Chapter 7. Advanced Frequent Pattern Mining: Sequential Patterns and Graph Patterns

Meng Jiang

CSE 40647/60647 Data Science Fall 2017

Introduction to Data Mining

Pattern Mining Methods

Pattern	Closed Pattern (Concepts)	Idea 1: Pattern candidate generation and pruning	Idea 2: Vertical format to accelerate mining	Idea 3: Pattern growth
Frequent pattern (itemset)	?	?	?	?
Sequential pattern	?	?	?	?
Graph pattern	?	?	x	?

Pattern Mining Methods

Pattern	**Closed Pattern (Concepts)**	**Idea 1: Pattern candidate generation and pruning**	**Idea 2: Vertical format to accelerate mining**	**Idea 3: Pattern growth**
Frequent pattern (itemset)	Closed frequent itemset	Apriori	ECLAT	FP-Growth
Sequential pattern	Closed seq. pattern	?	?	?
Graph pattern	Closed graph pattern	?	x	?

Advanced Frequent Pattern Mining

- Mining Diverse Patterns
- Constraint-Based Frequent Pattern Mining
- **Sequential Pattern Mining**
- Graph Pattern Mining

Sequential Patterns: Applications

- Sequential pattern mining has broad applications
  - Customer shopping sequences
    - Purchase a laptop first, then a digital camera, and then a smartphone, within 6 months
  - Medical treatments, natural disasters (e.g., earthquakes), science & engineering processes, stocks and markets, …
  - Weblog click streams, calling patterns, …
  - Software engineering: Program execution sequences, …
  - Biological sequences: DNA, protein, …

Sequential Pattern and Sequential Pattern Mining

- Sequential pattern mining: Given a set of sequences, find the complete set of frequent subsequences (i.e., satisfying the min_sup threshold)

A *sequence database*

SID	Sequence
10	<a(abc)(ac)d(cf)>
20	<(ad)c(bc)(ae)>
30	<(ef)(ab)(df)cb>
40	<eg(af)cbc>

A *sequence*: < (ef) (ab) (df) c b >

- An element may contain a set of items (also called events)
- Items within an element are unordered and we list them alphabetically

<a(bc)dc> is a *subsequence* of <a(abc)(ac)d(cf)>

- Given support threshold min_sup = 2, <(ab)c> is a sequential pattern

http://hanj.cs.illinois.edu/pdf/spano1.pdf

Sequence vs Element/Itemset/Event vs Item/Instance

Let $I = \{i_1, i_2, \ldots, i_n\}$ be a set of all **items**. An **itemset** is a subset of items. A **sequence** is an ordered list of itemsets. A sequence s is denoted by $\langle s_1 s_2 \cdots s_l \rangle$, where s_j is an itemset, i.e., $s_j \subseteq I$ for $1 \leq j \leq l$. s_j is also called an **element** of the sequence, and denoted as $(x_1 x_2 \cdots x_m)$, where x_k is an item, i.e., $x_k \in I$ for $1 \leq k \leq m$. For brevity, the brackets are omitted if an element has only one item. That is, element (x) is written as x. An item can occur at most once in an element of a sequence, but can occur multiple times in different elements of a sequence. The

Sequential Pattern Mining Algorithms

- Algorithm requirement: Efficient, scalable, finding complete set, incorporating various kinds of user-specific constraints
- The Apriori property still holds: If a subsequence s_1 is infrequent, none of s_1's super-sequences can be frequent
- Representative algorithms
  - Apriori-based Generalized Sequential Patterns: GSP (Srikant & Agrawal @ EDBT'96)
  - Vertical format-based mining: SPADE (Zaki@Machine Leanining'00)
  - Pattern-growth methods: PrefixSpan (Pei, et al. @TKDE'04)
- Mining **closed** sequential patterns: CloSpan (Yan, et al. @SDM'03)
- Constraint-based sequential pattern mining

GSP: Apriori-Based Sequential Pattern Mining

- Initial candidates: All singleton sequences
  - <a>, <b>, <c>, <d>, <e>, <f>, <g>, <h>
- Scan DB once, count support for each candidate
- Generate length-2 candidate sequences

SID	Sequence
10	<(bd)cb(ac)>
20	<(bf)(ce)b(fg)>
30	<(ah)(bf)abf>
40	<(be)(ce)d>
50	<a(bd)bcb(ade)>

min_sup = 2

Cand.	sup
<a>	3
<b>	5
<c>	4
<d>	3
<e>	3
<f>	2
<g>	1
<h>	1

	<a>	<b>	<c>	<d>	<e>	<f>
<a>	<aa>	<ab>	<ac>	<ad>	<ae>	<af>
<b>	<ba>	<bb>	<bc>	<bd>	<be>	<bf>
<c>	<ca>	<cb>	<cc>	<cd>	<ce>	<cf>
<d>	<da>	<db>	<dc>	<dd>	<de>	<df>
<e>	<ea>	<eb>	<ec>	<ed>	<ee>	<ef>
<f>	<fa>	<fb>	<fc>	<fd>	<fe>	<ff>

Length-2 candidates:
36 + 15= 51
Without Apriori pruning:
8*8+8*7/2=92 candidates

	<a>	<b>	<c>	<d>	<e>	<f>
<a>		<(ab)>	<(ac)>	<(ad)>	<(ae)>	<(af)>
<b>			<(bc)>	<(bd)>	<(be)>	<(bf)>
<c>				<(cd)>	<(ce)>	<(cf)>
<d>					<(de)>	<(df)>
<e>						<(ef)>
<f>						

GSP (Generalized Sequential Patterns): Srikant & Agrawal @ EDBT'96)

GSP Mining and Pruning

- Repeat (for each level (i.e., length-k))
  - Scan DB to find length-k frequent sequences
  - Generate length-(k+1) candidate sequences from length-k frequent sequences using Apriori
  - set k = k+1
- Until no frequent sequence or no candidate can be found

Sequential Pattern Mining in Vertical Data Format: The SPADE Algorithm

- A sequence database is mapped to: <SID, EID>
- Grow the subsequences (patterns) one item at a time by Apriori candidate generation

SID	Sequence
1	<a(abc)(ac)d(cf)>
2	<(ad)c(bc)(ae)>
3	<(ef)(ab)(df)cb>
4	<eg(af)cbc>

min_sup = 2

Ref: SPADE (Sequential PAttern Discovery using Equivalent Class) [M. Zaki 2001]

SID	EID	Items
1	1	a
1	2	abc
1	3	ac
1	4	d
1	5	cf
2	1	ad
2	2	c
2	3	bc
2	4	ae
3	1	ef
3	2	ab
3	3	df
3	4	c
3	5	b
4	1	e
4	2	g
4	3	af
4	4	c
4	5	b
4	6	c

a

SID	EID
1	1
1	2
1	3
2	1
2	4
3	2
4	3

b

SID	EID
1	2
2	3
3	2
3	5
4	5

ab

SID	EID (a)	EID(b)
1	1	2
2	1	3
3	2	5
4	3	5

ba

SID	EID (b)	EID(a)
1	2	3
2	3	4

aba

SID	EID (a)	EID(b)	EID(a)
1	1	2	3
2	1	3	4

https://pdfs.semanticscholar.org/39a0/8c17dec400a6f04af5fe5746dab3a5ebodc.pdf

PrefixSpan: A Pattern-Growth Approach

- Prefix and suffix
  - Given <a(abc)(ac)d(cf)>
  - Prefixes: <a>, <aa>, <a(ab)>, <a(abc)>, …
  - Prefixes-based projection
- PrefixSpan Mining: Prefix Projections
  - Step 1: Find length-1 sequential patterns
    - <a>, <b>, <c>, <d>, <e>, <f>
  - Step 2: Divide search space and mine each projected DB
    - <a>-projected DB,
    - <b>-projected DB,
    - …
    - <f>-projected DB, …

SID	Sequence
10	<a(abc)(ac)d(cf)>
20	<(ad)c(bc)(ae)>
30	<(ef)(ab)(df)cb>
40	<eg(af)cbc>

Prefix	Suffix (Projection)
<a>	<(abc)(ac)d(cf)>
<aa>	<(_bc)(ac)d(cf)>
<ab>	<(_c)(ac)d(cf)>

PrefixSpan (Prefix-projected Sequential pattern mining) Pei, et al. @TKDE'04

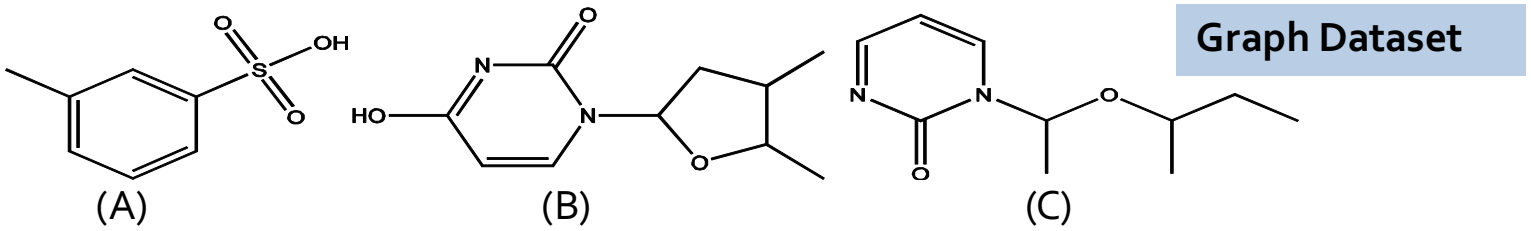
http://hanj.cs.illinois.edu/pdf/span01.pdf

Advanced Frequent Pattern Mining

- Mining Diverse Patterns
- Constraint-Based Frequent Pattern Mining
- Sequential Pattern Mining
- **Graph Pattern Mining**

Frequent (Sub)Graph Patterns

- Given a labeled graph dataset $D = \{G_1, G_2, \ldots, G_n\}$, the supporting graph set of a subgraph g is $D_g = \{G_i \mid g \subseteq G_i, G_i \in D\}$.
  - $support(g) = |D_g| / |D|$
- A (sub)graph g is ***frequent*** if $support(g) \geq min\_sup$ Ex.: Chemical structures
- Alternative:
  - Mining frequent subgraph patterns from a single large graph or network



Graph Pattern Mining: Applications

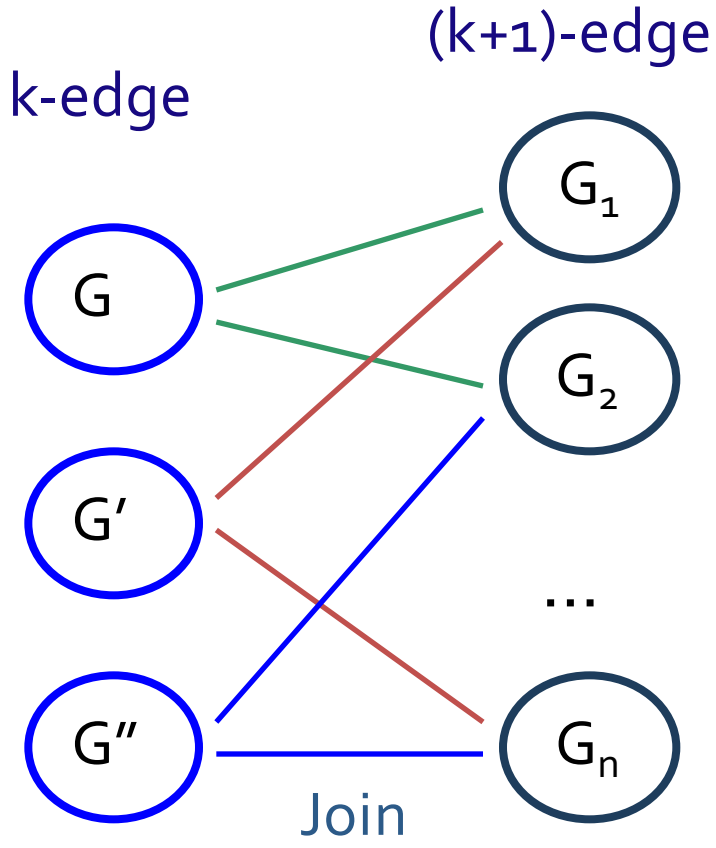
- Bioinformatics
  - Gene networks, protein interactions, metabolic pathways
- Chem-informatics: Mining chemical compound structures
- Social networks, web communities, tweets, …
- Cell phone networks, computer networks, …
- Web graphs, XML structures, semantic Web, information networks
- Software engineering: program execution flow analysis
- Building blocks for graph classification, clustering, compression, comparison, and correlation analysis
- Graph indexing and graph similarity search

Graph Pattern Mining Algorithms: Different Methodologies

- Generation of candidate subgraphs
  - Apriori vs. pattern growth (e.g., FSG vs. gSpan)
- Search order
  - Breadth vs. depth
- Elimination of duplicate subgraphs
  - Passive vs. active (e.g., gSpan (Yan&Han'02))
- Order of pattern discovery
  - Path → tree → graph (e.g., GASTON (Nijssen&Kok'04))

