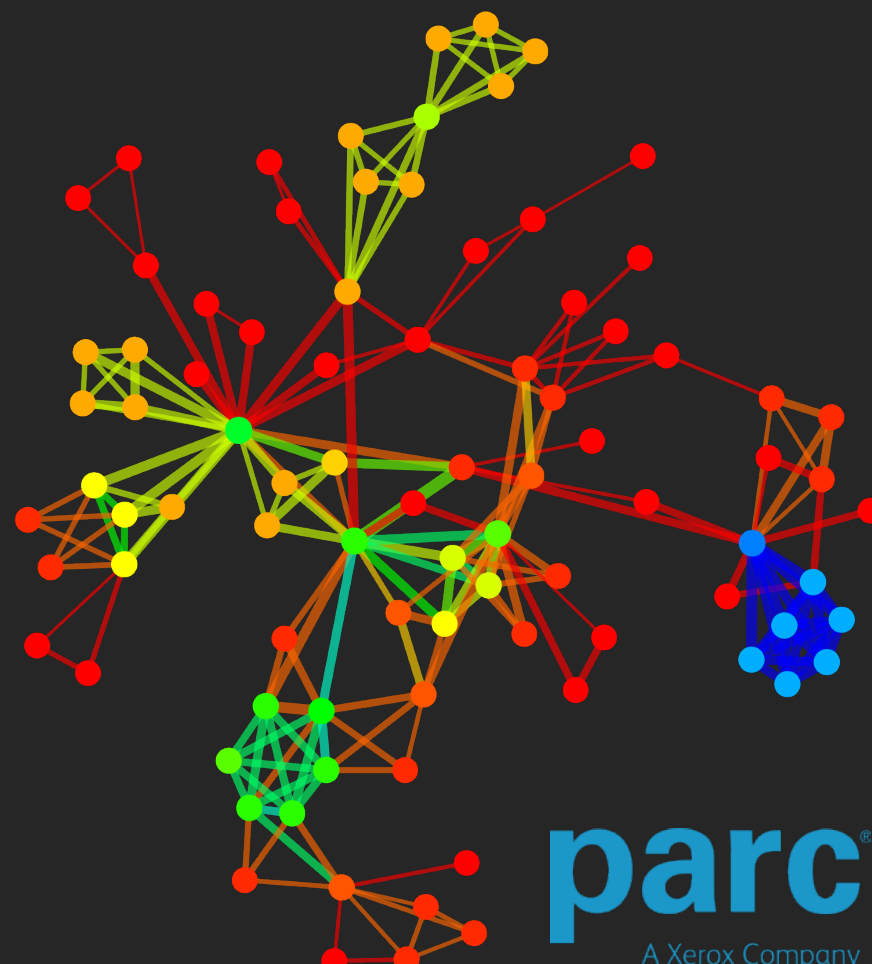
Leveraging Multiple GPUs and CPUs for Graphlet Counting in Large Networks

Ryan A. Rossi

Member of Research Staff
Palo Alto Research Center (PARC)



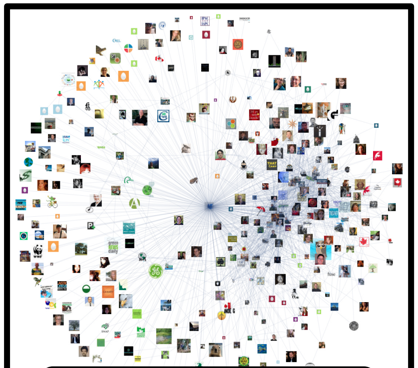
Joint work with:

Rong Zhou
PARC

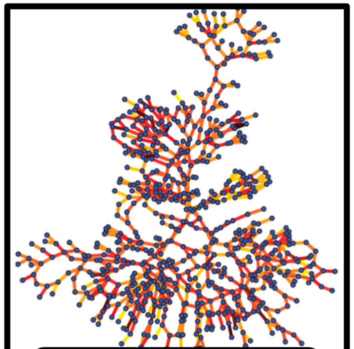
xerox

parc
A Xerox Company

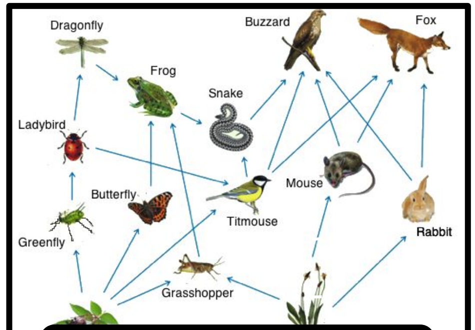
Graphs – rich and powerful data representation



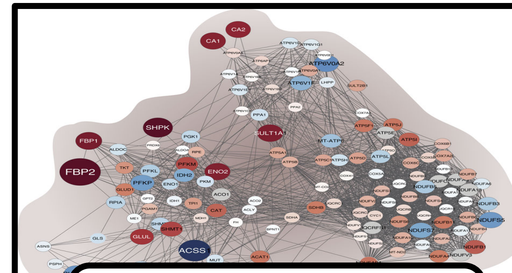
Social network



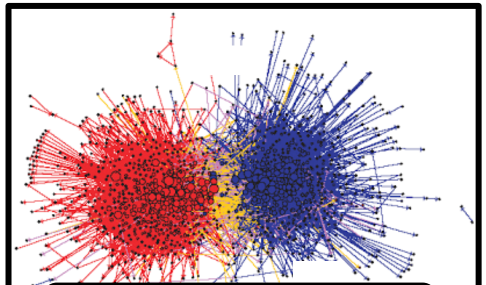
Power grid



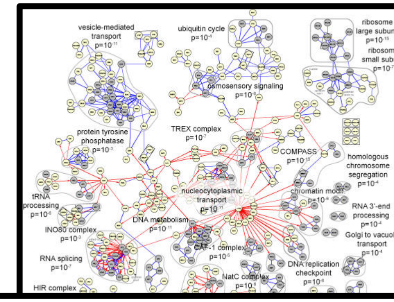
Food Web [2007]



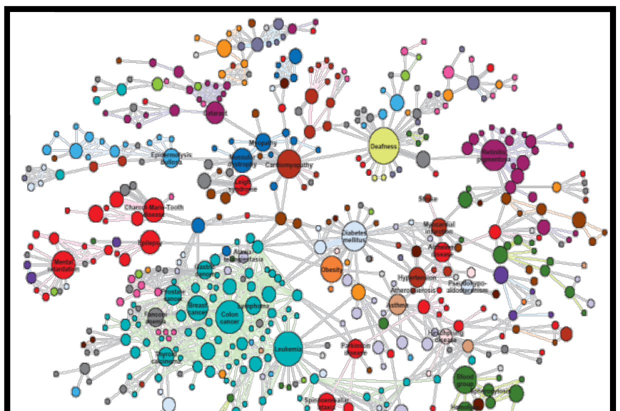
Protein Interactions [breast cancer]



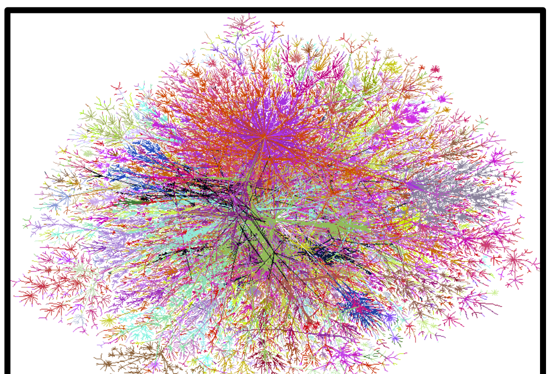
Political blogs



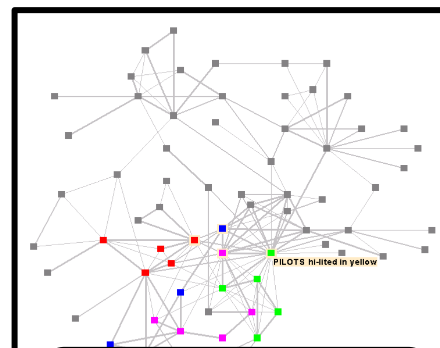
Gene Regulatory Network [Decourty 2008]



Human Disease Network [Barabasi 2007]



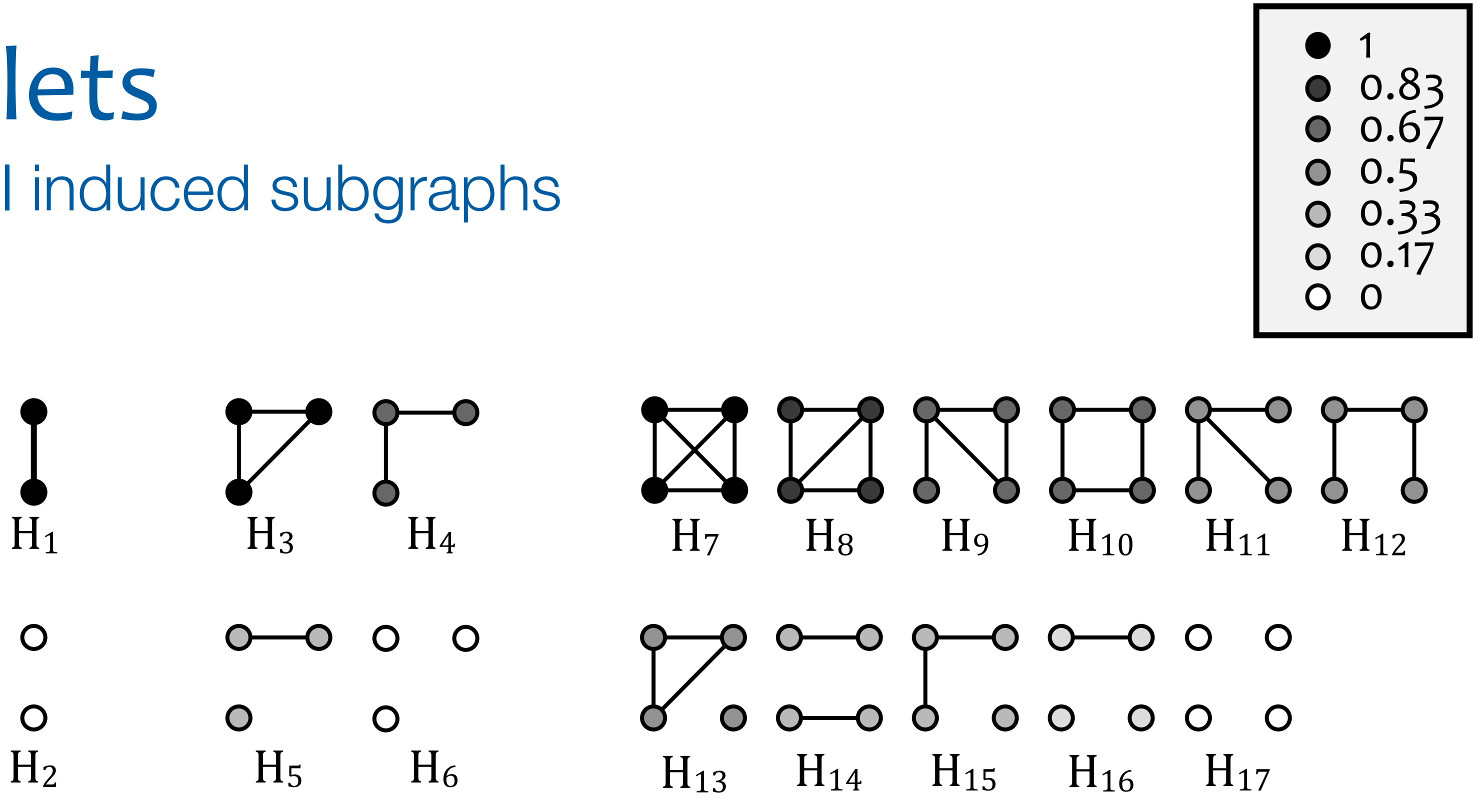
Internet (AS) [2005]



Terrorist Network [Krebs 2002]

