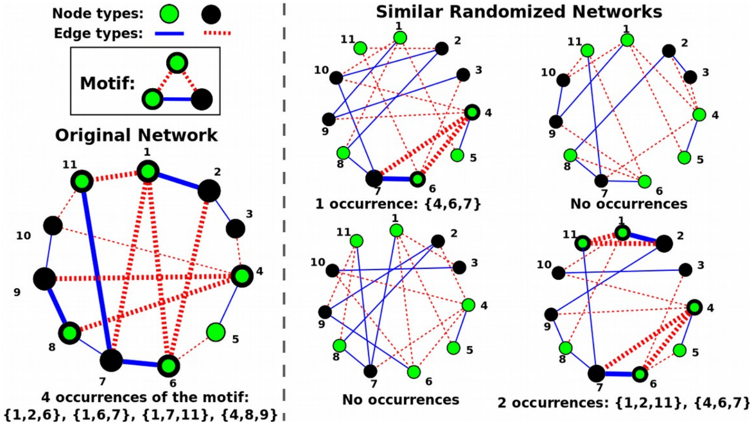
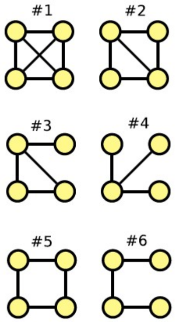
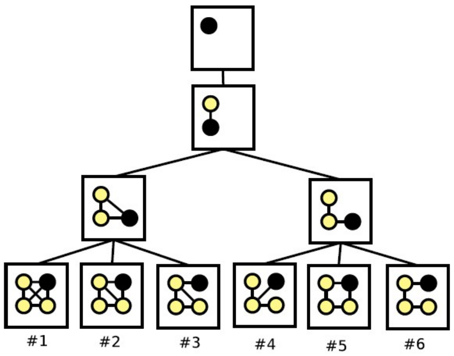






Subgraphs as Fundamental Ingredients of Complex Networks

Concepts, Methods and Applications

Pedro Ribeiro

DCC/FCUP & CRACS/INESC-TEC

Network Science 2019/2020 (DCC/FCUP)

Contents

1) **Motivation**

2) **Concepts**

3) **Computational Challenge**
(and sequential exact solutions)

4) **Sampling approach**

5) **Parallel Approach**

6) **Example Applications**

7) **Resources**

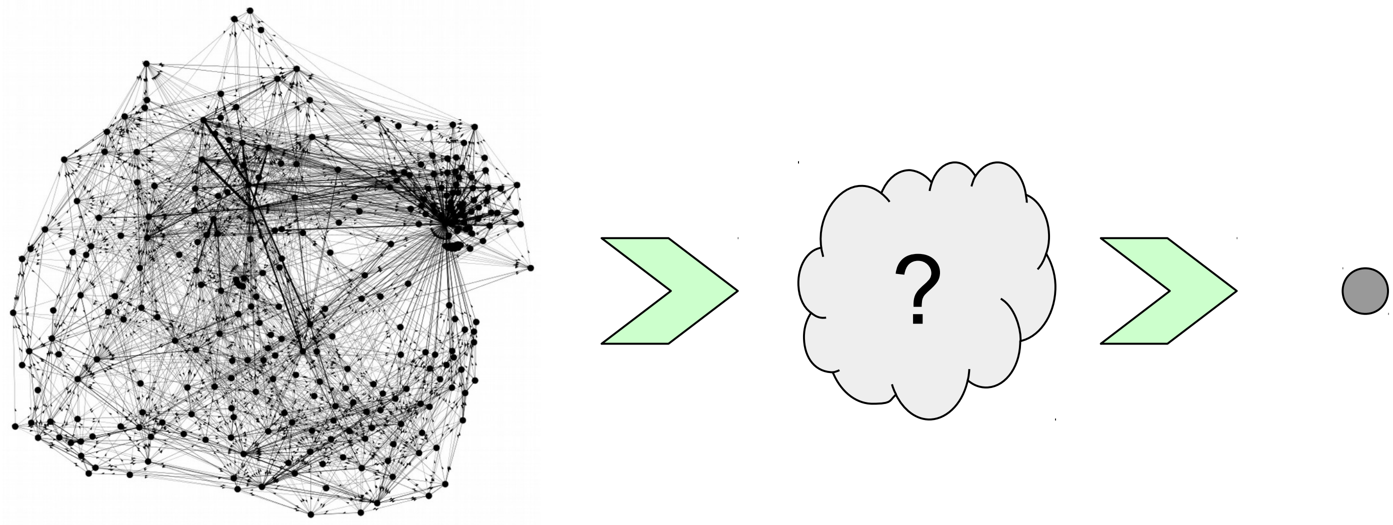
1) MOTIVATION

Network Metrics

- **There are many available metrics at the node level:**
  - E.g. degree, betweenness, closeness

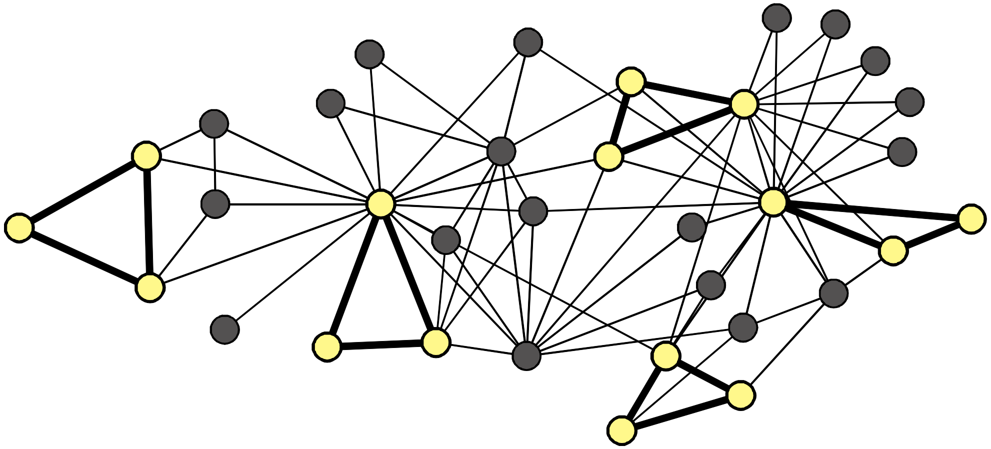
- **There are also many metrics at the global level:**
  - E.g. diameter, avg. distance, density, clustering coefficient

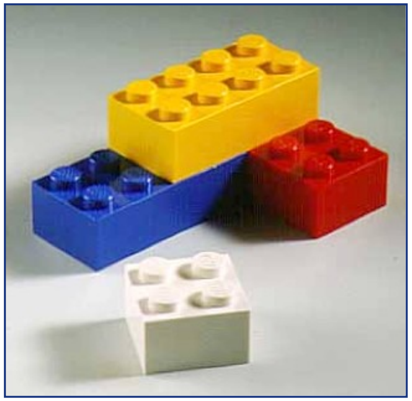
- **What about something inbetween?**



Building Blocks of Networks

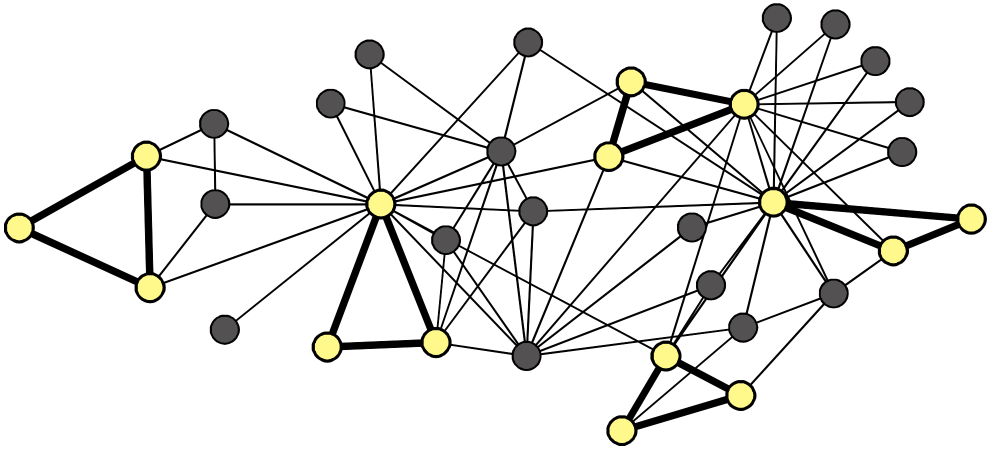
- **Subnetworks, or subgraphs, are the building blocks of networks**

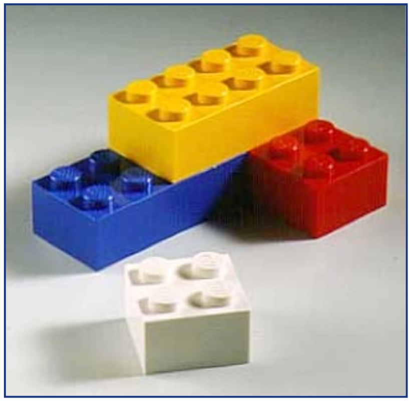


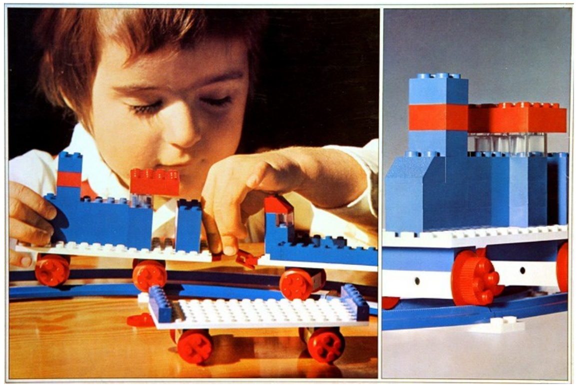


Building Blocks of Networks

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

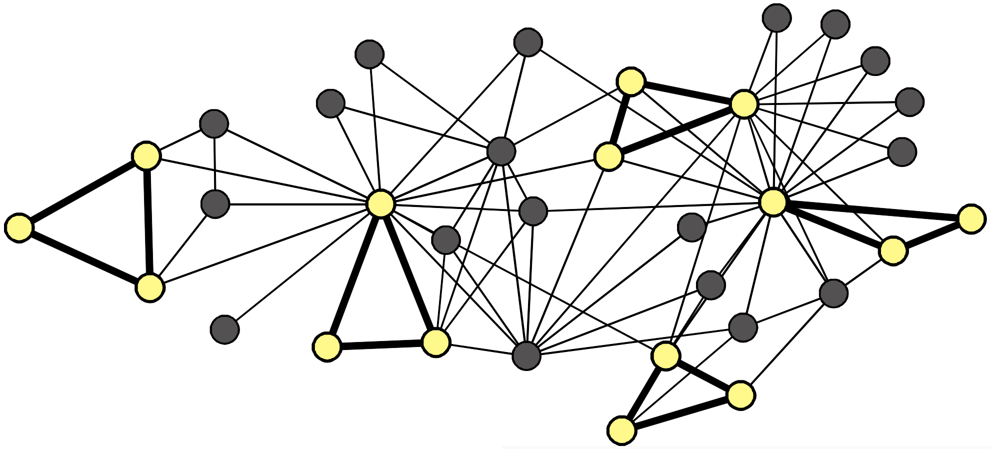


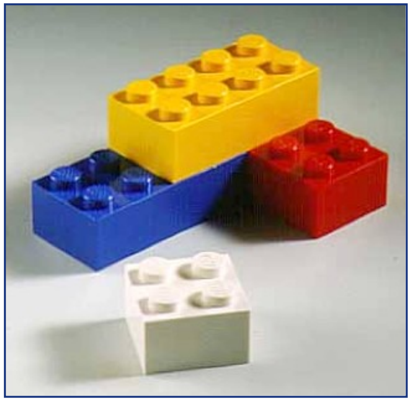


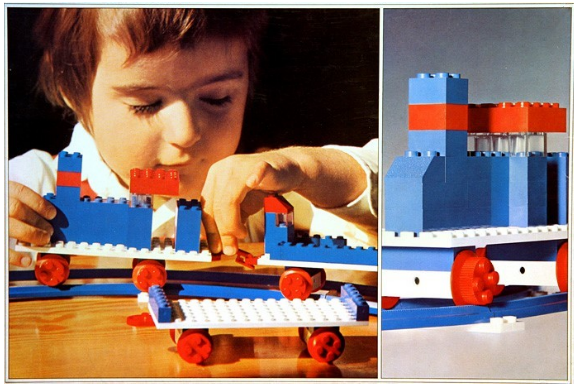


Building Blocks of Networks

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

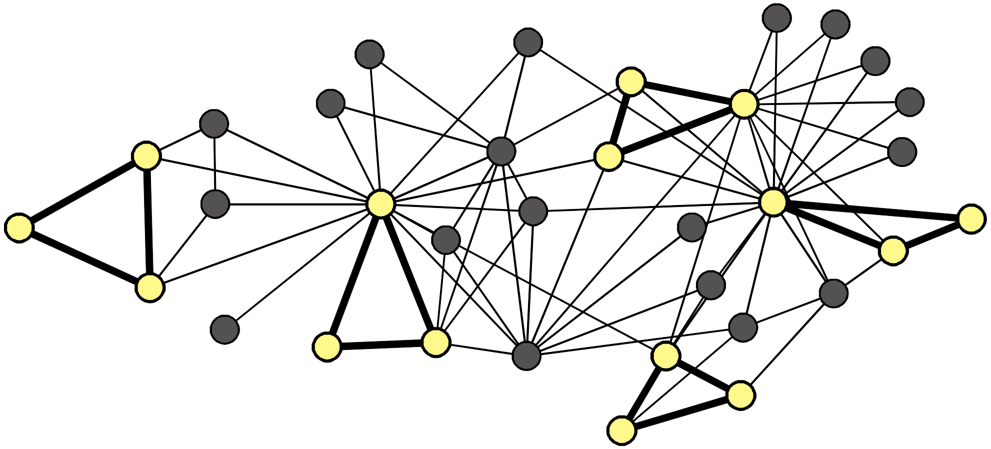


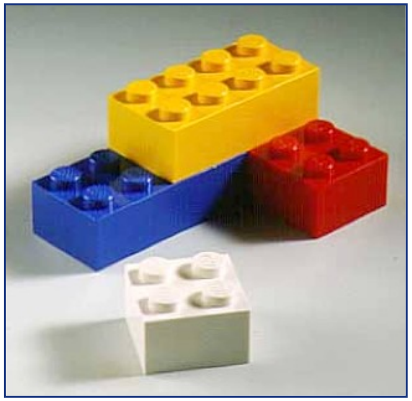




Building Blocks of Networks

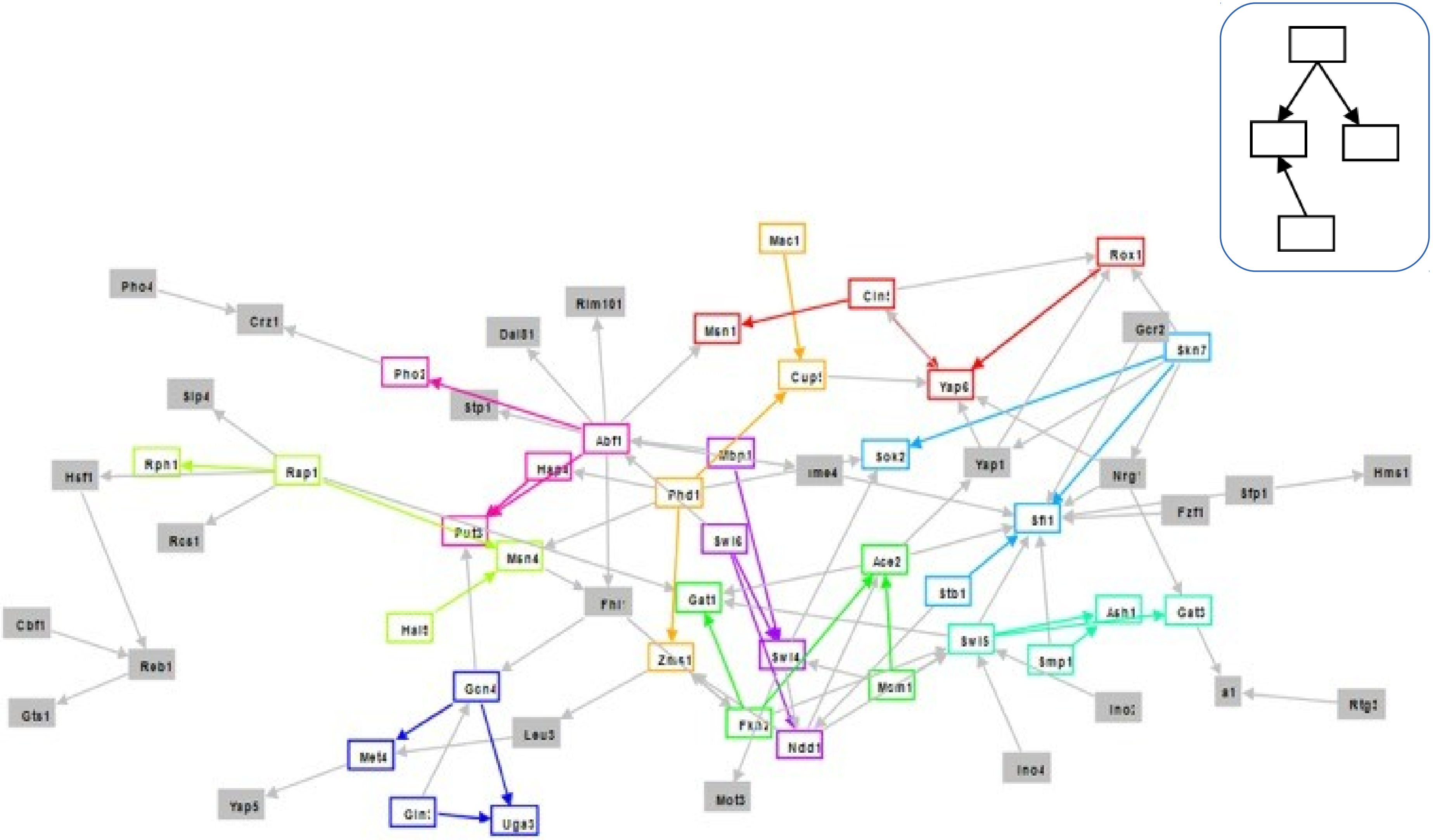
- **Subnetworks, or subgraphs, are the building blocks of networks**





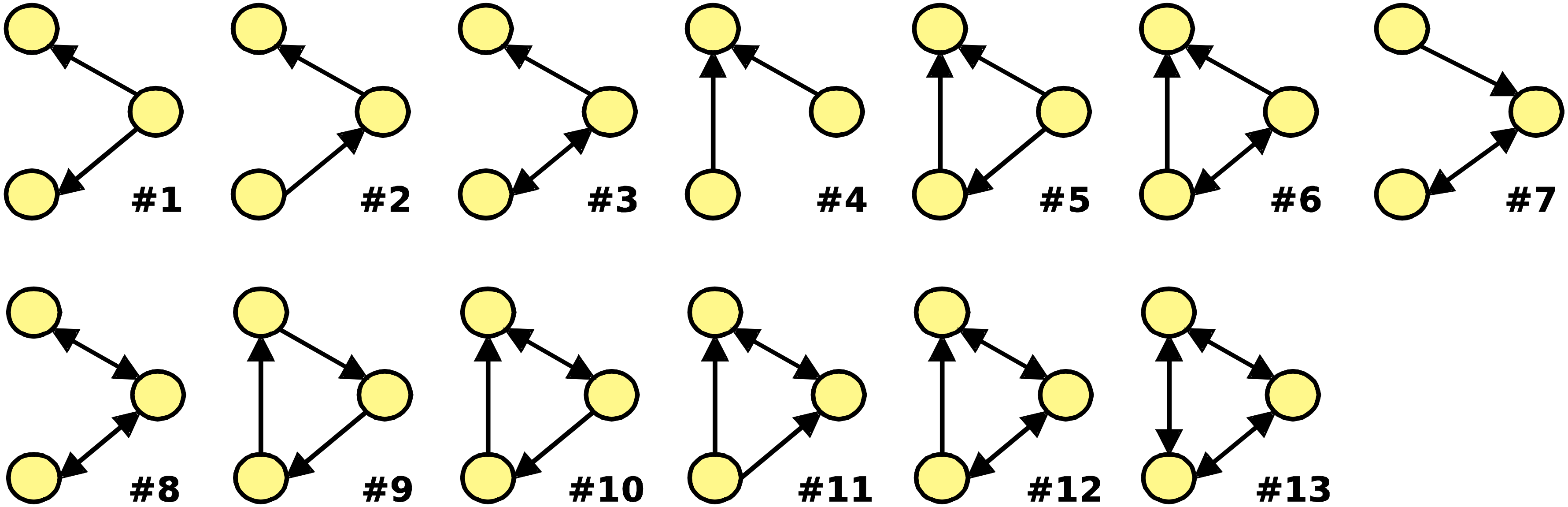
- **They have the power to characterize and discriminate networks**

Building Blocks of Networks



Example Application

- **Consider all possible directed subgraphs of size 3**



Example Application

- **For each subgraph type:**
  - Metric capable of classifying subgraph “significance” [more about that later]
  - Values in interval [-1,1]
    - Negative values indicate **underepresentation**
    - Positive values indicate **overrepresentation**

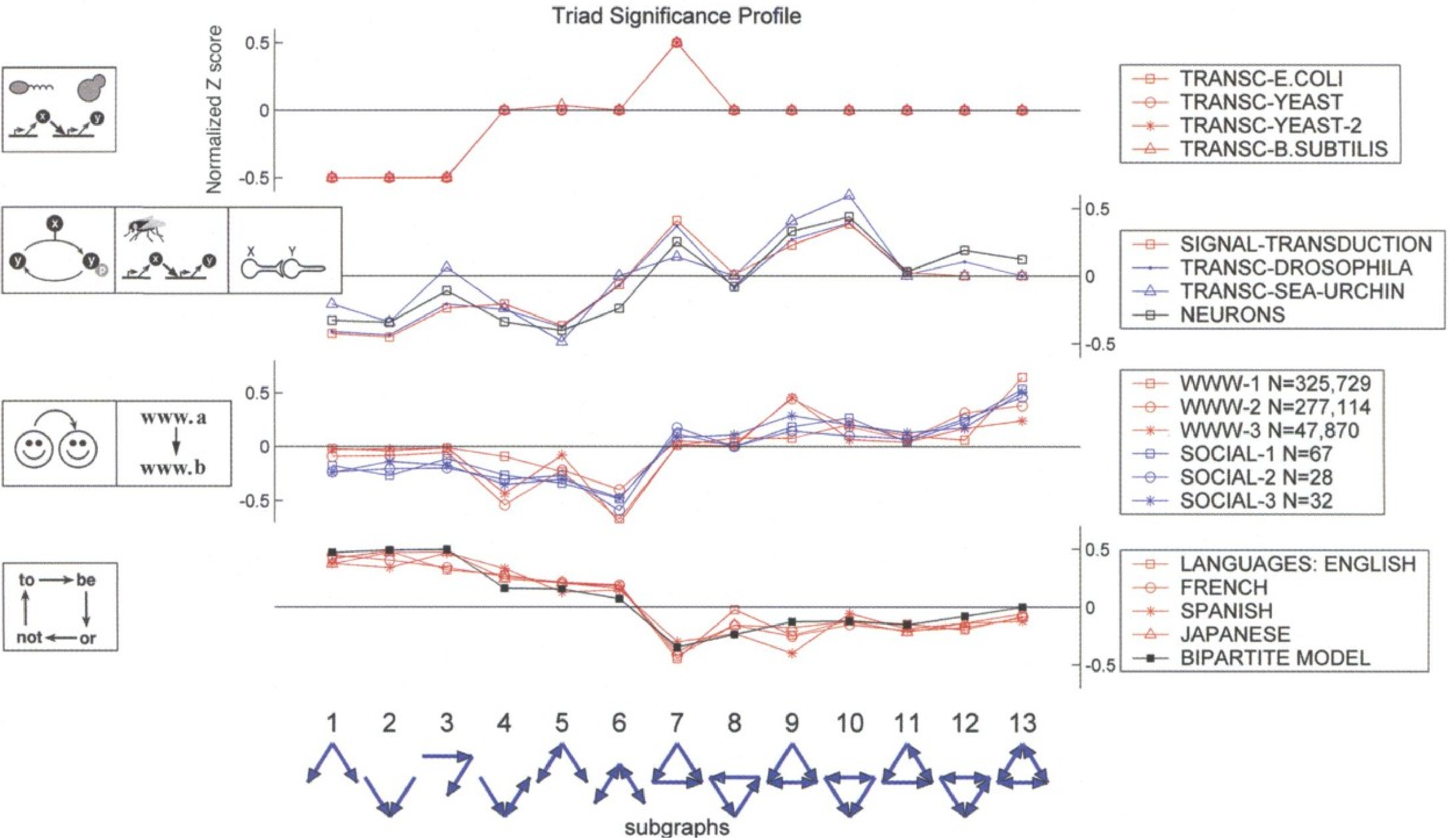
- **With this you could create a network fingerprint:**
  - Feature vector with all subgraph significances

Example Application

- **Consider the following varied types of networks:**
  - Regulatory Network (gene regulation)
  - Neuronal Network (synaptic connections)
  - World Wide Web (hyperlinks between pages)
  - Social network (friendships)
  - Semantic Networks (word adjacency)

- **What happens when we look at their fingerprints as defined before?**

Example Application

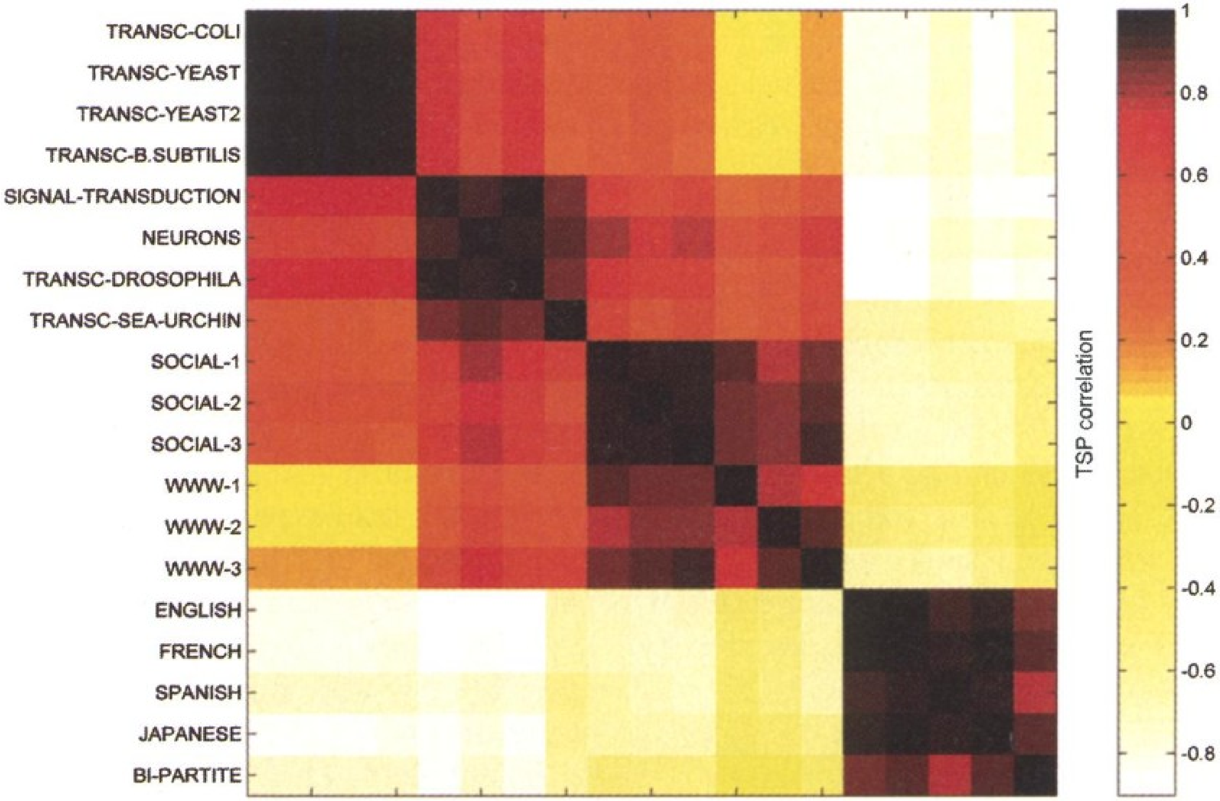


Different networks have similar fingerprints!