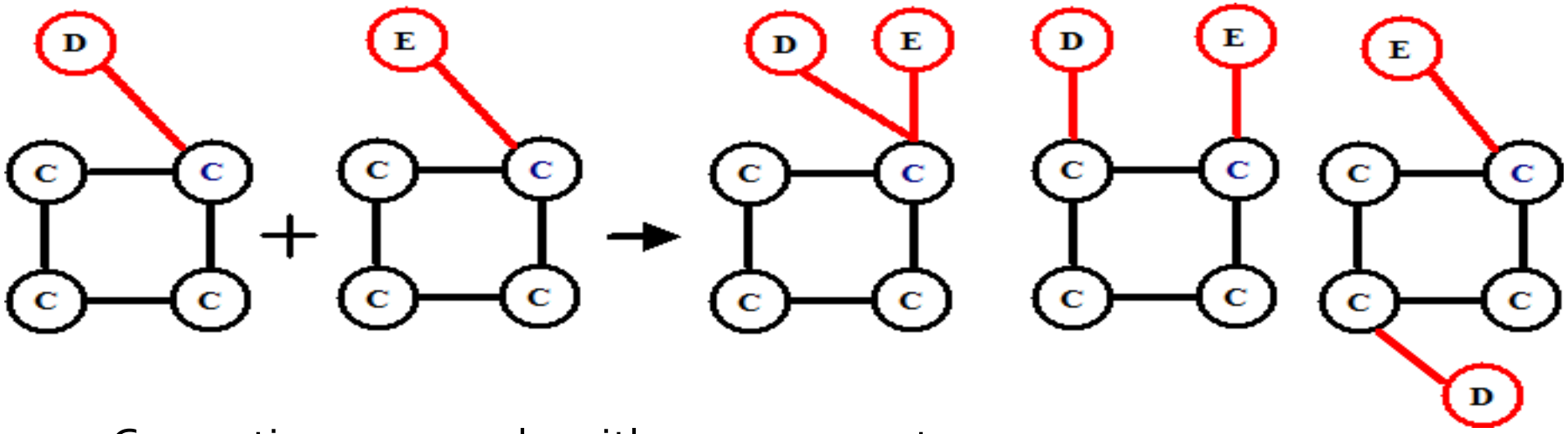
Apriori-Based Approach

- The Apriori property (anti-monotonicity): A size-*k* subgraph is frequent if and only if all of its subgraphs are frequent
- A candidate size-(*k*+1) edge/vertex subgraph is generated if its corresponding two *k*-edge/vertex subgraphs are frequent
- Iterative mining process:
  - Candidate-generation → candidate pruning → support counting → candidate elimination



Candidate Generation: Vertex Growing vs. Edge Growing

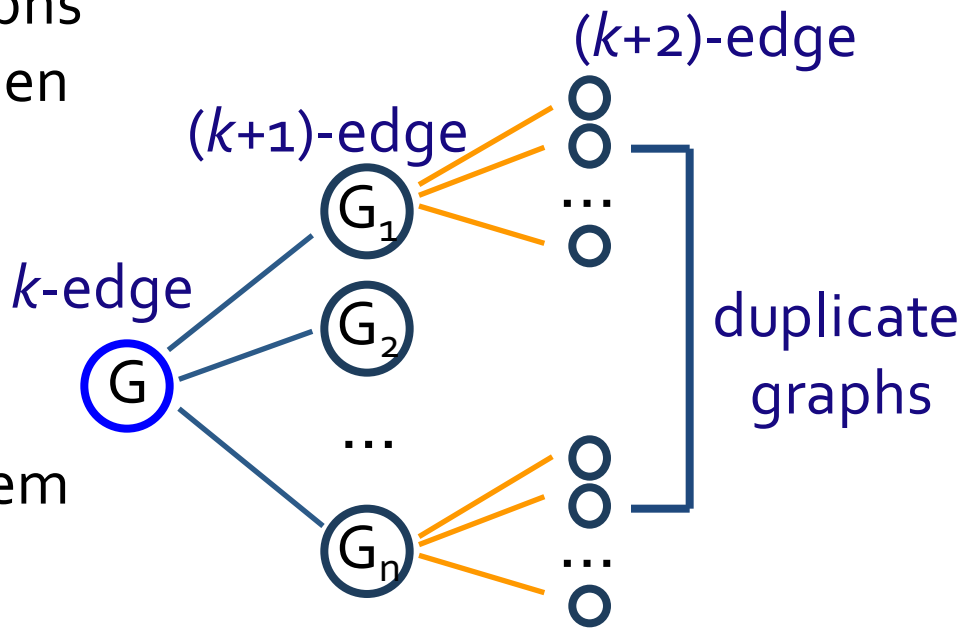
- Methodology: breadth-search, Apriori joining two size-k graphs
  - Many possibilities at generating size-(k+1) candidate graphs



- Generating new graphs with one more vertex
  - AGM (Inokuchi, et al., PKDD'00)
- Generating new graphs with one more edge
  - FSG (Kuramochi and Karypis, ICDM'01)
- Performance shows via edge growing is more efficient

Pattern-Growth Approach

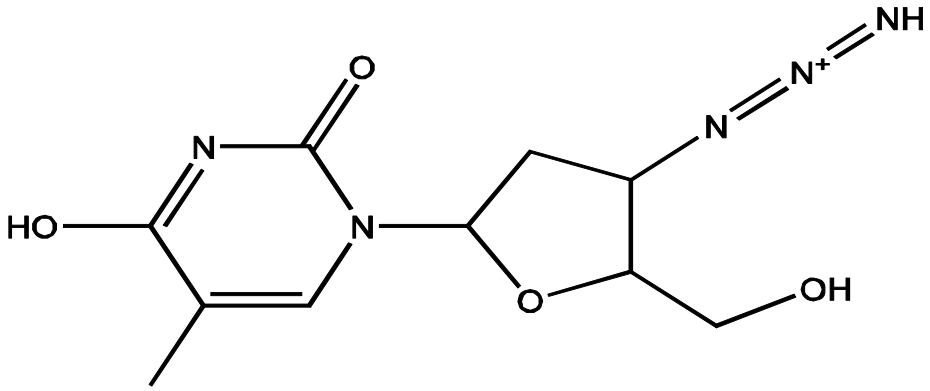
- Depth-first growth of subgraphs from *k*-edge to (*k*+1)-edge, then (*k*+2)-edge subgraphs
- Major challenge
  - Generating many duplicate subgraphs
- Major idea to solve the problem
  - Define an order to generate subgraphs
  - DFS spanning tree: Flatten a graph into a sequence using depth-first search
  - gSpan (Yan & Han: ICDM'02)



http://cs.ucsb.edu/~xyan/papers/gSpan-short.pdf

Why Mining Closed Graph Patterns?

- Challenge: An **n**-edge frequent graph may have 2^n subgraphs
- Motivation: Explore *closed frequent subgraphs* to handle graph pattern explosion problem
- A frequent graph G is *closed* if there exists no supergraph of G that carries the same support as G

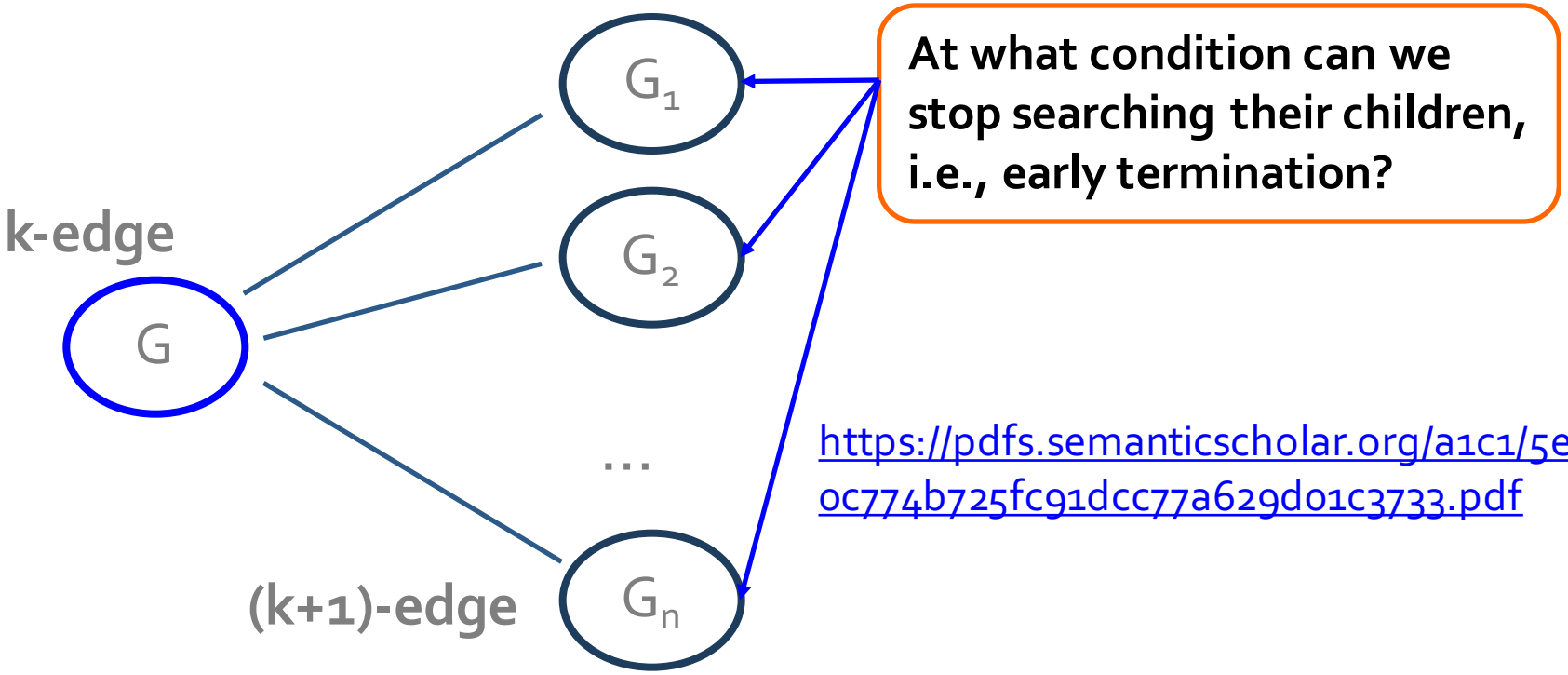


If this subgraph is *closed* in the graph dataset, it implies that none of its frequent super-graphs carries the same support

- *Lossless compression:* Does not contain non-closed graphs, but still ensures that the mining result is complete
- Algorithm CloseGraph: Mines closed graph patterns directly

CloseGraph: Directly Mining Closed Graph Patterns

- CloseGraph: Mining closed graph patterns by extending gSpan
- Suppose G and G_1 are frequent, and G is a subgraph of G_1
- If **in any part of the graph in the dataset where G occurs, G_1 also occurs**, then we need not grow G (except some special, subtle cases), since none of G's children will be closed except those of G_1

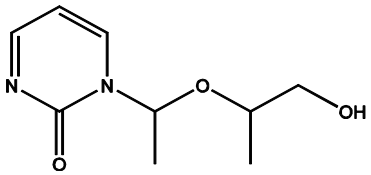


https://pdfs.semanticscholar.org/a1c1/5e6369oc774b725fc91dcc77a629d01c3733.pdf

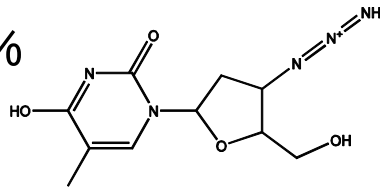
Experiment and Performance Comparison

- The AIDS antiviral screen compound dataset from NCI/NIH
- The dataset contains 43,905 chemical compounds
- Discovered Patterns: The smaller minimum support, the bigger and more interesting subgraph patterns discovered

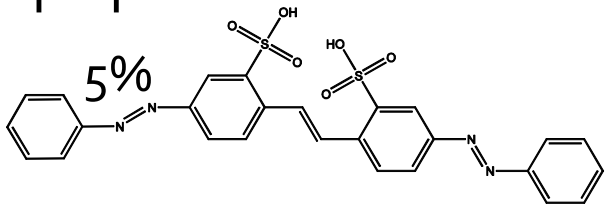
20%



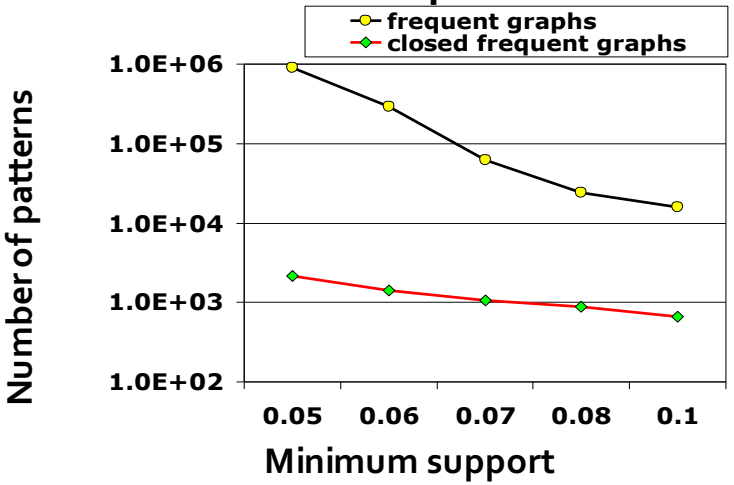
10%



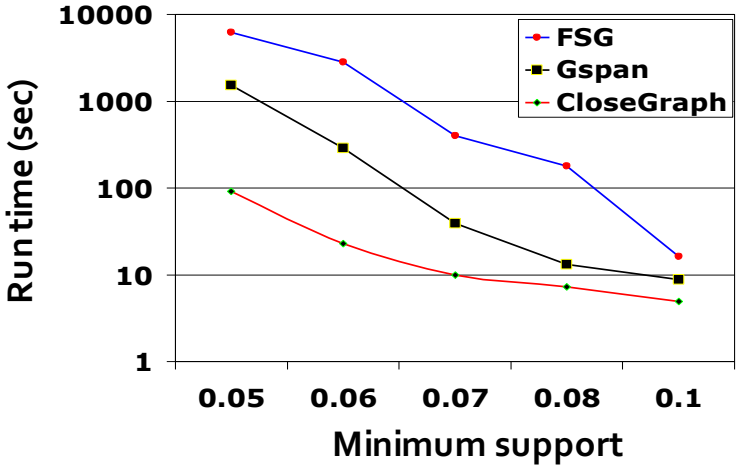
5%



# of Patterns: Frequent vs. Closed



Runtime: Frequent vs. Closed



References: Sequential Pattern Mining

- R. Srikant and R. Agrawal, "Mining sequential patterns: Generalizations and performance improvements", EDBT'96
- M. Zaki, "SPADE: An Efficient Algorithm for Mining Frequent Sequences", Machine Learning, 2001
- J. Pei, J. Han, B. Mortazavi-Asl, J. Wang, H. Pinto, Q. Chen, U. Dayal, and M.-C. Hsu, "Mining Sequential Patterns by Pattern-Growth: The PrefixSpan Approach", IEEE TKDE, 16(10), 2004
- X. Yan, J. Han, and R. Afshar, "CloSpan: Mining Closed Sequential Patterns in Large Datasets", SDM'03
- J. Pei, J. Han, and W. Wang, "Constraint-based sequential pattern mining: the pattern-growth methods", J. Int. Inf. Sys., 28(2), 2007
- M. N. Garofalakis, R. Rastogi, K. Shim: Mining Sequential Patterns with Regular Expression Constraints. IEEE Trans. Knowl. Data Eng. 14(3), 2002
- H. Mannila, H. Toivonen, and A. I. Verkamo, "Discovery of frequent episodes in event sequences", Data Mining and Knowledge Discovery, 1997