Graphlets

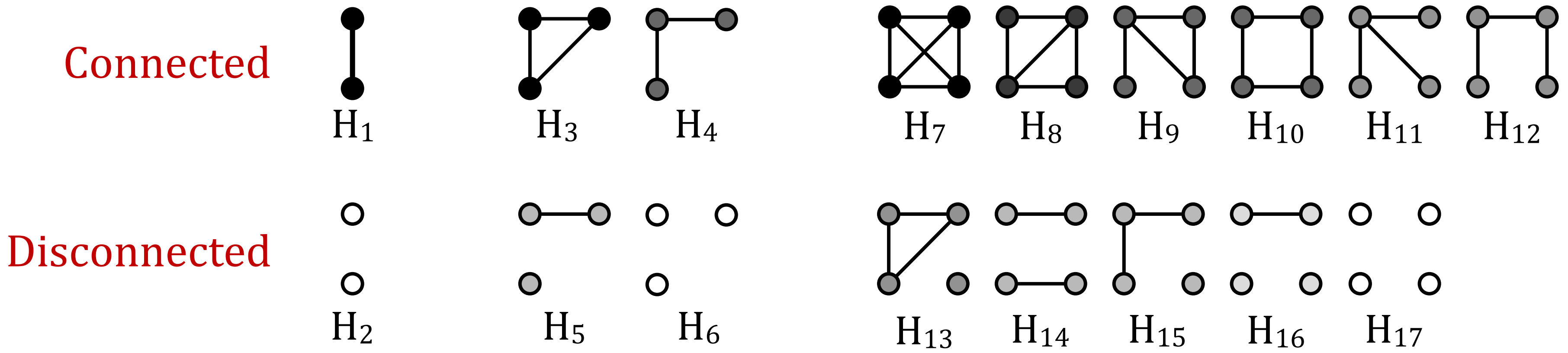
Small induced subgraphs



Network Motifs: Simple Building Blocks of Complex Networks [Milo et. al – Science 2002]
The Structure and Function of Complex Networks [Newman – Siam Review 2003]

Graphlets

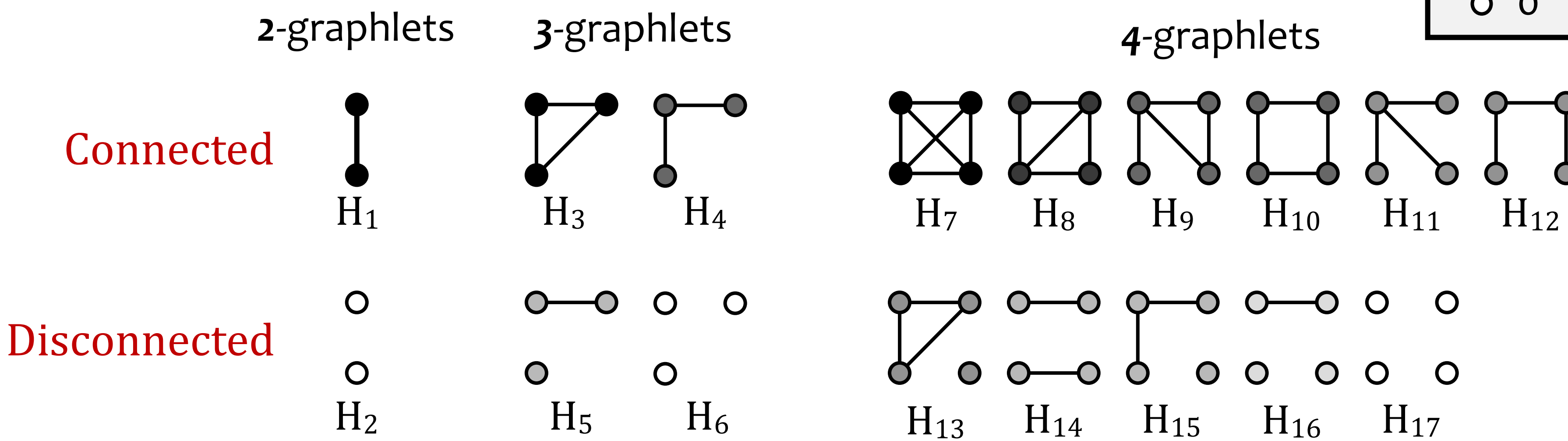
Small induced subgraphs



Network Motifs: Simple Building Blocks of Complex Networks [Milo et. al – Science 2002]
The Structure and Function of Complex Networks [Newman – Siam Review 2003]

Graphlets

Small induced subgraphs

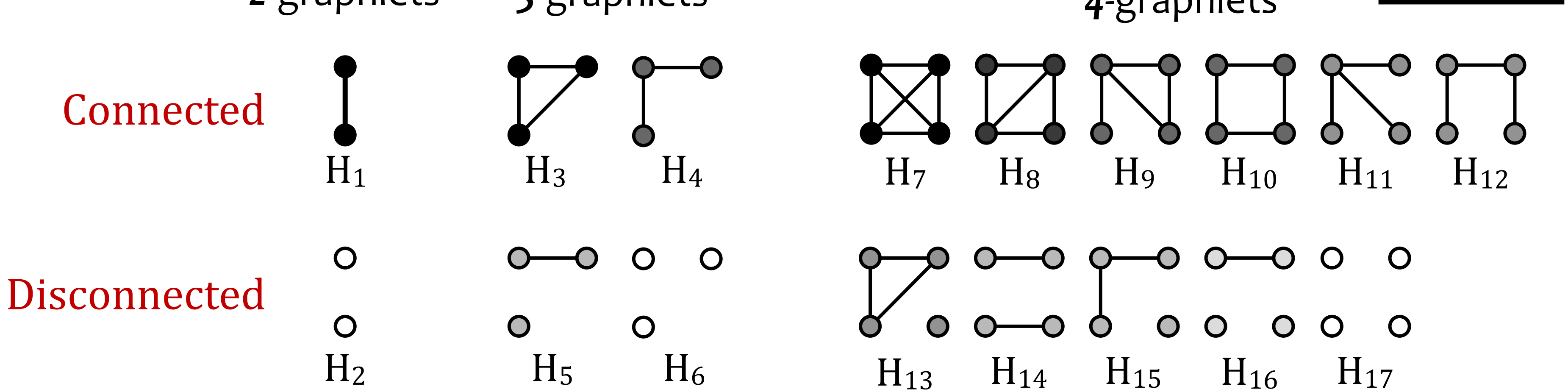


k-graphlets = family of graphlets of size k

Network Motifs: Simple Building Blocks of Complex Networks [Milo et. al – Science 2002]
The Structure and Function of Complex Networks [Newman – Siam Review 2003]

Graphlets

Small induced subgraphs



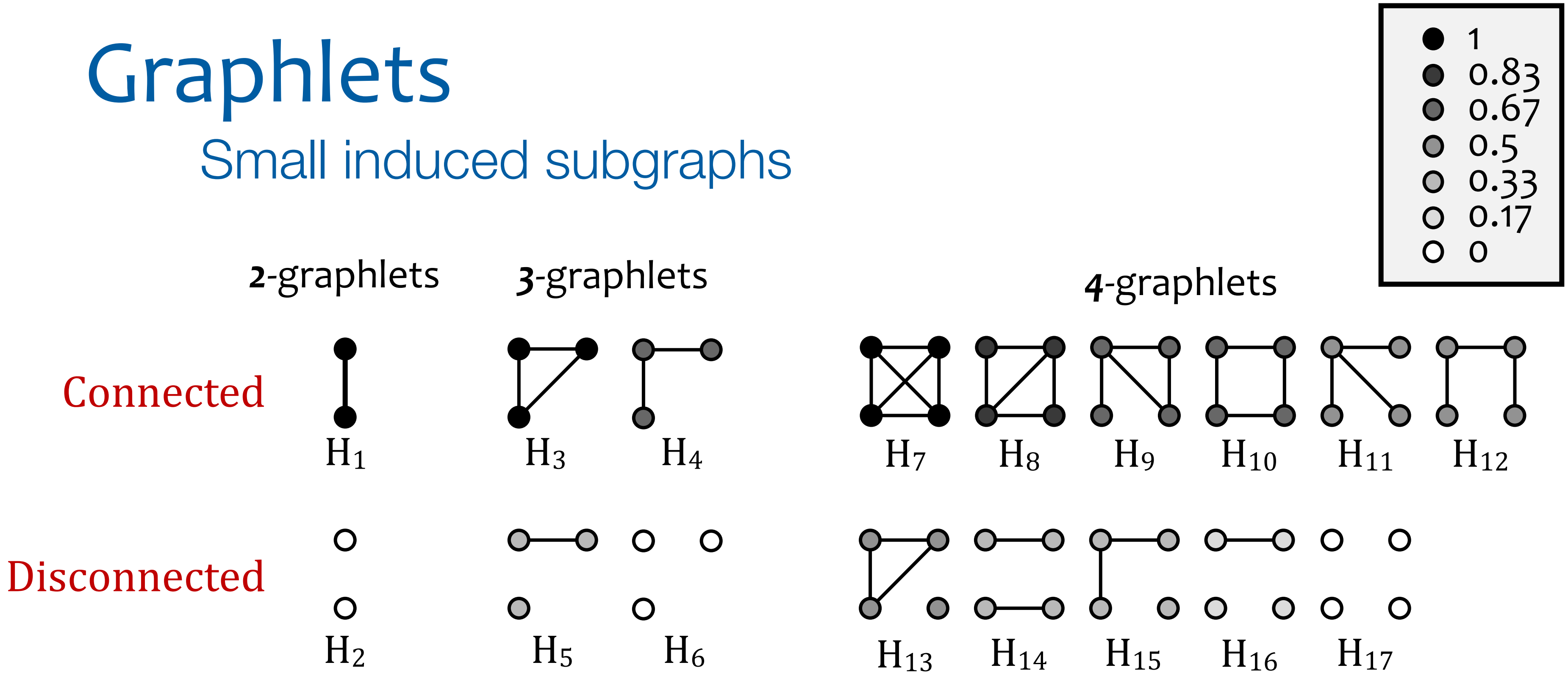
k-graphlets = family of graphlets of size k

motifs = frequently occurring subgraphs

Network Motifs: Simple Building Blocks of Complex Networks [Milo et. al – Science 2002]
The Structure and Function of Complex Networks [Newman – Siam Review 2003]

Graphlets

Small induced subgraphs



k-graphlets = family of graphlets of size k

motifs = frequently occurring subgraphs

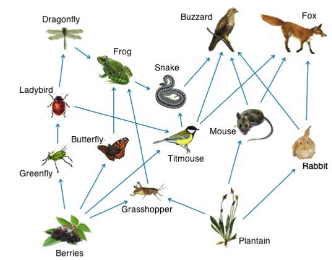
Applied to food web, genetic, neural, web, and other networks

Found distinct graphlets in each case

Network Motifs: Simple Building Blocks of Complex Networks [Milo et. al – Science 2002]

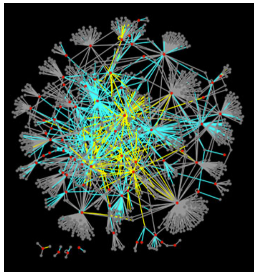
The Structure and Function of Complex Networks [Newman – Siam Review 2003]

Applications of Graphlets



- Biological Networks
  - network alignment, protein function prediction

    [Pržulj 2007][Milenković-Pržulj 2008] [Hulovatyy-Solava-Milenković 2014]
    [Shervashidze et al. 2009][Vishwanathan et al. 2010]

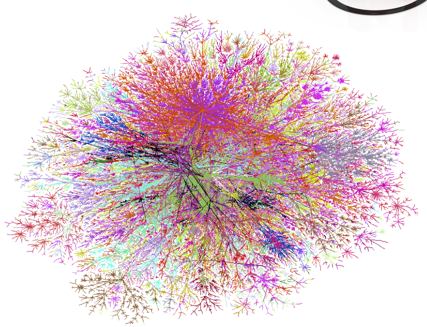


- Social Networks
  - Triad analysis, role discovery, community detection

    [Granovetter 1983][Holland-Leinhardt 1976][Rossi-Ahmed 2015]
    [Ahmed et al. 2015][Xie-Kelley-Szymanski 2013]

- Internet AS [Feldman et al. 2008]
- Spam Detection

  [Becchetti et al. 2008][Ahmed et al. 2016]



Useful for various machine learning tasks

e.g., Anomaly detection, Role Discovery, Relational Learning, Clustering etc.