Image: (Milo et al., 2004)

Example Application

Correlation



Different networks have similar fingerprints!

Image: (Milo et al., 2004)

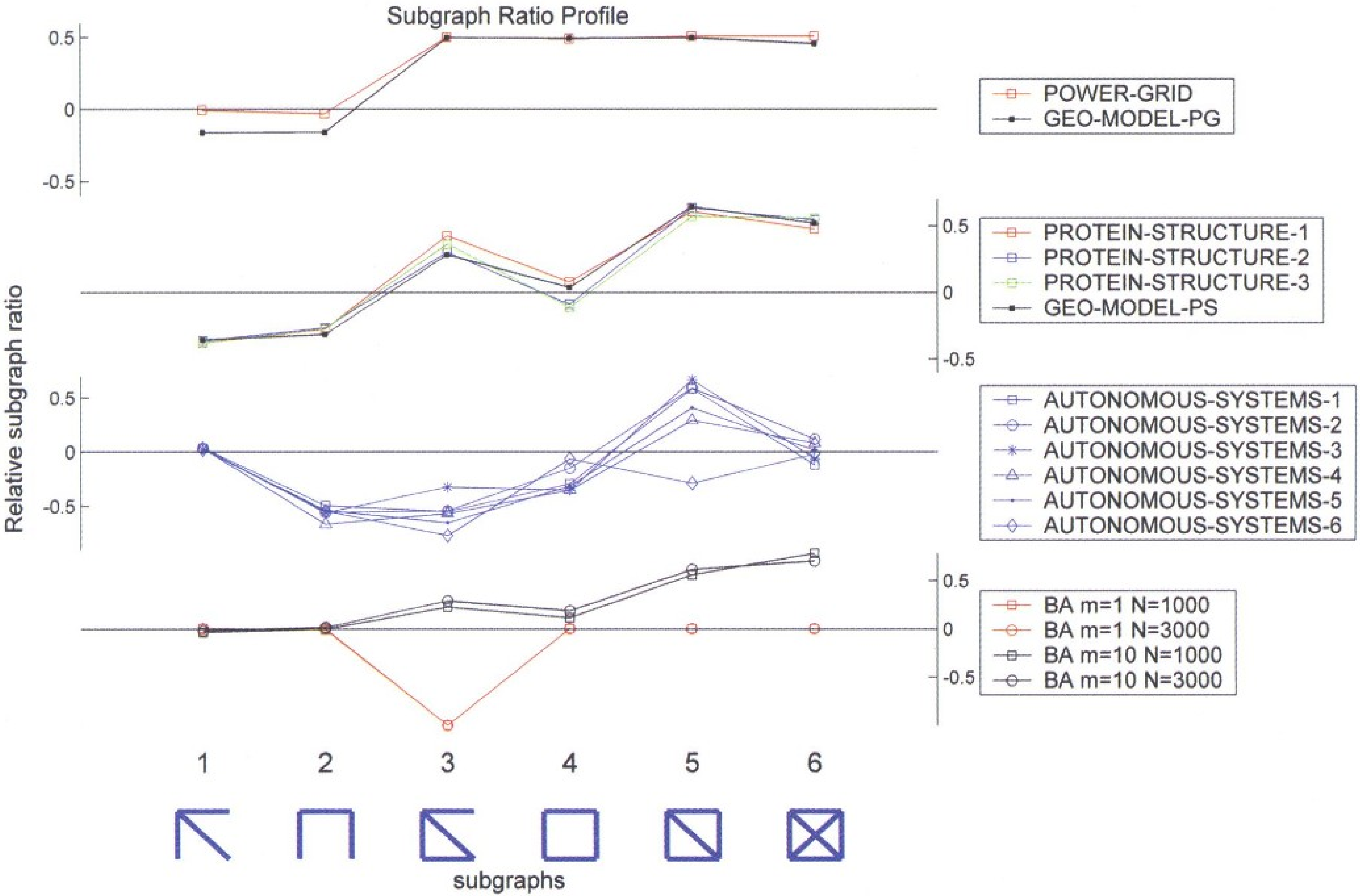
Example Application

- **What about undirected networks?**

- **Consider the following types of networks:**
  - Power Grid (electrical geographical power grid)
  - Protein Structure (seconday structure adjacency
  - Autonomous Systems (internet)

- **What happens when we look at their fingerprints as defined before?**

Example Application



Different networks have similar fingerprints!

Image: (Milo et al., 2004)

Subgraphs are powerful

Subgraphs have the power to **characterize** and **discriminate** networks

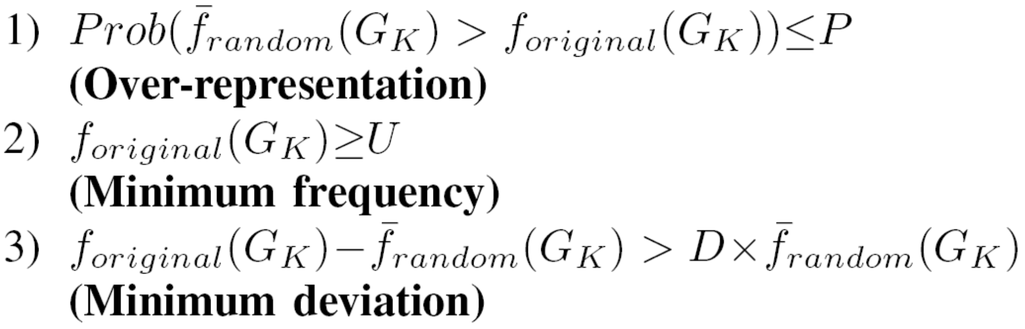
Their applicability is **general**

2) CONCEPTS

Network Motifs

- **Milo et al. (2002) came up with the definition of network motifs:**
  - "recurring, significant patterns of interconnections"

- **How to define:**
  - Pattern: induced subgraph
  - Recurring: found many times, i.e., high frequency
  - Significant: more frequent than it would be expected in similar networks (same degree sequence)



1) $Prob(\bar{f}_{random}(G_K) > f_{original}(G_K)) \leq P$
(Over-representation)

2) $f_{original}(G_K) \geq U$
(Minimum frequency)

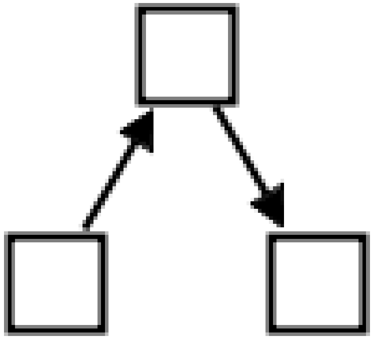
3) $f_{original}(G_K) - \bar{f}_{random}(G_K) > D \times \bar{f}_{random}(G_K)$
(Minimum deviation)

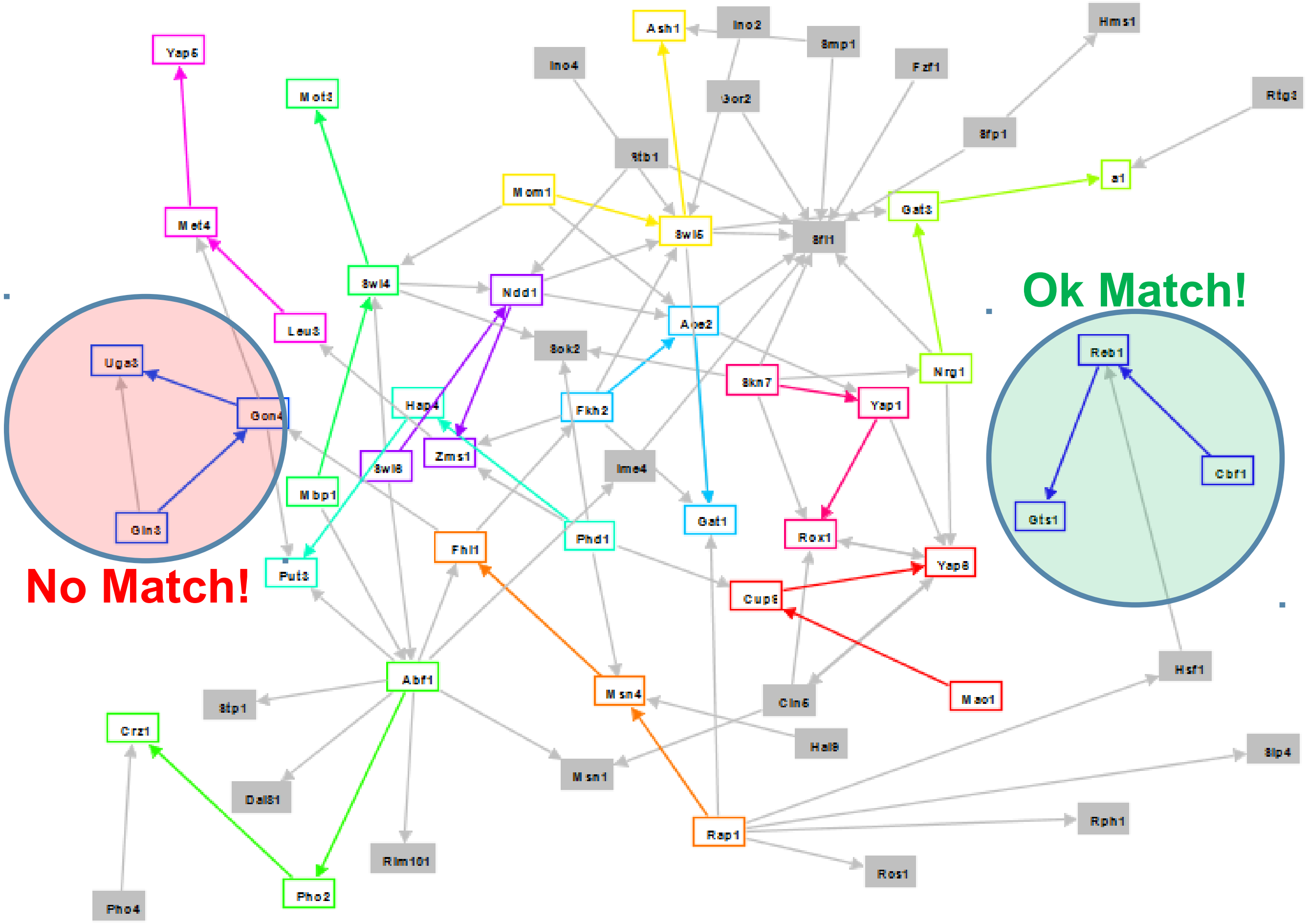
Parameters P, U, D, N control the definition (Milo et al., 2002, used {0.01, 4, 0.1, 1000})

Image: Adapted from (Milo et al., 2004)

Subgraph concepts - Induced

- **Induced Subgraphs**



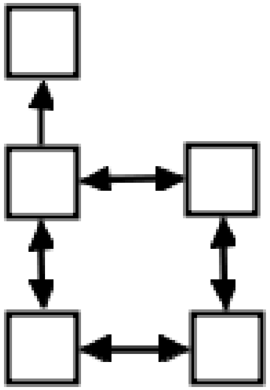


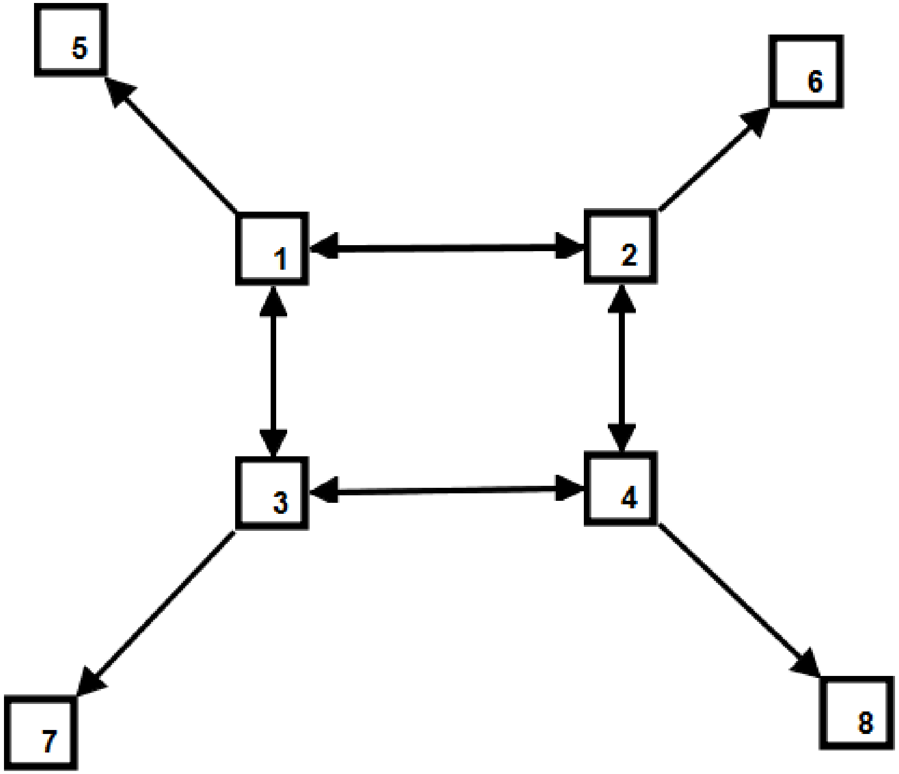
No Match!

Ok Match!

Subgraph concepts - Frequency

- **How to count?**
  - Allow **overlapping**



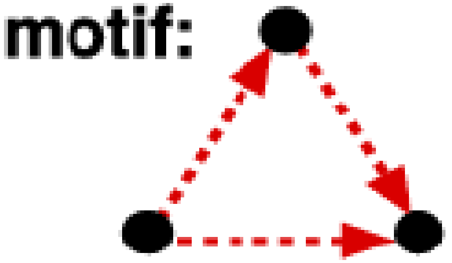


  - 4 occurrences:

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

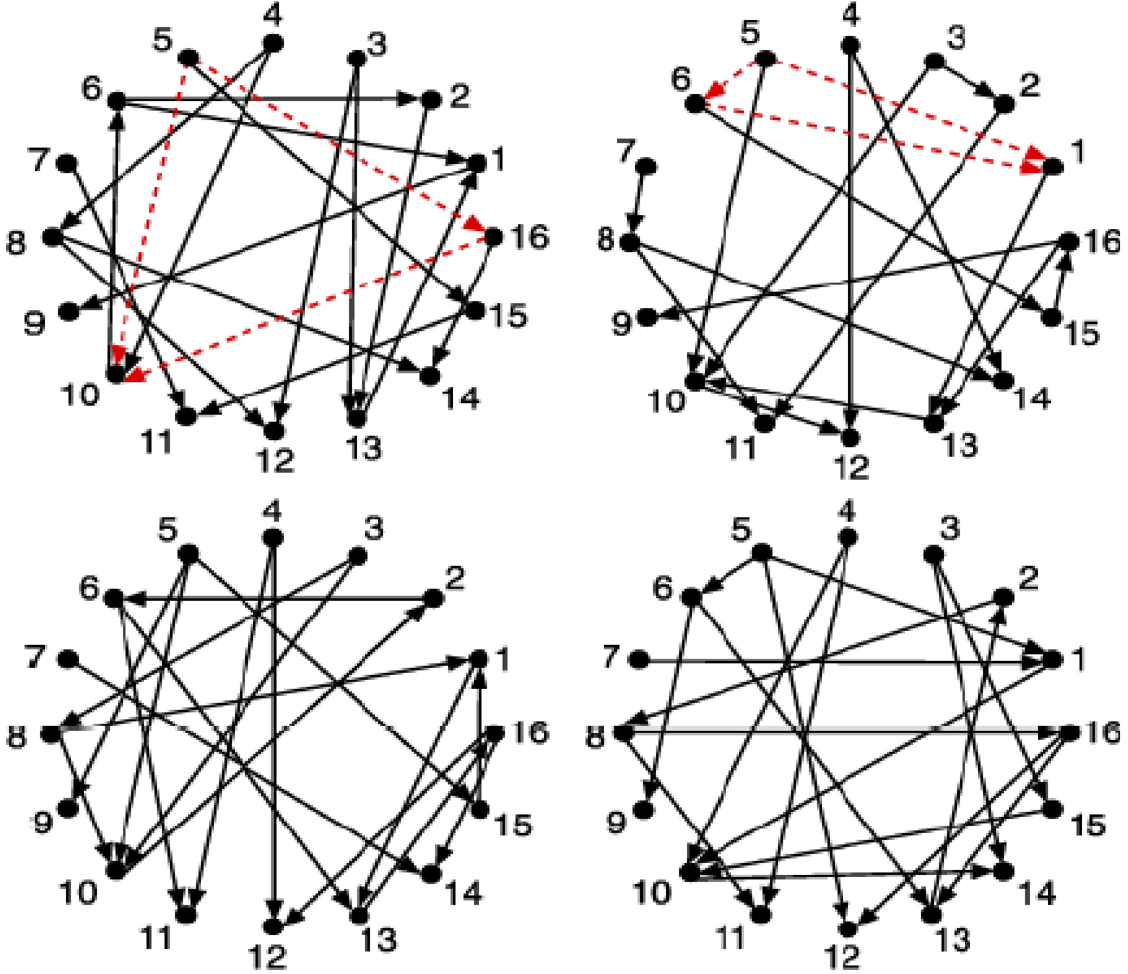
Subgraph concepts – Significance

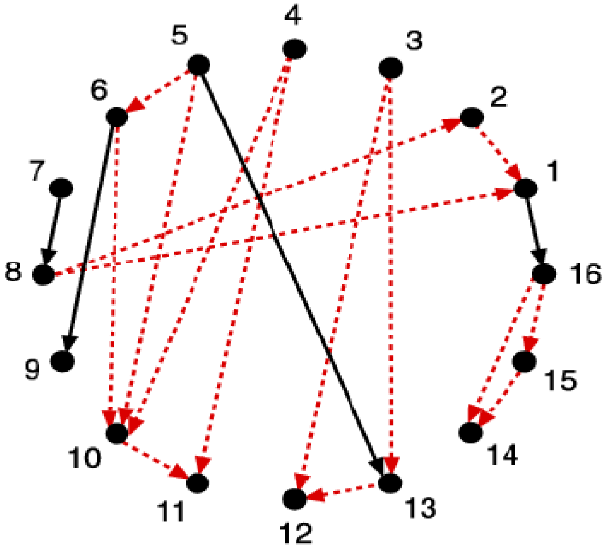
Traditional Null Model – keep **Degree Sequence**



Motif

Random Networks



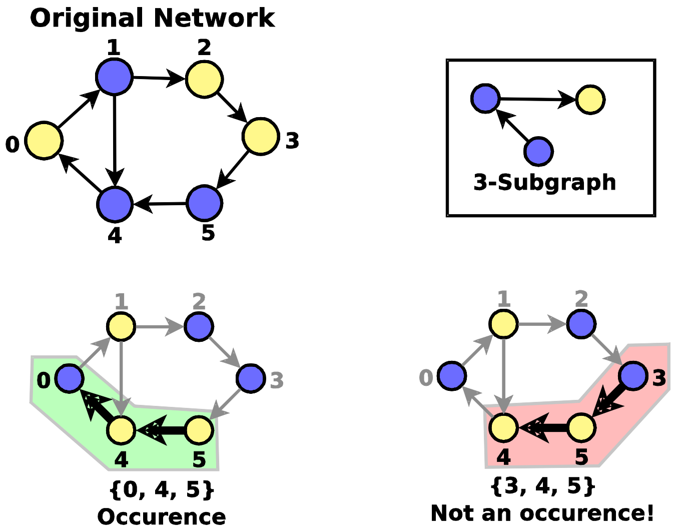


Original Network

Image: Adapted from (Milo et al., 2002)

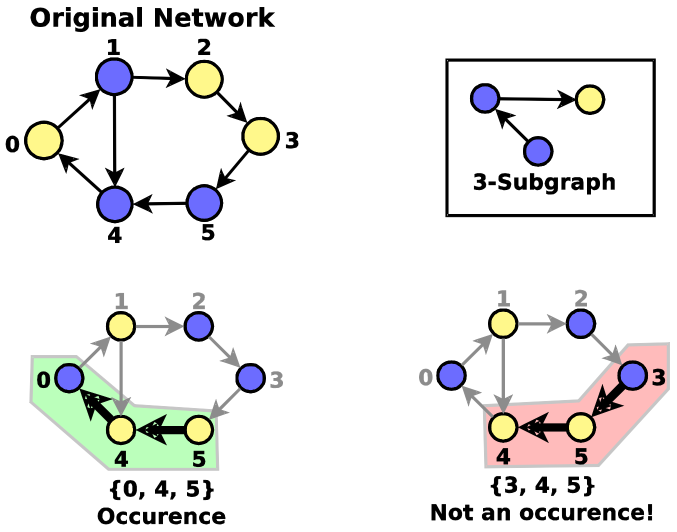
Network Motifs Applicability

- **Canon definition:**
  - Directed and Undirected
  - Colored and uncolored

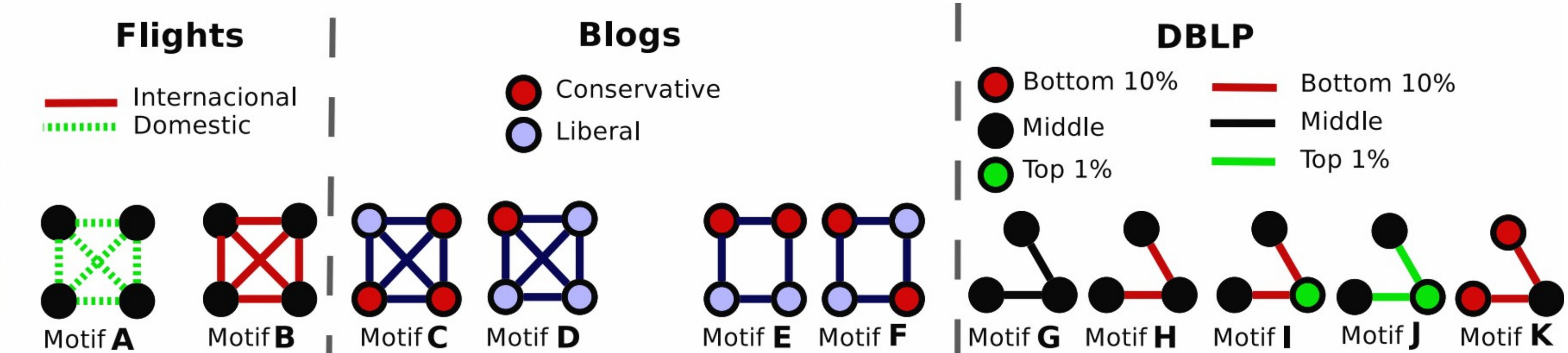


Network Motifs Applicability

- **Canon definition:**
  - Directed and Undirected
  - Colored and uncolored



Example application of colored motifs: [Ribeiro & Silva, Complenet'2014]



Overrepresentation of **A** much larger than **B**

Overrepresentation of **C** much larger than **D**

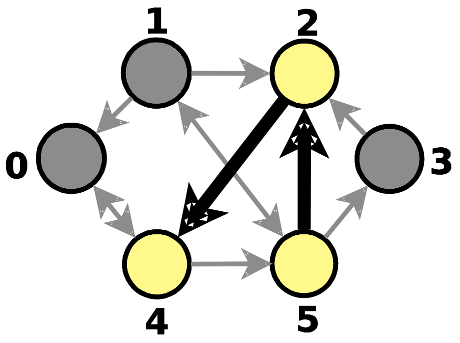
E is **over**represented
F is **under**represented

I and **K** are **over**represented
G and **H** are **under**represented
J is almost **neutral**

Network Motifs Applicability

Variations on the concept

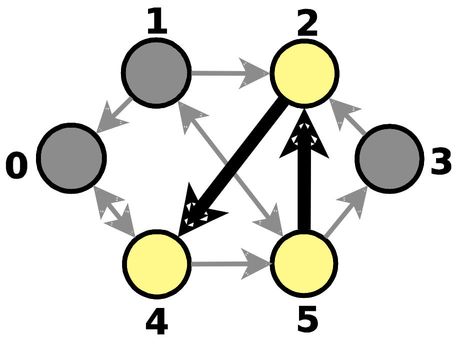
- Different frequency concepts
- Different significance metrics
- Under-Representation (**anti-motifs**)
- Weighted networks
- Different constraints for the null model



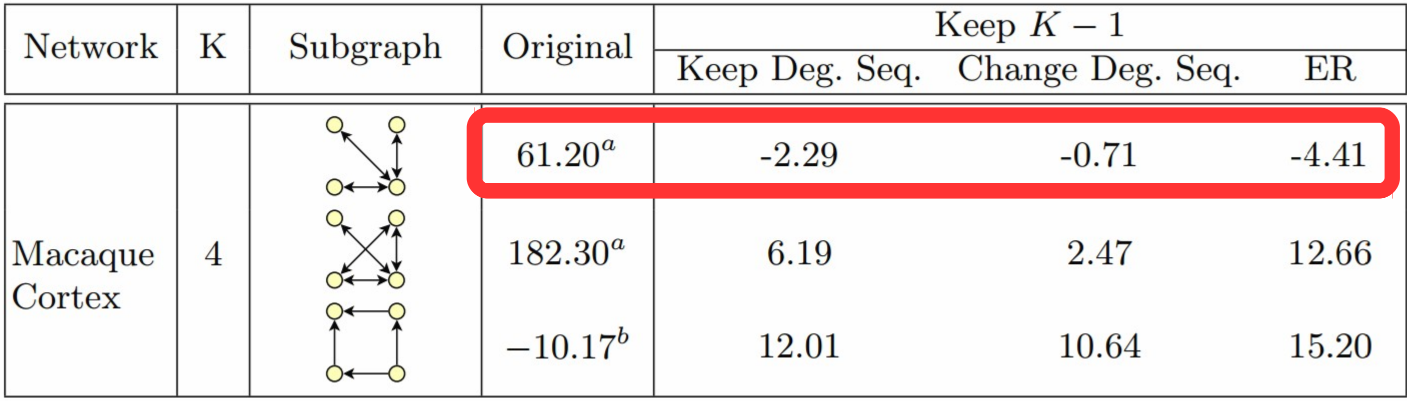
Network Motifs Applicability

Variations on the concept

- Different frequency concepts
- Different significance metrics
- Under-Representation (**anti-motifs**)
- Weighted networks
- Different constraints for the null model



Ex. application of different null model: [Silva, Paredes & Ribeiro, Complenet'2017]

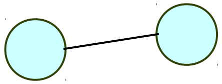


Network	K	Subgraph	Original	Keep $K-1$		
				Keep Deg. Seq.	Change Deg. Seq.	ER
Macaque Cortex	4		61.20^a	-2.29	-0.71	-4.41
			182.30^a	6.19	2.47	12.66
			-10.17^b	12.01	10.64	15.20

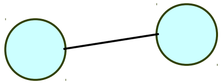
Random networks with prescribed degree frequencies

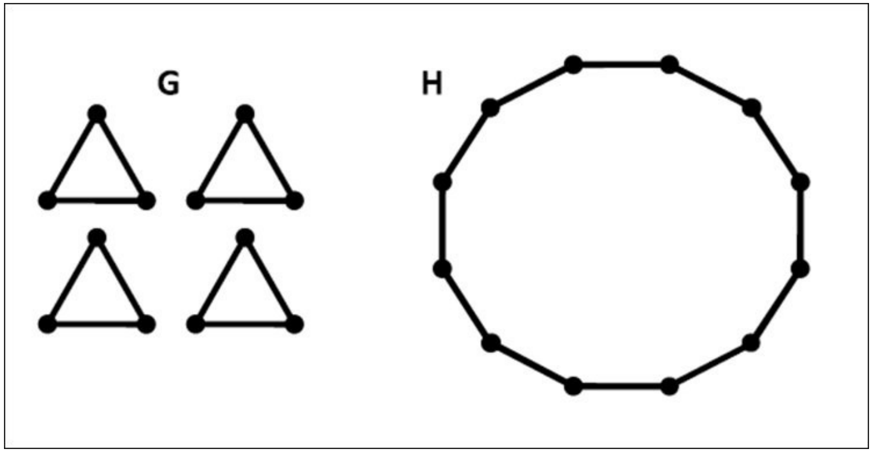
Graphlet Degree Distribution

- **What about a "node-level" subgraph metric?**

- **The degree distribution is in a way measuring participation in subgraphs of size 2** 
  - Can we generalize this?