References: Graph Pattern Mining

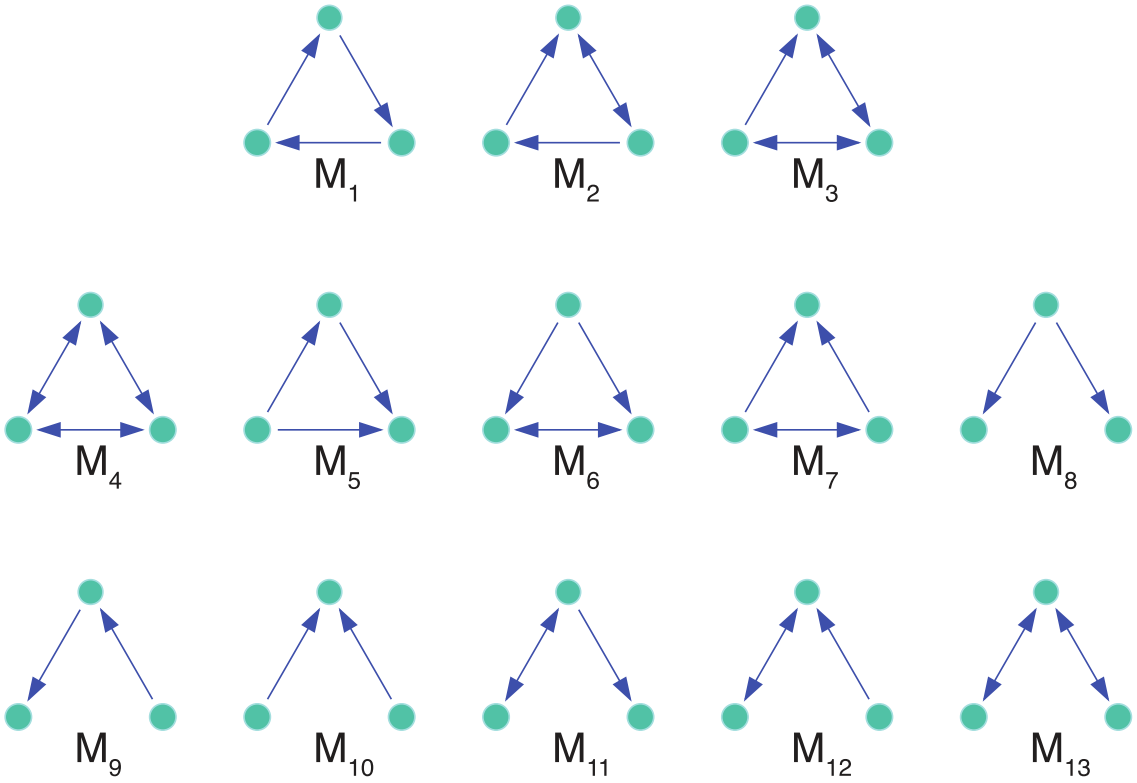
- C. Borgelt and M. R. Berthold, Mining molecular fragments: Finding relevant substructures of molecules, ICDM'02
- J. Huan, W. Wang, and J. Prins. Efficient mining of frequent subgraph in the presence of isomorphism, ICDM'03
- A. Inokuchi, T. Washio, and H. Motoda. An apriori-based algorithm for mining frequent substructures from graph data, PKDD'00
- M. Kuramochi and G. Karypis. Frequent subgraph discovery, ICDM'01
- S. Nijssen and J. Kok. A Quickstart in Frequent Structure Mining can Make a Difference. KDD'04
- N. Vanetik, E. Gudes, and S. E. Shimony. Computing frequent graph patterns from semistructured data, ICDM'02
- X. Yan and J. Han, gSpan: Graph-Based Substructure Pattern Mining, ICDM'02
- X. Yan and J. Han, CloseGraph: Mining Closed Frequent Graph Patterns, KDD'03
- X. Yan, P. S. Yu, J. Han, Graph Indexing: A Frequent Structure-based Approach, SIGMOD'04
- X. Yan, P. S. Yu, and J. Han, Substructure Similarity Search in Graph Databases, SIGMOD'05

Higher-Order Organization of Complex Networks

Austin Benson, David Gleich,
Jure Leskovec

Higher-Order Structures

Small subgraphs (motifs, graphlets) are building blocks of networks



Higher-Order Structures

Higher-order structures are fundamental for control and function of complex systems:

- Social networks [Watts, Strogatz '98]
- Metabolic networks [Milo et al. '02]
- Protein networks [Alon '07] [Vinayagama et al.'16]
- Transportation networks [Rosvall '14]
- Neural networks [Park, Friston '13]
- Food webs [Stouffer et al. '12]

Higher-Order Organization

Prior work is based on simply counting frequencies of individual subgraphs

We still lack understanding of higher-order organization of complex systems

This talk: Higher-order Organization

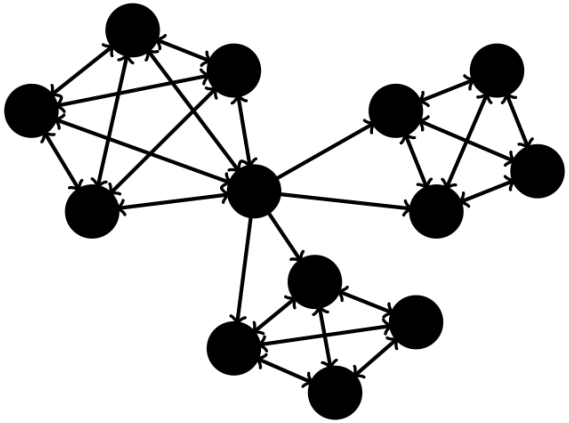
What is higher order organization of networks?

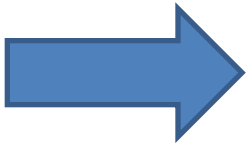
How do we extract it?

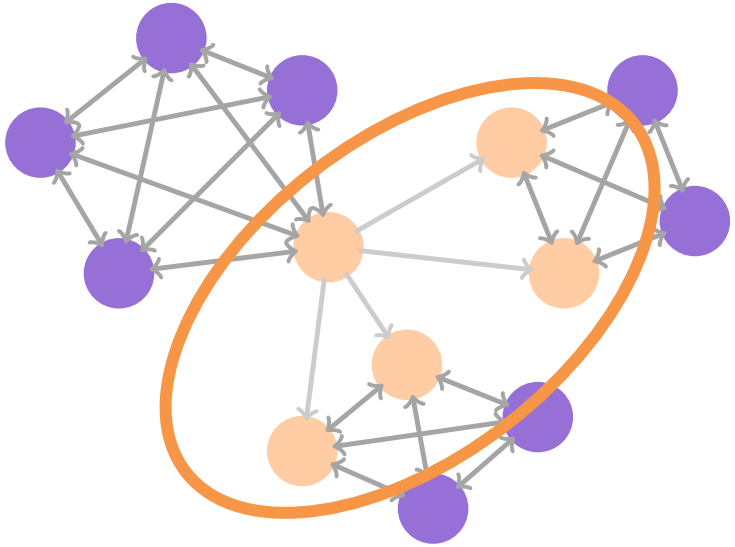
This talk: Modules of Motifs

Find modules based on motifs!

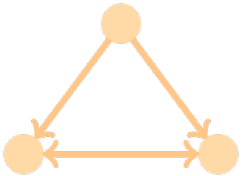
Network:





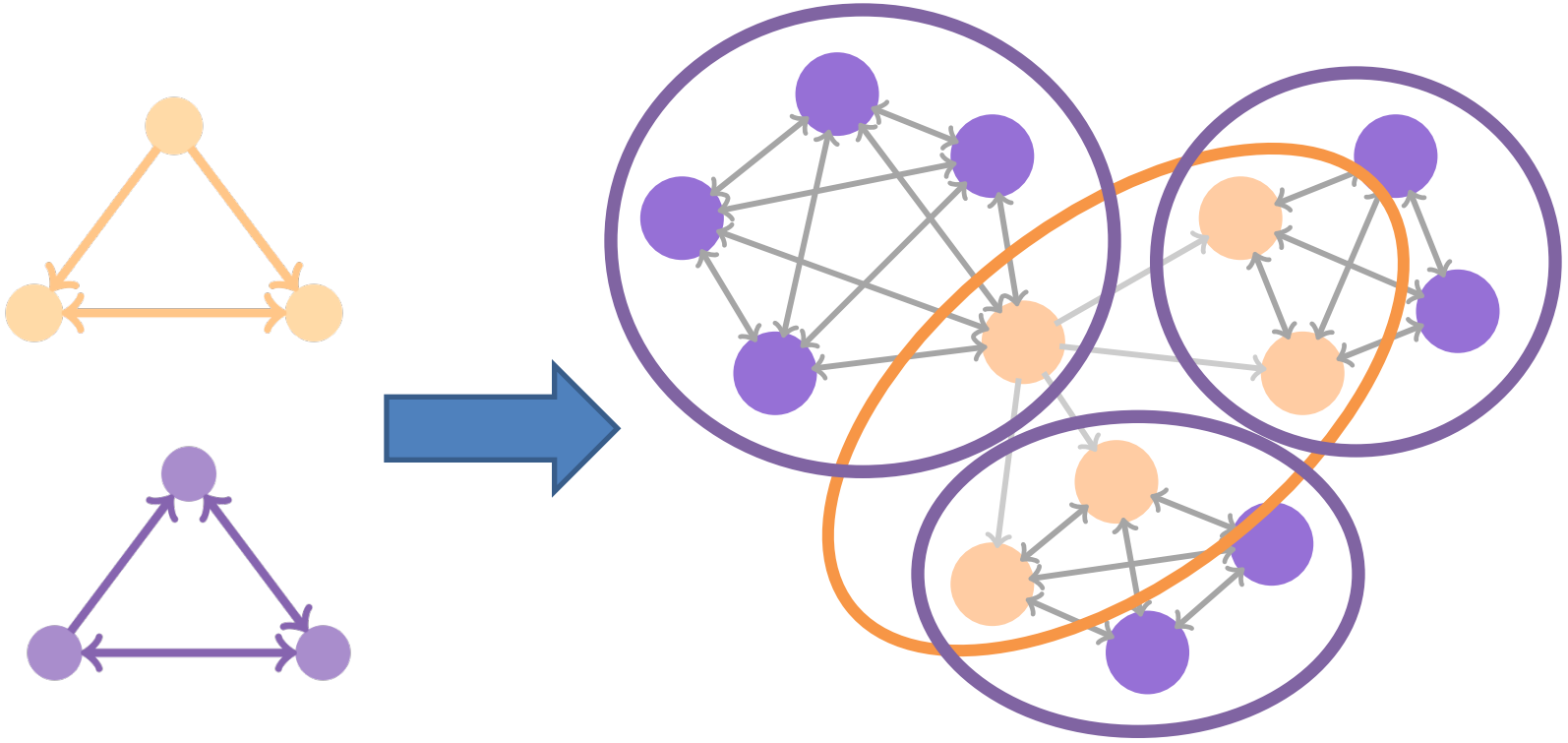


Motif:



[Serrour, Arenas, Gomez '11]