Useful for a variety of ML tasks

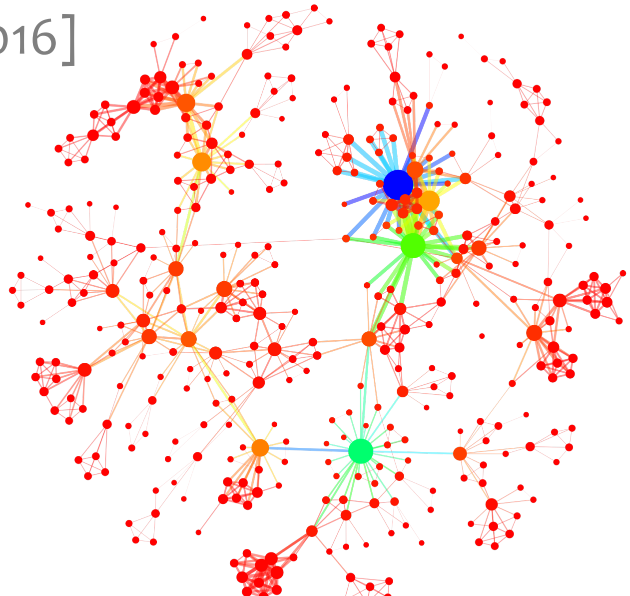
- **Graph-based anomaly detection**
  - Unusual/malicious behavior detection
  - Emerging event and threat identification, …
- **Graph-based semi-supervised learning, classification, …**
- **Link prediction and relationship strength estimation**
- **Graph similarity queries**
  - Find similar nodes, edges, or graphs
- **Subgraph detection and matching**

Applications: Higher-order network analysis *and* modeling

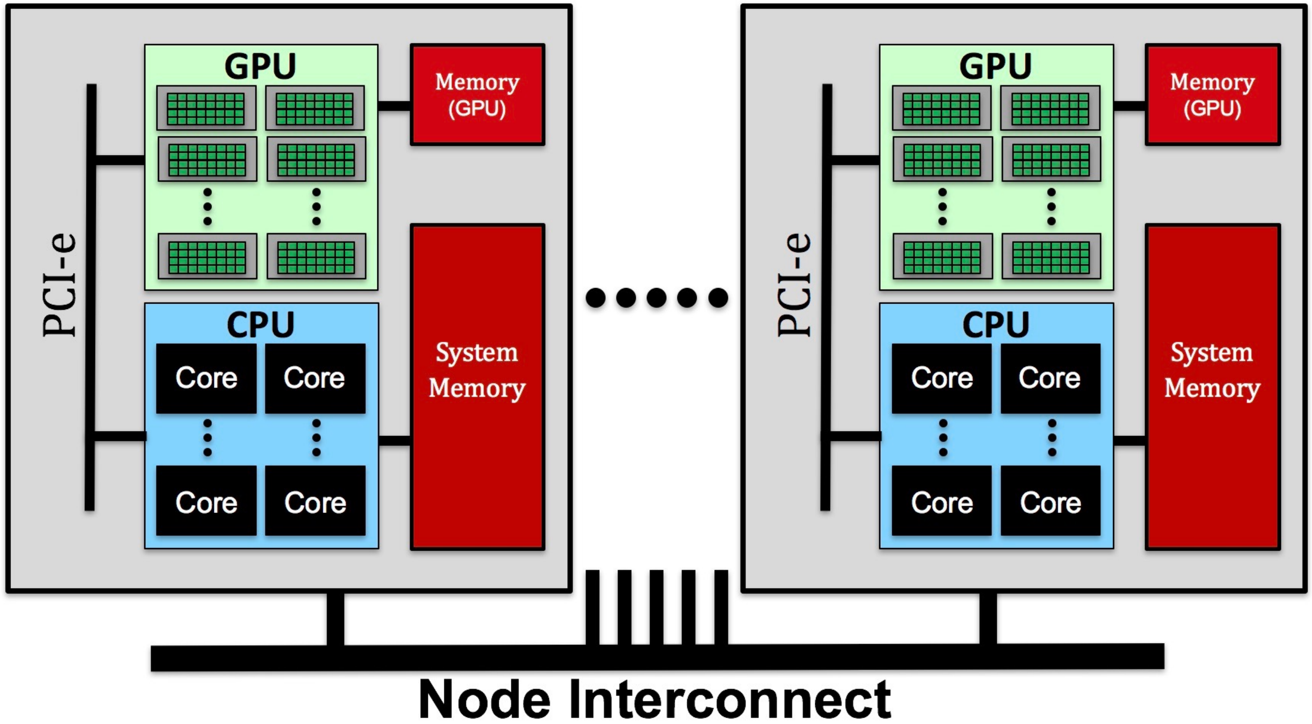
Higher-order network structures

- Visualization – "spotting anomalies" [Ahmed et al. ICDM 2014]
- Finding large cliques, stars, and other larger network structures [Ahmed et al. KAIS 2015]
- Spectral clustering [Jure et al. Science 2016]
- Role discovery [Ahmed et al. 2016]

…

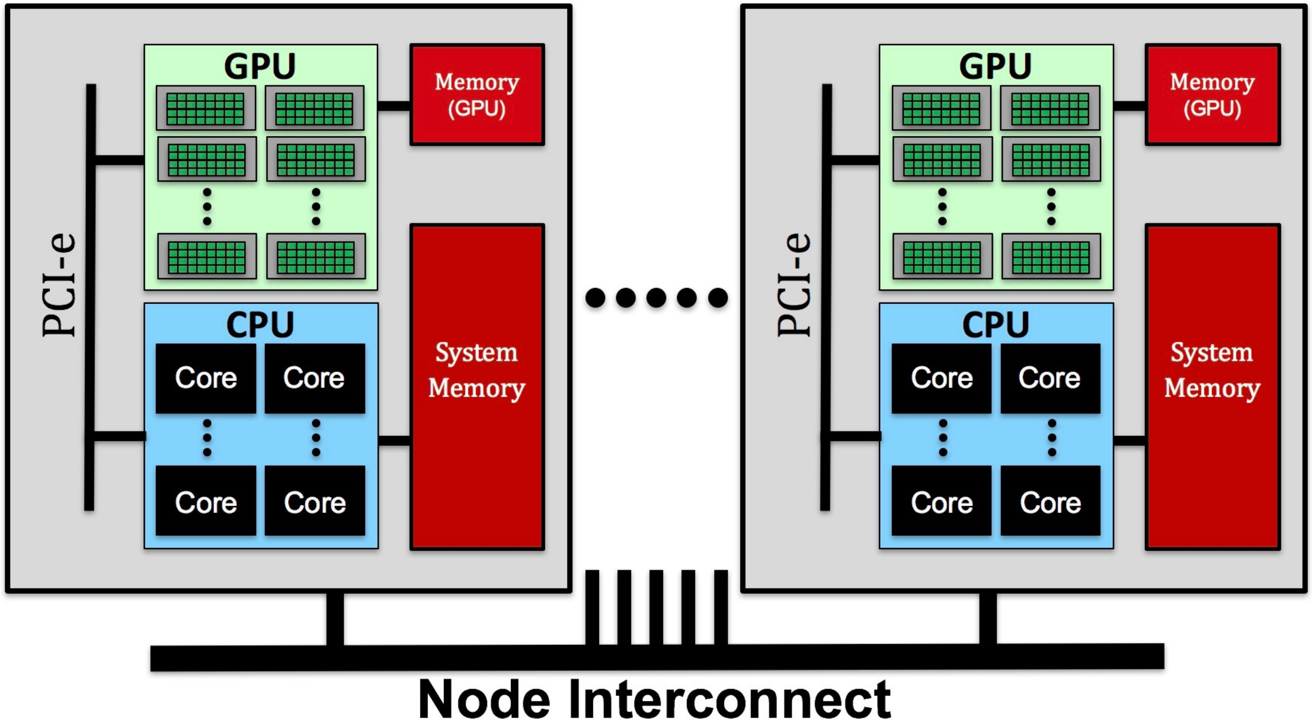


How CPU/GPUs compare



CPU	GPU
Large memory	Memory is very limited
Few fast/powerful processing units	Thousands of smaller processing units
Handles unbalanced jobs better	Performs best with "balanced" workloads
Optimized for general computations	Optimized for simple repetitive calculations at a very fast rate.

How CPU/GPUs compare



CPU	GPU
Large memory	Memory is very limited
Few fast/pow	rocessing units
Handles unb	anced" workloads
Optimized for general computations	Optimized for simple repetitive calculations at a very fast rate.

Combine advantages of both

Problem: global graphlet counting
(macro-level)

INPUT: a *large* graph **G**=(V,E), set of graphlets $\mathcal{H}$

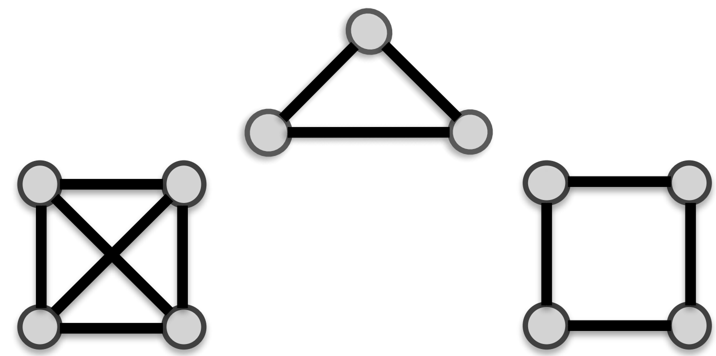
PROBLEM: Find the number of embeddings (appearances) of each graphlet $H_k \in \mathcal{H}$ in **G**

Problem: global graphlet counting
(macro-level)

INPUT: a *large* graph **G**=(V,E), set of graphlets $\mathcal{H}$

PROBLEM: Find the number of embeddings (appearances) of each graphlet $H_k \in \mathcal{H}$ in **G**

Given an input graph G
- How many triangles in G?
- How many cliques of size 4-nodes in G?
- How many cycles of size 4-nodes in G?

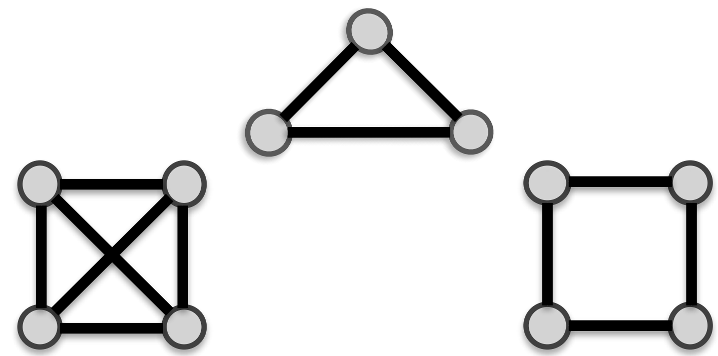


Problem: global graphlet counting
(macro-level)

INPUT: a *large* graph **G**=(V,E), set of graphlets $\mathcal{H}$

PROBLEM: Find the number of embeddings (appearances) of each graphlet $H_k \in \mathcal{H}$ in **G**

Given an input graph G
- How many triangles in G?
- How many cliques of size 4-nodes in G?
- How many cycles of size 4-nodes in G?