Graphlet Degree Distribution

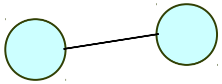
- **What about a "node-level" subgraph metric?**
- **The degree distribution is in a way measuring participation in subgraphs of size 2** 
  - Can we generalize this?

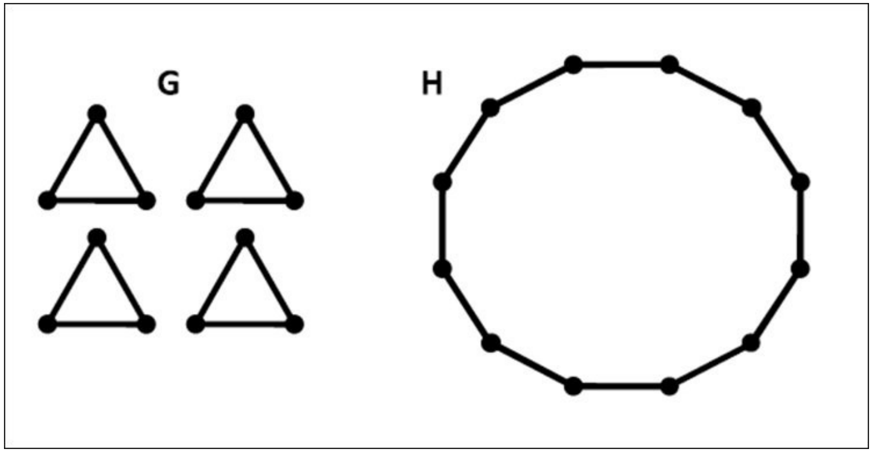


The same degree distribution can correspond to very different networks!

Graphlet Degree Distribution

- **What about a "node-level" subgraph metric?**

- **The degree distribution is in a way measuring participation in subgraphs of size 2** 
  - Can we generalize this?



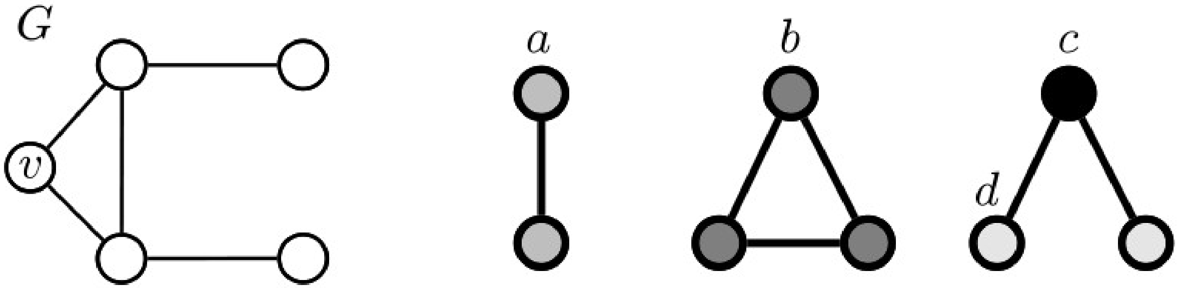
The same degree distribution can correspond to very different networks!

- **Przulj (2006) came up with the definition of graphlet degree distribution:**
  - Where does the node appear in **orbits** of subgraphs?

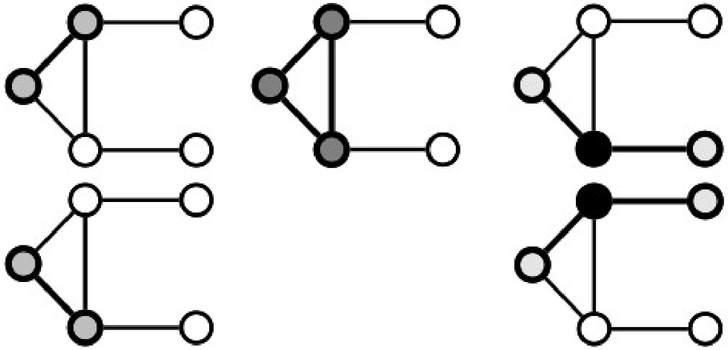
Graphlet Degree Vector

- An automorphism **"orbit"** takes into account the symmetries of the graph

- The **graphlet degree vector** is a feature vector with the frequency of the node in each orbit position

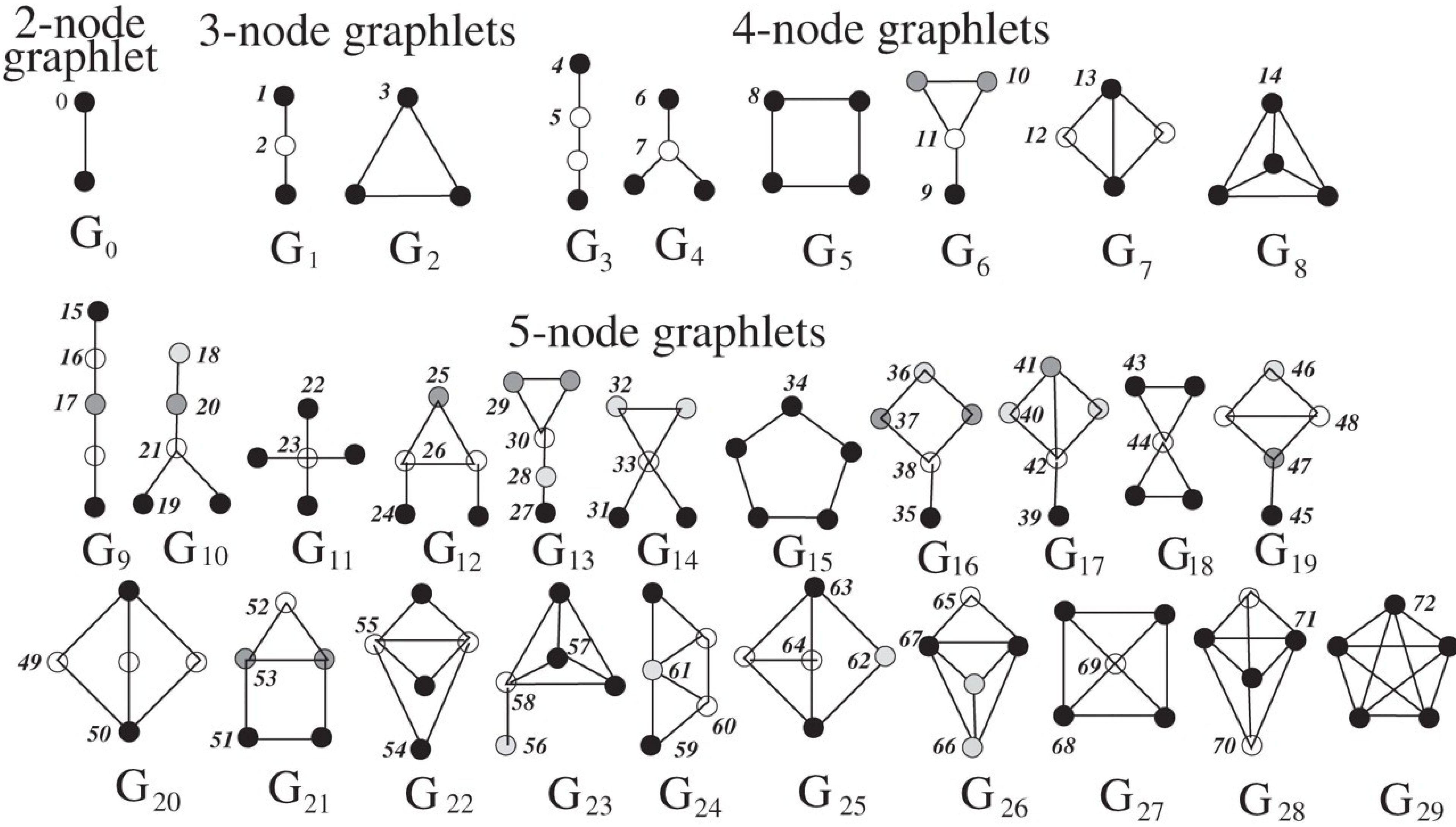


orbit	a	b	c	d
GDV(v)	2	1	0	2



Graphlet Degree Distribution

Equivalent to “degree distribution”



3) COMPUTATIONAL CHALLENGE

Computational Problem

- **In its core, finding motifs and graphlets its all about finding and counting subgraphs.**

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

- **Execution time grows exponentially as the size of the graph or the motif/graphlet increases**
  - Feasible motif size is usually small (3 to 8) and network size in the order of hundreds or thousands of nodes

What we have been doing

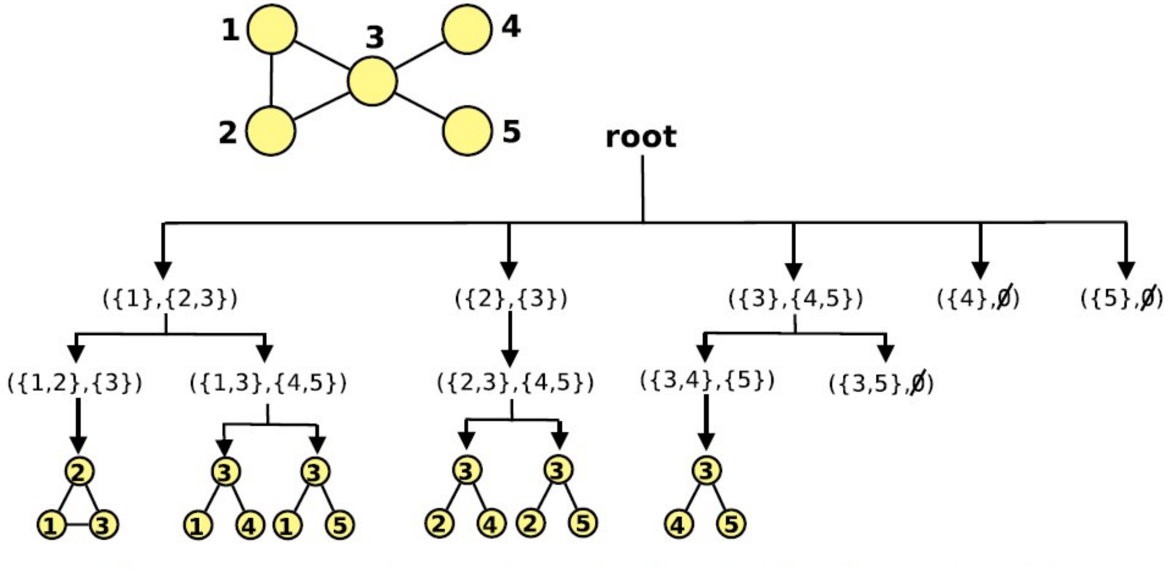
- **Our primary goal was to improve efficiency in network motif detection.**

Scale Up!

- **How?**
  - Novel **data structures** for the graphs and subgraphs
  - Novel faster **algorithms**
  - **Sampling** techniques
  - **Parallel** approaches (with different paradigms)

Previous Approaches

- **Network-centric approaches:**
  - Enumerate all *k*-connected sets of nodes and then compute isomorphisms (ex: ESU/Fanmod, Kavosh)



- **Subgraph-centric approaches:**
  - Find one subgraph at a time (ex: Grochow and Kellis)

A set-centric approach

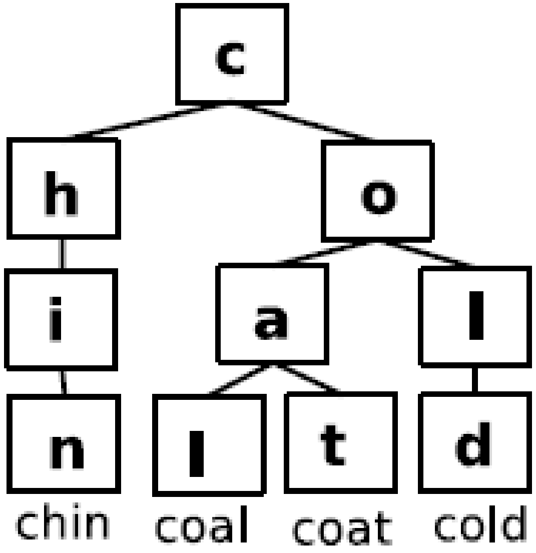
- **Key insight: can we do better looking for a given set of subgraphs?**
  - All *k*-subgraphs – even “uninteresting” subgraphs
  - One at a time – no re-usage of computation
  - Can we find what is **common between subgraphs** and use that?

- **Set-centric approach:**
  - Find a custom set of subgraphs
    (maybe one, maybe all, maybe something in between)

Inspiration

- **Sequences and prefix trees**

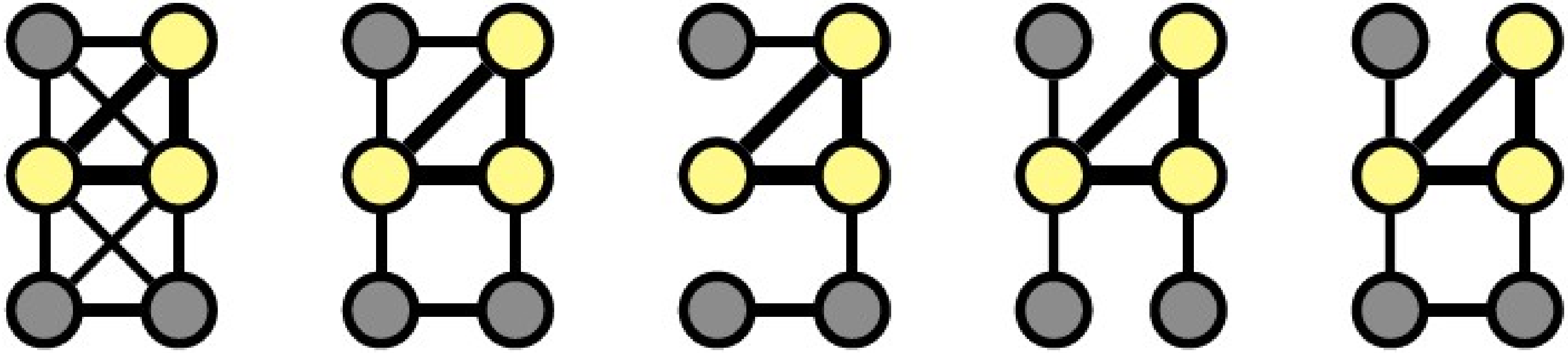
chin
coal
coat
cold



- **Can this concept be extended?**

Motivation and Concept

- Subgraphs have **common substructure**



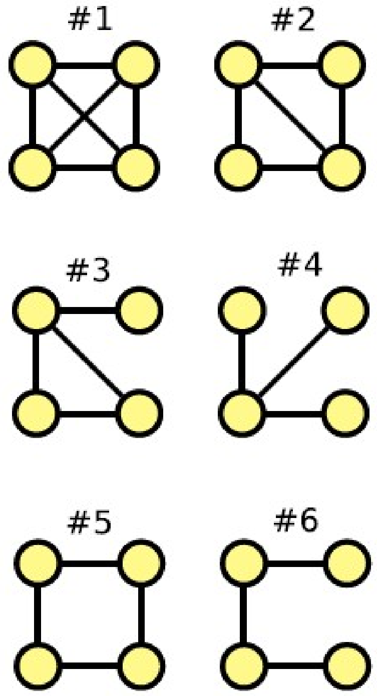
- Create a tree where each tree node corresponds to a single graph vertex

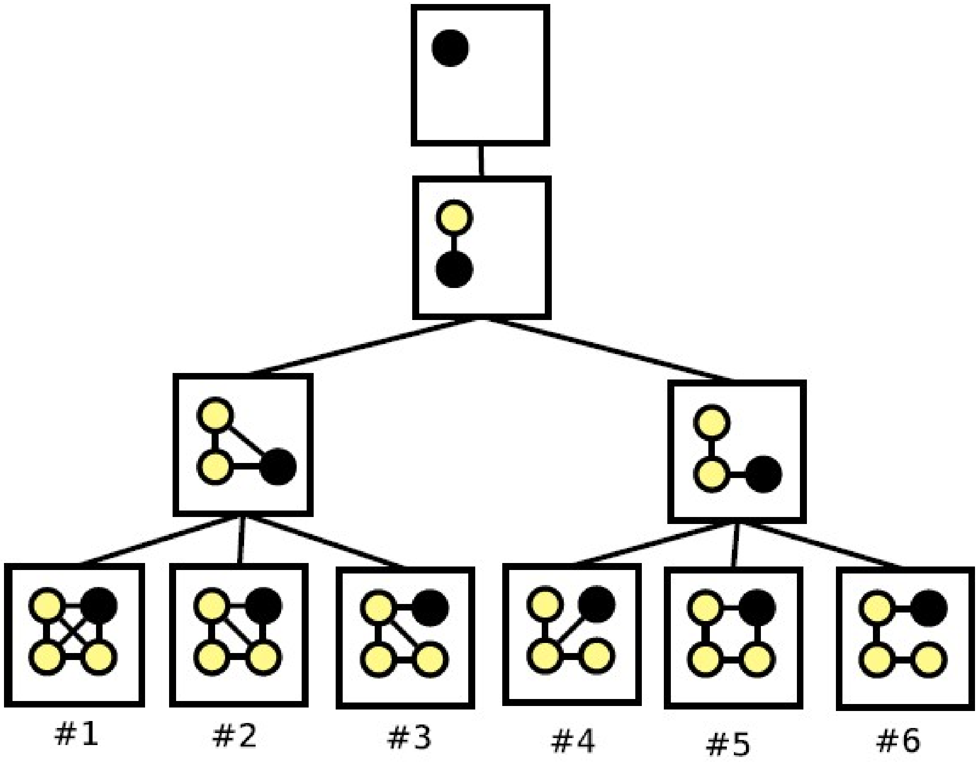
G-Tries

(etimology: **G**raph Ret**TRIE**val)

The G-Trie data structure

- **G-Tries:** (customized) collections of subgraphs
  - **Common substructures** are identified
  - Information is **"compressed"**

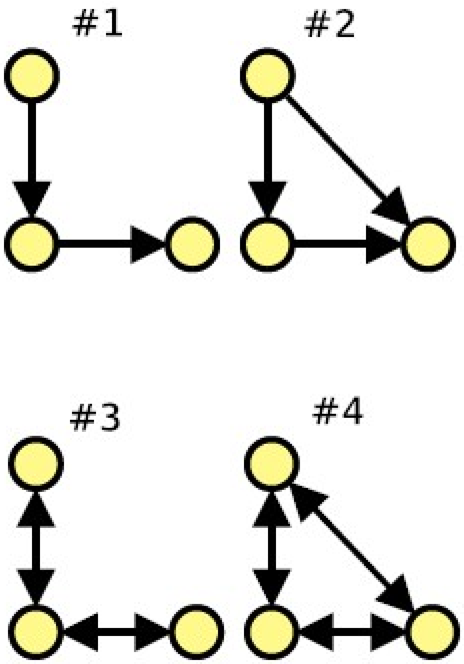


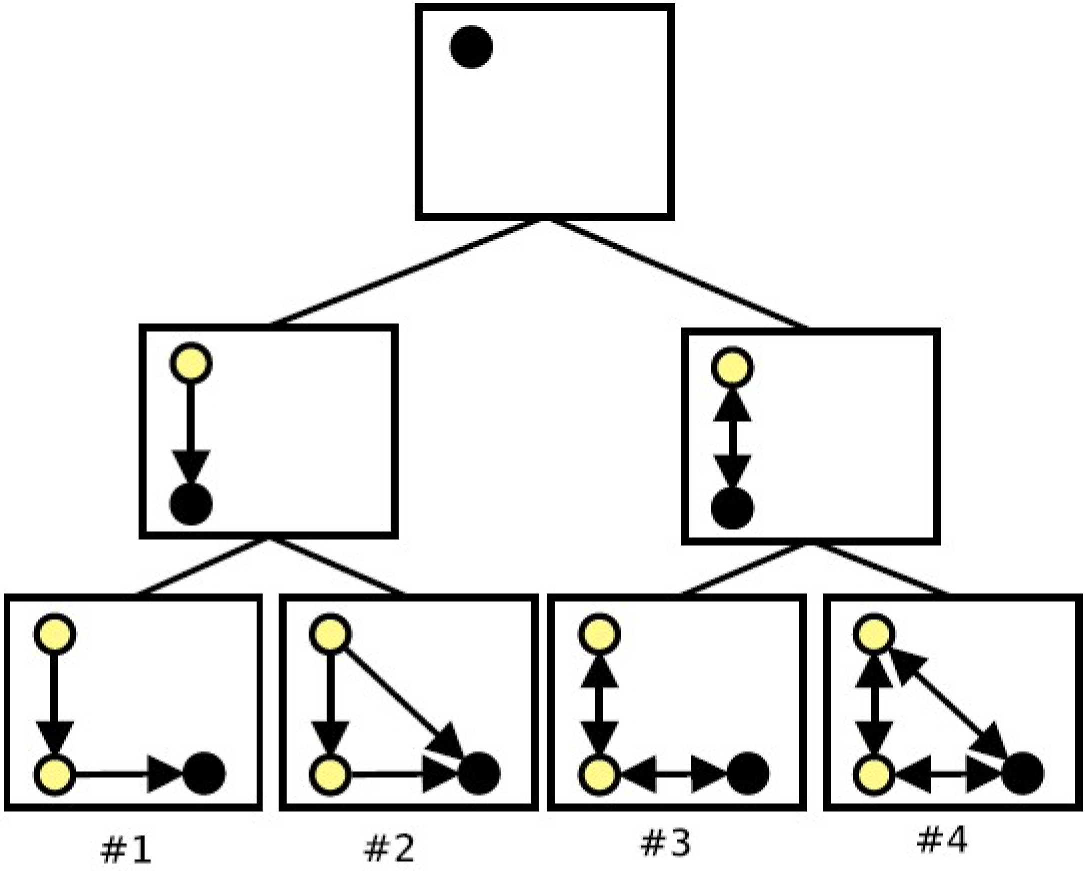


[Ribeiro & Silva, DMKD,2014]

The G-Trie data structure

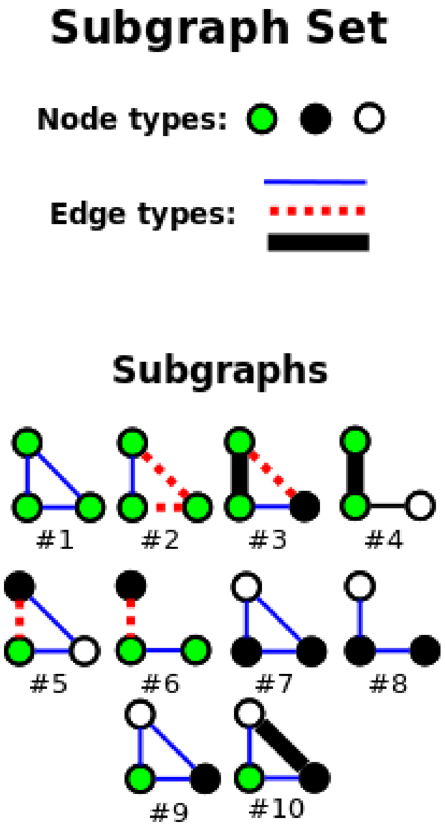
- **G-Tries:** also valid for directed networks