Modules of Motifs

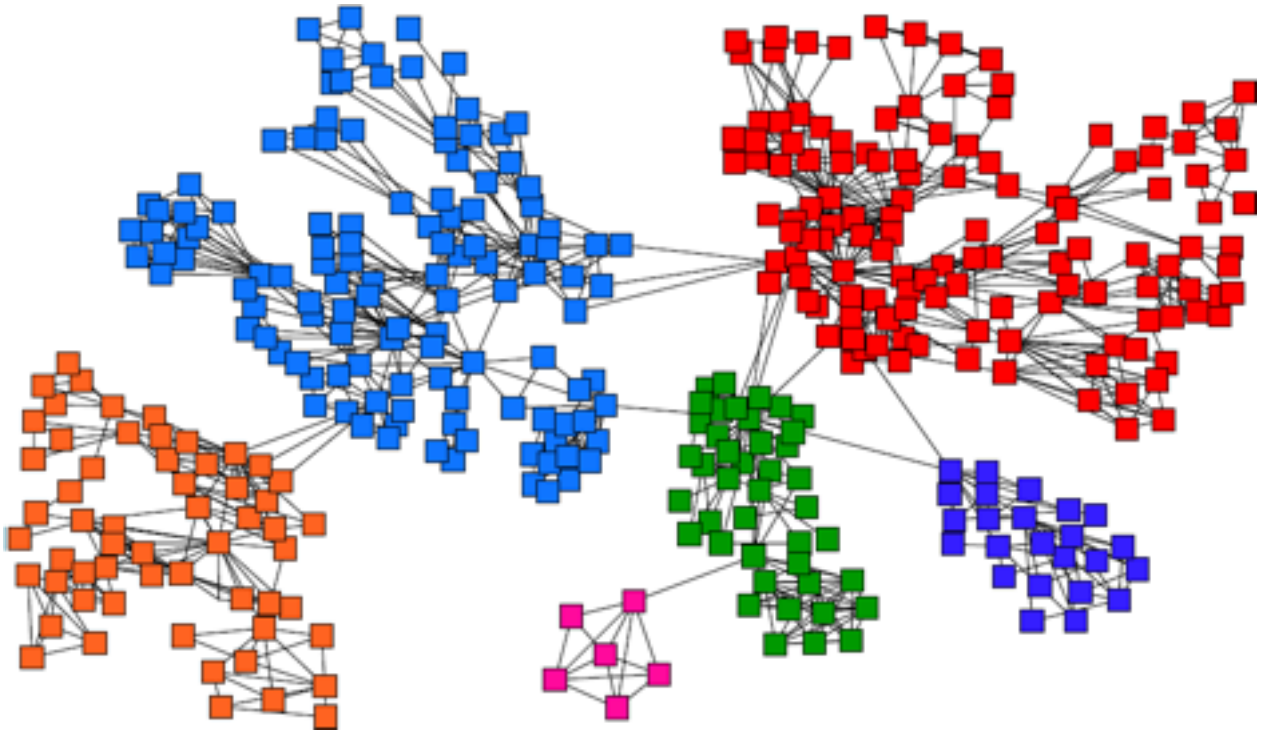


Different motifs reveal different modular structures!

How do we find modules of network motifs?

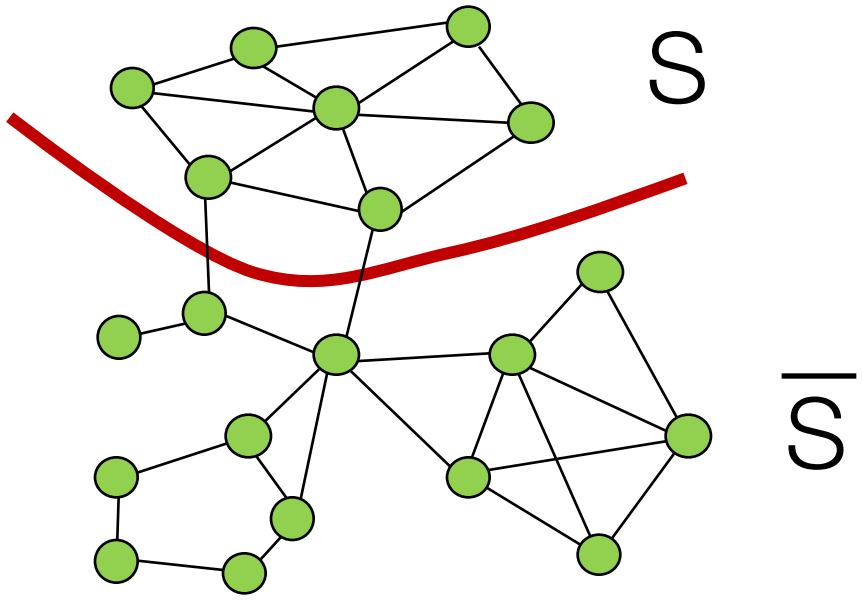
Background: Communities

- Old idea: Find densely connected sets of nodes



Background: Communities

- Define an objective function $\phi(S)$



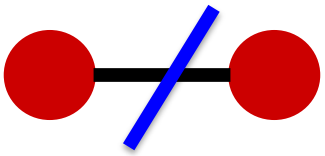
- Good community S cuts few edges while keeping many inside the set S

Defining an Objective Function

Conductance of a cluster S:

$$\phi(S) = \frac{\#(edges\ cut)}{vol(S)}$$

$edges\ cut$:



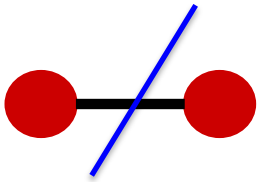
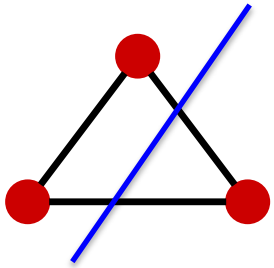
$vol(S)$ = #(edge end points in S)

Lower conductance means a better cluster

How do we generalize conductance to network motifs?

Defining Motif Conductance

Generalize Cut and Volume to motifs:

$edges\ cut$  $\rightarrow$ $motifs\ cut$ 

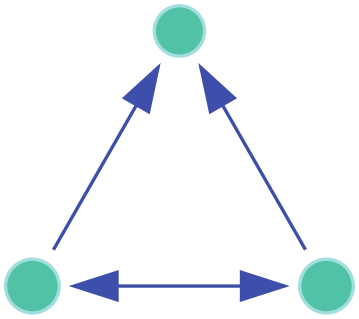
$vol(S)$ = #(edge end-points in S) $\rightarrow$ $vol_M(S)$ = #(motif end-points in S)

$$\phi(S) = \frac{\#(edges\ cut)}{vol(S)} \rightarrow \boxed{\phi(S) = \frac{\#(motifs\ cut)}{vol_M(S)}}$$

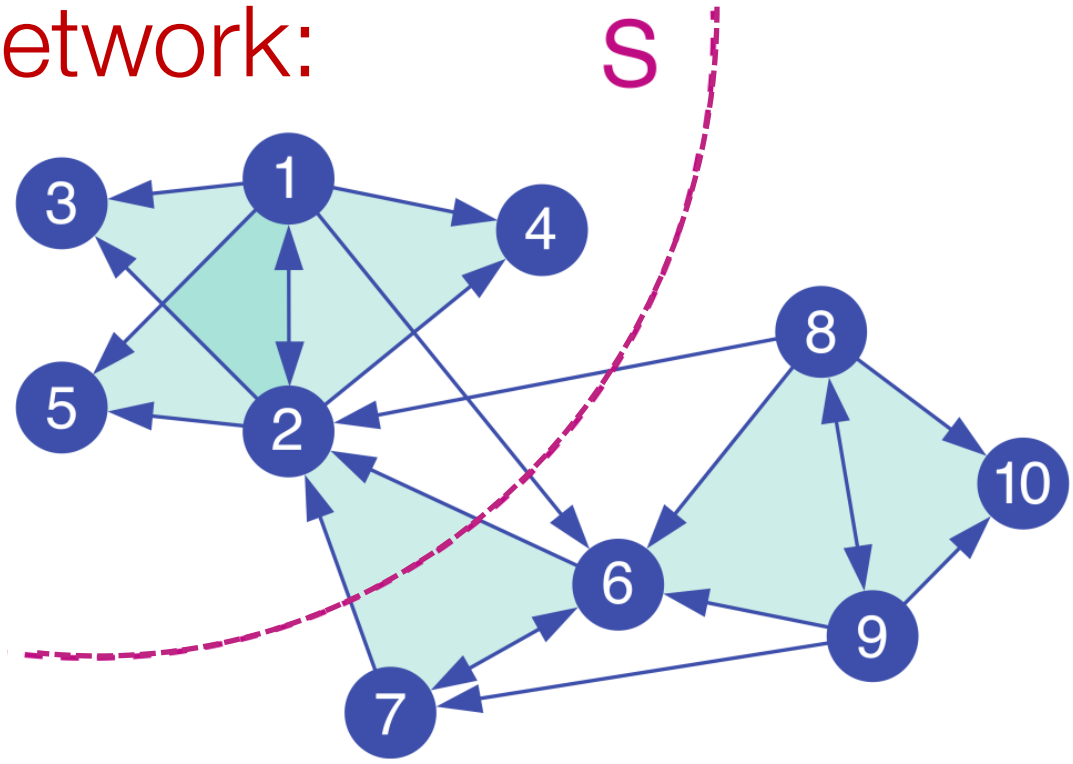
[Benson, Gleich, Leskovec, Science 2016]

Motif Conductance: Example

Motif:



Network:



$$\phi_M(S) = \frac{\text{motifs cut}}{\text{motif volume}}$$

Higher-order Partitioning

How do we find clusters of motifs?

- Given a motif M and a graph G
- Find a set of nodes S that minimizes motif conductance

Bad news: Finding set S with the minimal motif conductance is computationally intractable (NP-hard)!

[Benson, Gleich, Leskovec, Science 2016]

Motif Spectral Clustering

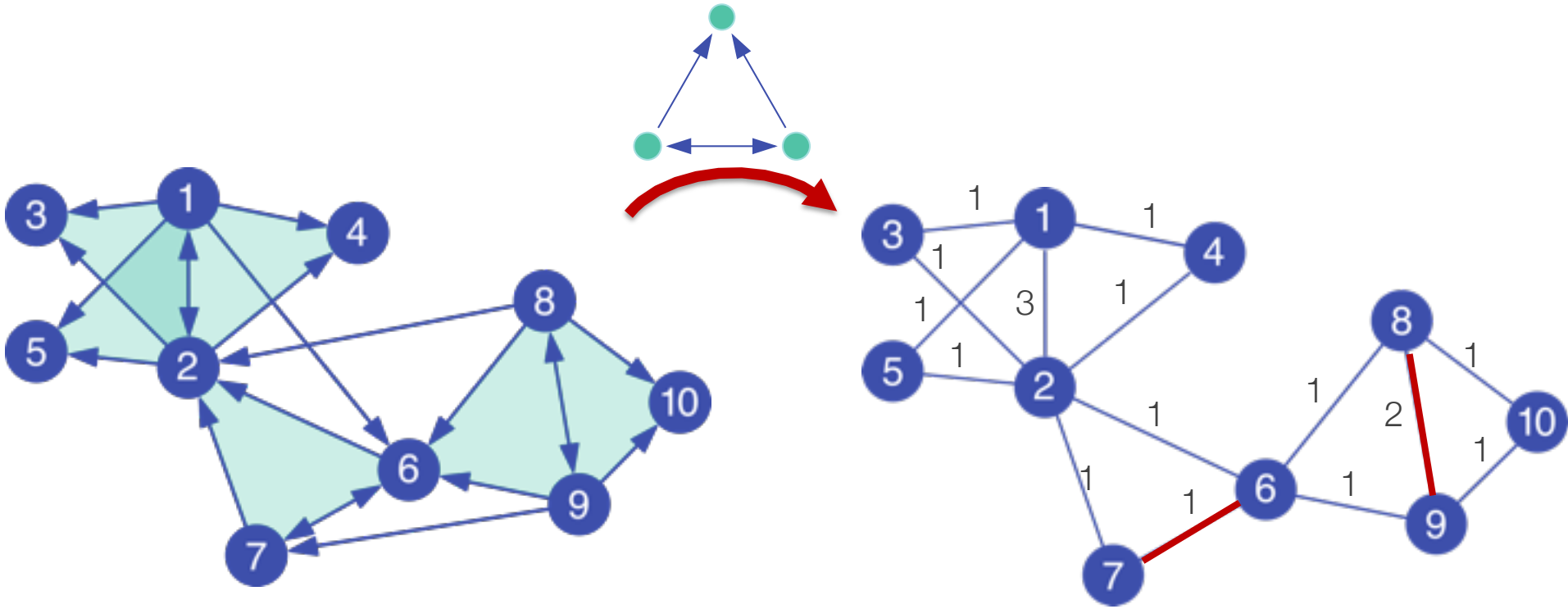
Solution: Motif Spectral Clustering

- Input: Graph G and motif M
- Using G form a new weighted graph W
- Apply spectral clustering on W
- Output the clusters

Theorem: Resulting clusters will obtain near optimal motif conductance

[Benson, Gleich, Leskovec, Science 2016]