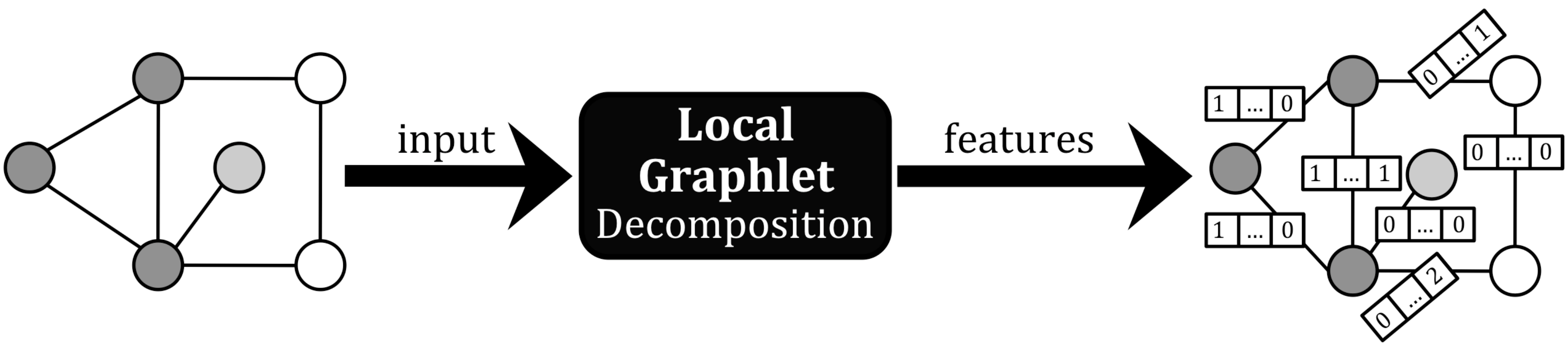
→ ***Many applications require counting** all k-vertex graphlets*

→ ***Recent research work***
- *Exact/approximation of global counts* [Rahman et al. TKDE14] [Jha et al. WWW15]
- *Scalable for massive graphs (billions of nodes/edges)*] [Ahmed et al. ICDM15,KAIS16]

Problem: local graphlet counting
(micro-level)

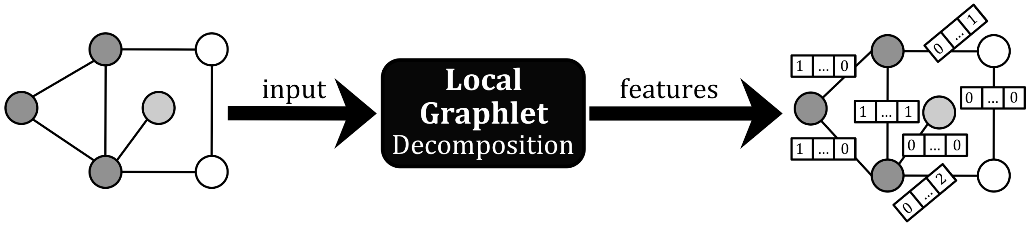
INPUT: a *large* graph G=(V,E), set of graphlets $\mathcal{H}$

PROBLEM: Find the number of occurrences that edge i is contained within H_k, for all $k = 1, \dots, |\mathcal{H}|$



Role discovery, Relational Learning, Multi-label Classification

Current work



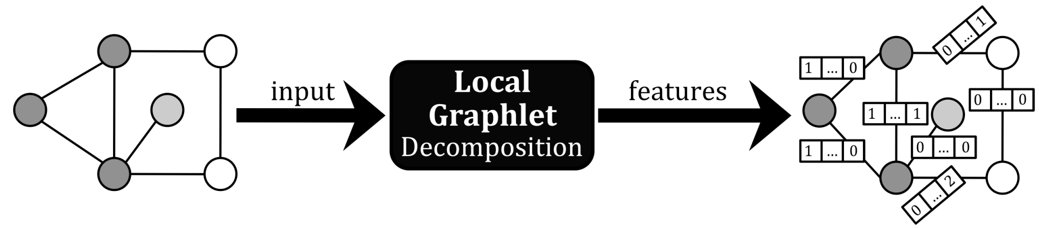
Sequential

- Enumerate all possible graphlets $\mathcal{O}(|V|^k)$
  - *- Exhaustive enumeration is too expensive*
- Count graphlets for each *node* $\mathcal{O}(|V|.\Delta^{k-1})$
  - *- Expensive for large k*

[Shervashidze et al. – AISTAT 2009]
[Hočevar et al. – Bioinfo. 13]

→ ***Not practical*** *– scales only for graphs with few hundred/thousand nodes/edges*

Current work



Sequential

- Enumerate all possible graphlets $\mathcal{O}(|V|^k)$
  - *Exhaustive enumeration is too expensive*
- Count graphlets for each *node* $\mathcal{O}(|V|.\Delta^{k-1})$
  - *Expensive for large k*

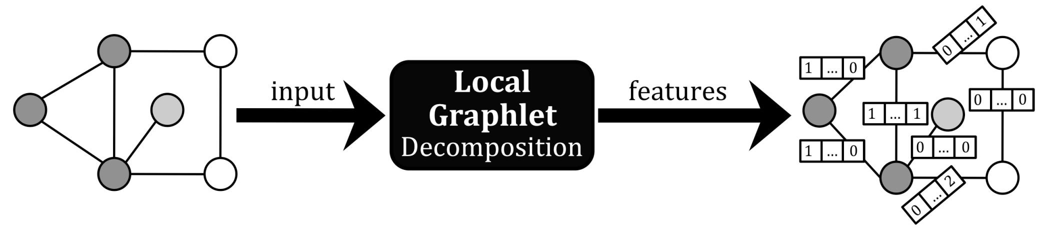
[Shervashidze et al. – AISTAT 2009]
[Hočevar et al. – Bioinfo. 13]

→ ***Not practical*** – *scales only for graphs with few hundred/thousand nodes/edges*

Parallel

- Edge-centric graphlet counting **(PGD)** [Ahmed *et al.* ICDM 14, KAIS 15]
  - Multi-core CPUs, large graphs

Current work



Sequential

- Enumerate all possible graphlets $\mathcal{O}(|V|^k)$
  - Exhaustive enumeration is too expensive
- Count graphlets for each *node* $\mathcal{O}(|V|.\Delta^{k-1})$
  - Expensive for large k

[Shervashidze et al. – AISTAT 2009]
[Hočevar et al. – Bioinfo. 13]

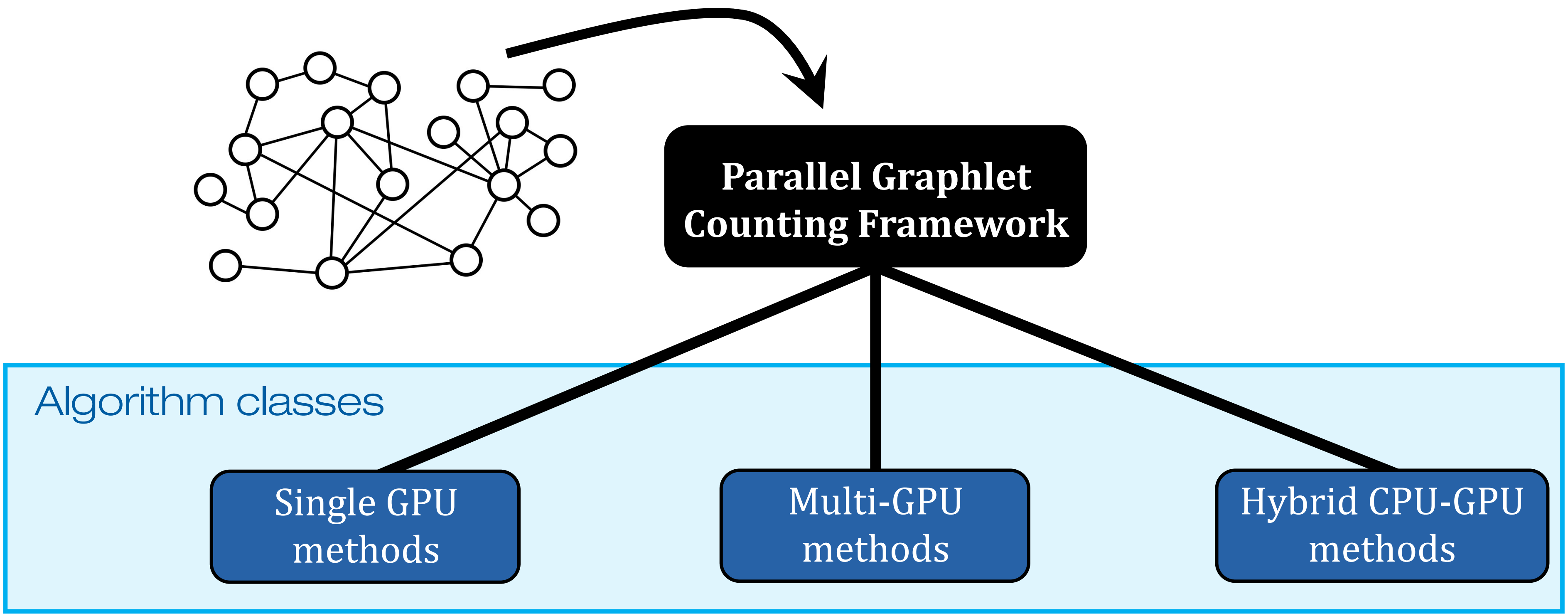
→ ***Not practical*** *– scales only for graphs with few hundred/thousand nodes/edges*

Parallel

- Edge-centric graphlet counting **(PGD)** [Ahmed *et al.* ICDM 14, KAIS 15]
  - Multi-core CPUs, large graphs
- Node-centric graphlet counting,
  - Single GPU, Handles only *tiny graphs* **(ORCA-GPU)** [Milinković *et al.*]

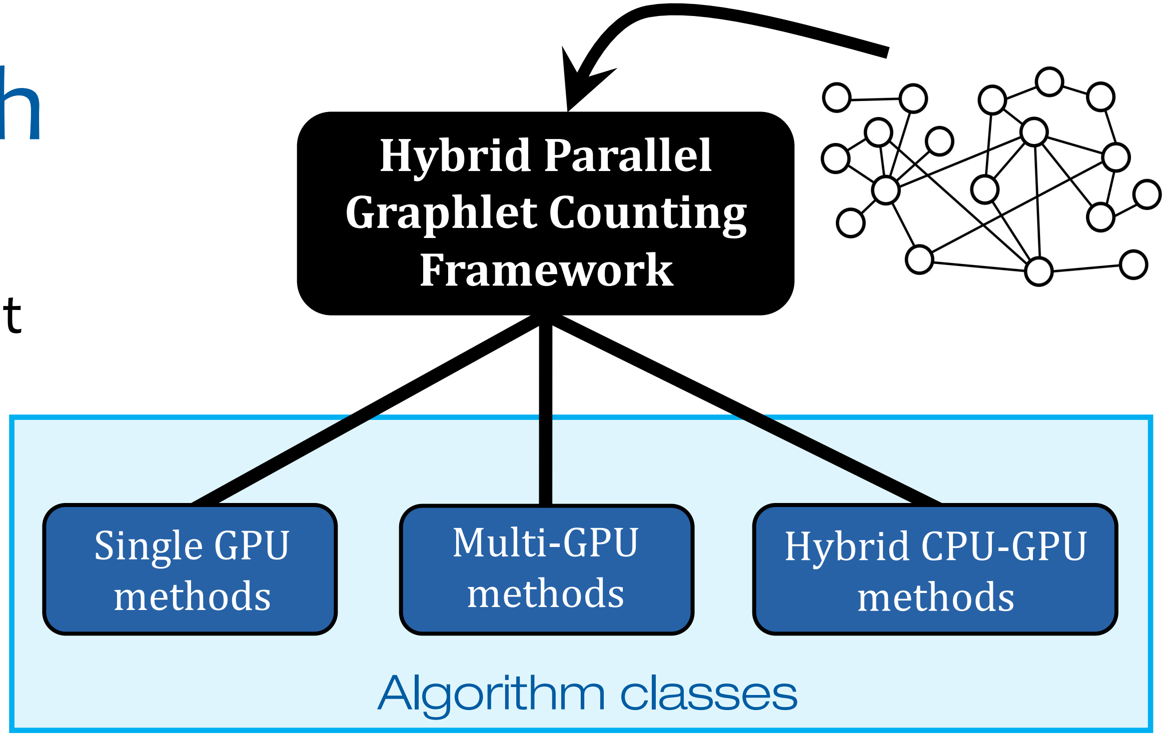
Our approach

Hybrid parallel graphlet counting framework that leverages all available CPUs and GPUs



Our approach

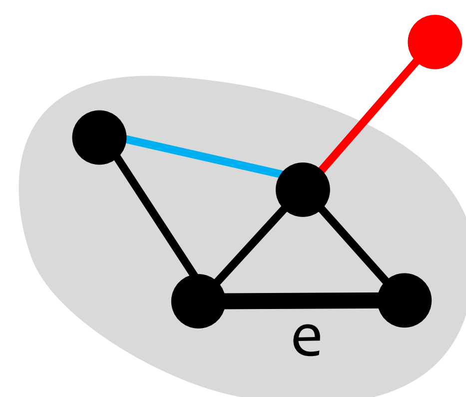
Hybrid parallel graphlet counting framework that *leverages all available CPUs & GPUs*



Other key advantages:

- Edge-centric parallelization
  - Improved load balancing & lock-free
- Global *and* local graphlet counts
- Connected *and* disconnected graphlets
- Fine-grained parallelization
- Space-efficient