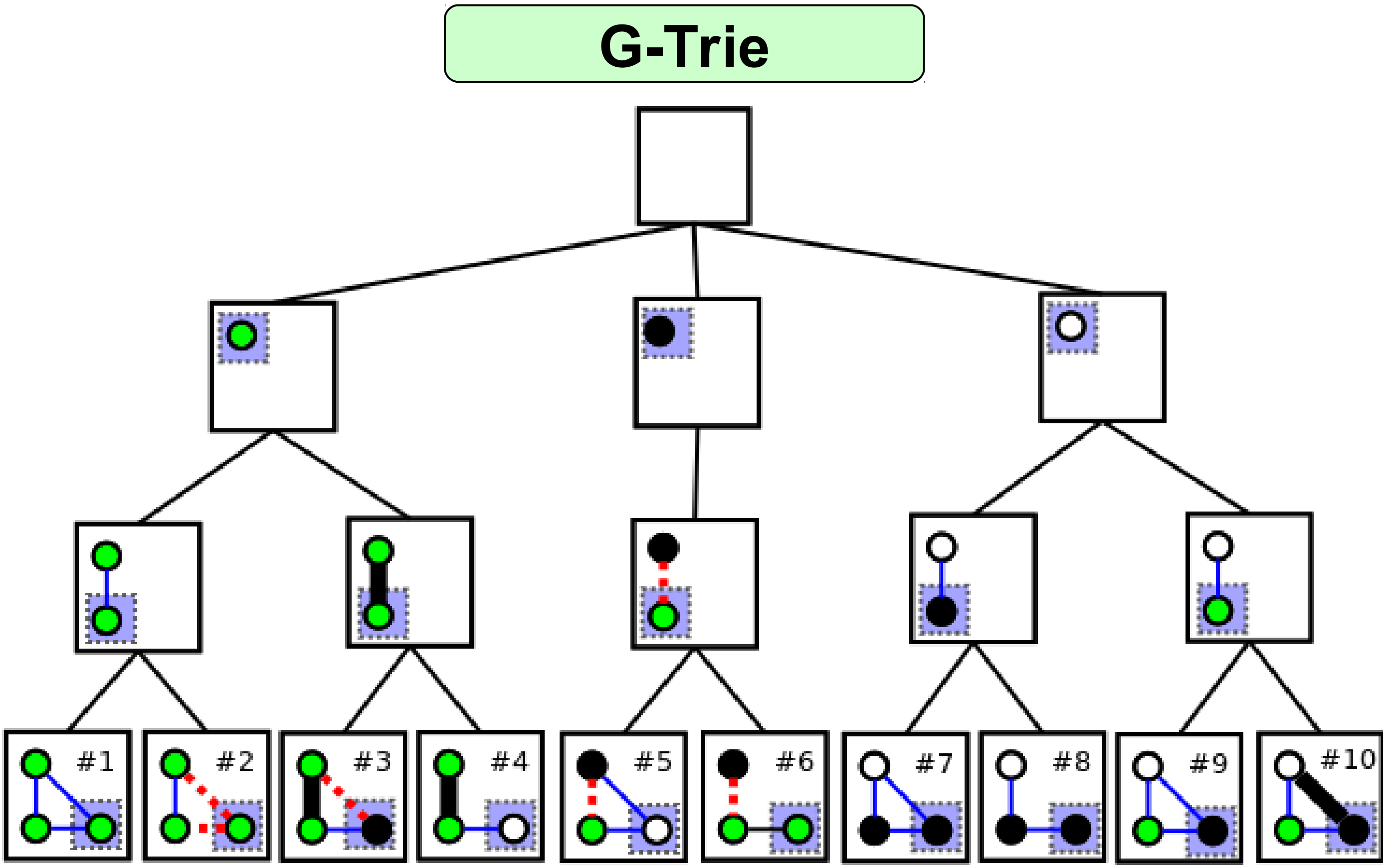




The G-Trie data structure

- **G-Tries:** also valid for colored/labeled networks

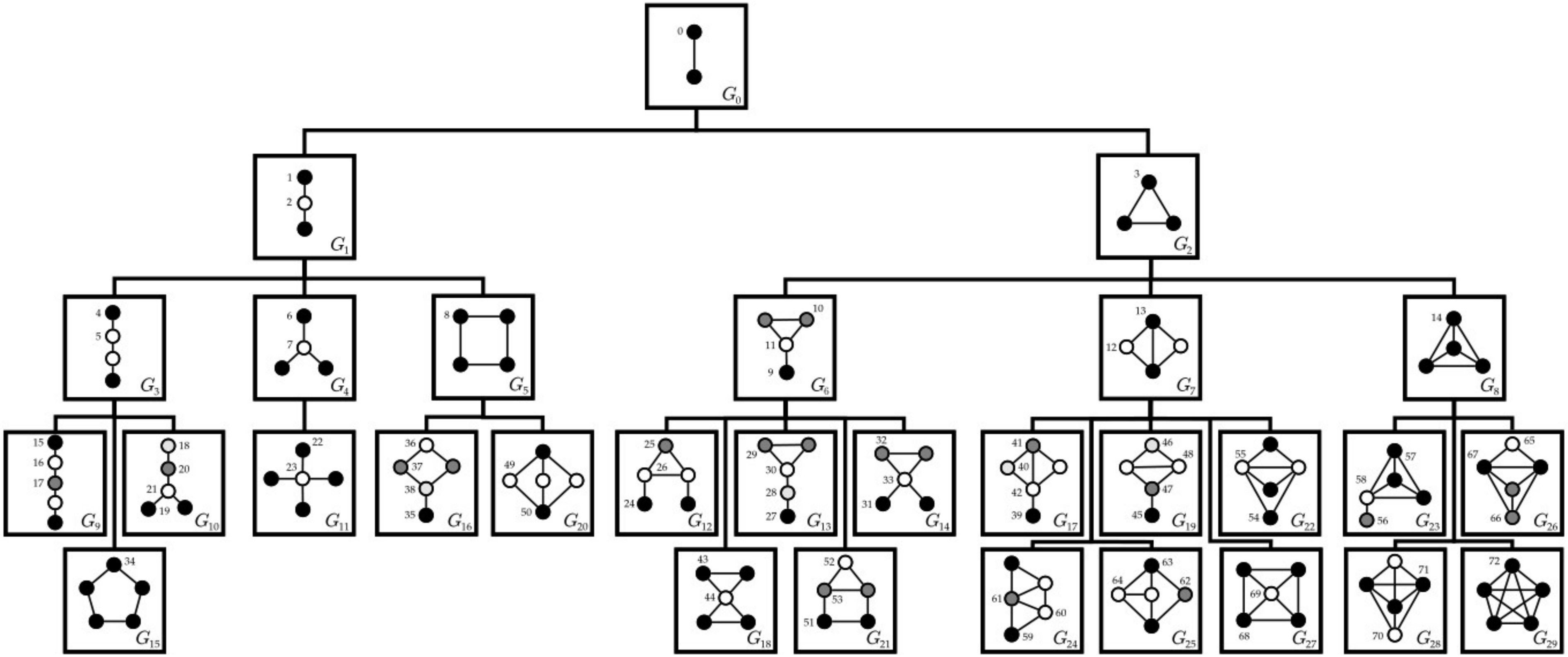




[Ribeiro & Silva, Complenet'2014]

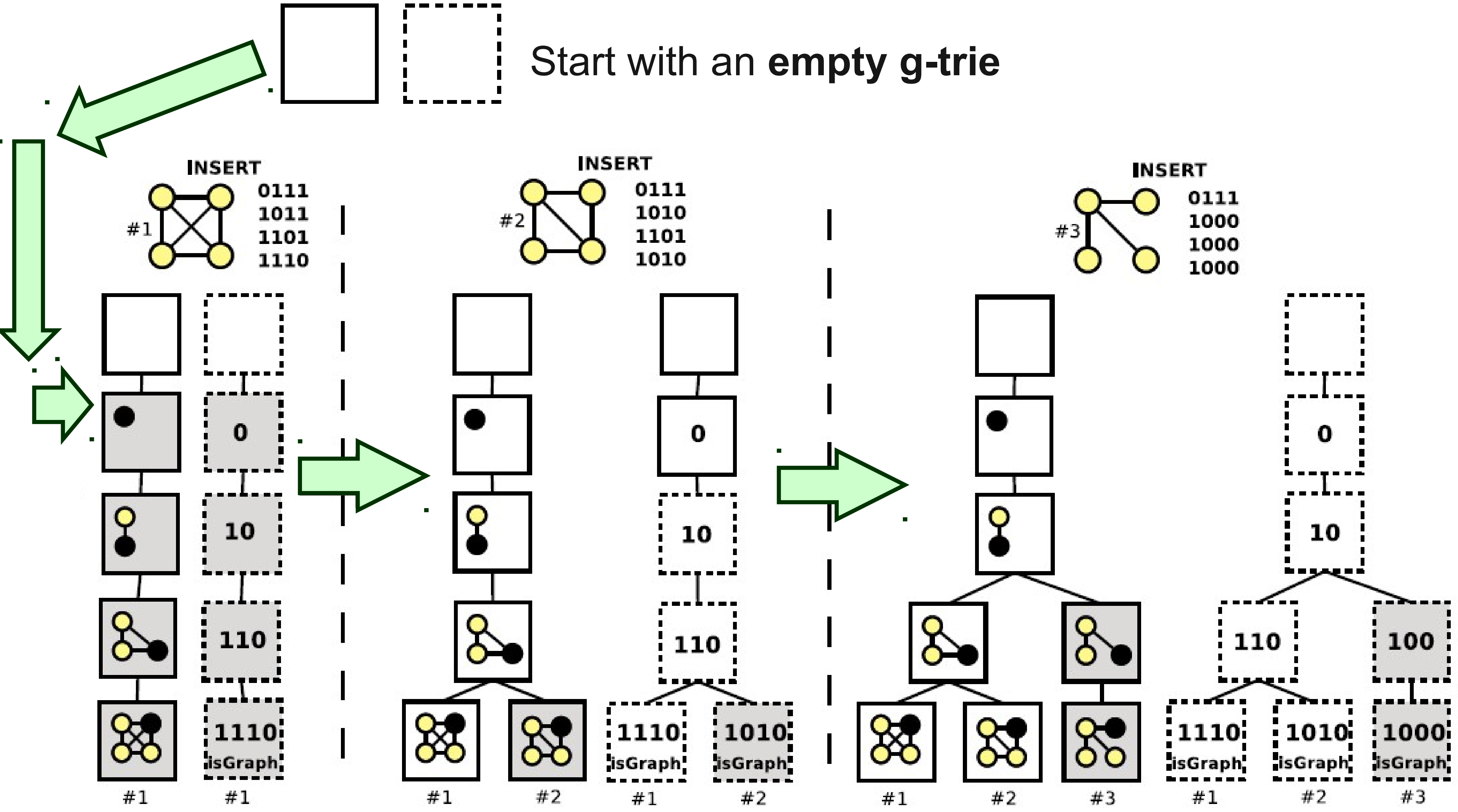
The G-Trie data structure

- **G-Tries:** can also incorporate **orbit** information



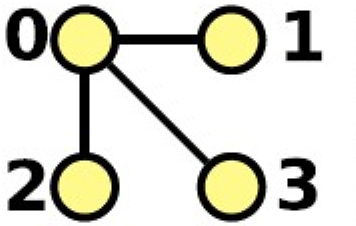
Creating a G-Trie

- **Iterative insertion**

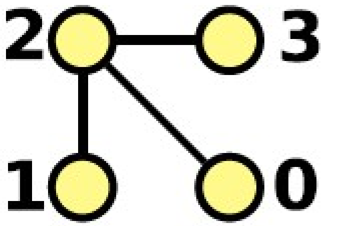


The Need for a Canonical Form

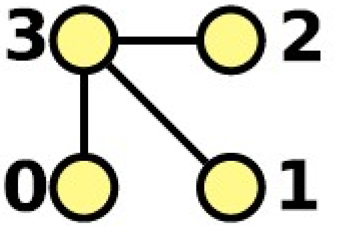
- There are different **node orderings** representing the same subgraph



```
0111
1000
1000
1000
```



```
0010
0010
1101
0010
```

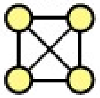
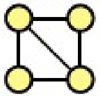
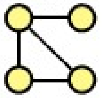
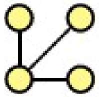
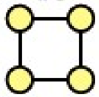
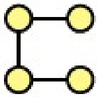


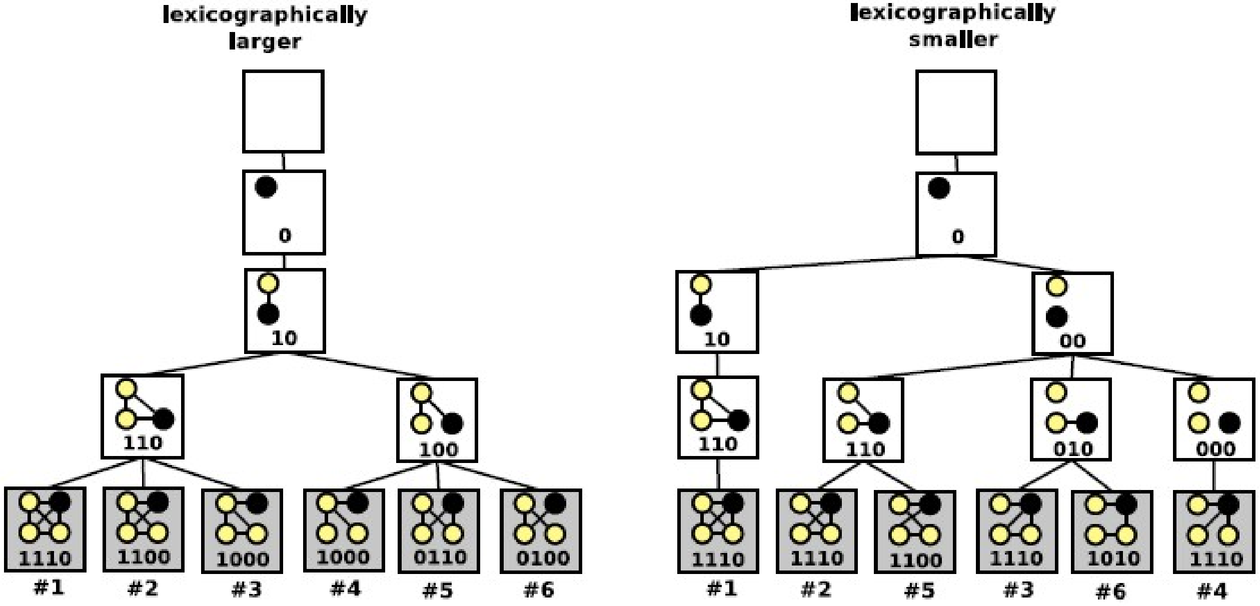
```
0001
0001
0001
1110
```

- **Canonical form** for a getting an unique g-trie

- Different canon will give origin to different g-tries

Impact of Canonical Form

	#1	#2	#3	#4	#5	#6
Graph						
lexicographically larger	0111 1011 1101 1110	0111 1011 1100 1100	0111 1010 1100 1000	0111 1000 1000 1000	0110 1001 1001 0110	0110 1001 1000 0100
lexicographically smaller	0111 1011 1101 1110	0011 0011 1101 1110	0001 0011 0101 1110	0001 0001 0001 1110	0011 0011 1100 1100	0001 0010 0101 1010

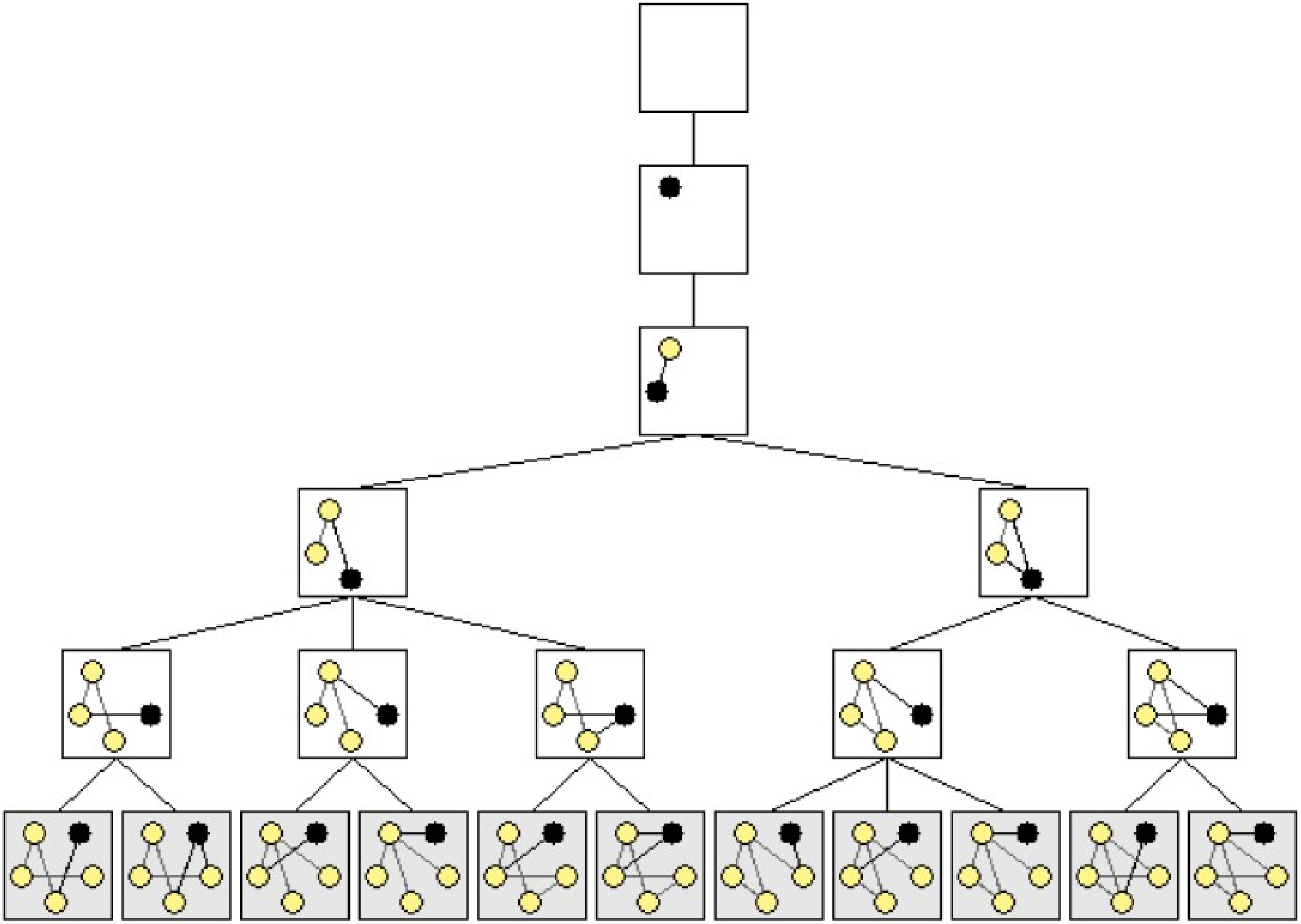


Custom Canonical Form

- **Connectivity**
  - Path induces connected subgraph
- **Compressibility**
  - More common substructure, less g-tries nodes
- **Constraining**
  - As many connections as possible to ancestor nodes (limit possible matches)

GTCanon

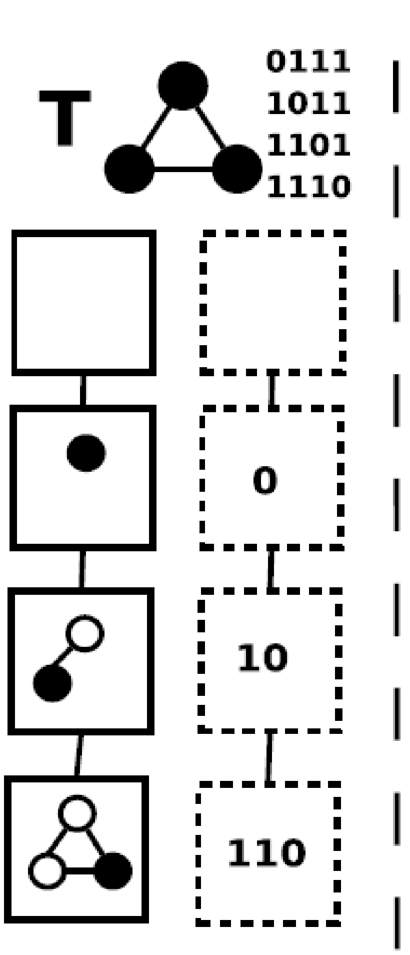
GTCanon Example

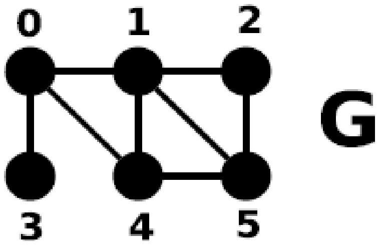


Searching with G-Tries

Backtracking Procedure

- Searching **at the same time** for several subgraphs

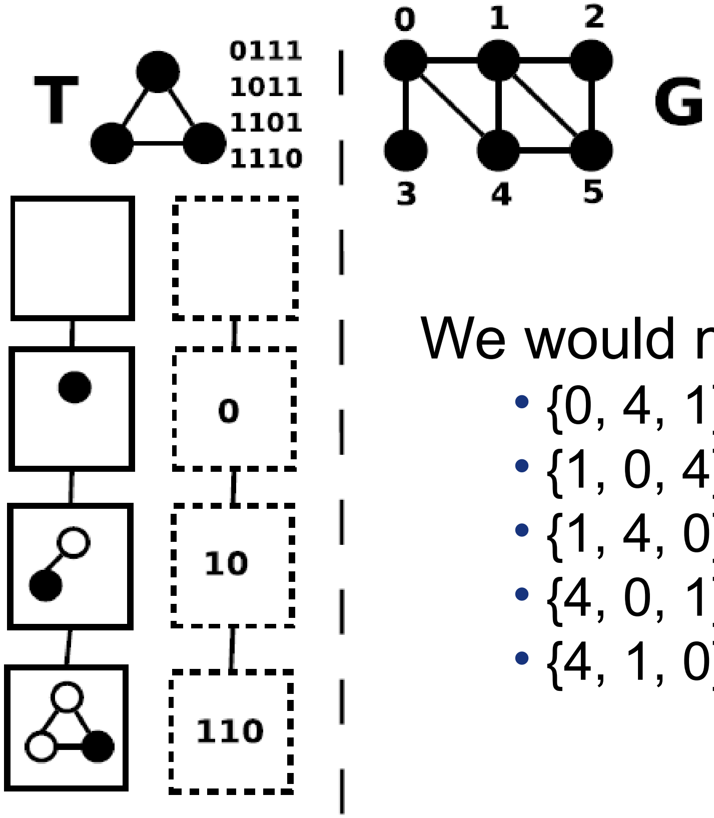




Candidates for node 1: {0, 1, 2, 3, 4, 5}
Try 0: Match = {0}, Neighb. = {1,3,4}
Try 1: Match = {0,1}, Neighb. = {2,3,4,5}
Try 2: no edge from 2 to 0! FAIL
Try 3: no edge from 3 to 1! FAIL
Try 4: Match = {0, 1, 4} FOUND!
Try 5: no edge from 5 to 1! FAIL

Searching with G-Tries

- **The same subgraph could be found several times due to automorphisms (symmetries)**

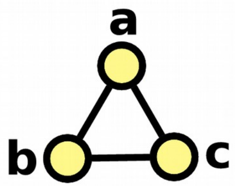


We would not only find {0,1,4} but also:

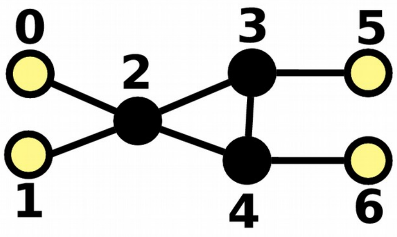
- {0, 4, 1}
- {1, 0, 4}
- {1, 4, 0}
- {4, 0, 1}
- {4, 1, 0}

Symmetry Breaking Conditions

- **Conditions on node labels**



Symmetry Breaking Conditions: **{a<b, b<c}**



Possible Matches of {a,b,c} in the graph of size 7:

{2,3,4} - OK!

~~{2,4,3}~~ - No match (b>c)

~~{3,2,4}~~ - No match (a>b)

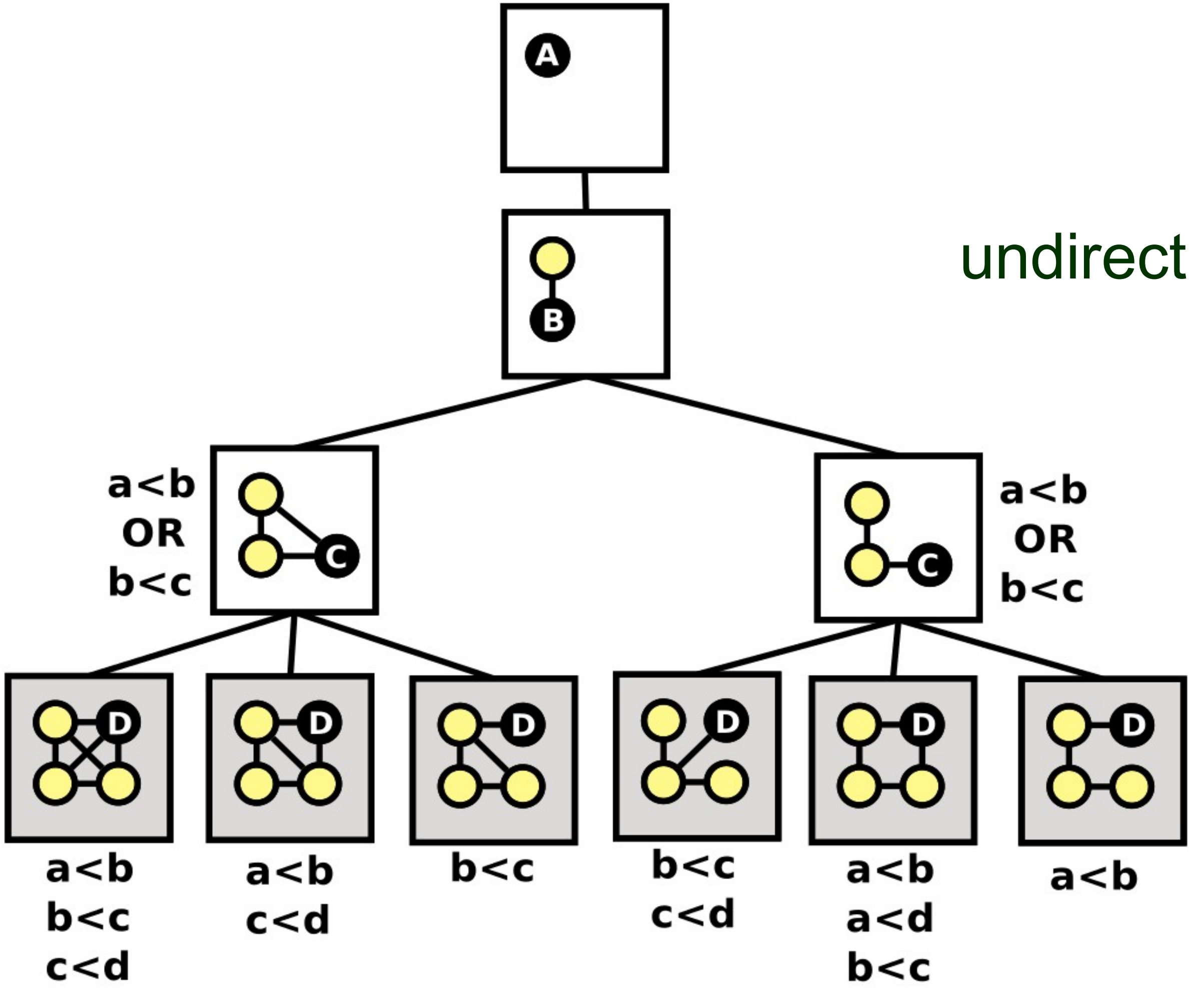
~~{3,4,2}~~ - No match (b>c)

~~{4,2,3}~~ - No match (a>b)

~~{4,3,2}~~ - No match (a>b, b>c)

- **Augment g-trie with these conditions**
  - Match only when conditions of at least one descendant are respected
- **Filter conditions to ensure minimum work**
  - Ex: transitive property (a<b,a<c,b<c leads to a<b, b<c); assured descendants, only store relevant to node, etc