Motif Spectral Clustering



Graph G

Weighted graph $W^{(M)}$

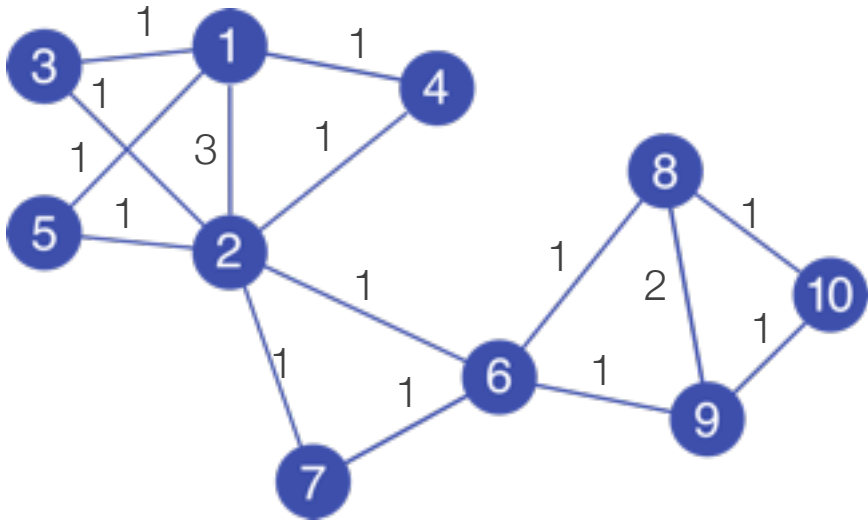
$W_{ij}^{(M)}$ = # of times edge (i,j) participates in motif M

Motif Spectral Clustering

Insight:

Spectral clustering on weighted graph $W^{(M)}$ finds clusters of low motif conductance:

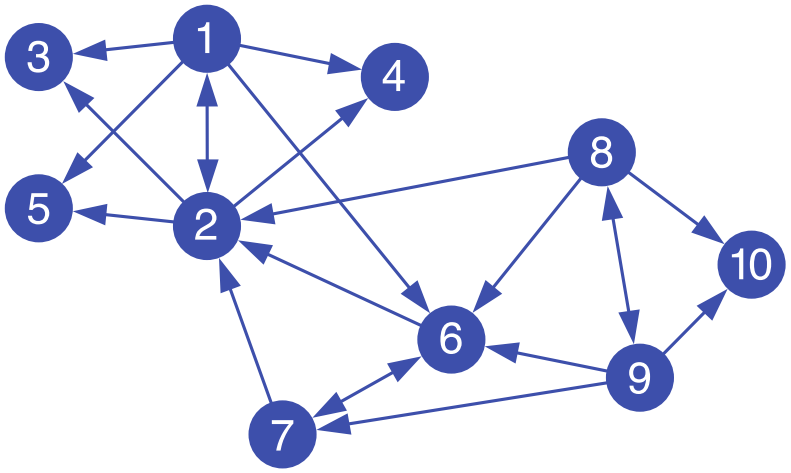
$$\phi_M(S) = \frac{\text{motifs cut}}{\text{motif volume}}$$

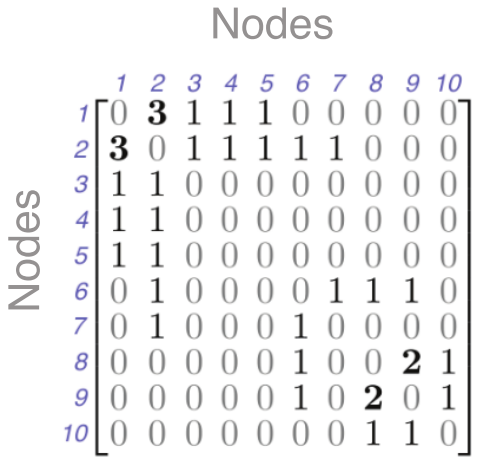


Weighted graph $W^{(M)}$

$W_{ij}^{(M)}$ = # of times edge (i,j) participates in motif M

Motif Spectral Clustering: 1





Step 1: Form weighted graph $W^{(M)}$

Motif Spectral Clustering: 2

$$\mathcal{L}^{(M)} = D^{-1/2}(D - W^{(M)})D^{-1/2}$$

$$\mathcal{L}^{(M)}\mathbf{z} = \lambda_2\mathbf{z}$$

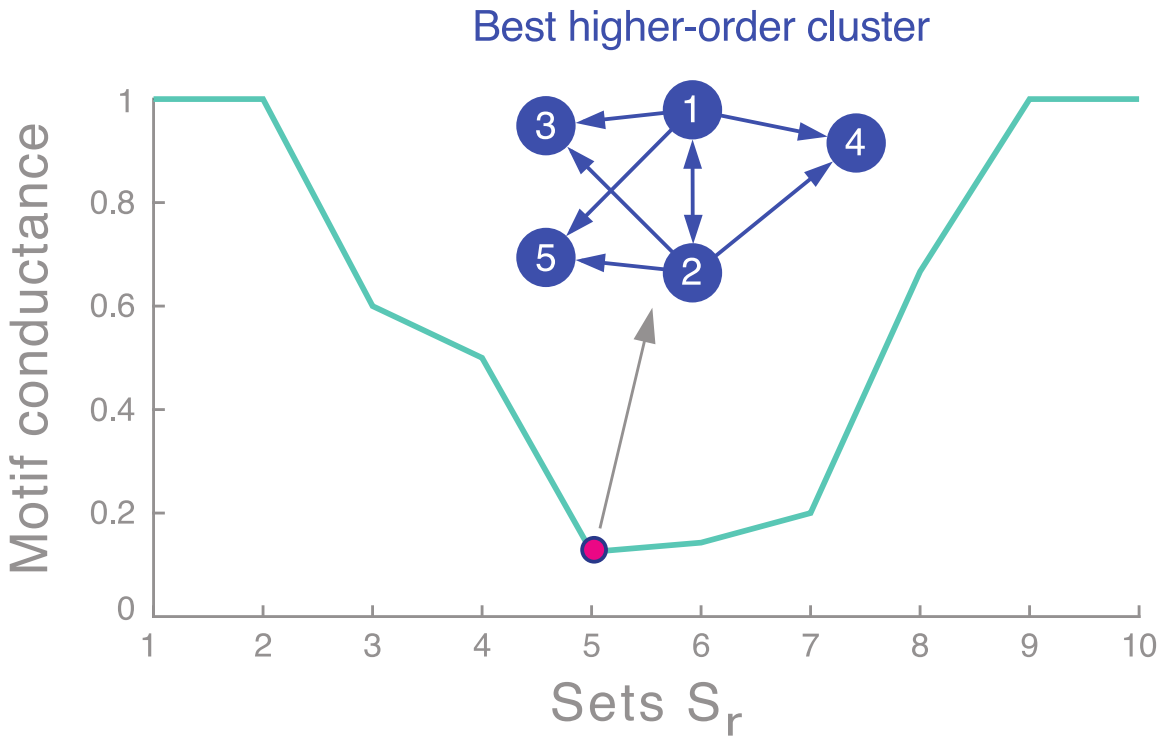
$$\mathbf{f}^{(M)} = D^{-1/2}\mathbf{z}$$

$D = \text{diag}(W^{(M)}\mathbf{e})$

Diagonal degree matrix

Step 2: Compute Fiedler vector $f^{(M)}$ associated with λ_2 of the Laplacian of $W^{(M)}$

Motif Spectral Clustering: 3



Step 3: Sort nodes by values in $f^{(M)}$: $f_1, f_2, \ldots f_n$

Let $S_r = \{f_1, \ldots, f_r\}$ and compute the motif conductance of each S_r

Motif Cheeger Inequality

Theorem: The algorithm finds a set of nodes S for which

$$\phi_M(S) \leq 4\sqrt{\phi_M^*}$$

$\phi_M(S)$… motif conductance of S found by our algorithm

ϕ_M^* … motif conductance of optimal set S*

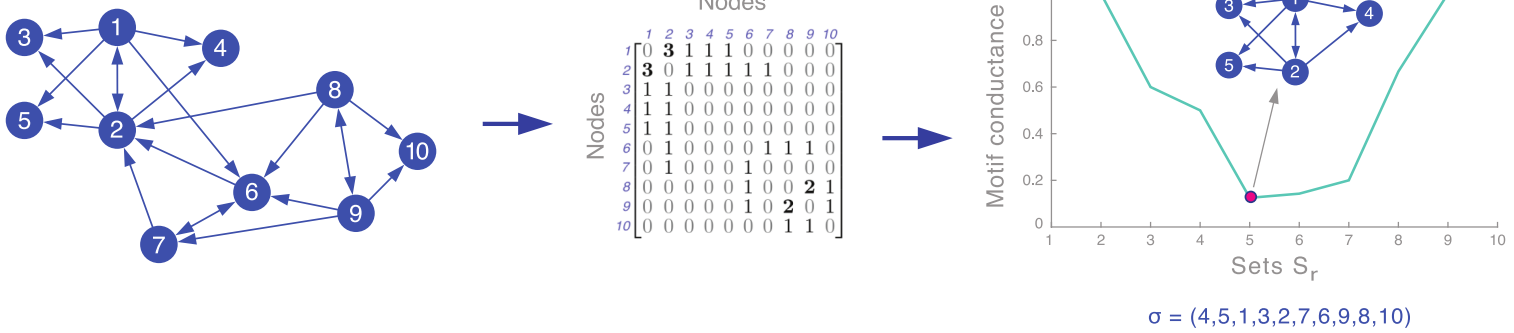
In other words: Clusters S found by our method are provably near optimal

Advantages

- Easy to implement: ~10 lines of Matlab
- Scalable to large networks
  (1.4B edge graphs)
- General: Can be applied to directed, signed, and weighted networks
- Code available:
  http://snap.stanford.edu/higher-order/

Summary

- Generalization of community detection to higher-order structures
- Motif-conductance objective admits a motif Cheeger inequality
- Simple, fast, and scalable:



Applications

1) We don't know a motif of interest
   - Food webs and new applications
2) We know the motif of interest
   - Regulatory transcription networks, connectome, social networks
3) We seek richer information from data
   - Transportation networks

Application 1:

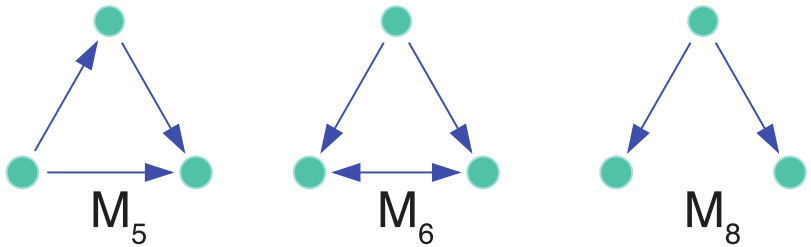
We do not know the motif of interest

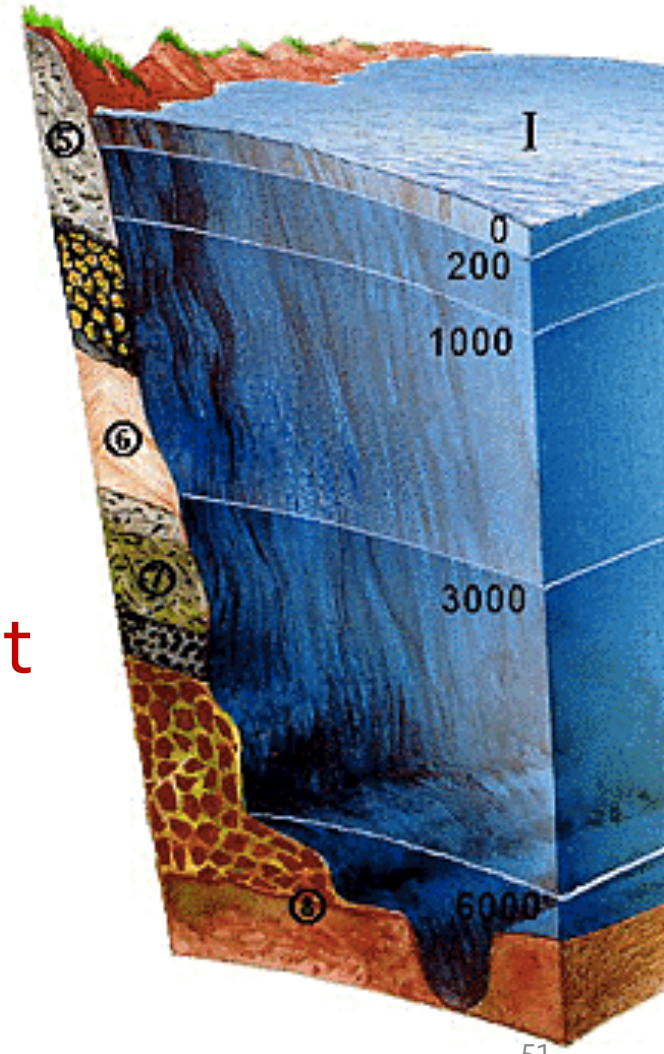
Application 1: Food webs

Florida Bay food web:

- Nodes: species in the ecosystem
- Edges: carbon exchange (who eats whom)

Different motifs capture different energy flow patterns:





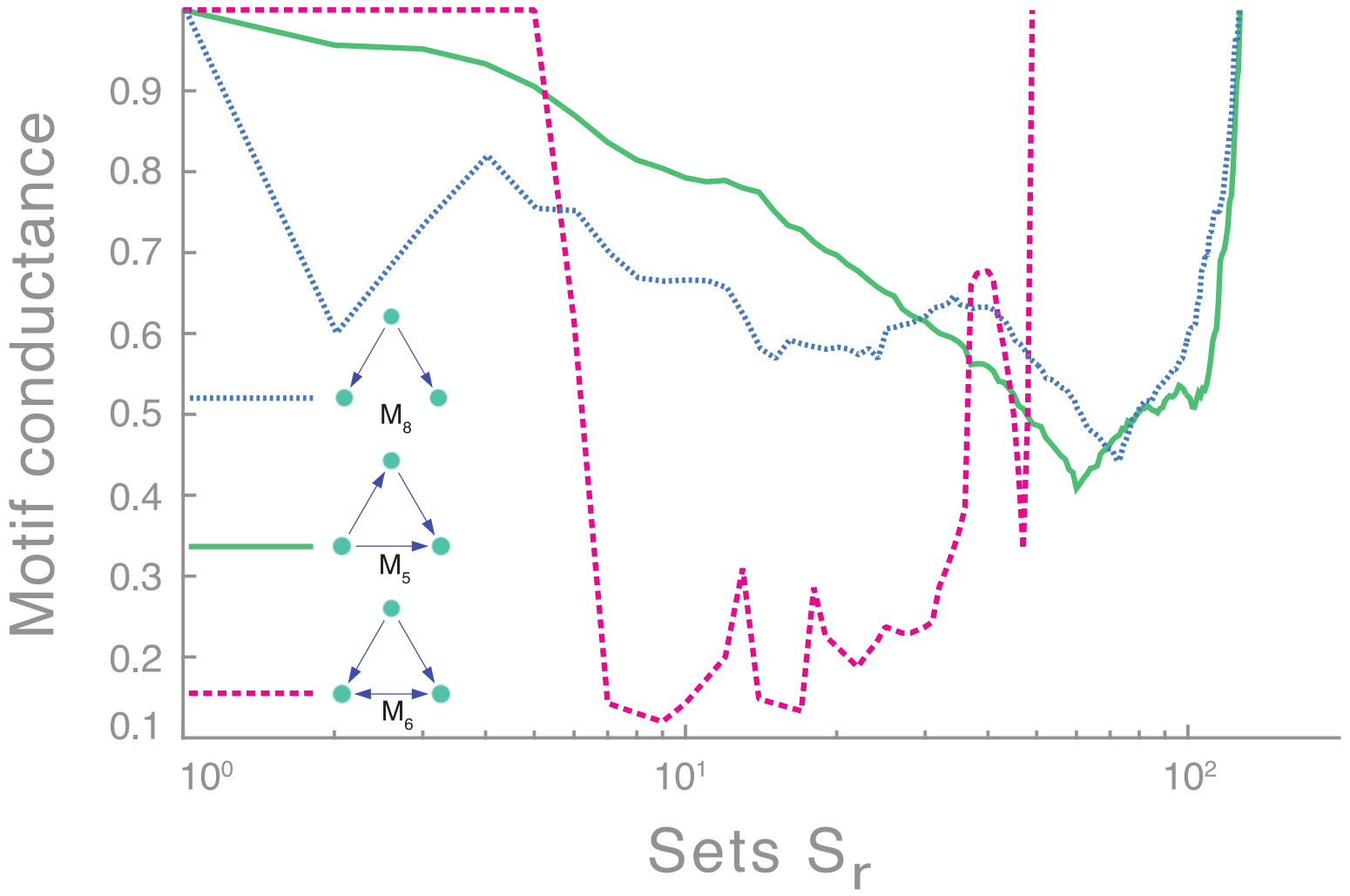
Florida Bay Food Web

Which motif organizes the food web?

Approach:

- Run motif spectral clustering separately for each of the 13 motifs
- Examine the Sweep profile to see which motif gives best clusters

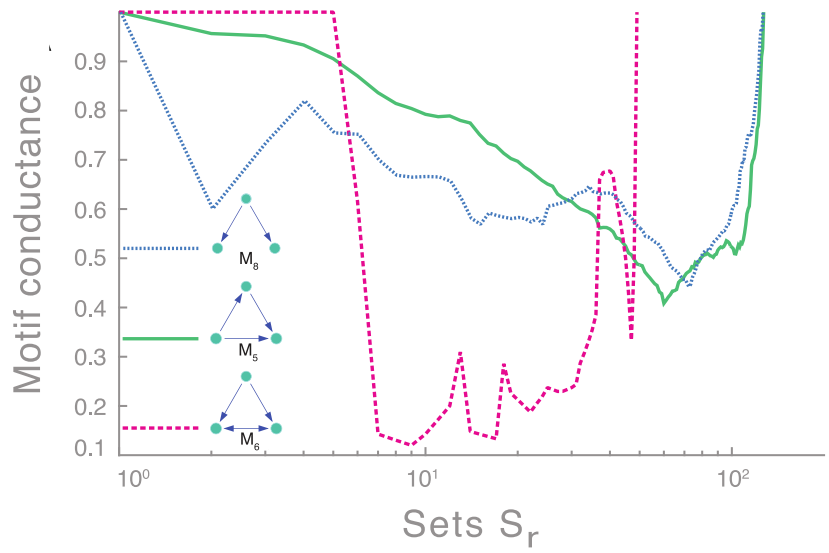
Florida Bay Food Web

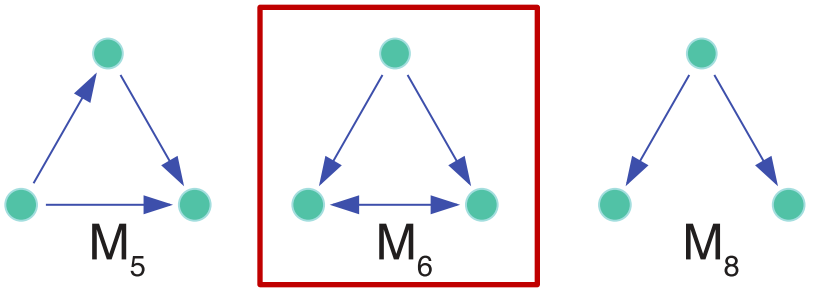


Food Web: Observations

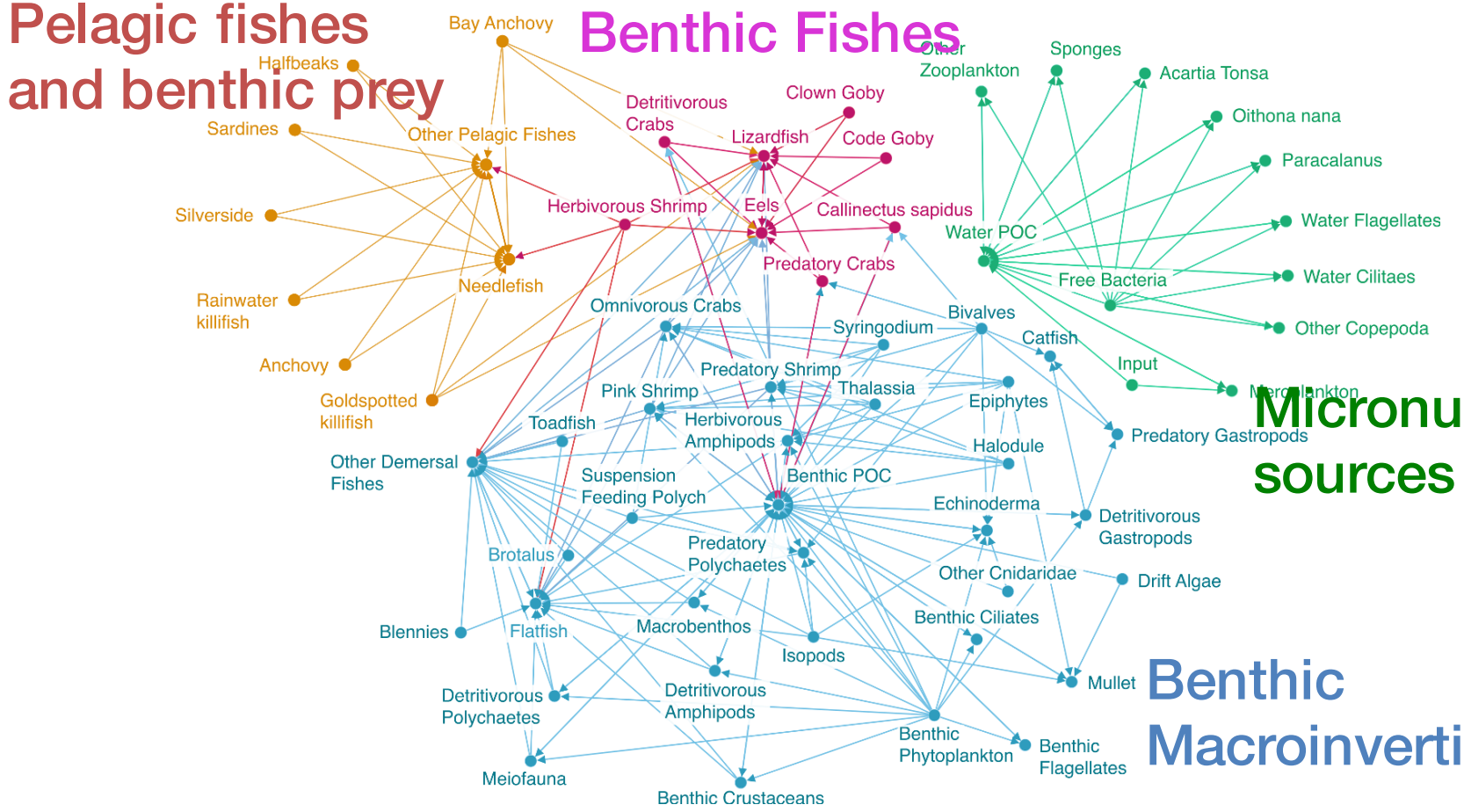
Observation:
Network organizes based on motif M_6 (but not M_5 or M_8)

- There exist good cuts for M_6 but not for M_5 or M_8



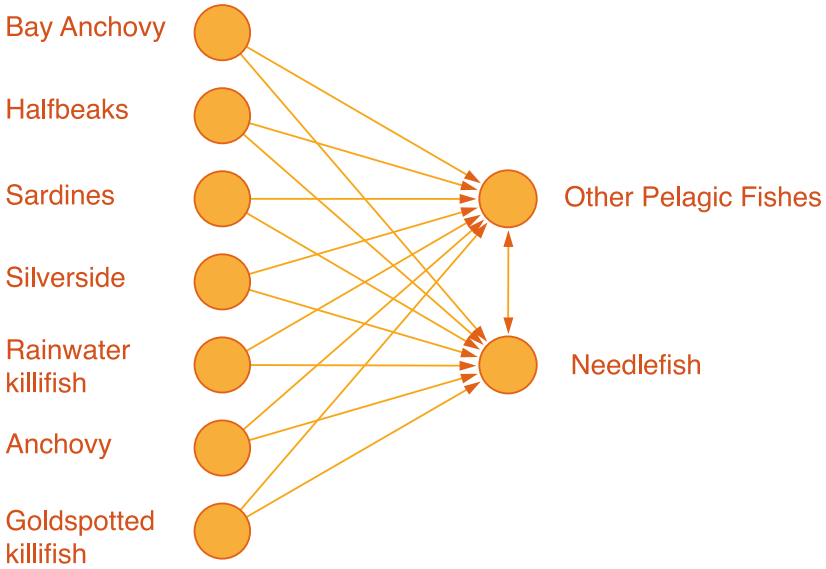


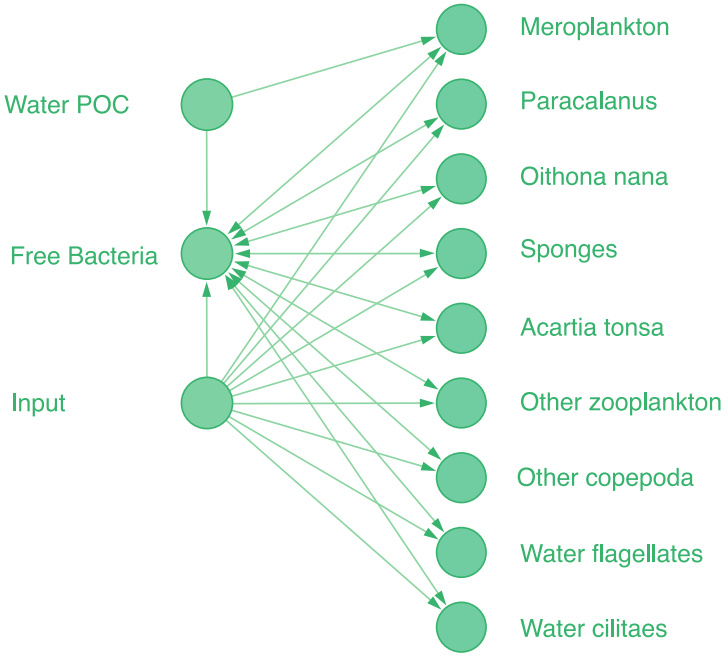
Food Web: Clusters



M_6 reveals known aquatic layers with higher accuracy (84% vs. 65%)