⋮

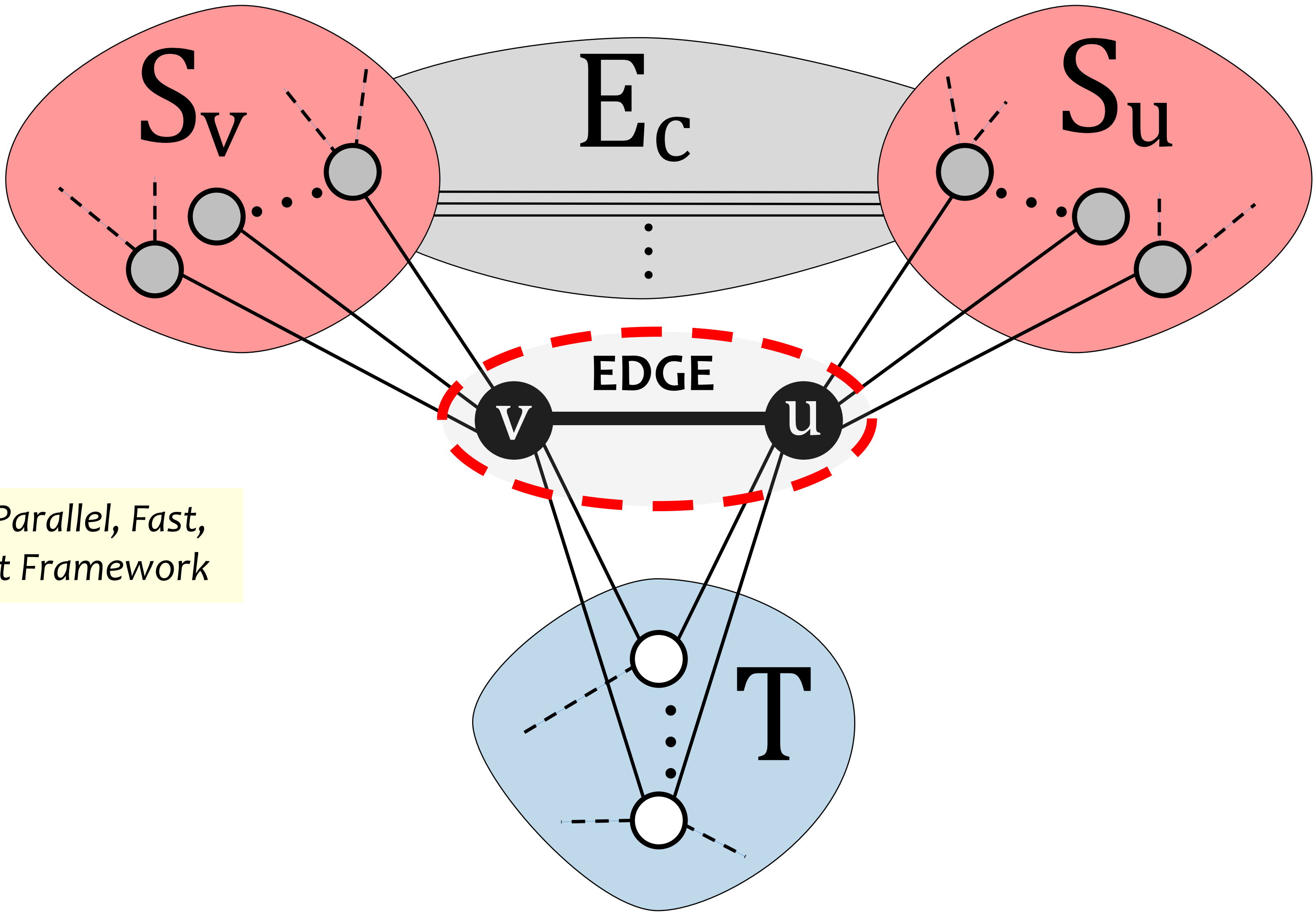


$$T = \{\underbrace{w_1, \cdots, w_i}_{T_{1:i}}, \underbrace{w_{i+1}, \cdots, w_t}_{T_{i+1:t}}\}$$

Overview of our approach

Overview

S_v (S_u) = nodes that form a **2-star** with **v** (u)



Edge-centric, Parallel, Fast, Space-efficient Framework

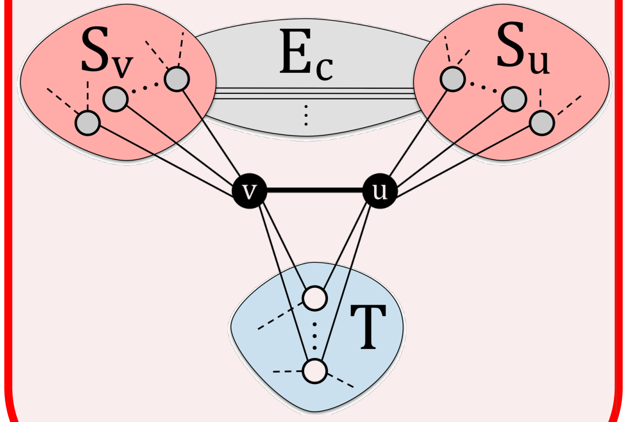
T = nodes completing a **triangle** with edge (v, u)

Our Approach –
(Edge-centric, parallel, space-efficient)

Step 1

Searching Edge Neighborhoods

For each edge
Find the triangles



Our Approach –

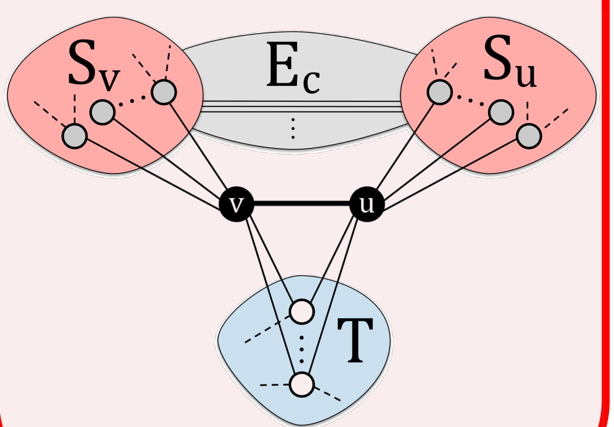
(Edge-centric, parallel, space-efficient)

Step 1

Searching Edge Neighborhoods

For each edge

Find the triangles



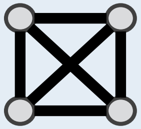
Step 2

Count a few k-graphlets

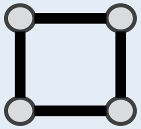
For each edge,

***count only*:**

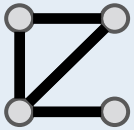
k-cliques



k-cycles



tailed-triangles

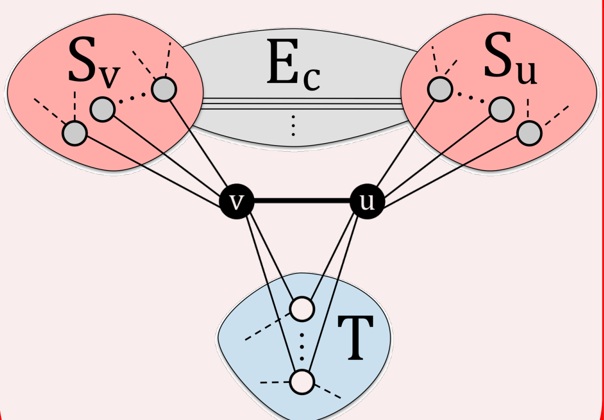


Our Approach –
(Edge-centric, parallel, space-efficient)

Step 1

Searching Edge Neighborhoods

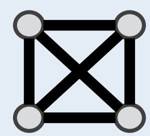
For each edge
Find the triangles

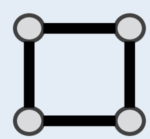


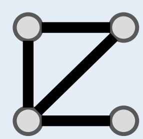
Step 2

Count a few k-graphlets

For each edge,
***count only*:**

k-cliques 

k-cycles 

tailed-triangles 

Step 3

Count all other graphlets

For each edge, use

combinatorial relationships

to derive counts of other graphlets in ***constant time o(1)***

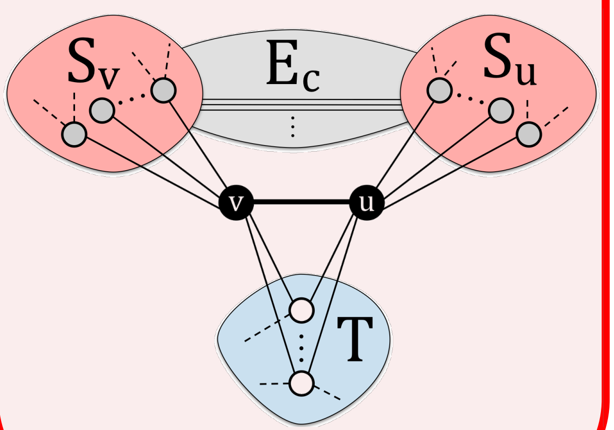
Our Approach – (Edge-centric, parallel, space-efficient)