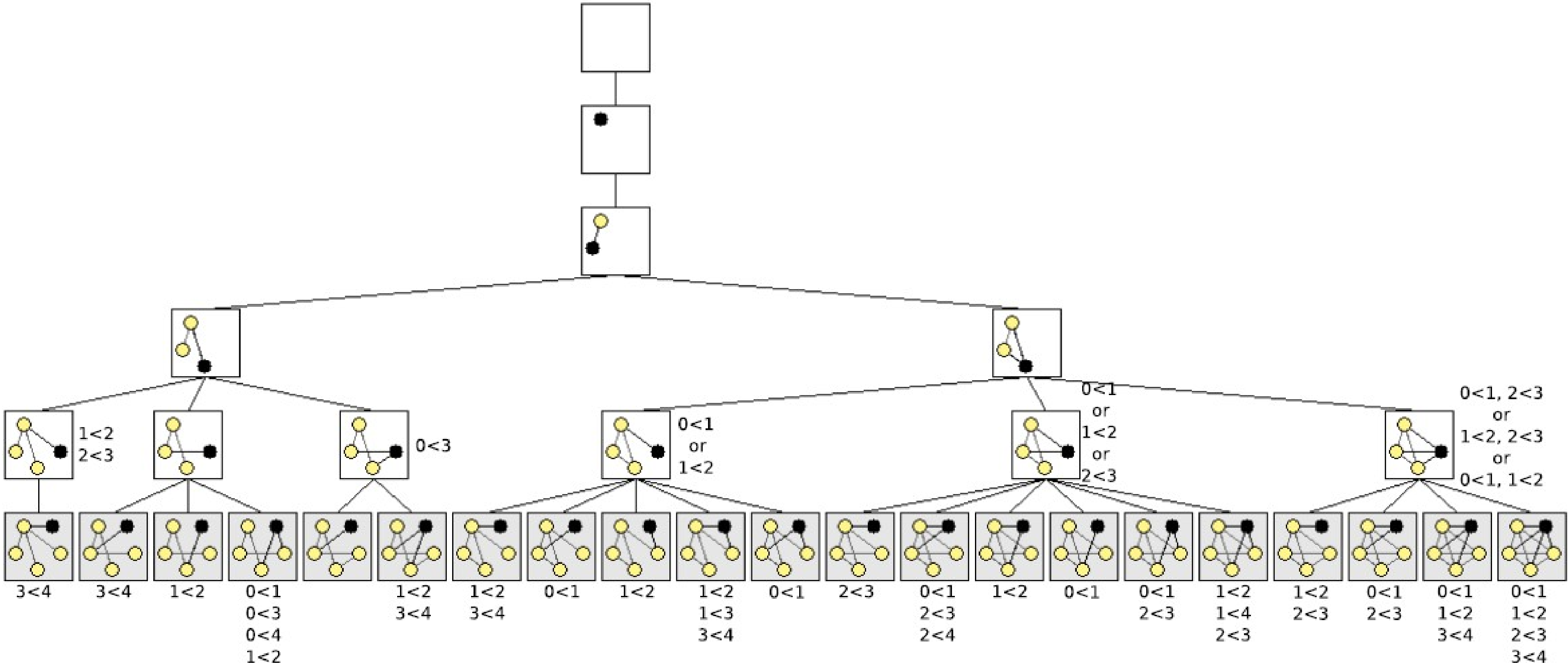
Complete G-Trie Example



All six undirected 4-subgraphs

Complete G-Trie Example

All 21 undirected 5-subgraphs



Sequential version: some results

- **Comparison with main competing algorithms**
  - ESU & Kavosh **(network-centric)**
  - Grochow and Kellys **(subgraph-centric)**

- **Implemented in common framework**
  - Implementation at least as efficient as original
  - C++ as the programming language
  - Efficient graph primitives
  - More "fair" comparison

Sequential version: some results

- **Set of 12 representative networks**

Network	Group	Directed	|V(G)|	|E(G)|	Nr. Neighbours	
					Average	Max
dolphins	social	no	62	159	5.1	12
circuit	physical	no	252	399	3.2	14
neural	biological	yes	297	2,345	14.5	134
metabolic	biological	yes	453	2,025	8.9	237
links	social	yes	1,490	19,022	22.4	351
coauthors	social	no	1,589	2,742	3.5	34
ppi	biological	no	2,361	6,646	5.6	64
odlis	semantic	yes	2,909	18,241	11.3	592
power	physical	no	4,941	6,594	2.7	19
company	social	yes	8,497	6,724	1.6	552
foldoc	Semantic	yes	13,356	120,238	13.7	728
internet	Physical	no	22,963	48,436	4.2	2,390

Sequential version: some results

- **On both directed and undirected graphs we were from 1 to 2 orders of magnitude faster than existing state of the art at that time**
  - From 10x to 200x

Example results for **full census of size *k***
(***speedup*** *on a set of **undirected** networks)*

Network	k	ESU	Kavosh	Grochow
dolphins	8	28.9	26.9	39.5
circuit	9	53.2	52.0	39.4
coauthors	6	64.4	66.3	39.7
ppi	5	61.8	62.1	25.6
power	7	38.2	38.0	285.9
internet	4	46.9	45.5	14.7

Sequential version: some results

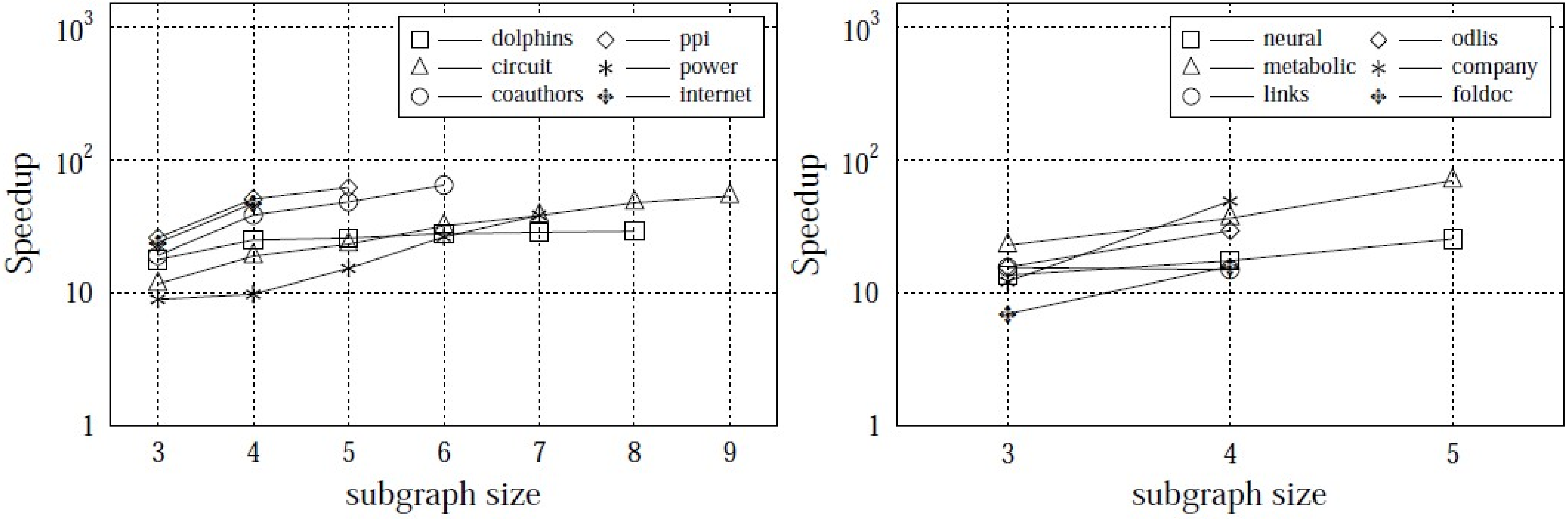
- **On both directed and undirected graphs we were from 1 to 2 orders of magnitude faster than existing state of the art at that time**
  - From 10x to 200x

Example results for **full census of size *k***
(***speedup*** *on a set of* ***directed*** *networks*)

Network	K	ESU	Kavosh	Grochow
neural	5	25.3	25.5	28.8
metabolic	5	69.9	68.9	15.4
links	4	14.9	15.2	13.2
odlis	4	29.3	29.7	22.6
company	4	48.9	50.1	25.3
foldoc	4	15.8	16.0	50.5

Sequential version: some results

- **On both directed and undirected graphs we were from 1 to 2 orders of magnitude faster than existing state of the art at that time**
  - From 10x to 200x

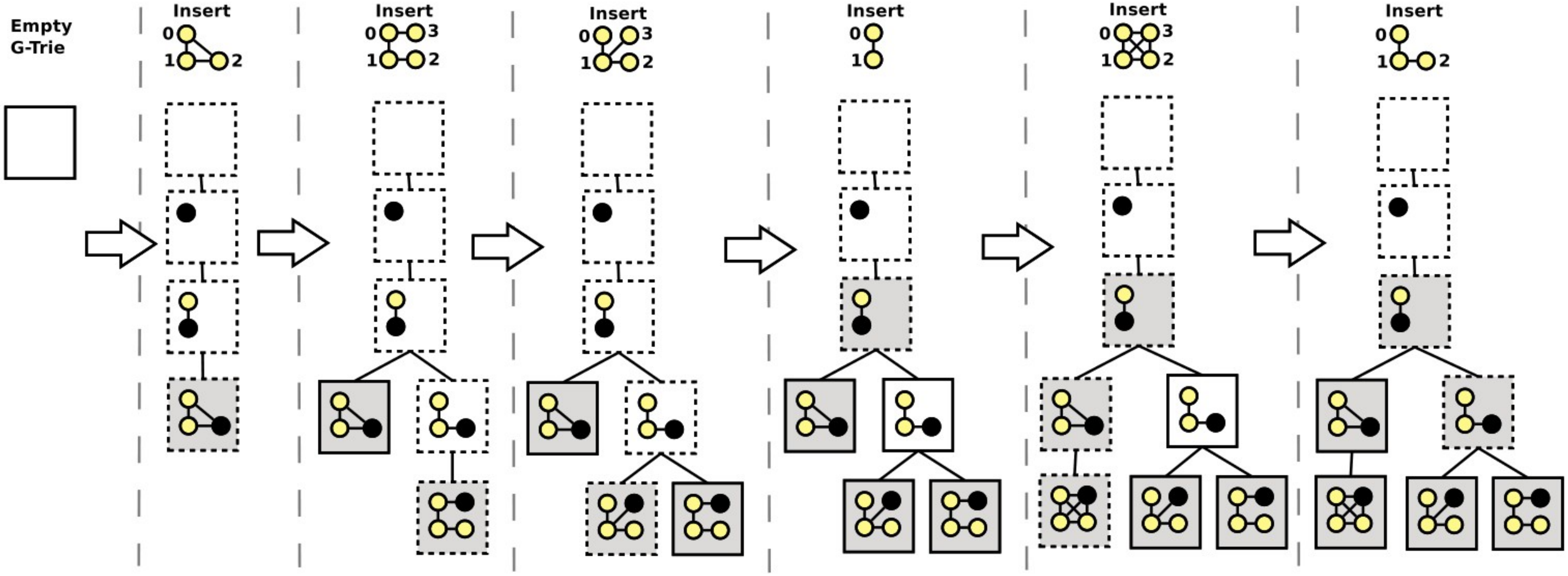


(a) vs ESU on undirected networks

(b) vs ESU on directed networks

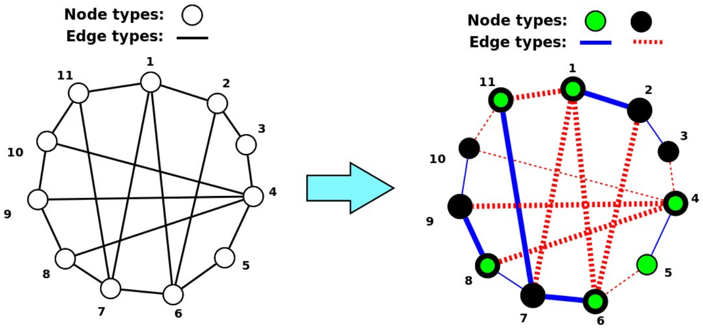
Sequential version: some results

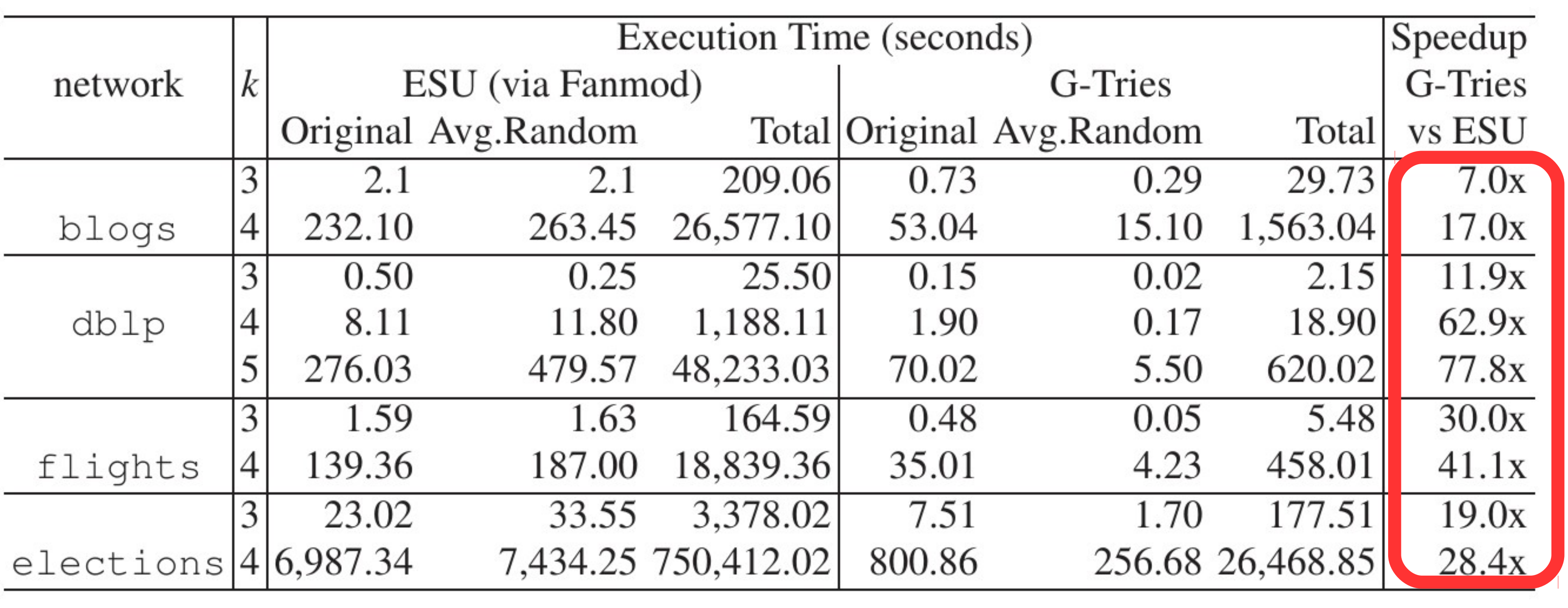
- **Speedup also when looking for different sets of subgraphs (other than full census of size k)**
  - Better speedup as more subgraphs are being searched at the same time **(set-centric)**



Sequential version: some results

- **Speedup** also when using colored networks

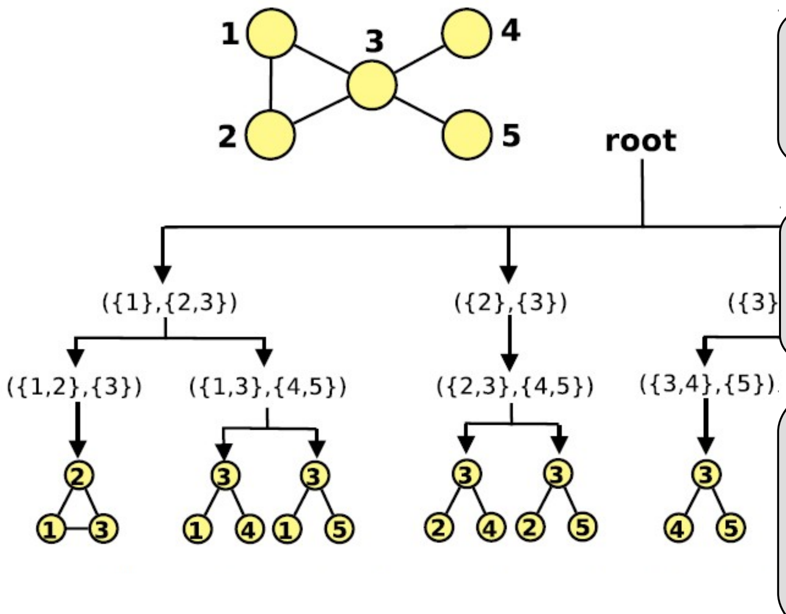




network	*k*	Execution Time (seconds)						Speedup
		ESU (via Fanmod)			G-Tries			G-Tries
		Original	Avg.Random	Total	Original	Avg.Random	Total	vs ESU
blogs	3	2.1	2.1	209.06	0.73	0.29	29.73	7.0x
	4	232.10	263.45	26,577.10	53.04	15.10	1,563.04	17.0x
dblp	3	0.50	0.25	25.50	0.15	0.02	2.15	11.9x
	4	8.11	11.80	1,188.11	1.90	0.17	18.90	62.9x
	5	276.03	479.57	48,233.03	70.02	5.50	620.02	77.8x
flights	3	1.59	1.63	164.59	0.48	0.05	5.48	30.0x
	4	139.36	187.00	18,839.36	35.01	4.23	458.01	41.1x
elections	3	23.02	33.55	3,378.02	7.51	1.70	177.51	19.0x
	4	6,987.34	7,434.25	750,412.02	800.86	256.68	26,468.85	28.4x

Dynamic G-Tries

- **Speedup also when adapting to network-centric methodology**
  - Use as base any enumeration method (e.g. ESU)



While enumerating, embed occurrences in a **g-trie**!

Different leafs may represent the same subgraph due to symmetries!

Use **nauty** for discovering isomorphic classes of leafs (but occurrences are grouped, avoiding redundancy)

FaSE – Fast Subgraph Enumeration

G-Trie Dynamically Built

[Paredes & Ribeiro, ASONAM' 2013]

Graph Representations

- **Core graph primitive is edge verification**
  - Adjacency Matrix (AdjMat) gives that in O(1)
  - Used when $O(n^2)$ fits in memory

- **For larger sparse graphs we use an hybrid representation:**
  - Combine linear search + hash tables + trie
  - Low-level optimizations (cache, bitwise ops, ...)

[Paredes & Ribeiro, NetSciX'2016]

- **Overhead with AdjMat is small !**
  - From 4x more with binary search

⇩

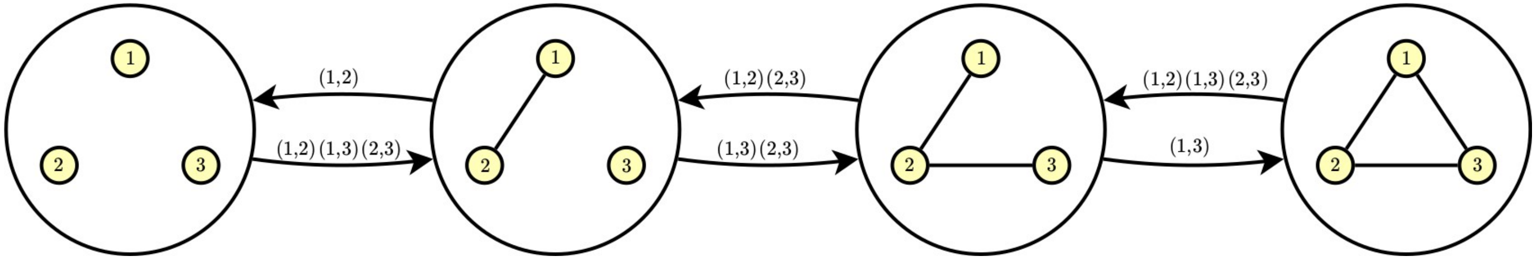
  - **Less than 1.5x on average with hybrid approach**

Iterative updates

- **Update subgraph counts after edge deletion or removal**
  - Take into account only the subgraphs that touch(ed) that particular edge

[Silva, Paredes & Ribeiro, Complenet'2017]

- **Add the capability of following the isomorphic type of a set of nodes**
  - Edge updates change the type of subgraph

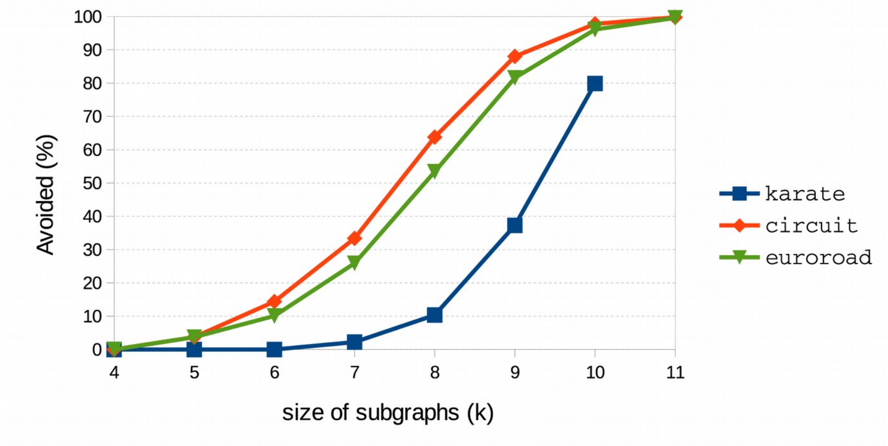


Automaton to keep subgraph type as “state”

[Paredes & Ribeiro, Complenet'2018]

Improve motif discovery

- **Iterative deepening of subgraph size**
  - Start with smaller sizes and keep incrementing
  - Discard supergraphs that contain *non-interesting* subgraphs (ex: frequency > 0)
  - Generate only supergraphs of *interesting* subgraphs

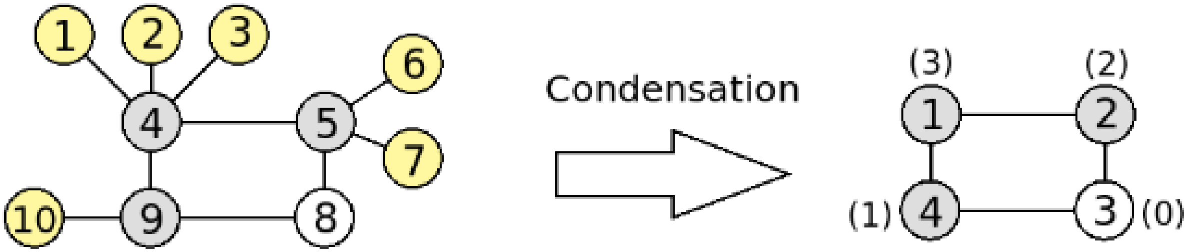


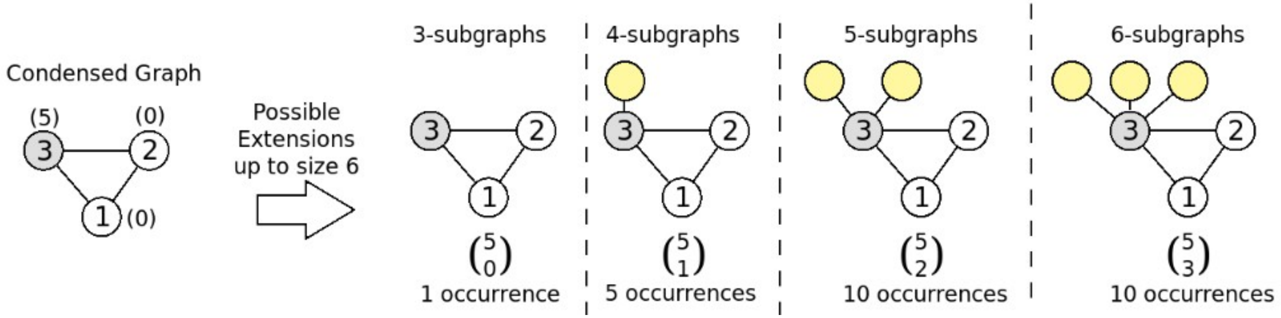
Improve candidate subgraph generation

[Grácio & Ribeiro, Complenet'2019]

Improve motif discovery

- **Combinatorial optimizations**
  - Lossless compression of original graph
  - Count on reduced graph; extrapolate results



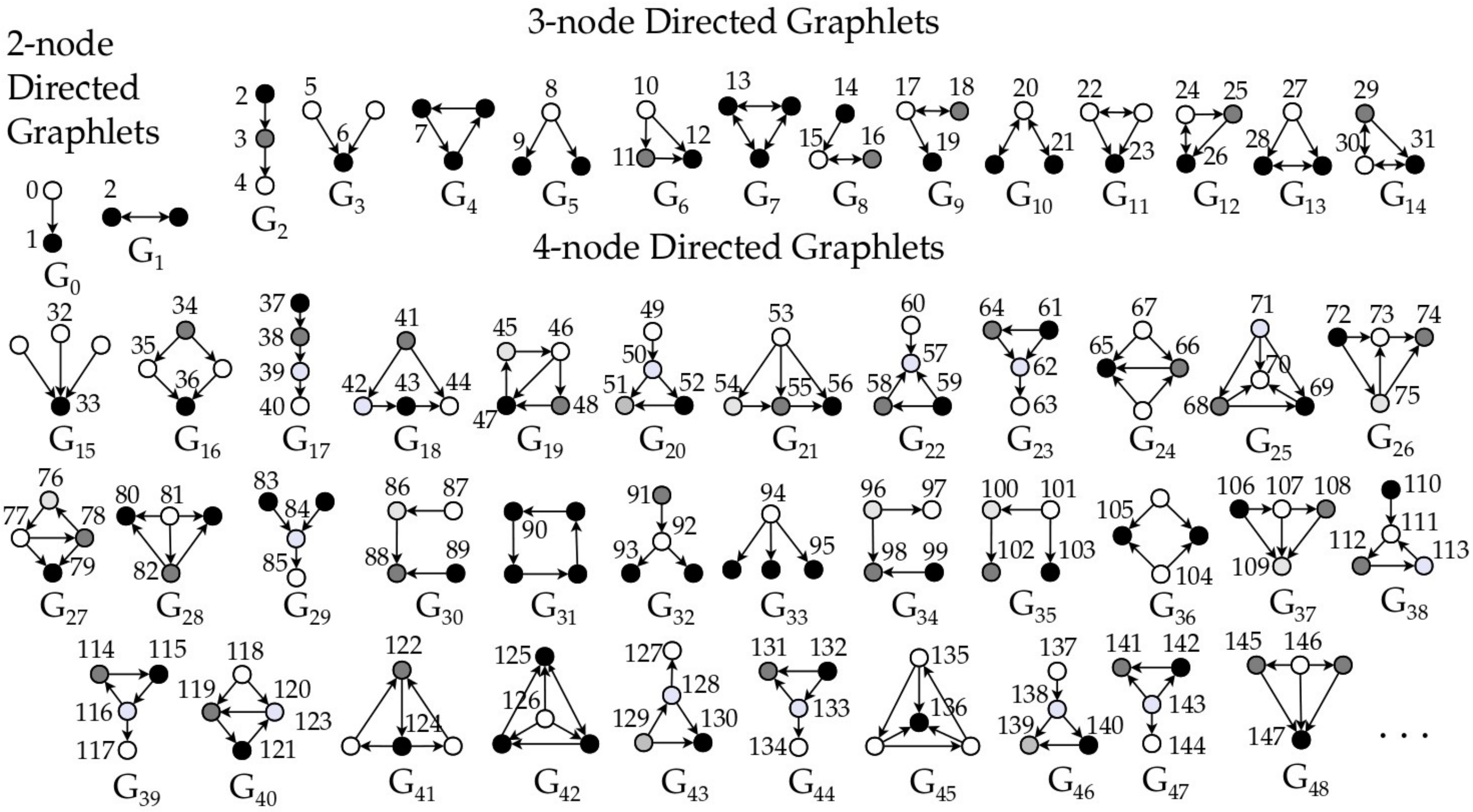


Account for multiple occurrences once

[Martins & Ribeiro, Complenet'2020]