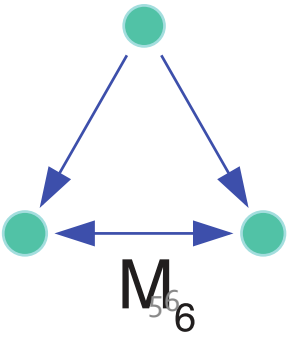
Structure of Aquatic Layers





Aquatic layers organize based on M_6

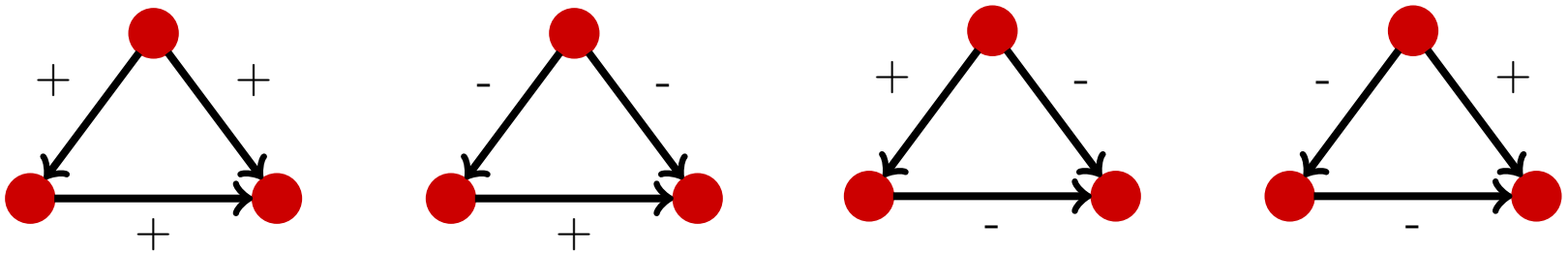
- Many instances of M_6 inside
- Few instances of M_6 cross



Application 2:
Signed networks

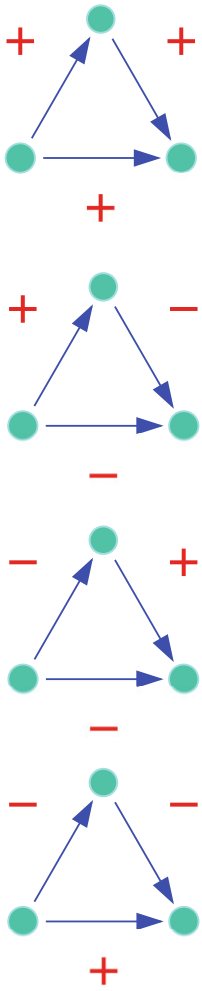
(2) Gene Regulatory Networks

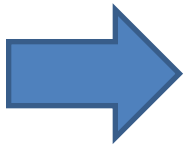
- Nodes are groups of genes in mRNA
- Edges are directed transcriptional regulation relationships

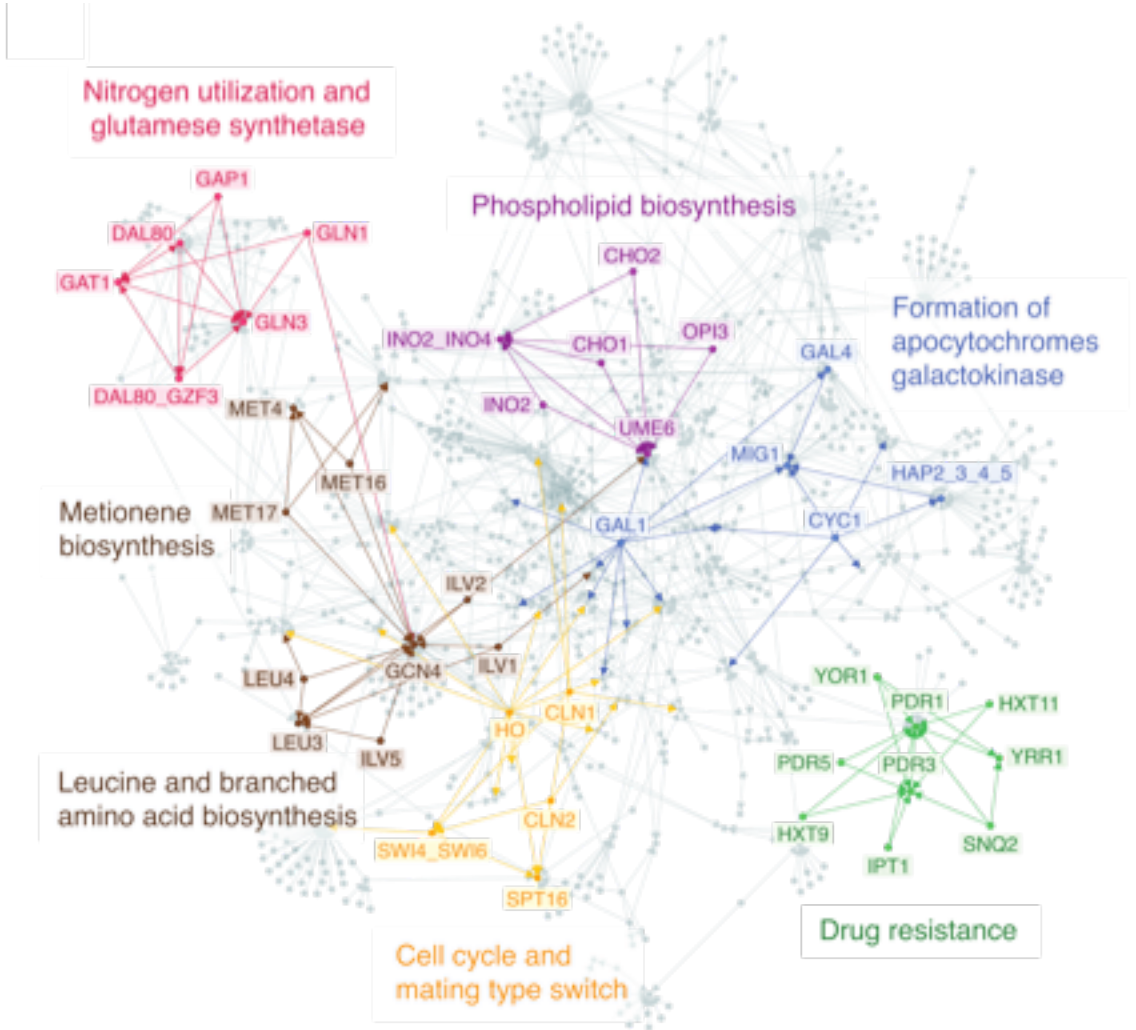


- The "feedforward loop" motif represents biological function [Alon '07]

Yeast Regulatory Network



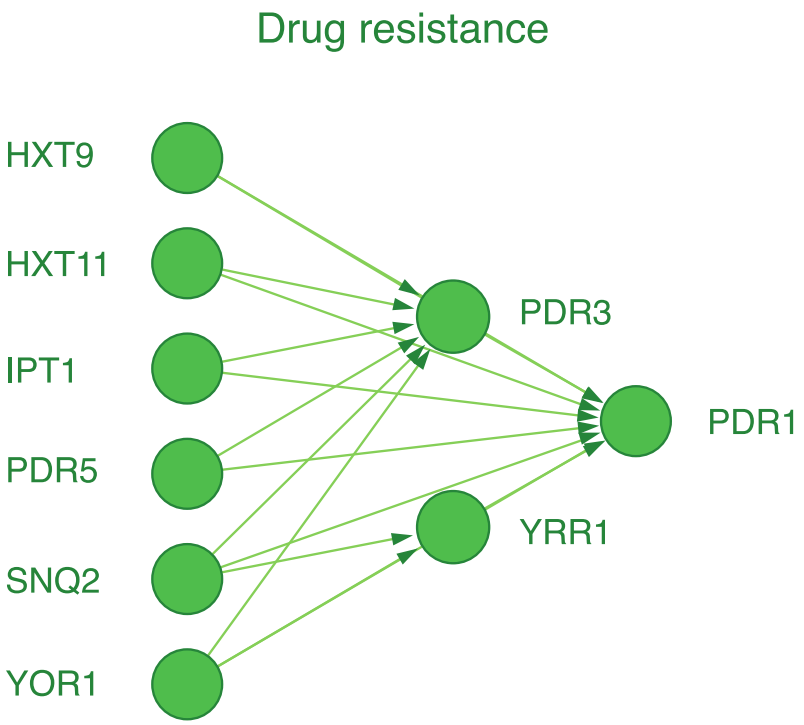


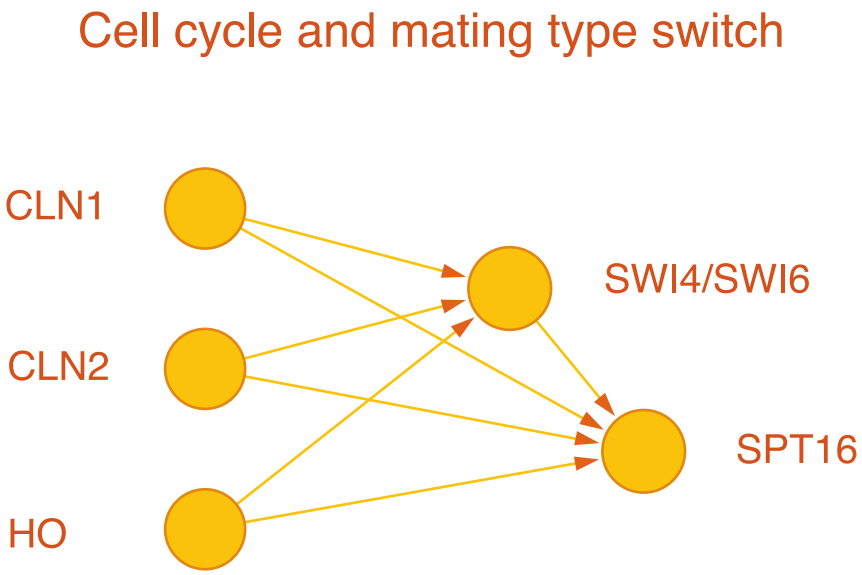


97% detection accuracy (vs. 68-82%)

Structure of Modules

- Feed forward loops:



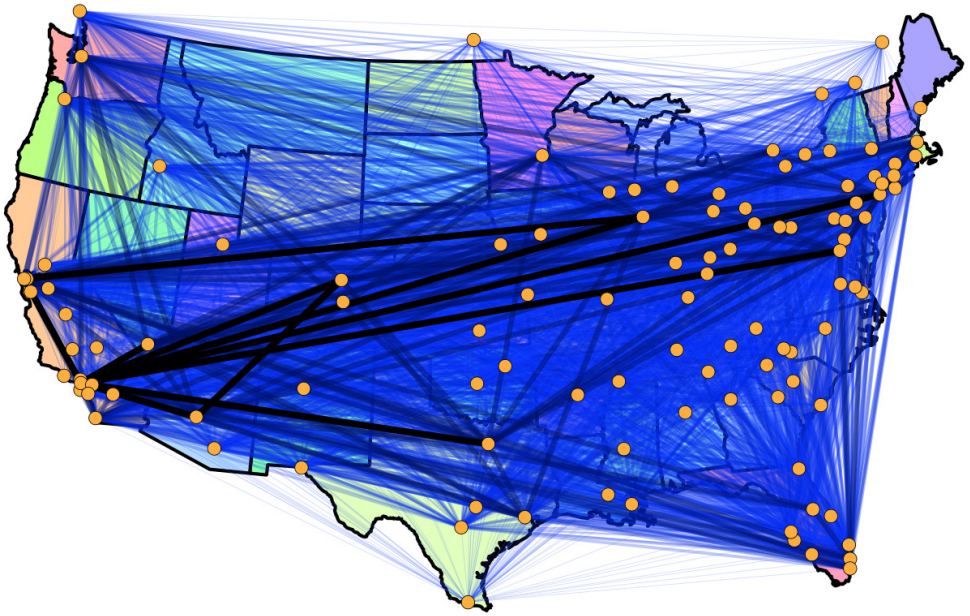


Application 3:
Exploring richer information from data

(4) Transportation Networks

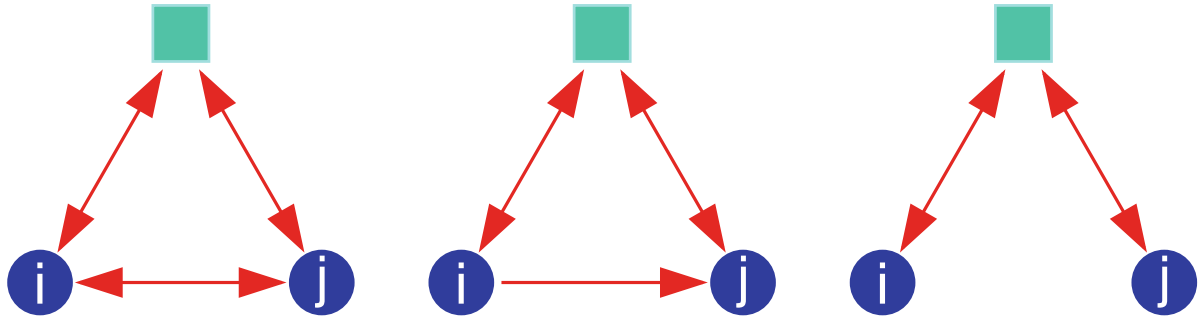
North American air transport network:

- Nodes are airports
- Unweighted directed edges reflect reachability
- (Based on Freyet al.'s 2007)