Step 1

Searching Edge Neighborhoods

For each edge

Find the triangles



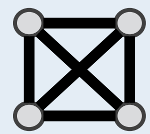
Step 2

Count a few k-graphlets

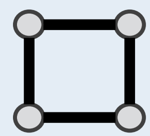
For each edge,

***count only*:**

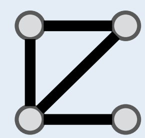
k-cliques



k-cycles



tailed-triangles



Step 3

Count all other graphlets

For each edge, use

combinatorial relationships

to derive counts of other graphlets

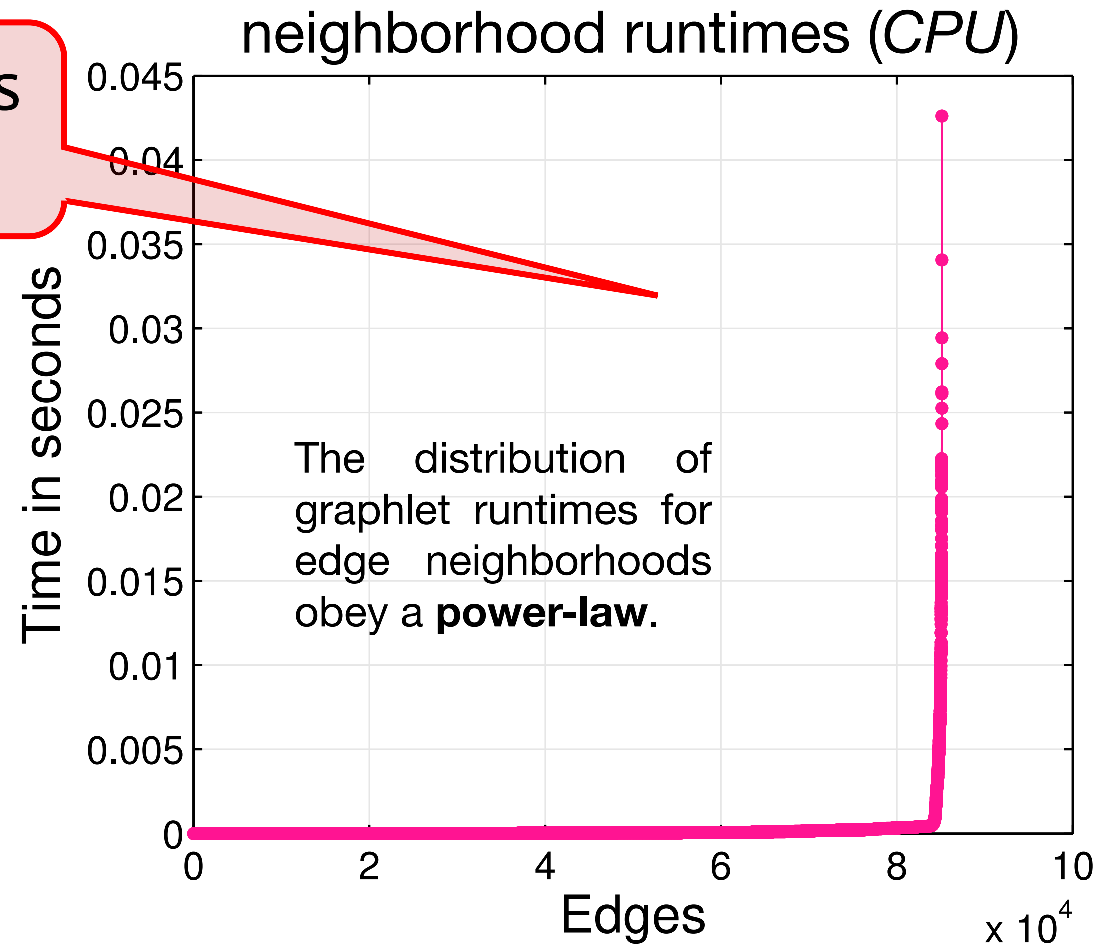
in ***constant time o(1)***

Step 4

Merge all counts

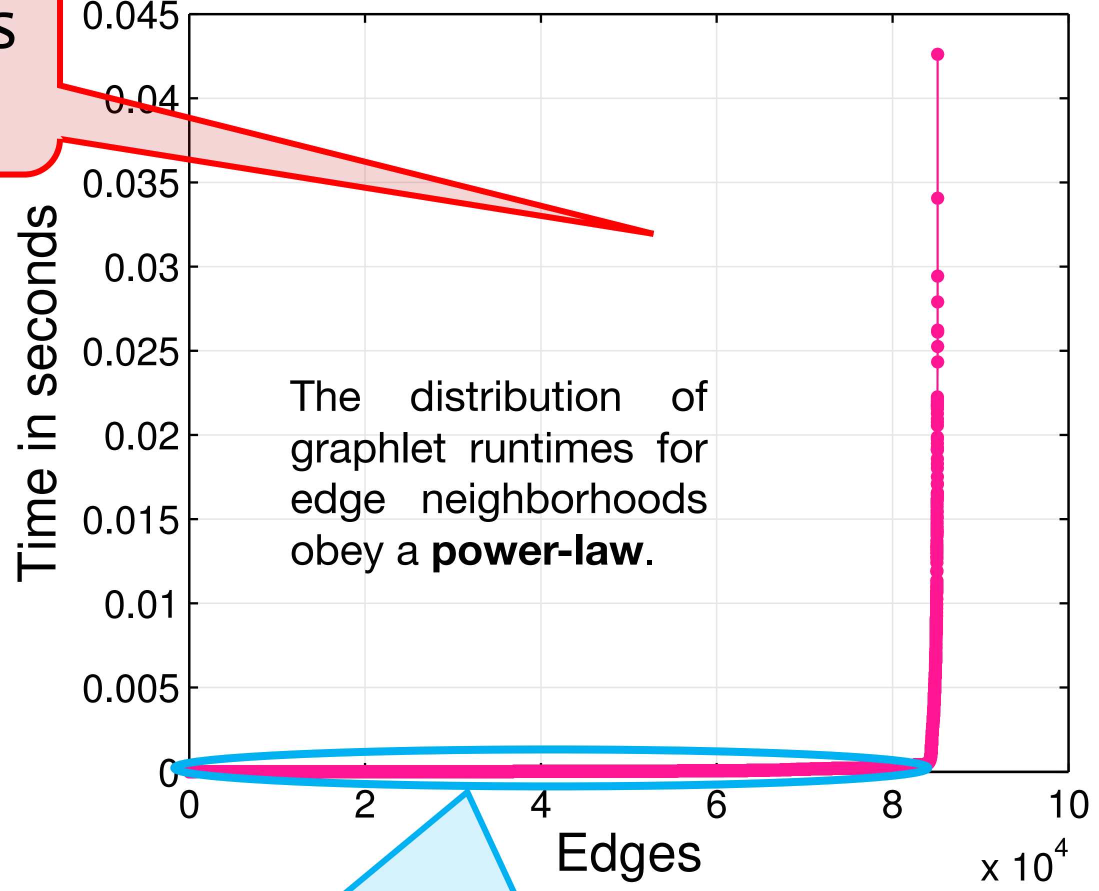
Key Observations

Neighborhood runtimes are **power-lawed**



Key Observations

Neighborhood runtimes are **power-lawed**

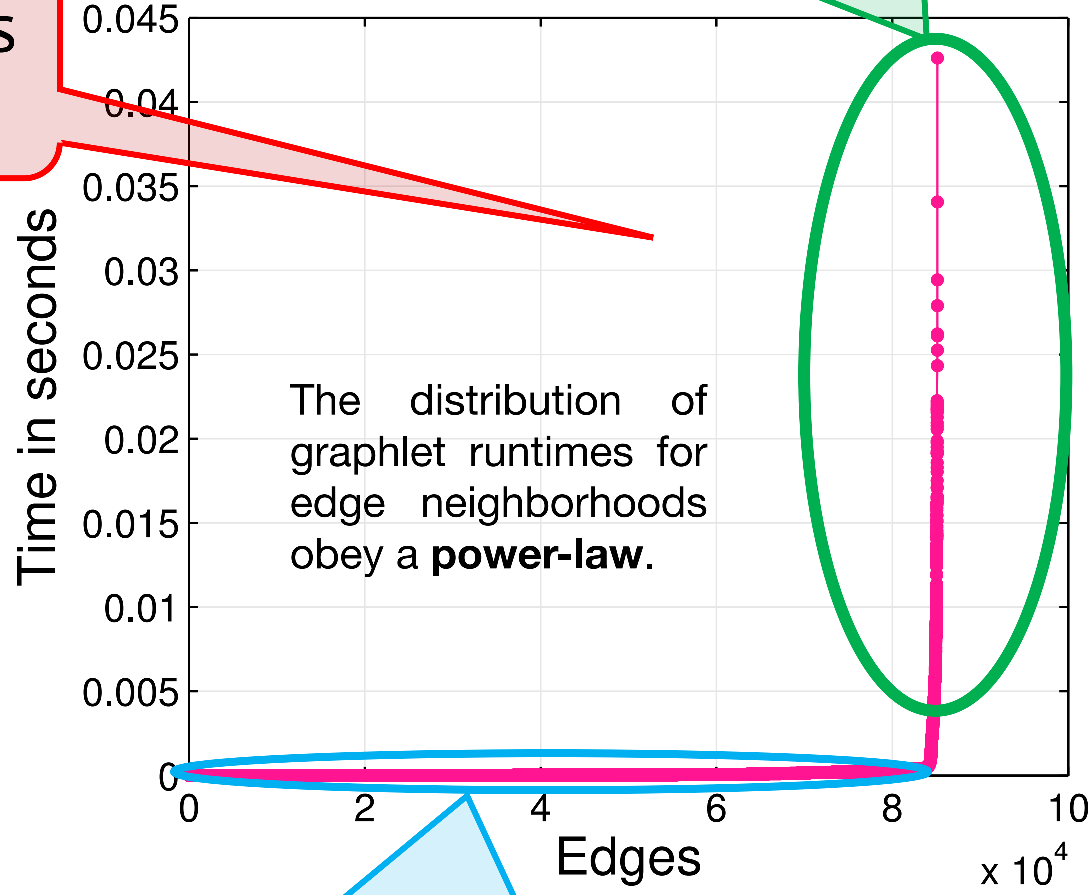


Most edge neighborhoods are fast with runtimes that are approximately equal.

Key Observations

HOWEVER, a handful of neighborhoods are hard and take significantly longer.

Neighborhood runtimes are **power-lawed**

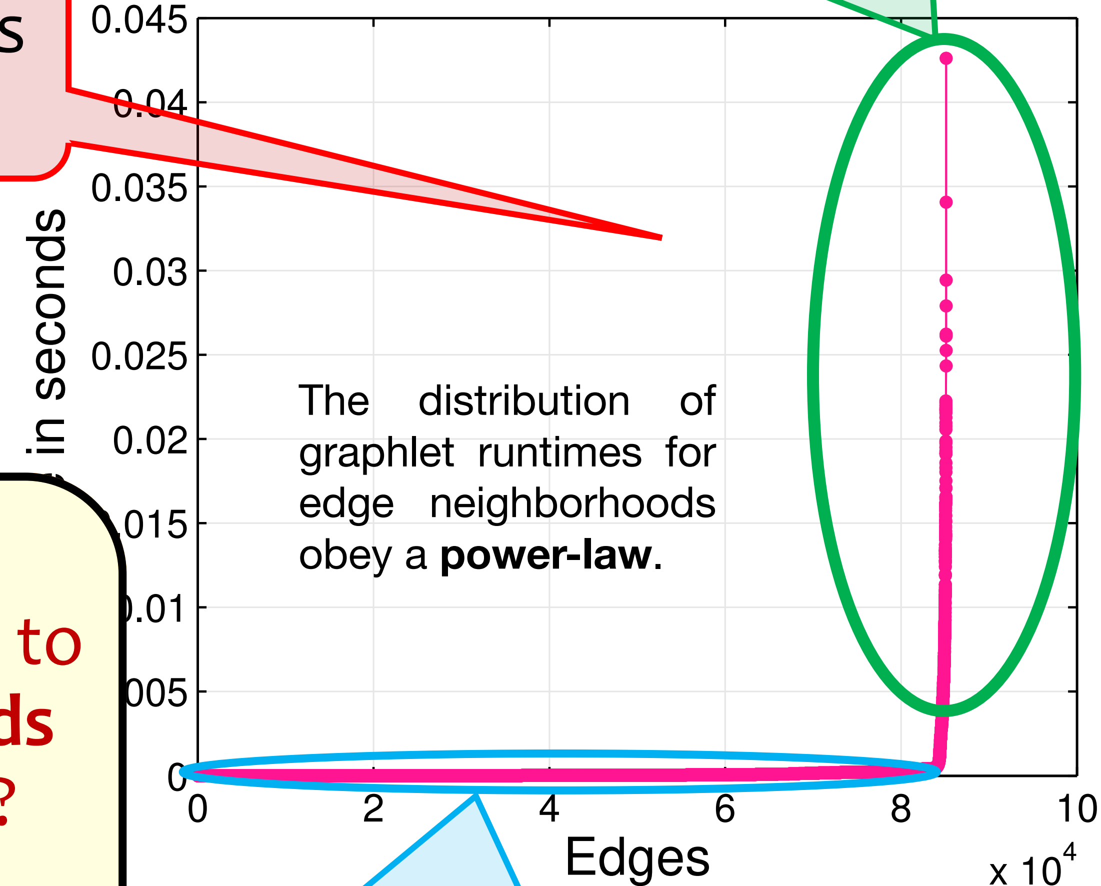


Most edge neighborhoods are fast with runtimes that are approximately equal.

Key Observations

Neighborhood runtimes are **power-lawed**

HOWEVER, a handful of neighborhoods are hard and take significantly longer.



Most edge neighborhoods are fast with runtimes that are approximately equal.

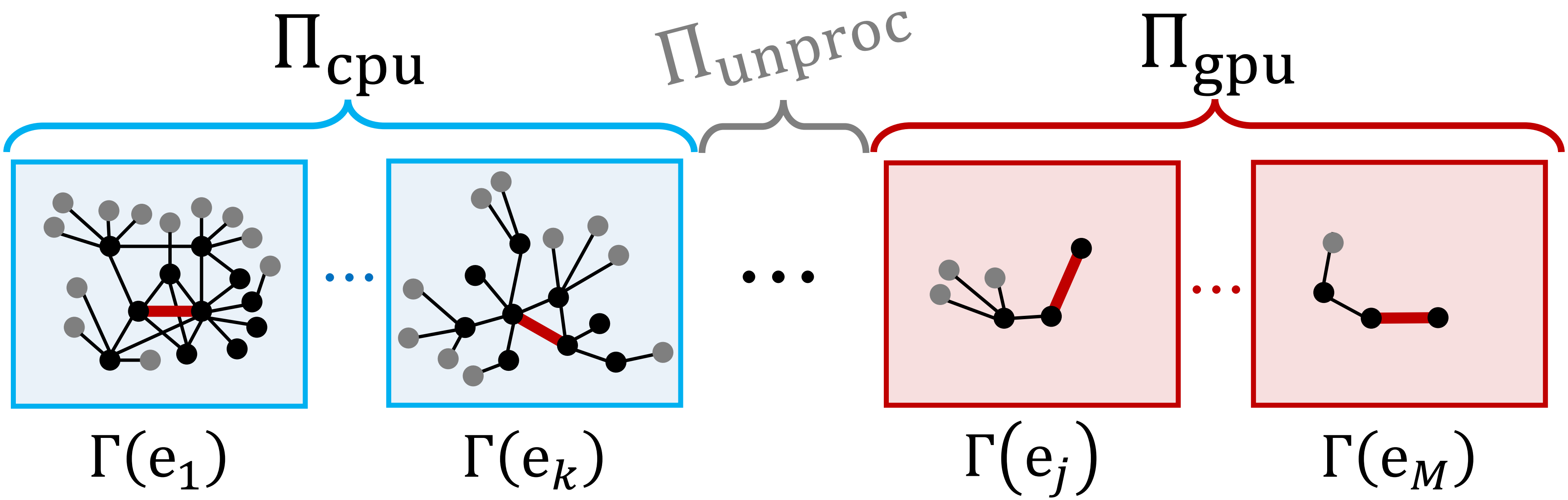
QUESTION:

What is the “best” way to **partition neighborhoods** among CPUs and GPUs?