Extending existing metrics

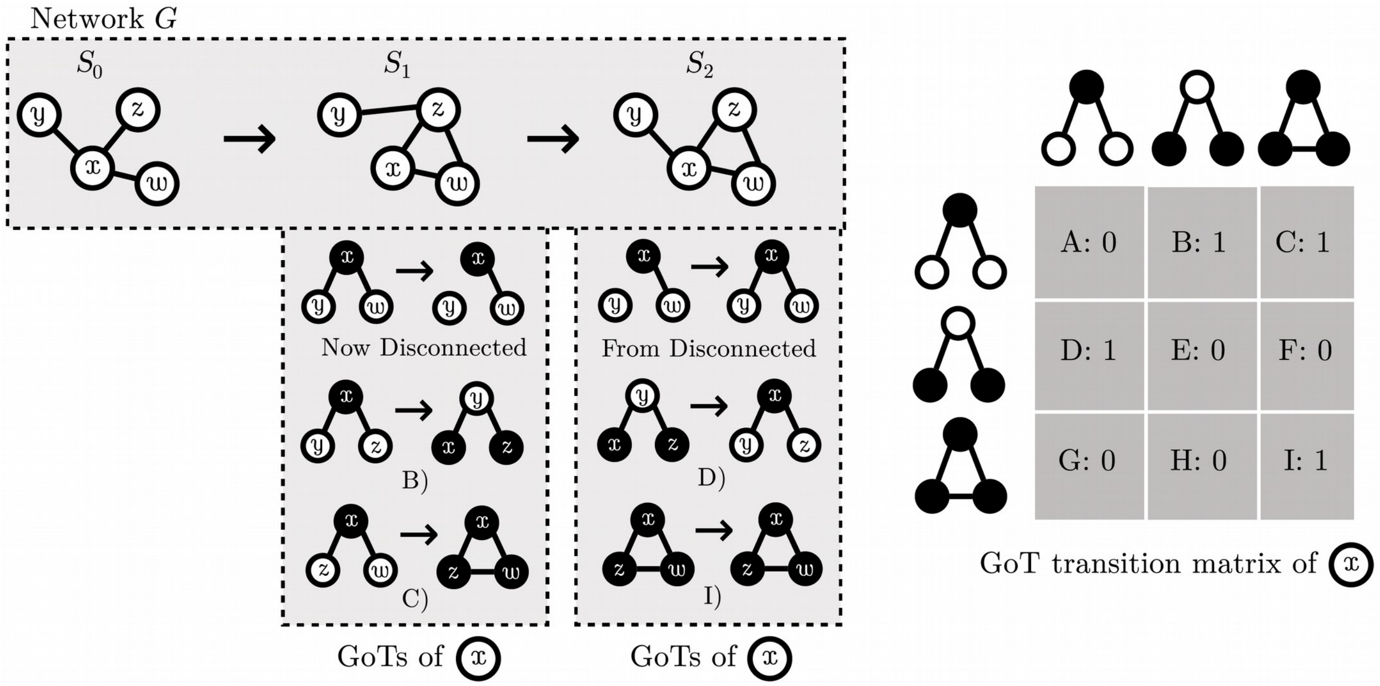
- **Extending the applicability of graphlets to directed networks**



[Aparício, Ribeiro & Silva, TCBB, 2017]

Temporal networks

- **Study evolution of subgraphs**



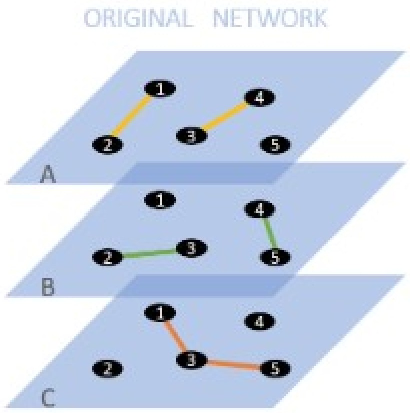
A: 0	B: 1	C: 1
D: 1	E: 0	F: 0
G: 0	H: 0	I: 1

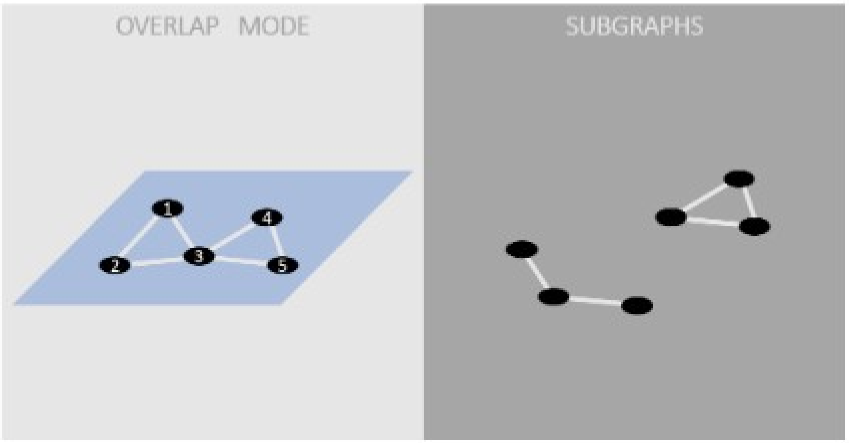
Graphlet-Orbit Transitions (**GoT**): fingerprints for temporal network comparison

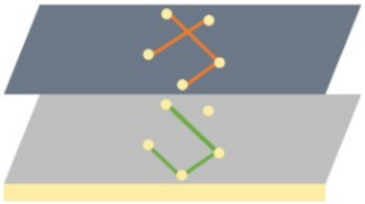
[Aparício, Ribeiro & Silva, PloS One, 2018]

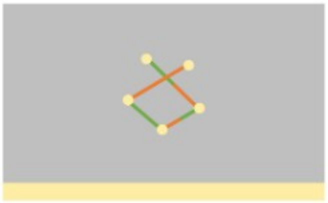
Multilayer Networks

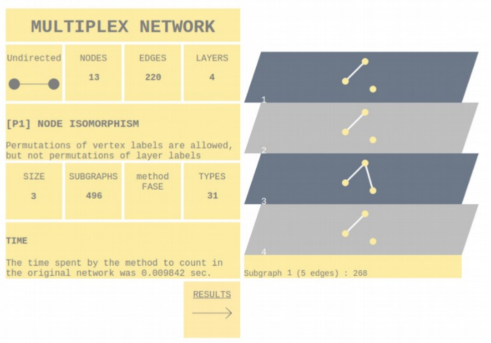
- **Motifs in networks with multiple layers**











MULTIPLEX NETWORK

Undirected	NODES	EDGES	LAYERS
●—●	13	220	4

[P1] NODE ISOMORPHISM

Permutations of vertex labels are allowed,
but not permutations of layer labels

SIZE	SUBGRAPHS	method	TYPES
3	496	FASE	31

TIME

The time spent by the method to count in
the original network was 0.009842 sec.

Subgraph 1 (5 edges) : 268

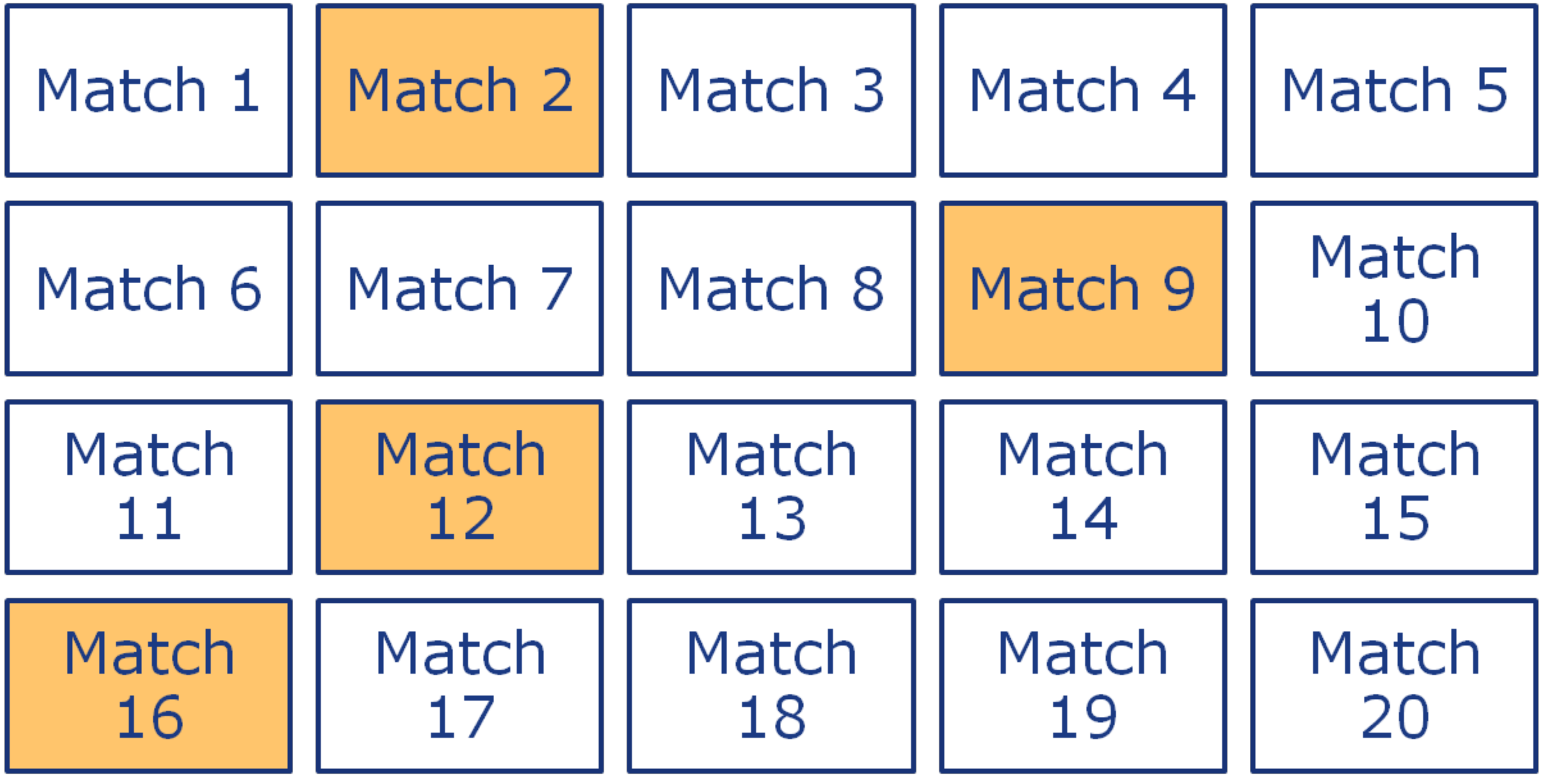
Journal submission being prepared

[Meira & Ribeiro, *in preparation*]

4) SAMPLING APPROACH

Approximating results

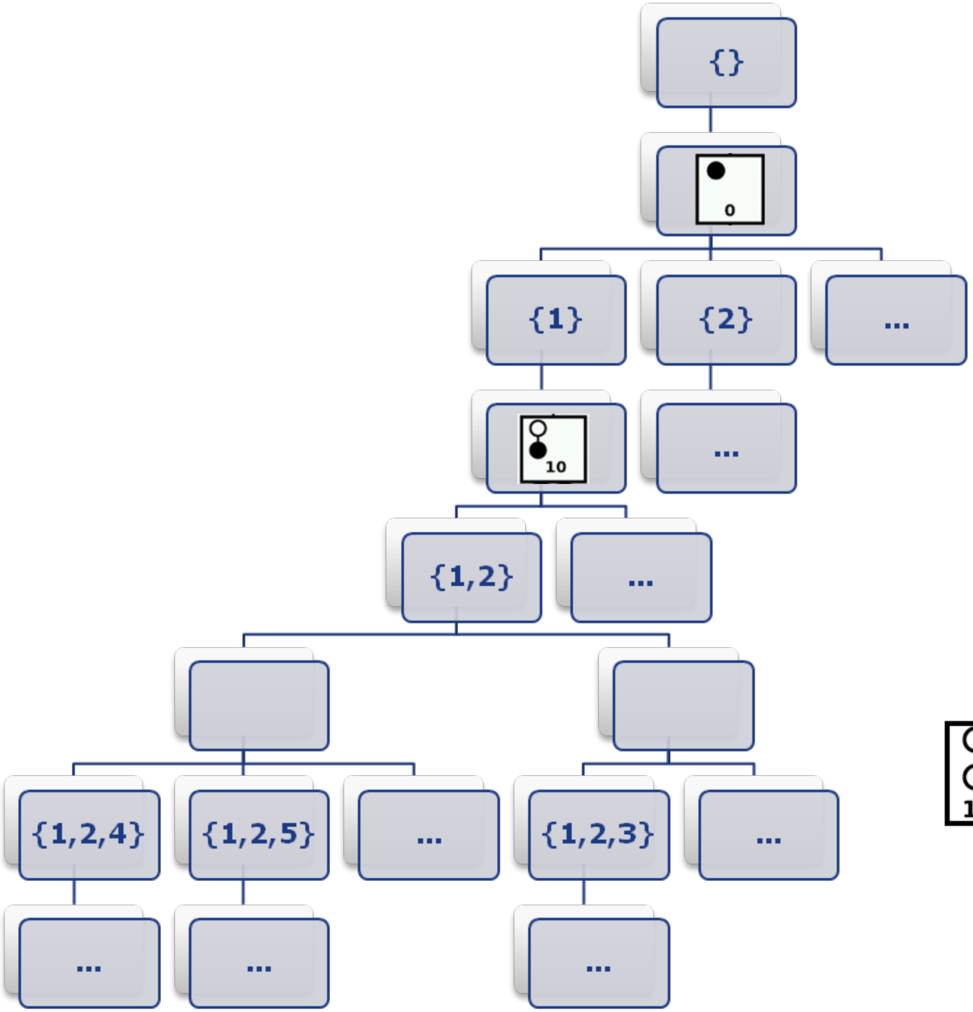
- **Sample subgraph occurrences**
  - Compute approximate results
  - Trade **accuracy** for **speed**

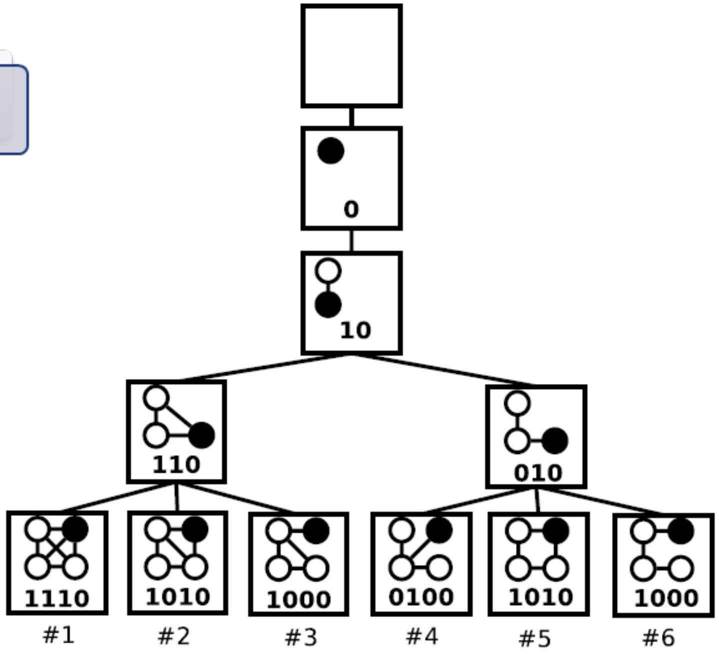


Match 1	Match 2	Match 3	Match 4	Match 5
Match 6	Match 7	Match 8	Match 9	Match 10
Match 11	Match 12	Match 13	Match 14	Match 15
Match 16	Match 17	Match 18	Match 19	Match 20

Sampling approach

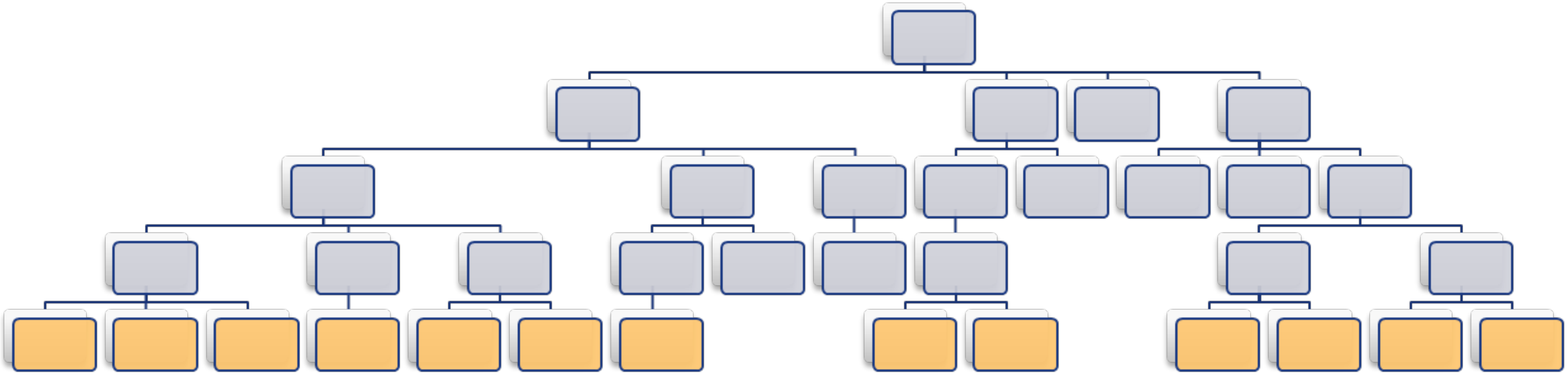
- **Backtracking procedure produces search tree**





Sampling approach

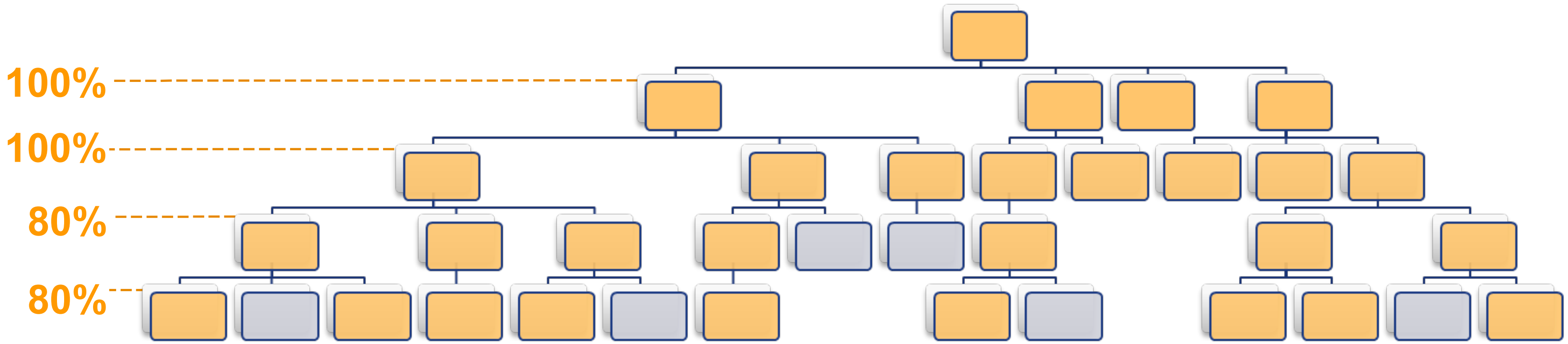
- **Original:** unbalanced search tree
- **Goal:** **uniform** sampling of occurrences



Subgraph Occurrences on *k*-census are in the last tree level

Sampling approach

- **Original:** unbalanced search tree
- **Goal: uniform** sampling



Associate a probability with traversing each search tree depth

Sampling approach

- **Probabilities associated with each depth:**
  - $\{P_0, P_1, P_2, \ldots, P_{max}\}$
- **Sampling is uniform:**
  - Probability of finding any occurrence is $P_0 \times P_1 \times P_2 \times \ldots \times P_{max}$
- **We can produce an unbiased estimator:**
  - Estimate of frequency of subgraph S =

$$\frac{\text{Nr of sampled occurrences of S}}{P_0 \times P_1 \times P_2 \times \ldots \times P_{max}}$$

Sampling approach

- The probabilities P_i control the search
- Regarding **accuracy**: avoid small values of probability close to the root
  - Entire search branches disregarded → more variance
- Regarding **execution times**: avoid high values if probability close to the root
  - More search branches explored → more time
- Choice should be **balanced**

Sampling approach: some results

- **90% accuracy for motif detection in less than 20% of time**

[Ribeiro & Silva, WABI'2010]

- **First sampling process for customized sets of subgraphs**
  - Only sample the subgraphs we want

- **Many parametrization choices**
  - Adaptable for different use cases
  - Possible to refine prediction for desired set of subgraphs

Adaptive sampling: ongoing work

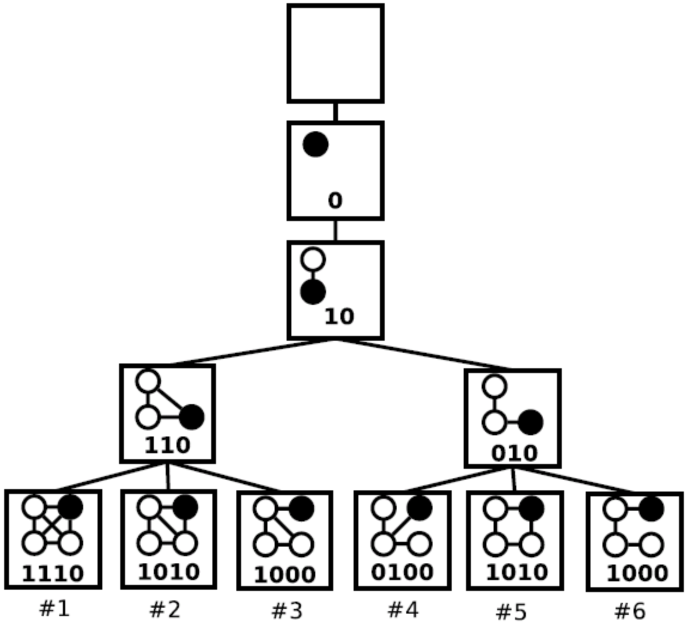
- **Adapt the sampling process:**
  - To the network
  - To the subgraphs being searched
  - To the available running time

- **High level ideas of the algorithm:**
  - Do several sampling iterations and look at how estimations are converging
    Ex: frequent subgraphs are easier to estimate
  - Change sampling weights
  - Changesubgraphs in the g-trie

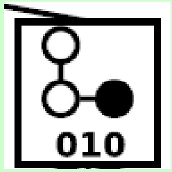
5) PARALLEL APPROACH

Opportunities for parallelization

- Sequential version produces a **tree-shaped** search tree
- Search tree nodes are **independent** from each other



{0,1,3}



If we know where we are,
we can continue from there

Tree Nodes -> Work Units

Initial Parallel Problem

- **Input:** set of **work units**
  - G-Trie: (Network, G-Trie Node, Partial Match)
  - ESU: (Network, Partial Match, Possible Extensions*)*

- **Goal: efficiently distribute** work units among processors

- **Initial target: distributed memory** with **message passing** [Ribeiro, Silva & Lopes, Cluster'2010]

- **Constraints:** Tree highly **unbalanced**
  - Pre-determined static allocation is very hard!
  - Requires **dynamic load balancing**

Distributed Snapshot

Receiver-Initiated Strategy

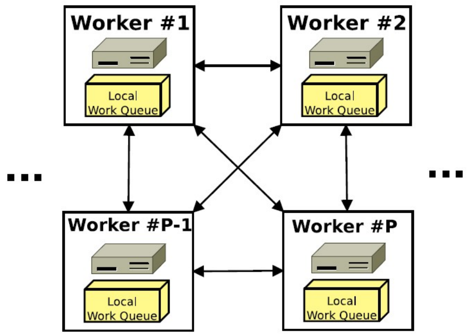
1) While computation not ended

- If work units available

. Process work unit

- Someone asked for work?