Transportation Networks

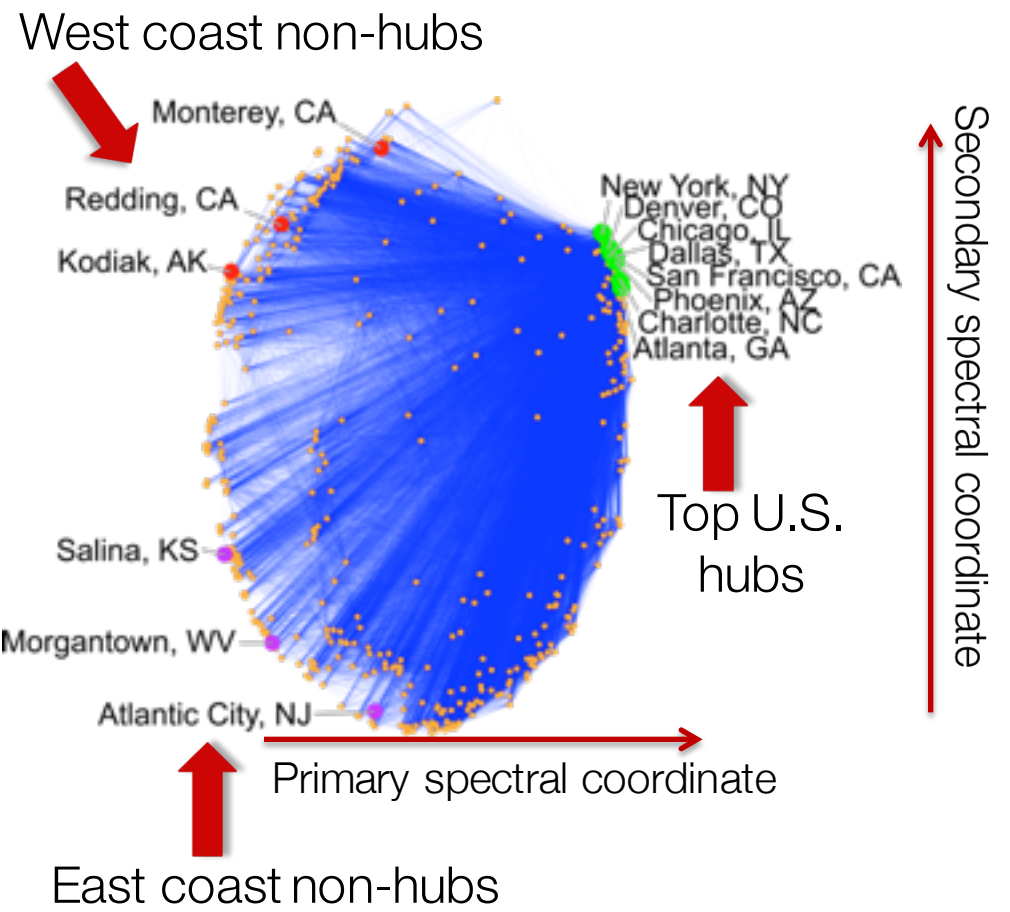
- Want to understand hub-and-spoke structure of the network
- Use "anchored" motifs:



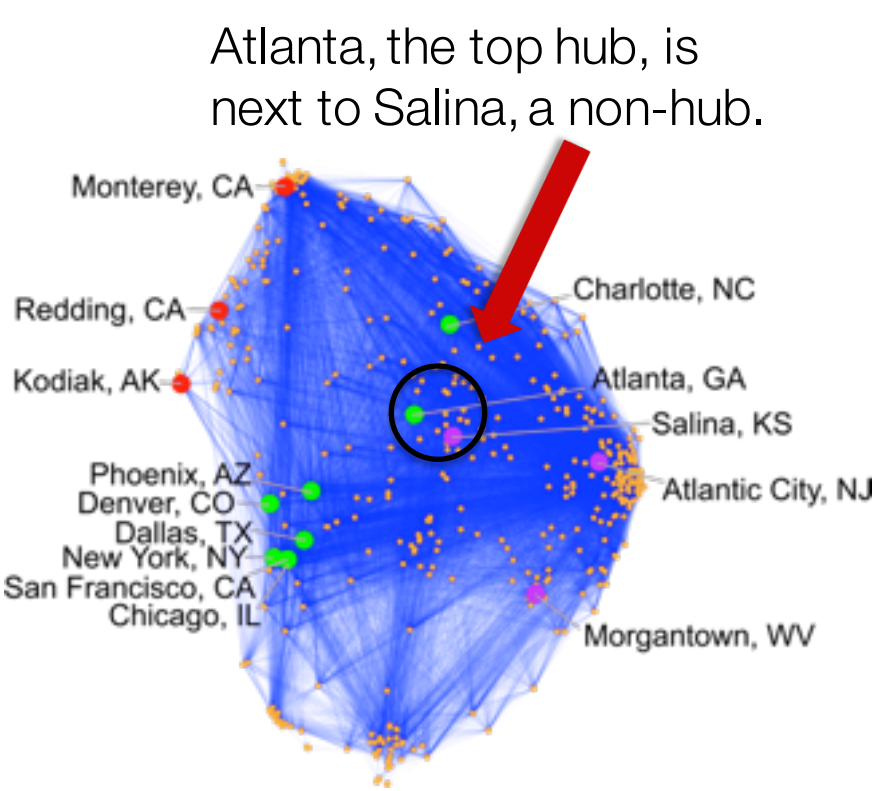
$(W_M)_{ij}$ = #{bidirectional 2-paths from i to j}

Transportation Networks

Motif-based spectral embedding

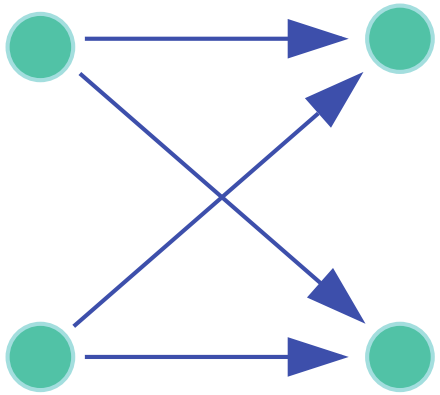


Edge-based spectral embedding



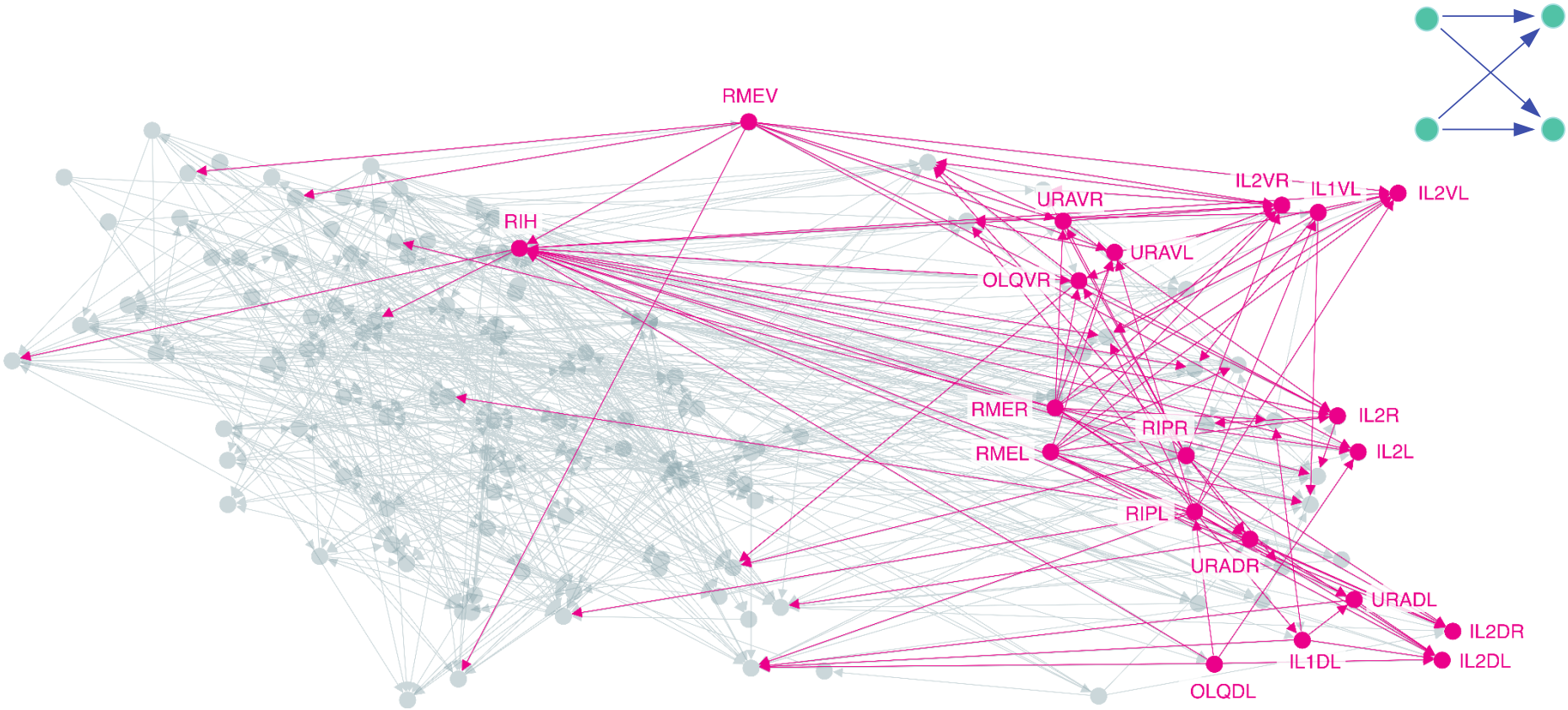
(3) Neuronal Networks

Bi-Fan motif is overexpressed
in *C. Elegans* network [Milo '02]



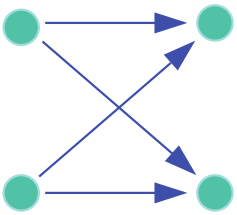
Cluster the network based on this motif!

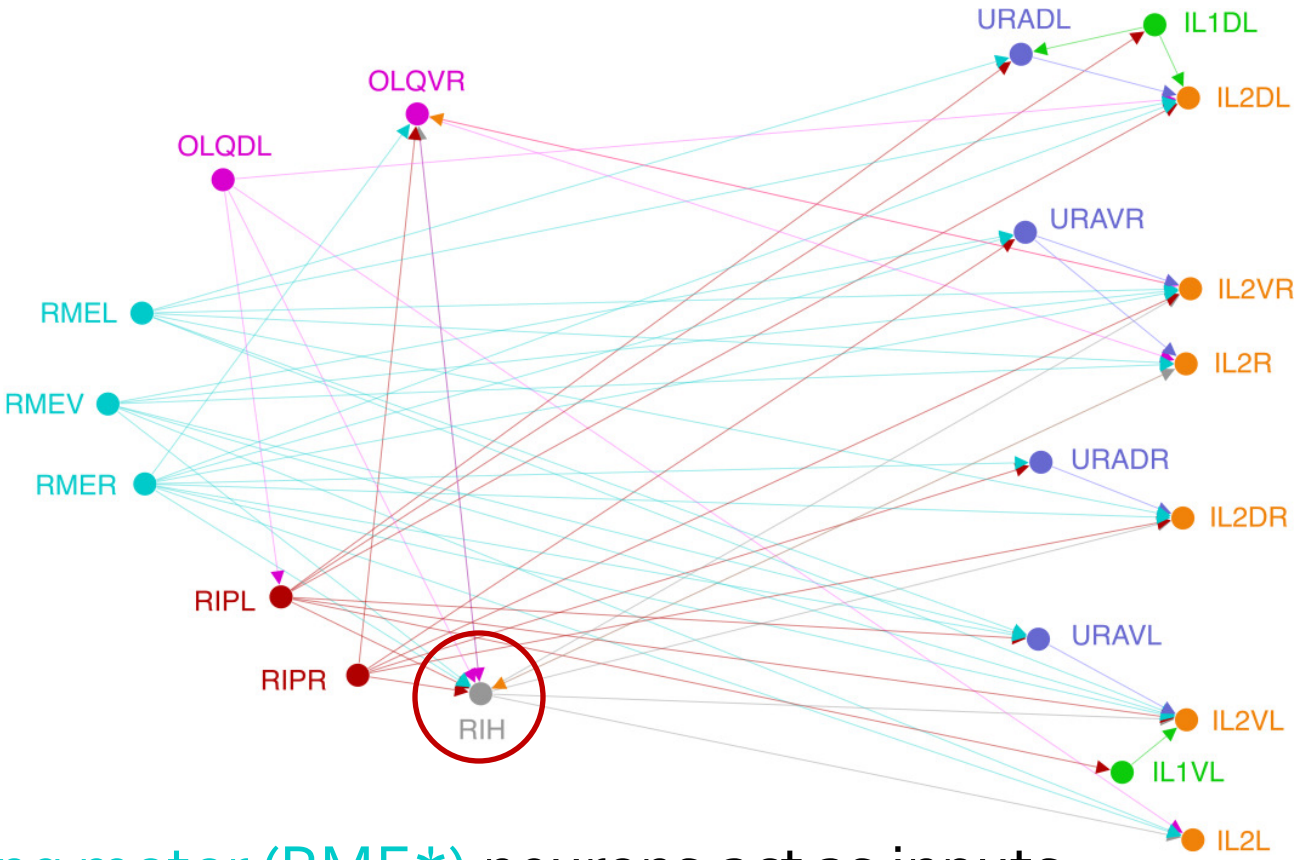
C. Elegans: Best Cluster



Node locations represent spatial coordinates of corresponding neurons

C. Elegans Network





- Ring motor (RME*) neurons act as inputs
- Inner labial sensory (IL2*) neurons are the destinations
- URA neurons act as intermediaries

Summary

- Community detection of higher-order structures
- Provably finds good clusters
- Simple, fast, and scalable
- Can be applied to directed, signed, and weighted networks

Paper, Data, Code

- Paper:
  Higher-order Organization of Complex Networks.
  Austin R. Benson, David F. Gleich, and Jure Leskovec.
  Science, 2016.

- Code & Data:
  http://snap.stanford.edu/higher-order/