- “hardness” proxy → edge deg., vol., …

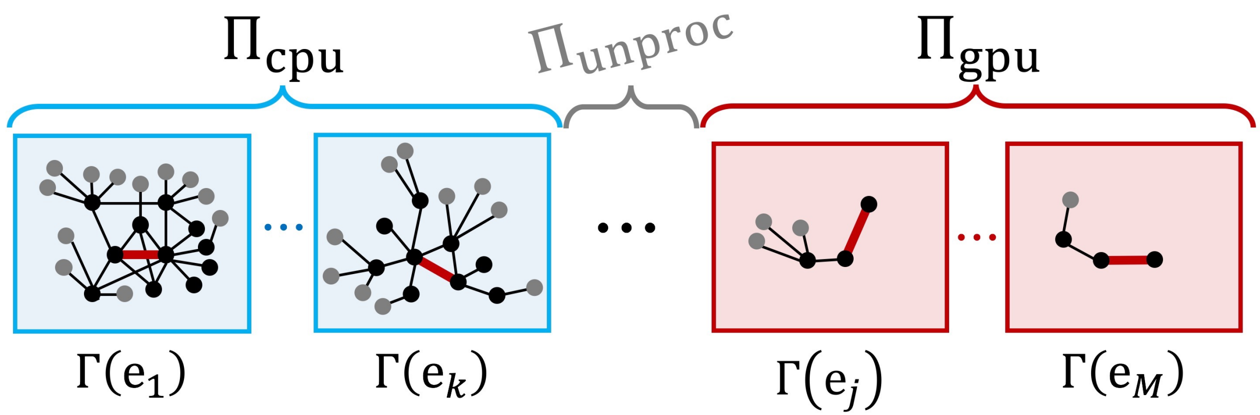
Our approach

- Order edges by "*hardness*" and partition into 3 sets:



Our approach

- Order edges by "*hardness*" and partition into 3 sets:



- Compute induced subgraphs centered at each edge
  - **CPU Workers:** use ***hash table*** for o(1) lookups, O(N)
  - **GPU Workers:** use ***binary search*** for o(log d) lookups

- When finished, dequeue next b edges:
  - **CPU:** get b edges from **FRONT** of Π_{unproc}
  - **GPU:** get b edges from **BACK** of Π_{unproc}

Preprocessing steps

Three simple and efficient preprocessing steps:

1) Sort vertices from smallest to largest **degree** $f(\cdot)$ and **relabel** them s.t. $f(v_1) \leq \cdots \leq f(v_N)$

Preprocessing steps

Three simple and efficient preprocessing steps:

1) Sort vertices from smallest to largest **degree** $f(\cdot)$ and **relabel** them s.t. $f(\mathrm{v}_1) \leq \cdots \leq f(\mathrm{v}_N)$

2) For each $\Gamma(\mathrm{v}_i), \forall i = 1, \ldots, \mathrm{N}$, order the set of neighbors $\Gamma(\mathrm{v}_i) = \{\ldots, \mathrm{w}_j, \ldots, \mathrm{w}_k, \ldots\}$ from smallest to largest **deg.**

Preprocessing steps

Three simple and efficient preprocessing steps:

1) Sort vertices from smallest to largest **degree** $f(\cdot)$ and **relabel** them s.t. $f(\mathrm{v}_1) \leq \cdots \leq f(\mathrm{v}_N)$

2) For each $\Gamma(\mathrm{v}_i), \forall i = 1, \ldots, \mathrm{N}$, order the set of neighbors $\Gamma(\mathrm{v}_i) = \{\ldots, \mathrm{w}_j, \ldots, \mathrm{w}_k, \ldots\}$ from smallest to largest **deg.**

3) Given edge $(\mathrm{v}, \mathrm{u}) \in \mathrm{E}$, ensure that $f(\mathrm{v}) \geq f(\mathrm{u})$
   - hence, v is always the vertex with largest degree, $d_v \geq d_u$

Preprocessing steps

Three simple and efficient preprocessing steps:

1) Sort vertices from smallest to largest **degree** $f(\cdot)$ and **relabel** them s.t. $f(\mathrm{v}_1) \leq \cdots \leq f(\mathrm{v}_N)$

2) For each $\Gamma(\mathrm{v}_i)$, $\forall i = 1, \ldots,$ N, order the set of neighbors $\Gamma(\mathrm{v}_i) = \{\ldots, \mathrm{w}_j, \ldots, \mathrm{w}_k, \ldots\}$ from smallest to largest **deg.**

3) Given edge (v, u) ∈ E, ensure that $f(\mathrm{v}) \geq f(\mathrm{u})$
   - hence, v is always the vertex with largest degree, $d_v \geq d_u$

- All of these steps are not required, but significantly improve
- Each step is extremely fast and lends itself to easy parallelization

Fine Granularity & Work Stealing

For a single edge (v, u) ∈ E,

I. Compute the sets $\mathbf{T}$ and $\mathbf{S_u}$

II. Find the total **4-cliques** using **T**

III. Find the total **4-cycles** using $\mathbf{S_u}$

NOTE: **(II)** and **(III)** are **independent** → **parallelize**

$$T = \{\underbrace{w_1, \cdots, w_i}_{T_{1:i}}, \underbrace{w_{i+1}, \cdots, w_t}_{T_{i+1:t}}\}$$

Unrestricted counts

$$|S_u| = d_u - |T_e| - 1$$
$$|S_v| = d_v - |T_e| - 1$$

$$C_3 = \sum_{e_k=(v,u)\in E} \mathbf{X}_{k,3} = \sum_{e_k=(v,u)\in E} |T|$$

$$C_4 = \sum_{e_k=(v,u)\in E} |S_v| + |S_u|$$

$$C_5 = \sum_{e_k=(v,u)\in E} N - (|S_v| + |S_u| + |T|) - 2$$

3-graphlets

$$C_7 = \sum_{e_k=(v,u)\in E} \mathbf{X}_{k,7}$$

$$C_8 = \sum_{e_k=(v,u)\in E} \binom{T}{2}$$

$$C_9 = \sum_{e_k=(v,u)\in E} |T| \cdot |S_v| \cdot |S_u|$$

$$C_{10} = \sum_{e_k=(v,u)\in E} \mathbf{X}_{k,10}$$

$$C_{11} = \sum_{e_k=(v,u)\in E} \binom{|S_v|}{2} + \binom{|S_u|}{2}$$

$$C_{12} = \sum_{e_k=(v,u)\in E} |S_v| \cdot |S_u|$$

$$C_{13} = \sum_{e_k=(v,u)\in E} |T| \cdot D_e$$

$$C_{14} = \sum_{e_k=(v,u)\in E} M - d_v - d_u + 1$$

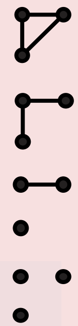
$$C_{15} = \sum_{e_k=(v,u)\in E} (|S_v| + |S_u|) \cdot D_e$$

$$C_{16} = \sum_{e_k=(v,u)\in E} \binom{D_e}{2}$$

$D_e = N - (|S_v| + |S_u| + |T|) - 2$

4-graphlets

Global counts



$$X_3 = {}^1\!/_3 \cdot C_3$$

$$X_4 = {}^1\!/_2 \cdot C_4$$

$$X_5 = C_5$$

$$X_6 = \binom{N}{3} - \left(X_3 + X_4 + X_5\right)$$

3-graphlets

$$X_7 = {}^1\!/_6 \cdot C_7$$

$$X_8 = C_8 - C_7$$

$$X_9 = {}^1\!/_2\left(C_9 - 4X_8\right)$$

$$X_{10} = {}^1\!/_4 \cdot C_{10}$$

$$X_{11} = {}^1\!/_3\left(C_9 - X_9\right)$$

$$X_{12} = C_{12} - C_{10}$$

$$X_{13} = {}^1\!/_3 \cdot \left(C_{13} - X_9\right)$$

$$X_{14} = {}^1\!/_2 \cdot \left(C_{14} - 6X_7 - 4X_8 - 2X_9 - 4X_{10} - 2X_{12}\right)$$

$$X_{15} = {}^1\!/_2 \cdot \left(C_{15} - 2X_{12}\right)$$

$$X_{16} = C_{16} - 2X_{14}$$

$$X_{17} = \binom{N}{4} - \sum X_i \quad \text{for } i = 7, \ldots, 16$$

4-graphlets

Time Complexity

4-clique	$\mathcal{O}(K\Delta T_{\max})$
4-cycle	$\mathcal{O}(K\Delta S_{\max})$
tailed-tri	$\mathcal{O}(K\Delta S_{\max})$
all	$\mathcal{O}(K\Delta(S_{\max} + T_{\max}))$

K = number of edges

Δ = max degree

T_{max} = max number of **triangles** incident to an edge in G

S_{max} = max number of **2-stars** incident to an edge in G

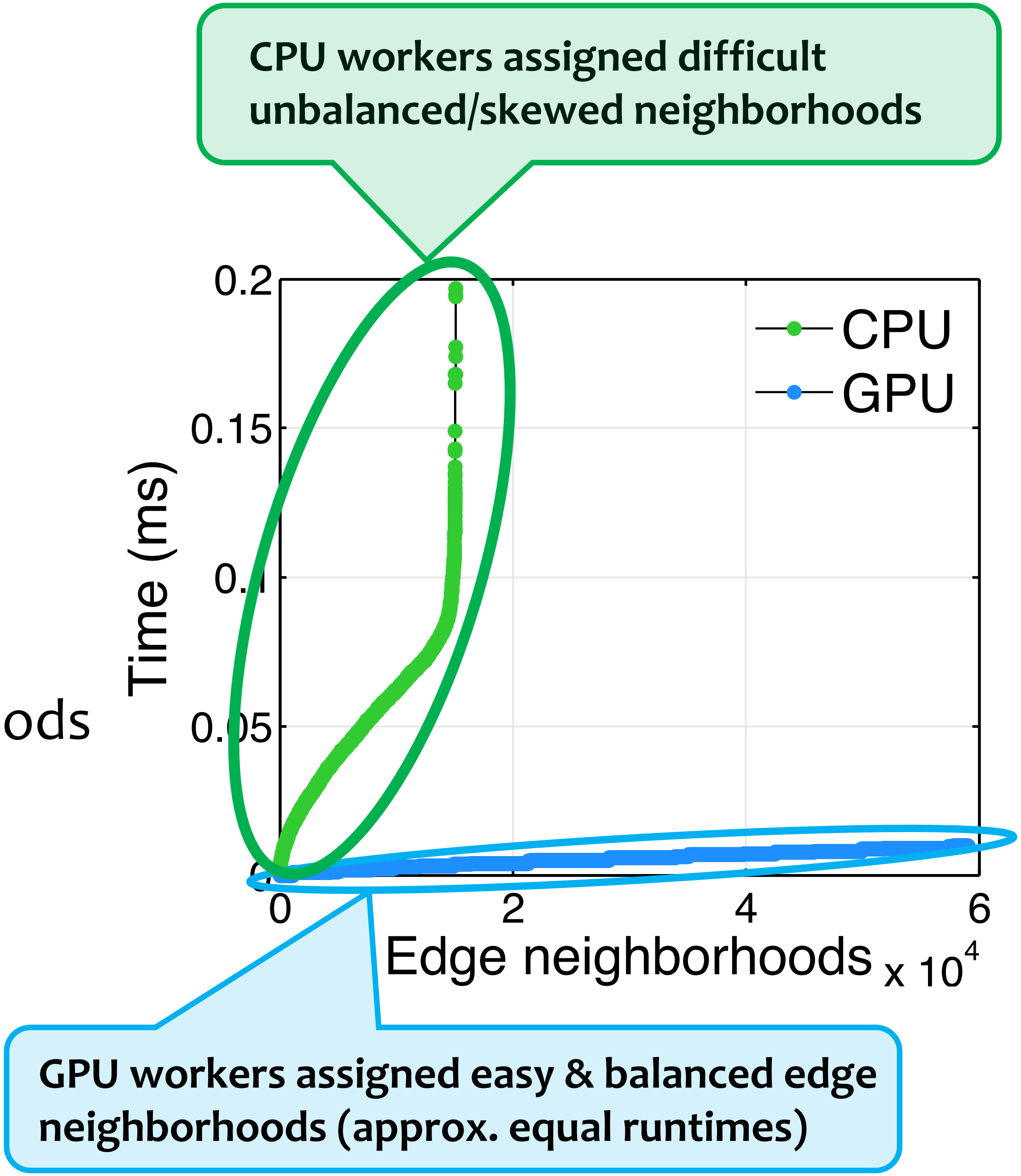
Experiments

Connected 4-graphlet frequencies for a variety of the real-world networks investigated from different domains.