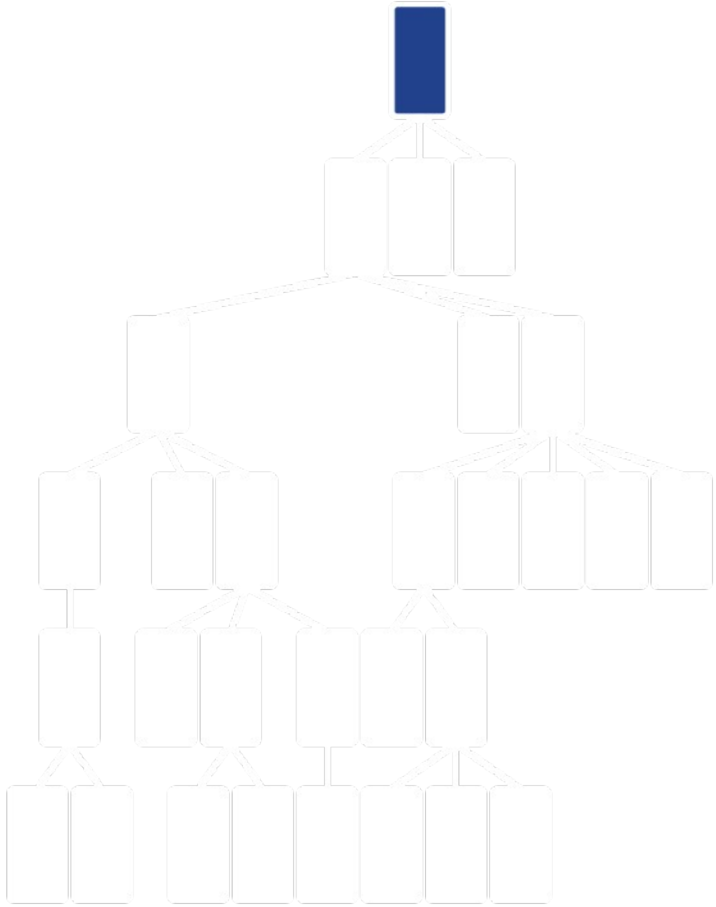
> Stop my computation
> Divide work in 2 similar halves
> Send half to requester
> Return to computation



- Else
  - Request work units from other processor

Running Computation

Example Computation



🟦 G-Trie Node

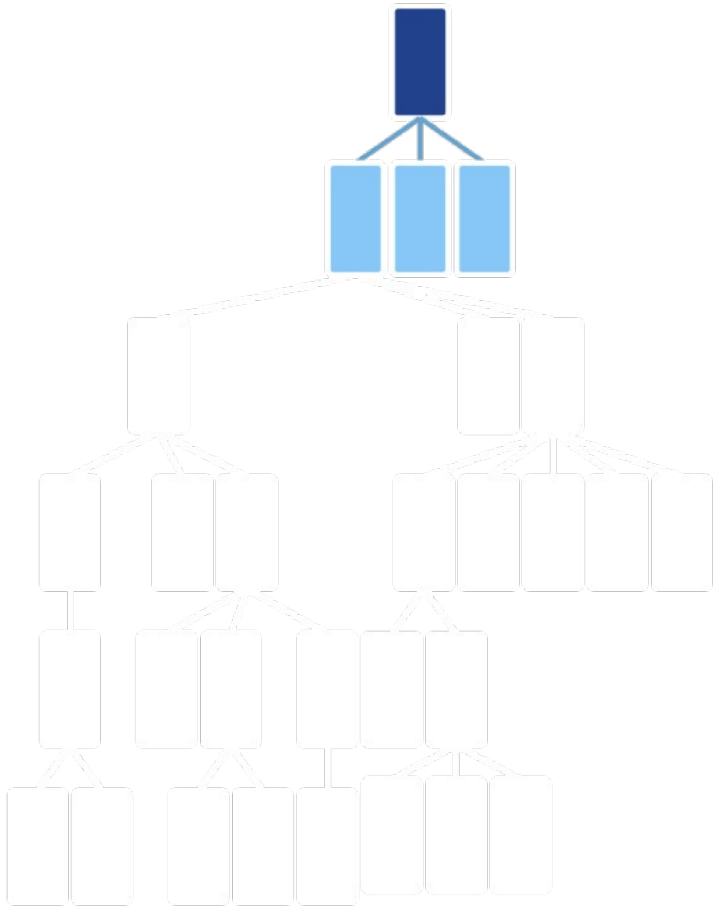
🟦 Graph Vertex

Running Computation

Example Computation

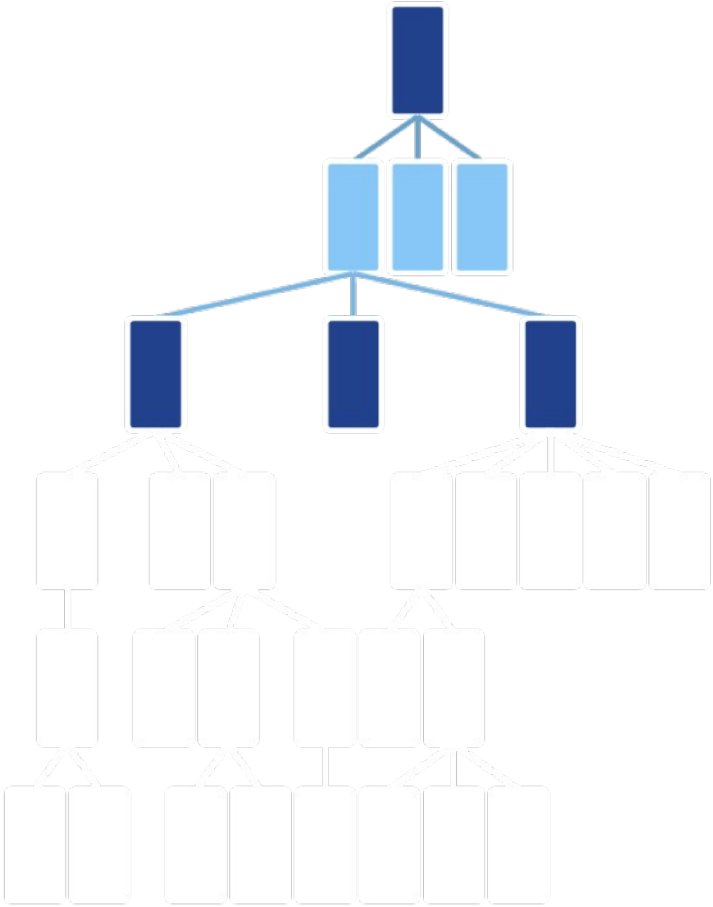
■ G-Trie Node

🟦 Graph Vertex



Running Computation

Example Computation



🟦 G-Trie Node

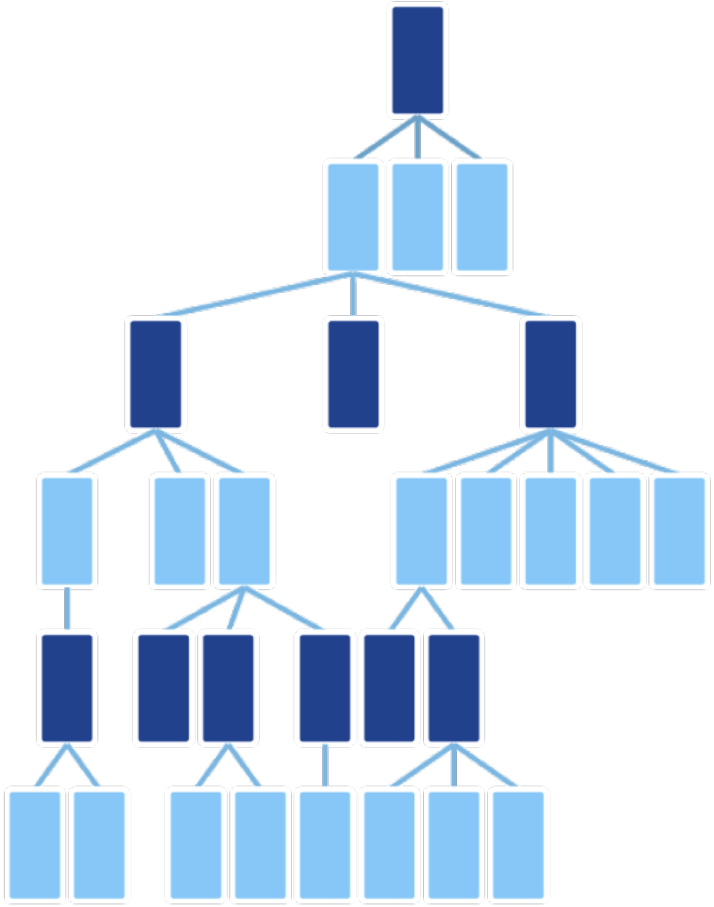
🟦 Graph Vertex

Running Computation

Example Computation

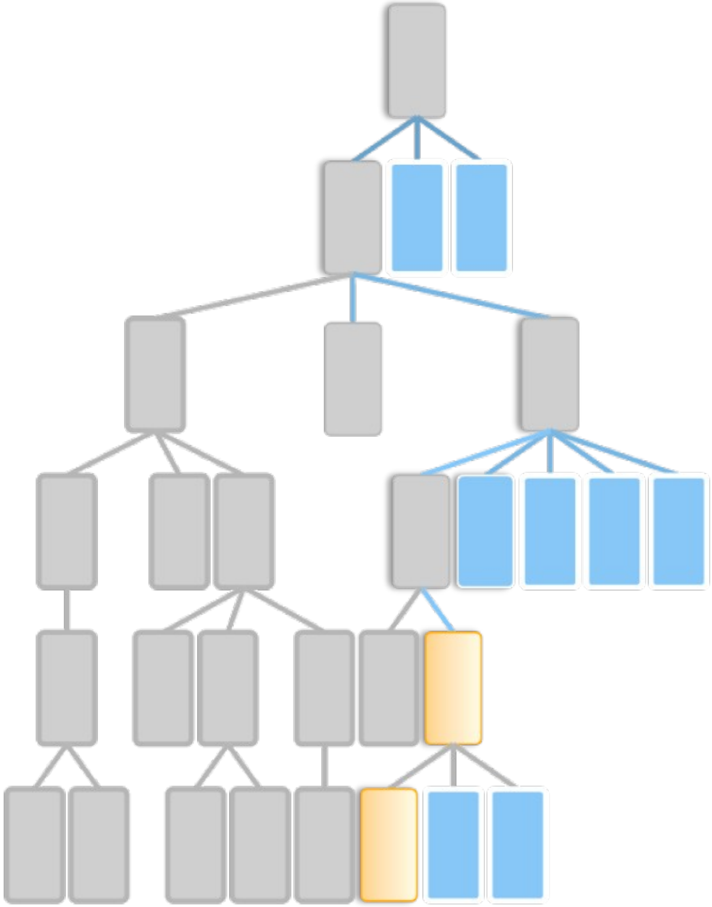
🟦 G-Trie Node

🟦 Graph Vertex



Stopping Computation

Example Computation



🟦 G-Trie Node

🟦 Graph Vertex

🟧 Current Work Unit

⬜ Explored Work Units

STOP

Dividing Computation

- **Goal:** divide work in **two “equal” halves**

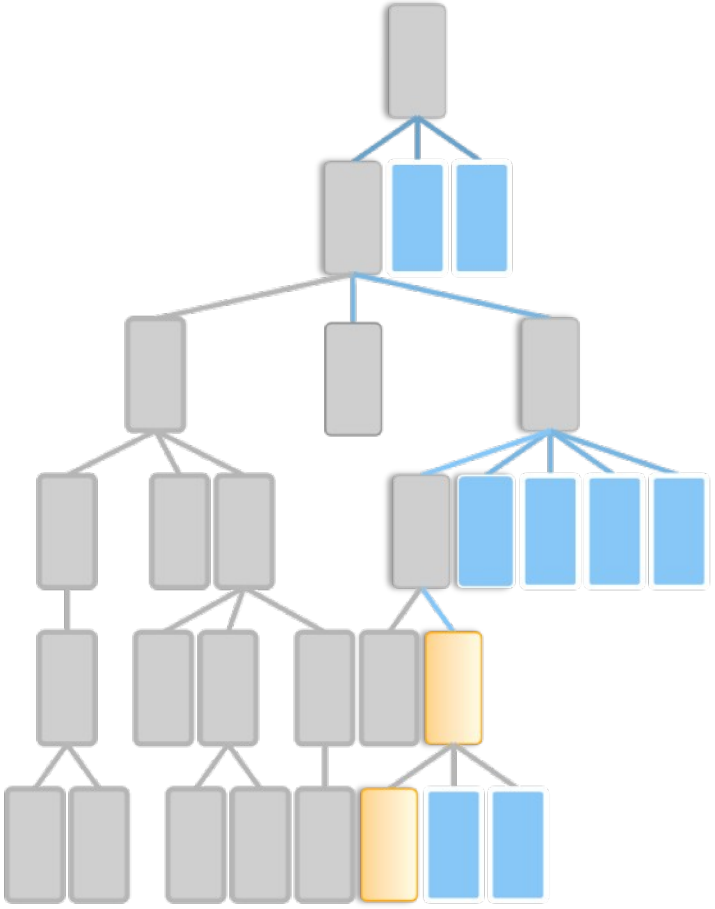
- **We create a compact representation of the search staten *(tree-shaped)***
  - Take advantage of common substructure in work units
  - Efficient methods for: stopping, dividing, resuming

- **We stop dividing when units are too small**
  - Threshold in distance to search tree leaf

- **We do a diagonal split**
  - Round-robin scheme

Dividing Computation

Example Computation



🟦 G-Trie Node

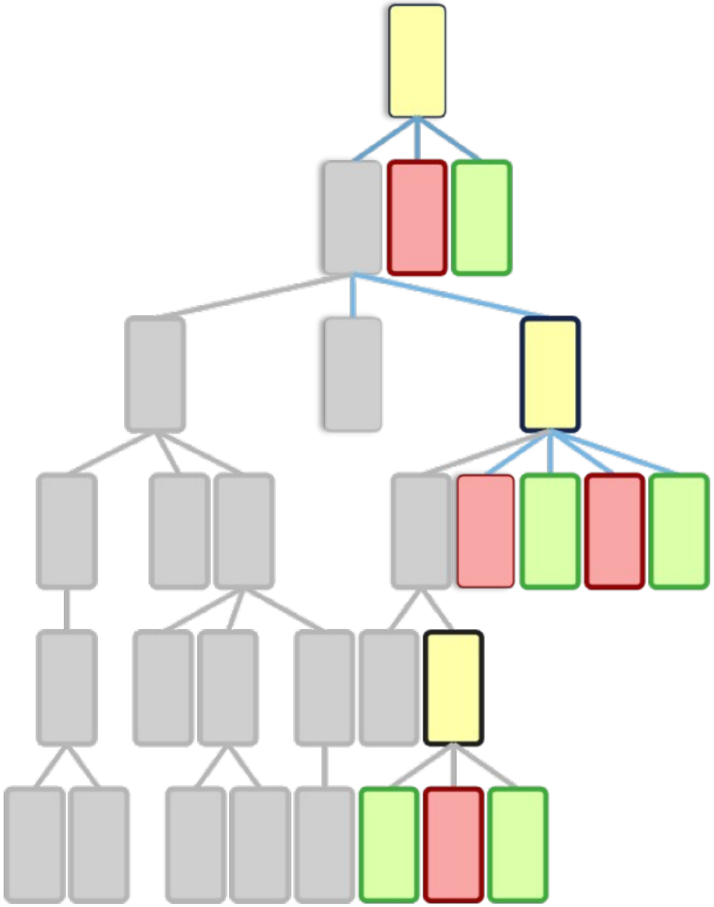
🟦 Graph Vertex

🟧 Current Work Unit

⬜ Explored Work Units

Dividing Computation

Example Computation



🟩 Keep

🟥 Give to requester

🟨 Both

Work Request

- **When we do not have work, which processor should we contact?**
  - No data locality
  - Search trees completely unbalanced

✓ **Ask a random processor!**
  - Random polling ([Sanders 1994])

Some Parallel Results

- **Absolute Speedup** *(distributed snapshots)*

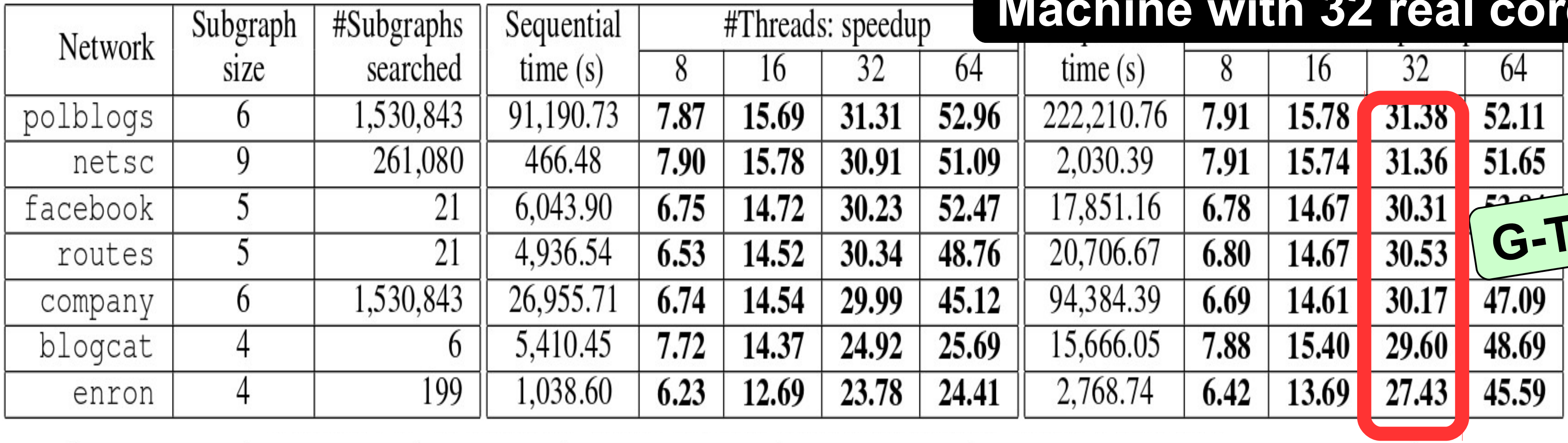
Network	K	#CPUs: Speedup		
		32	64	128
dolphins	10	30.8	59.4	112.7
circuit	11	31.3	61.7	121.2
neural	6	31.4	62.5	122.8
metabolic	6	31.5	62.9	126.0
links	4	30.0	57.1	95.9
coauthors	8	31.4	62.6	123.9
ppi	6	31.4	62.0	122.1
odlis	4	29.7	55.9	90.2
power	9	31.1	61.0	118.8
company	5	31.3	62.8	125.2
foldoc	4	30.9	60.6	116.9
internet	4	31.4	62.9	125.7

Almost linear speedup up to 128 cores!

Some Parallel Results

[Aparício, Ribeiro & Silva, ISPA'2014]

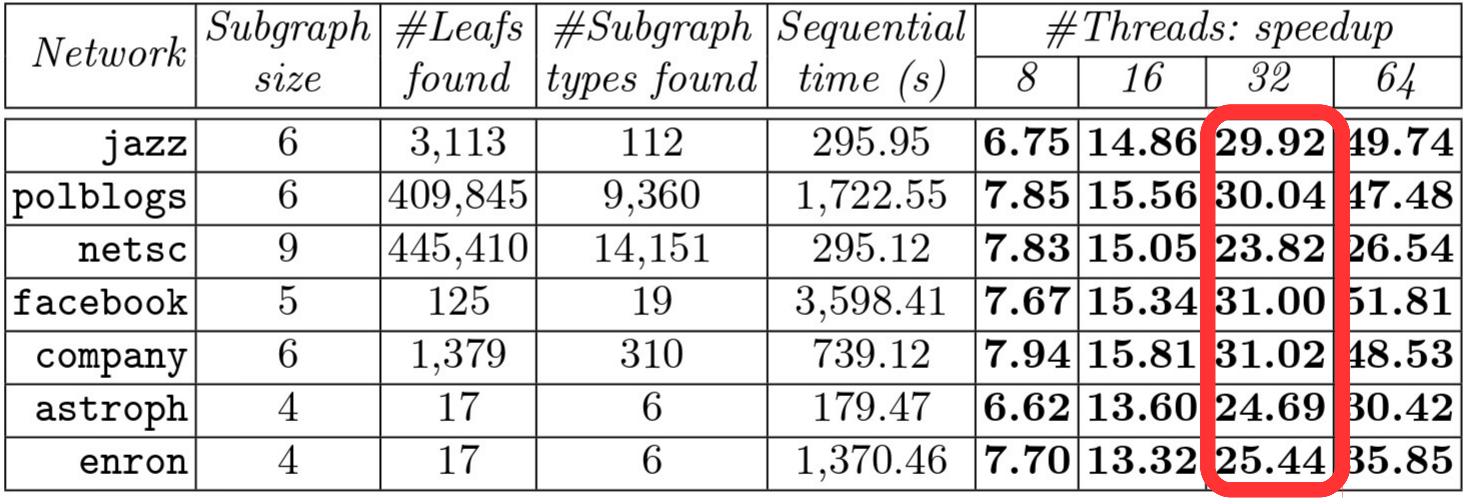
Shared memory implementation with similar results



Network	Subgraph size	#Subgraphs searched	Sequential time (s)	#Threads: speedup								
				8	16	32	64	time (s)	8	16	32	64
polblogs	6	1,530,843	91,190.73	**7.87**	**15.69**	**31.31**	**52.96**	222,210.76	**7.91**	**15.78**	**31.38**	**52.11**
netsc	9	261,080	466.48	**7.90**	**15.78**	**30.91**	**51.09**	2,030.39	**7.91**	**15.74**	**31.36**	**51.65**
facebook	5	21	6,043.90	**6.75**	**14.72**	**30.23**	**52.47**	17,851.16	**6.78**	**14.67**	**30.31**	[illegible]
routes	5	21	4,936.54	**6.53**	**14.52**	**30.34**	**48.76**	20,706.67	**6.80**	**14.67**	**30.53**	[illegible]
company	6	1,530,843	26,955.71	**6.74**	**14.54**	**29.99**	**45.12**	94,384.39	**6.69**	**14.61**	**30.17**	**47.09**
blogcat	4	6	5,410.45	**7.72**	**14.37**	**24.92**	**25.69**	15,666.05	**7.88**	**15.40**	**29.60**	**48.69**
enron	4	199	1,038.60	**6.23**	**12.69**	**23.78**	**24.41**	2,768.74	**6.42**	**13.69**	**27.43**	**45.59**

Machine with 32 real cores

G-Tries



Network	*Subgraph size*	*#Leafs found*	*#Subgraph types found*	*Sequential time (s)*	*#Threads: speedup*			
					8	*16*	*32*	*64*
jazz	6	3,113	112	295.95	**6.75**	**14.86**	**29.92**	**49.74**
polblogs	6	409,845	9,360	1,722.55	**7.85**	**15.56**	**30.04**	**47.48**
netsc	9	445,410	14,151	295.12	**7.83**	**15.05**	**23.82**	**26.54**
facebook	5	125	19	3,598.41	**7.67**	**15.34**	**31.00**	**51.81**
company	6	1,379	310	739.12	**7.94**	**15.81**	**31.02**	**48.53**
astroph	4	17	6	179.47	**6.62**	**13.60**	**24.69**	**30.42**
enron	4	17	6	1,370.46	**7.70**	**13.32**	**25.44**	**35.85**

FaSE

Almost linear speedup up to 32 cores!

Final Improvements

Combining:

G-Trie Complete Sequential Improvement

+

Time Gains of Sampling Approach

+

Scalability of Parallel Approach

↓

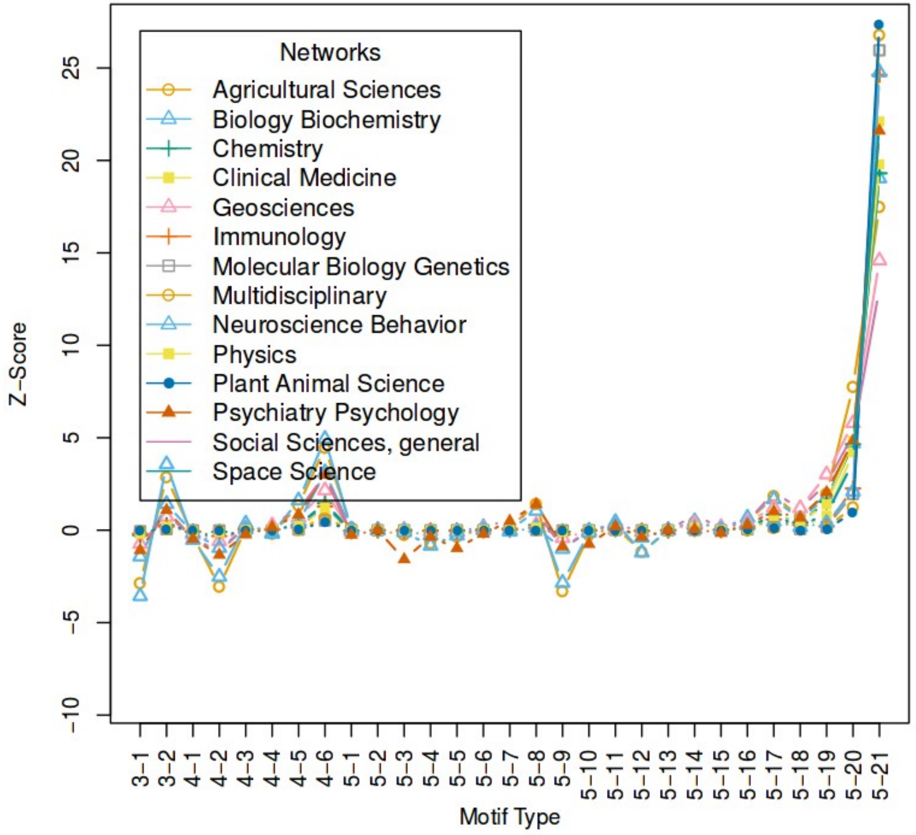
Over 2000x faster than previous state-of-the-art

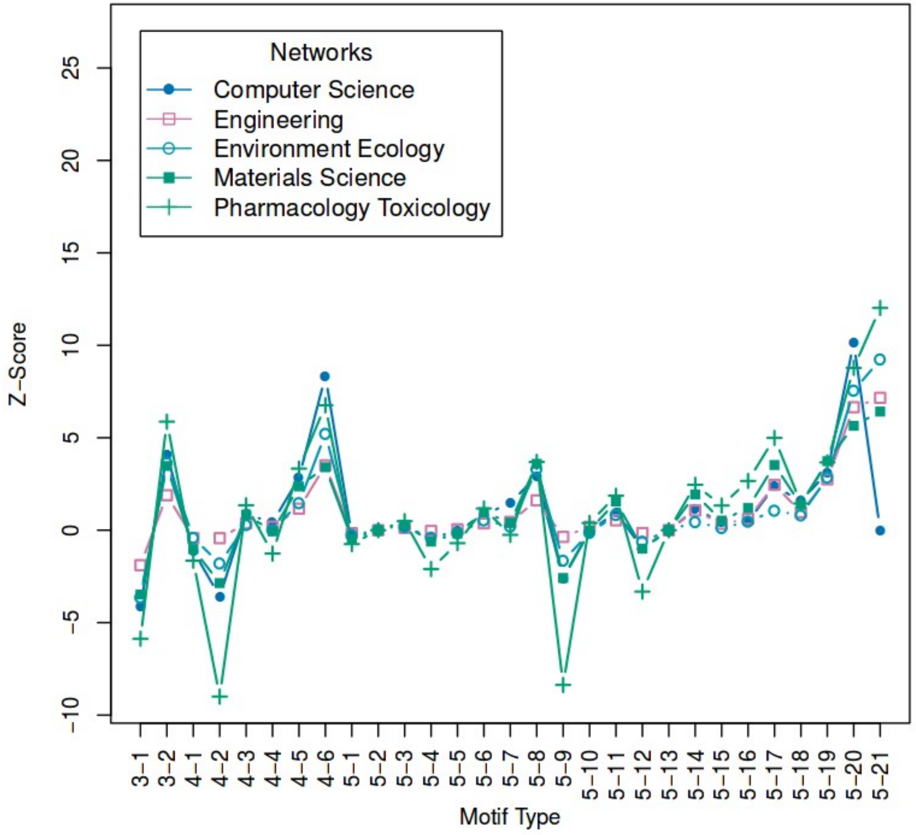
Larger Networks
Larger Subgraphs
New Insight

6) EXAMPLE APPLICATIONS

Co-Authorship Networks

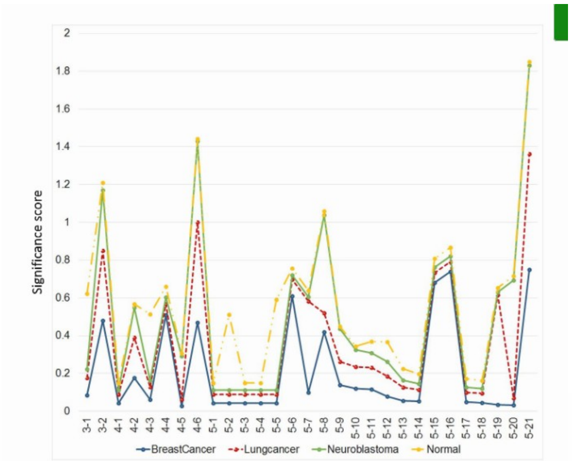
Undirected Network Motifs



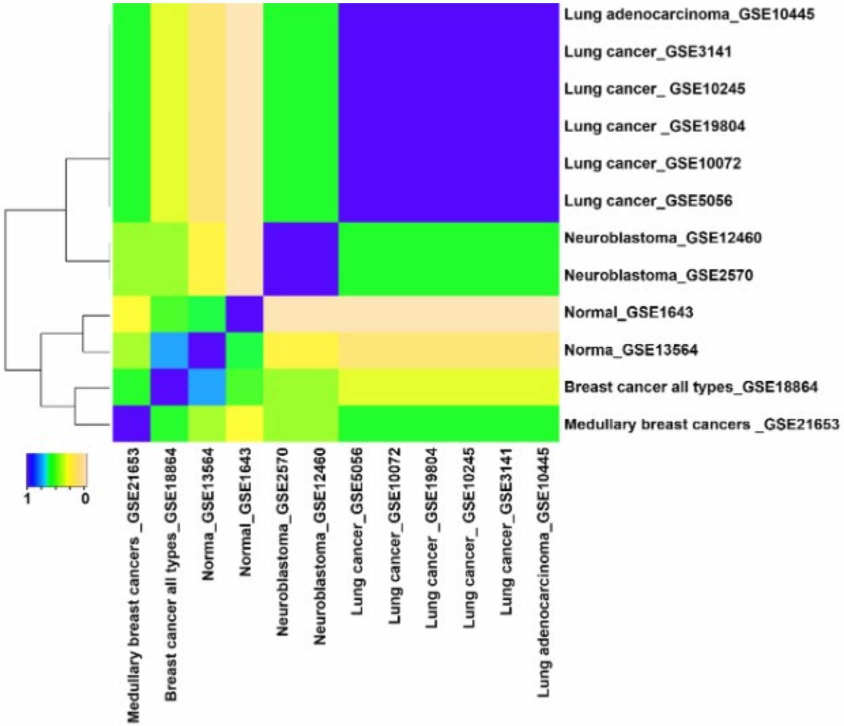


[Choobdar, Ribeiro & Silva, ASONAM'2012]

