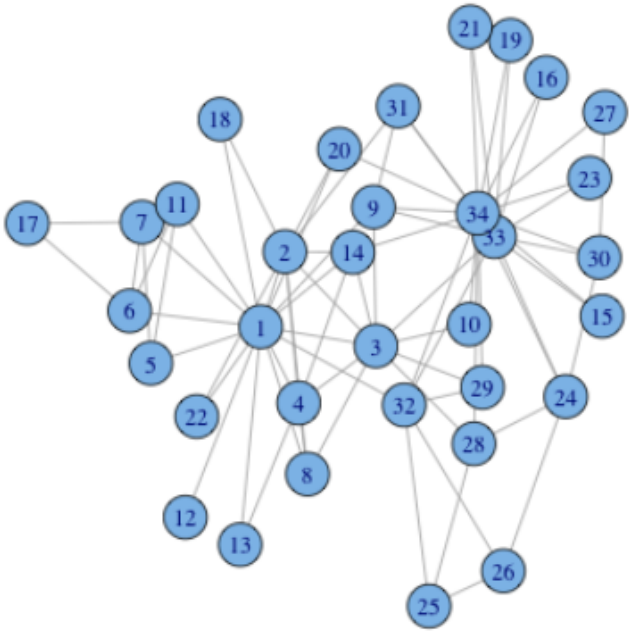
- Unweighted undirected graph $A_{ik} = A_{ki}$, binary matrix, only 0 and 1
- $\sum_k A_{ik} = \sum_k A_{ik}^2 = k_i$ - node degree
- $\sum_k A_{ik}A_{kj} = (A^2)_{ij} = n_{ij}$ - number of shared neighbors
- Cosine similarity (vectors in n-dim space)

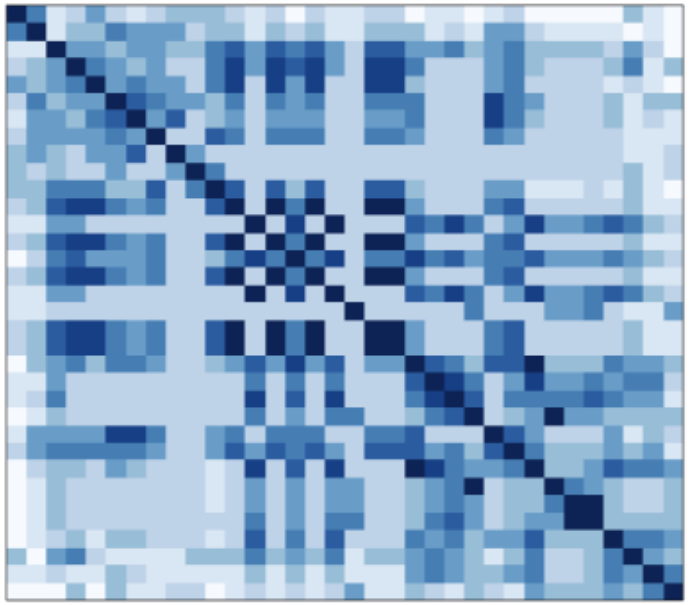
$$\sigma(v_i, v_j) = cos(\theta_{ij}) = \frac{n_{ij}}{\sqrt{k_i k_j}}$$

- Pearson correlation coefficient:

$$r_{ij} = \frac{n_{ij} - \frac{k_i k_j}{n}}{\sqrt{k_i - \frac{k_i^2}{n}}\sqrt{k_j - \frac{k_j^2}{n}}}$$

Similarity matrix





Graph

Node similarity matrix