Gene Co-Expression Networks



Weighted Network Motifs



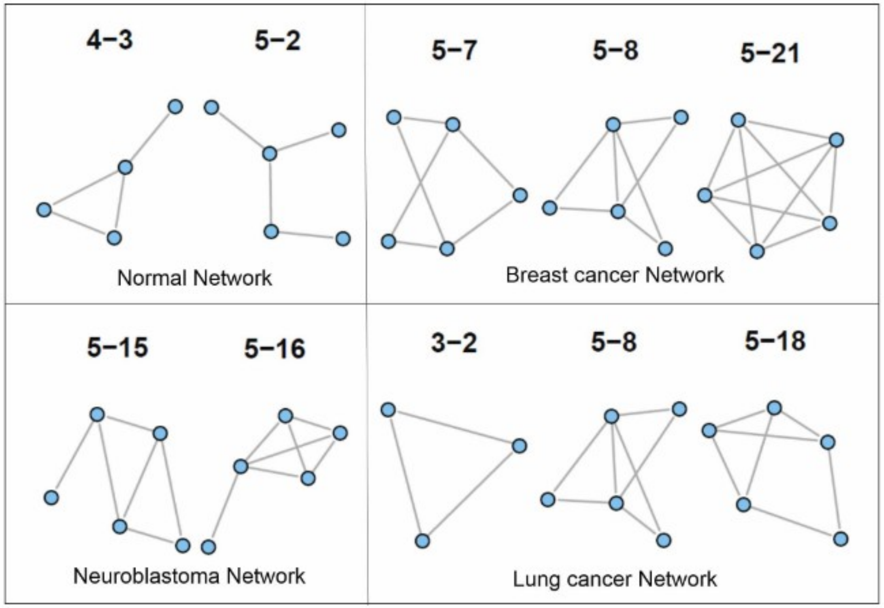
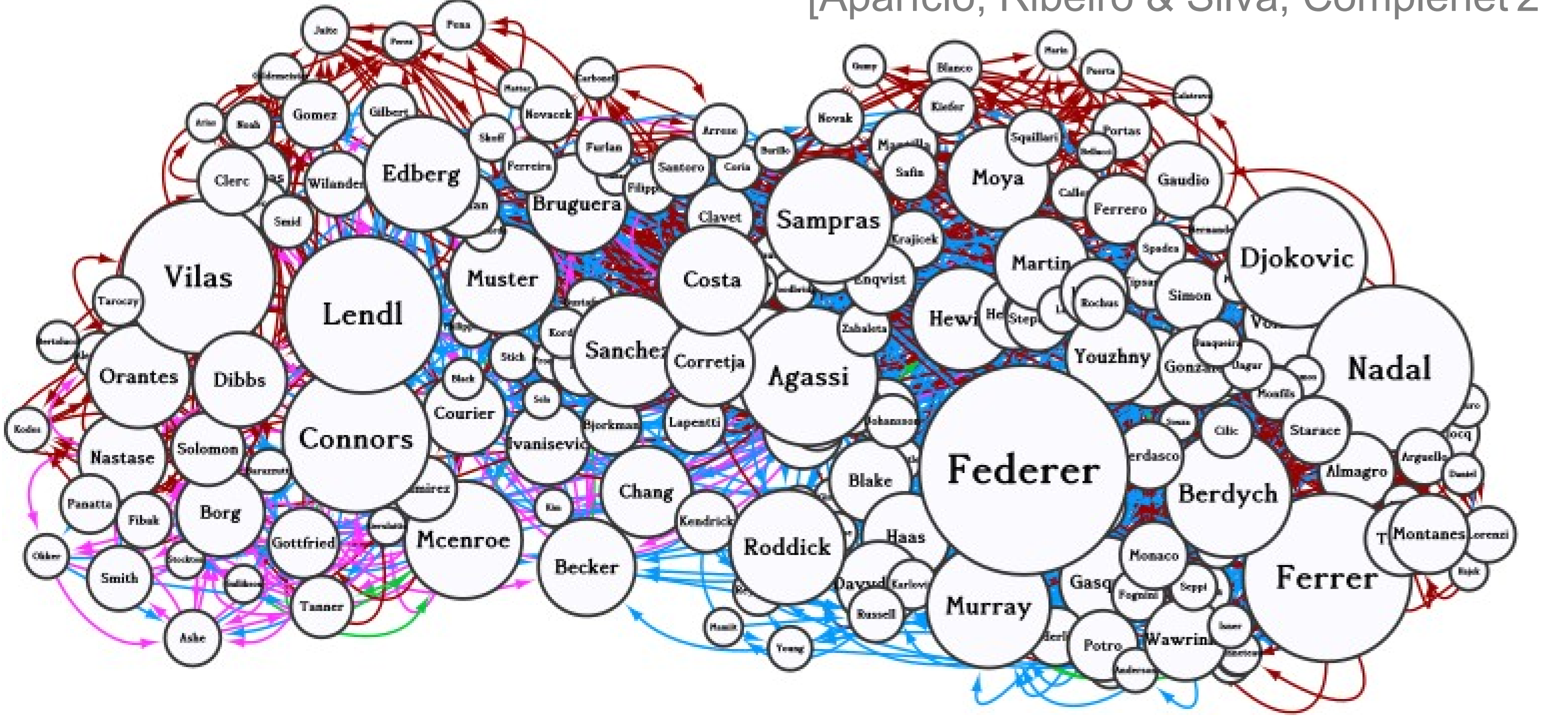


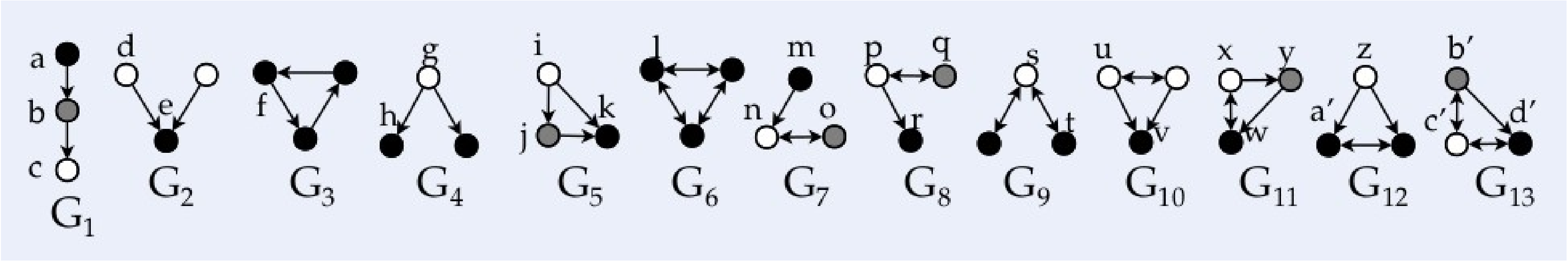
Figure 5: Discriminating subgraphs for each type of networks.

[Choobdar, Ribeiro & Silva, SAC'2015]

Tennis Networks

[Aparício, Ribeiro & Silva, Complenet'2016]

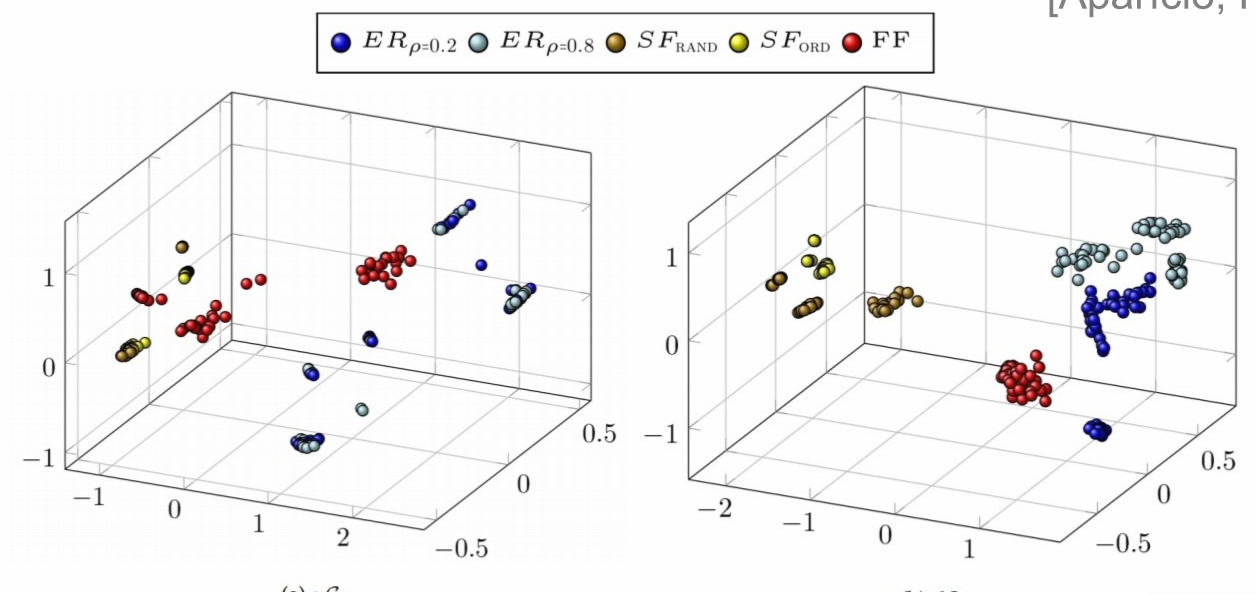


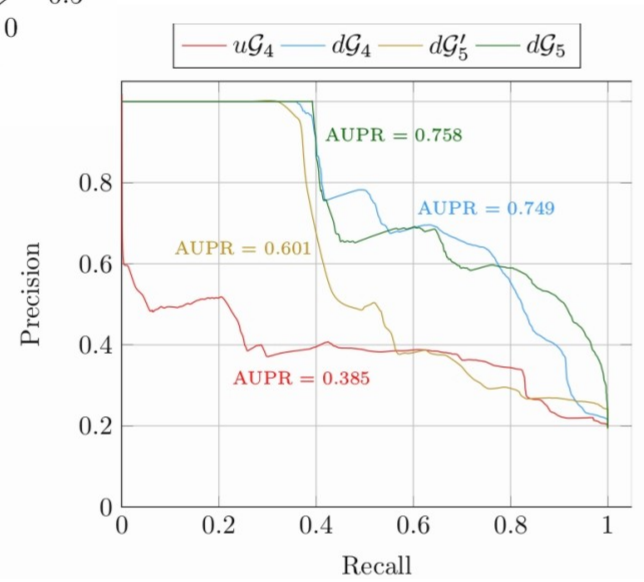


Dominance Patterns based on Directed Graphlets

Classifying and clustering

[Aparício, Ribeiro & Silva, TCBB, 2017]





Directed Graphlets

Classifying and clustering

[Aparício, Ribeiro & Silva, PLoS, 2018]

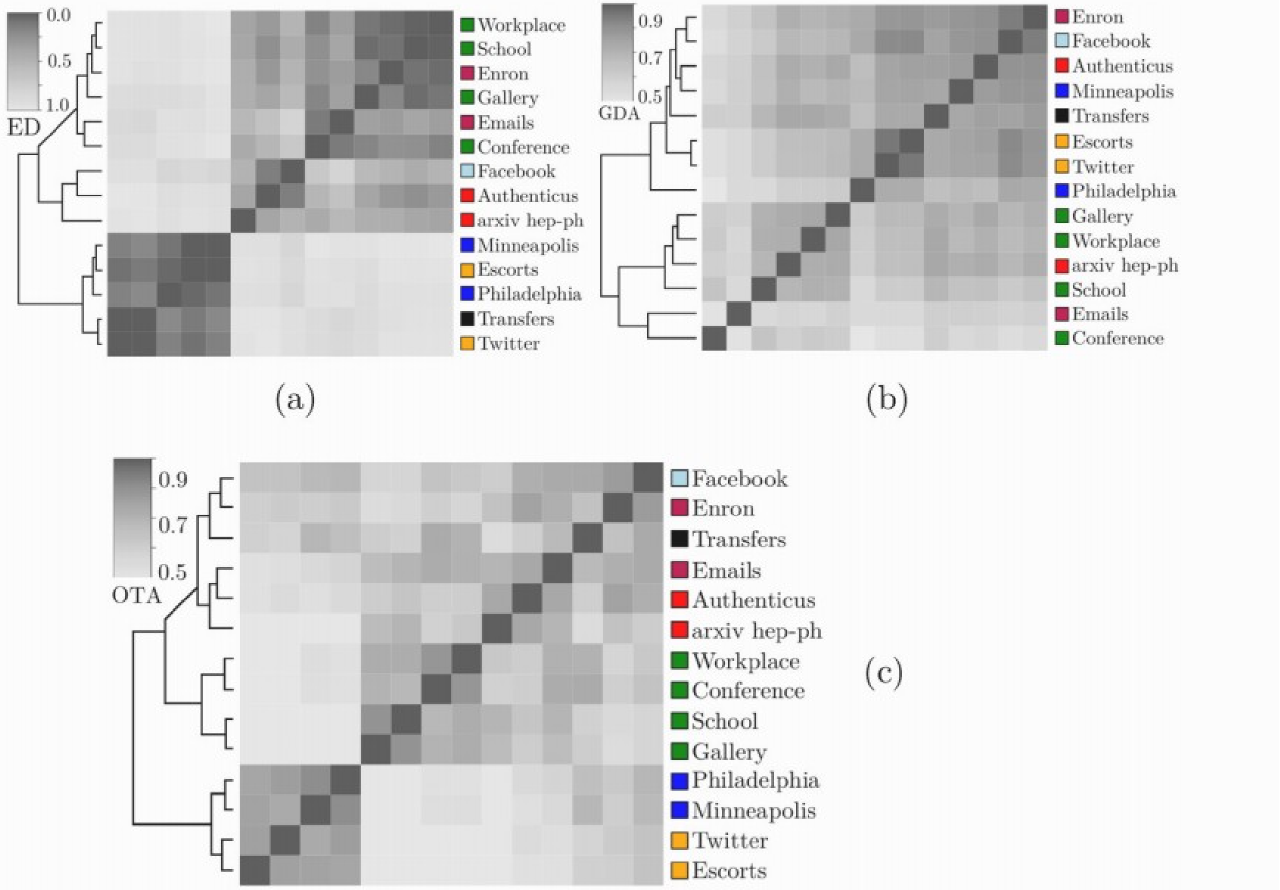
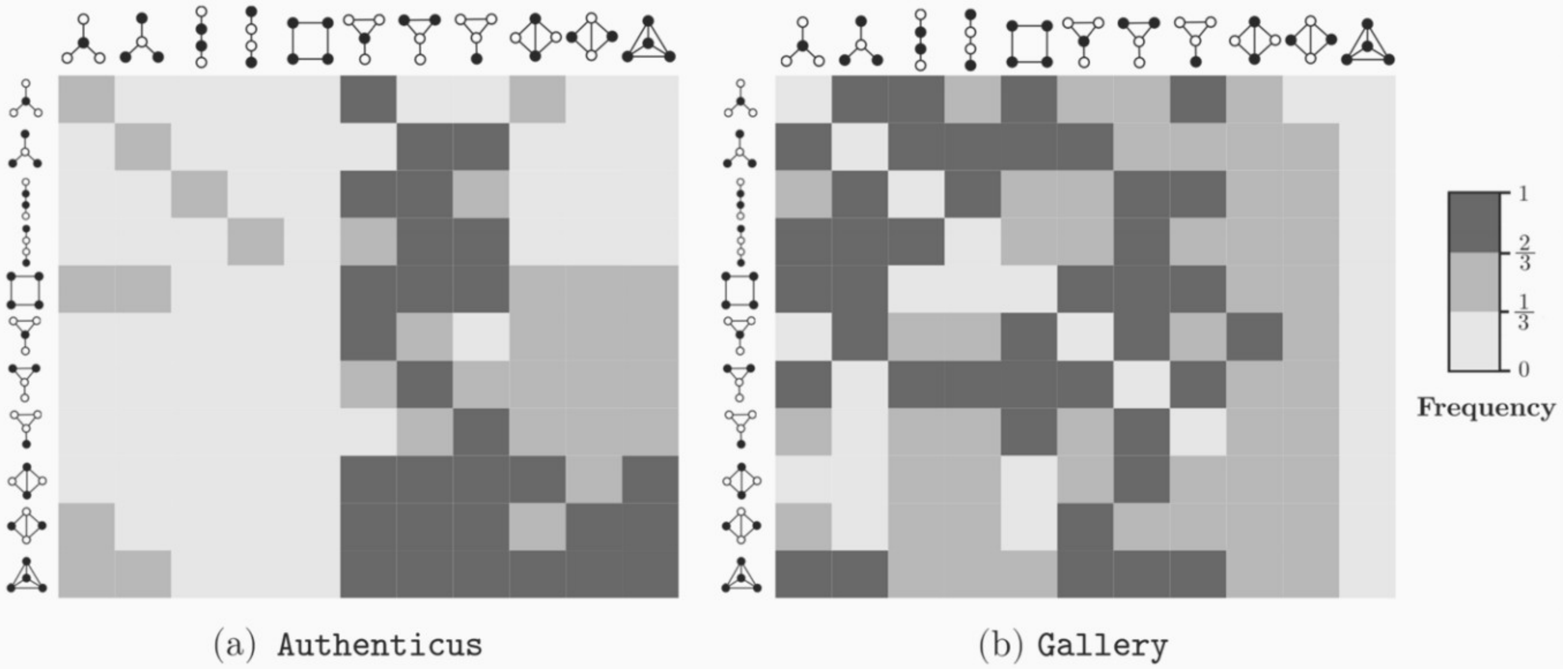


Fig 10. Similarity matrices according to (a) motif-fingerprints' Euclidean distance (*ED*), (b) graphlet-degree-agreement (*GDA*) and (c) orbit-transition-agreement (*OTA*). Clustering is performed using hierarchical clustering with complete linkage.

Graphlet-Orbit Transitions

Classifying and clustering

[Aparício, Ribeiro & Silva, PLoS, 2018]



(a) Authenticus

(b) Gallery

Fig 11. Orbit-transition matrices of (a) a collaboration network and a (b) physical interaction network for al 4-node orbits.

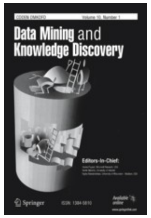
Graphlet-Orbit Transitions

7) RESOURCES

Some publications

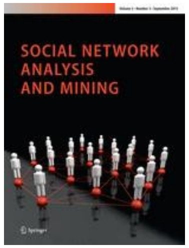
Core complete sequential algorithms

- **Large Scale Graph Representations for Subgraph Census.** NetSciX'2016
- **G-Tries: a data structure for storing and finding subgraphs.** Data Mining and Know. Discovery, 2014.
- **Towards a faster network-centric subgraph census.** ASONAM'2013
- **Querying Subgraph Sets with G-Tries**. DBSocial'2012 *(best paper award)*
- **Strategies for Network Motifs Discovery**. E-Science 2009.



Sampling approach

- **Rand-Fase: Fast Approximate Subgraph Census.** SNAM'2015.
- **Efficient Subgraph Frequency Estimation with G-Tries**. WABI'2010.



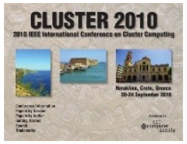
Some publications

Parallel approach

- **Scalable Subgraph Counting using MapReduce.** ACM'SAC 2017
- **Parallel subgraph counting for multicore architectures.** ISPA'2014
- **A Scalable Parallel Approach for Subgraph Census Computation.** MuCoCos'2014
- **Parallel Discovery of Network Motifs**. Journal of Parallel and Distributed Computing. 2012.
- **Efficient Parallel Subgraph Counting using G-Tries**. IEEE Cluster'2010.

Euro-Par 2014: Parallel Processing Workshops

Journal of Parallel and Distributed Computing



CLUSTER 2010

2010 IEEE International Conference on Cluster Computing

Heraklion, Crete, Greece 20-24 September 2010

acm SIGAPP

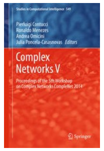
IEEE ISPA

International Symposium on Parallel and Distributed Processing with Applications

26-28 August 2014, Milan

Concept variations and applications

- **Graphlet-orbit Transitions (GoT): A fingerprint for temporal network comparison.** PloS One, 2018
- **Fast streaming small graph canonization.** CompleNet'2018
- **Network motifs detection using random networks with prescribed subgraph frequencies.** CompleNet'2017
- **Extending the applicability of Graphlets to Directed Networks.** T C Biology and Bioinformatics, 2016
- **A subgraph-based ranking system for professional tennis players.** CompleNet'2016
- **Discovering weighted motifs in gene co-expression networks.** ACM-SAC'2015
- **Discovering Colored Network Motifs.** CompleNet'2014
- **Co-authorship network comparison across research fields using motifs.** ASONAM'2012.
- **Motif Mining in Weighted Networks.** Damnet'2012



Complex Networks V

IEEE/ACM TRANSACTIONS ON COMPUTATIONAL BIOLOGY AND BIOINFORMATICS

ICDMW 2012

The 12th IEEE International Conference on Data Mining Workshops

10 December 2012 • Brussels, Belgium

acm SAC'15

30th Symposium On Applied Computing

ASONAM 2012

2012 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining

Istanbul, Turkey 26-29 August 2012

PLOS | ONE

Software

- **Reference sequential implementation** (C++)

  http://www.dcc.fc.up.pt/~pribeiro/gtries/

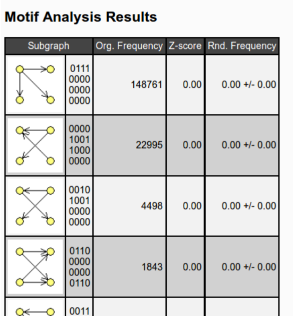
- **Parallel Implementation** (C++ pthreads, multicores)

  http://www.dcc.fc.up.pt/~daparicio/software.html

- **Cytoscape App** (Java, “alpha” version)

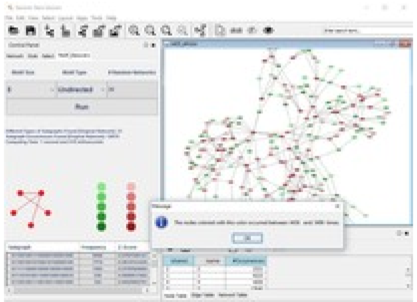
  http://apps.cytoscape.org/apps/motifdiscovery

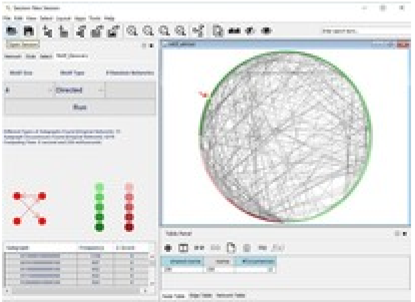
gtrieScanner



Motif Analysis Results

Subgraph		Org. Frequency	Z-score	Rnd. Frequency
	0111 0000 0000 0000	148761	0.00	0.00 +/- 0.00
	0000 1001 1000 0000	22995	0.00	0.00 +/- 0.00
	0010 1001 0000 0000	4498	0.00	0.00 +/- 0.00
	0110 0000 0000 0110	1843	0.00	0.00 +/- 0.00
	0011			





Network Science Group

Pedro Ribeiro

Fernando Silva

Phd Students

Jorge Silva
Expertise Finding
Bibliographic Nets

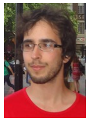
Vanessa Silva
Time Series &
Complex Networks

Alberto Barbosa
Sports
Analytics

Ahmad Naser eddin
Entity Resolution
Graph Databases

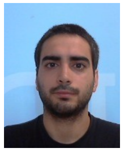
Sarvenaz Choobdar
(PhD *alumnus*)

Miguel Araújo
(PhD *alumnus*)


David Aparício
(PhD *alumnus*)

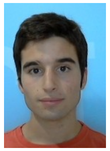
Msc Students

2018/2019


Luciano Grácio

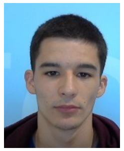
Miguel Martins

André Meira


Miguel Amaral

2019/2020

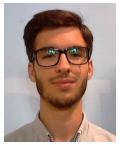
Henrique Branquinho


Bruno Casteleiro

Francisco Bento

Bsc Students

2018/2019


Ricardo
Pereira

Gonçalo
Paredes

2019/2020

Rúben
Dhanaraju

Complex Networks Group from DCC/FCUP

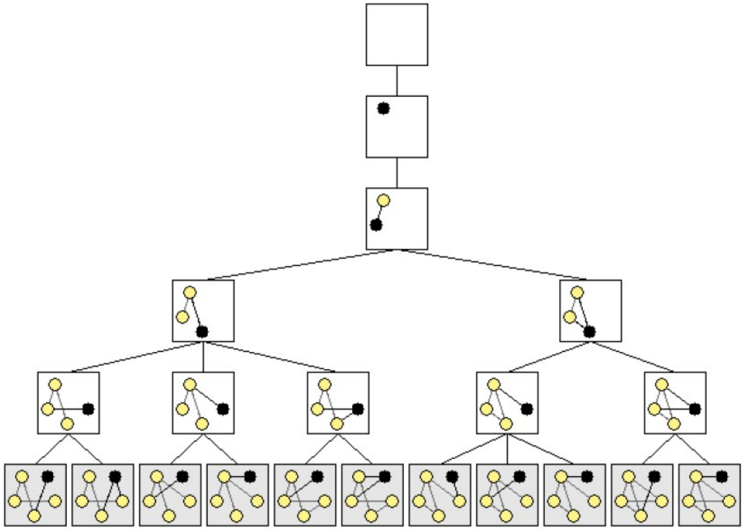
Porto, Portugal

GitHub

Subgraphs as Fundamental Ingredients of Complex networks

Pedro Ribeiro

Thank you for your attention!



Contacts:

Pedro Ribeiro
pribeiro@dcc.fc.up.pt
http://www.dcc.fc.up.pt/~pribeiro/