Network type	graph	Connected Graphlets					
							
Facebook networks	socfb-Texas84	70.7M	376M	1.2B	215M	664M	3.9B
	socfb-UF	98M	433M	708M	186M	778M	874M
	socfb-MIT	13.7M	88.5M	909M	50.9M	498M	3.8B
	socfb-Stanford3	37.1M	226M	659M	151M	600M	1.8B
	socfb-Wisc87	23M	121M	1.9B	59.3M	1.3B	3.8B
	socfb-Indiana	60.2M	269M	1.6B	141M	495M	3.9B
Social networks	soc-flickr	311M	1B	208M	252M	1.2B	3.7B
	soc-google-plus	186M	994M	204M	463M	668M	3.7B
	soc-youtube	3.8M	156M	1.2B	162M	1B	2.3B
	soc-livejournal	307M	1.9B	1.8B	465M	778M	3.5B
	soc-twitter	430M	2.3B	1.7B	990M	314M	1.9B
	soc-orkut	280M	3.2B	953M	595M	520M	2B
Interaction networks	ia-enron-large	2.3M	22.5M	376M	6.8M	185M	1.4B
	ia-wiki-Talk	2.2M	32.3M	668M	33.8M	766M	1.5B
Collaboration networks	ca-HepPh	150M	35.2M	462M	821k	143M	204M
Brain networks	brain-mouse-ret1	71.4M	303M	1.1B	47.4M	1.1B	1.1B
Web graphs	web-baidu-baike	27.8M	248M	476M	653M	1.3B	1.2B
	web-arabic05	232M	3.4M	26.5M	79.2k	490M	27.3M
Technological/IP networks	tech-as-skitter	149M	2.4B	571M	817M	808M	2.8B
Dense hard benchmark graphs	C500-9	656M	909M	201M	50.2M	7.3M	22.3M
	p-hat1000-1	20.3M	265M	1.3B	282M	1.2B	3B

Validating edge partitioning

- Edges partitioned by "hardness"
- GPUs assigned sparser neighborhoods
- Assigns edge neighborhoods to "best" processor type
- Importance of initial ordering



Experiments: Improvement

GPU: Uses a single multi-core GPU
Multi-GPU: Uses all available GPUs
Hybrid: Leverages all multi-core CPUs & GPUs

Runtime improvement over state-of-the-art

graph	$\mathbb{K}$	Δ	Δ_{gpu}	α	Speedup (times faster) GPU	Multi-GPU	Hybrid
socfb-Texas84	81	6312	450	0.031	4.65x	21.91x	263.26x
socfb-UF	83	8246	370	0.05	1.6x	55.65x	165.63x
socfb-MIT	72	708	266	0.7	11.98x	28.47x	106.14x
socfb-Stanford3	91	1172	365	0.05	21.07x	63x	133.15x
socfb-Wisc87	60	3484	300	0.04	17.88x	142.41x	189.08x
socfb-Indiana	76	1358	329	0.04	22.25x	96.89x	207.11x
soc-flickr	309	4369	4196	0.04	7.32x	31.85x	102.24x
soc-google-plus	135	1790	328	0.07	4.95x	11.98x	56.03x
soc-youtube	49	25409	1079	0.07	3.87x	26.82x	180.64x
soc-brightkite	52	1134	132	0.12	2.51x	8.09x	17.67x
soc-livejournal	213	2651	157	0.05	8.92x	70.01x	98.83x
soc-twitter	125	51386	13533	0.05	2.68x	21.76x	372.72x
soc-orkut	230	27466	646	0.05	6.12x	57.71x	129.26x
ia-enron-large	43	1383	243	0.176	2.94x	10.79x	28.30x
ia-wiki-Talk	58	1220	1034	0.02	23.35x	37.50x	85.46x
ca-HepPh	238	491	169	0.35	1.42x	6.62x	17.14x
brain-mouse-ret1	121	744	712	0.26	3.21x	5.14x	32.71x
web-baidu-baike	78	97848	11919	0.03	4.83x	39.55x	156.45x
web-arabic05	101	1102	49	0.14	5.19x	29.51x	60.02x

2 Intel Xeon CPUs (E5-2687) –
- 8 cores (3.10Ghz)

8 Titan Black NVIDIA GPUs –
- 2880 cores (889 Mhz), ~6GB

Experiments: Improvement

GPU: Uses a single multi-core GPU
Multi-GPU: Uses all available GPUs
Hybrid: Leverages all multi-core CPUs & GPUs

Runtime improvement over state-of-the-art

Improvement: *significant* at $\alpha = 0.01$

graph	$\mathbb{K}$	Δ	Δ_{gpu}	α	Speedup (times faster) GPU	Multi-GPU	Hybrid
socfb-Texas84	81	6312	450	0.031	4.65x	21.91x	263.26x
socfb-UF	83	8246	370	0.05	1.6x	55.65x	165.63x
socfb-MIT	72	708	266	0.7	11.98x	28.47x	106.14x
socfb-Stanford3	91	1172	365	0.05	21.07x	63x	133.15x
socfb-Wisc87	60	3484	300	0.04	17.88x	142.41x	189.08x
socfb-Indiana	76	1358	329	0.04	22.25x	96.89x	207.11x
soc-flickr	309	4369	4196	0.04	7.32x	31.85x	102.24x
soc-google-plus	135	1790	328	0.07	4.95x	11.98x	56.03x
soc-youtube	49	25409	1079	0.07	3.87x	26.82x	180.64x
soc-brightkite	52	1134	132	0.12	2.51x	8.09x	17.67x
soc-livejournal	213	2651	157	0.05	8.92x	70.01x	98.83x
soc-twitter	125	51386	13533	0.05	2.68x	21.76x	372.72x
soc-orkut	230	27466	646	0.05	6.12x	57.71x	129.26x
ia-enron-large	43	1383	243	0.176	2.94x	10.79x	28.30x
ia-wiki-Talk	58	1220	1034	0.02	23.35x	37.50x	85.46x
ca-HepPh	238	491	169	0.35	1.42x	6.62x	17.14x
brain-mouse-ret1	121	744	712	0.26	3.21x	5.14x	32.71x
web-baidu-baike	78	97848	11919	0.03	4.83x	39.55x	156.45x
web-arabic05	101	1102	49	0.14	5.19x	29.51x	60.02x

Experiments: Improvement

GPU: Uses a single multi-core GPU
Multi-GPU: Uses all available GPUs
Hybrid: Leverages all multi-core CPUs & GPUs

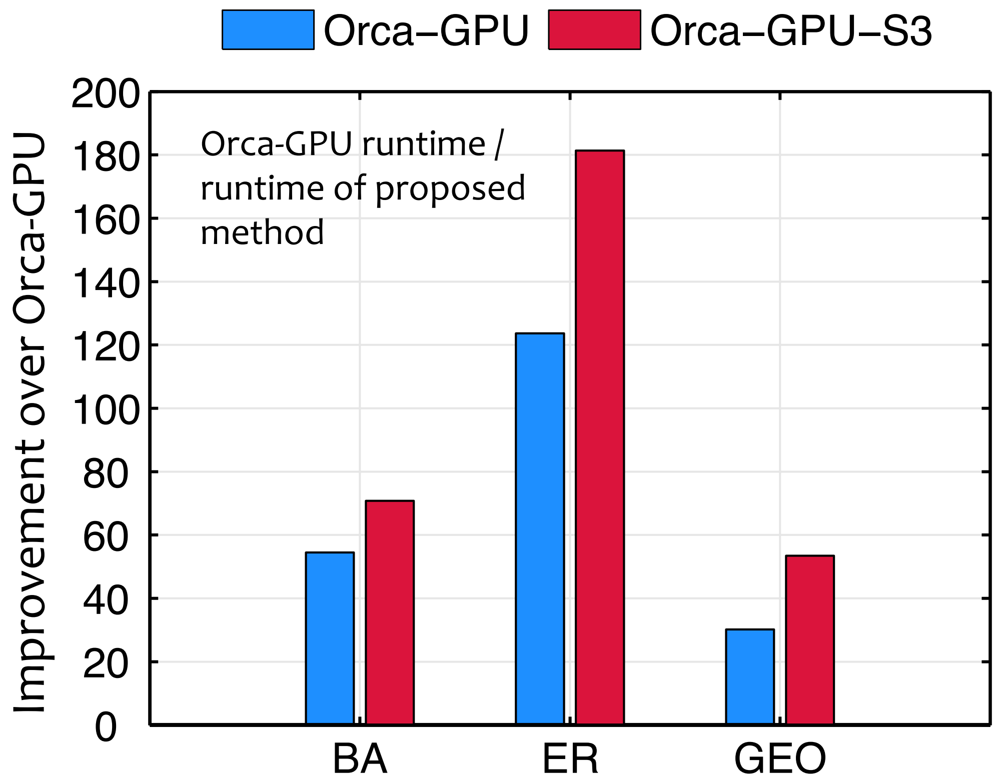
Runtime improvement over state-of-the-art

graph	$\mathbb{K}$	Δ	Δ_{gpu}	α	Speedup (times faster) GPU	Multi-GPU	Hybrid
socfb-Texas84	81	6312	450	0.031	4.65x	21.91x	263.26x
socfb-UF	83	8246	370	0.05	1.6x	55.65x	165.63x
socfb-MIT	72	708	266	0.7	11.98x	28.47x	106.14x
socfb-Stanford3	91	1172	365	0.05	21.07x	63x	133.15x
socfb-Wisc87	60	3484	300	0.04	17.88x	142.41x	189.08x
socfb-Indiana	76	1358	329	0.04	22.25x	96.89x	207.11x
soc-flickr	309	4369	4196	0.04	7.32x	31.85x	102.24x
soc-google-plus	135	1790	328	0.07	4.95x	11.98x	56.03x
soc-youtube	49	25409	1079	0.07	3.87x	26.82x	180.64x
soc-brightkite	52	1134	132	0.12	2.51x	8.09x	17.67x
soc-livejournal	213	2651	157	0.05	8.92x	70.01x	98.83x
soc-twitter	125	51386	13533	0.05	2.68x	21.76x	372.72x
soc-orkut	230	27466	646	0.05	6.12x	57.71x	129.26x
ia-enron-large	43	1383	243	0.176	2.94x	10.79x	28.30x
ia-wiki-Talk	58	1220	1034	0.02	23.35x	37.50x	85.46x
ca-HepPh	238	491	169	0.35	1.42x	6.62x	17.14x
brain-mouse-ret1	121	744	712	0.26	3.21x	5.14x	32.71x
web-baidu-baike	78	97848	11919	0.03	4.83x	39.55x	156.45x
web-arabic05	101	1102	49	0.14	5.19x	29.51x	60.02x
MEAN					**8x**	**40x**	**126x**

Improvement: *significant* at $\alpha = 0.01$

Comparing ORCA-GPU methods

- **Significant improvement** over Orca-GPU (at $\alpha = 0.01$)



Many problems with Orca-GPU:

- No "effective parallelization", many parts dependent
- Requires synchronization throughout, locks
- No fine-grained parallelization

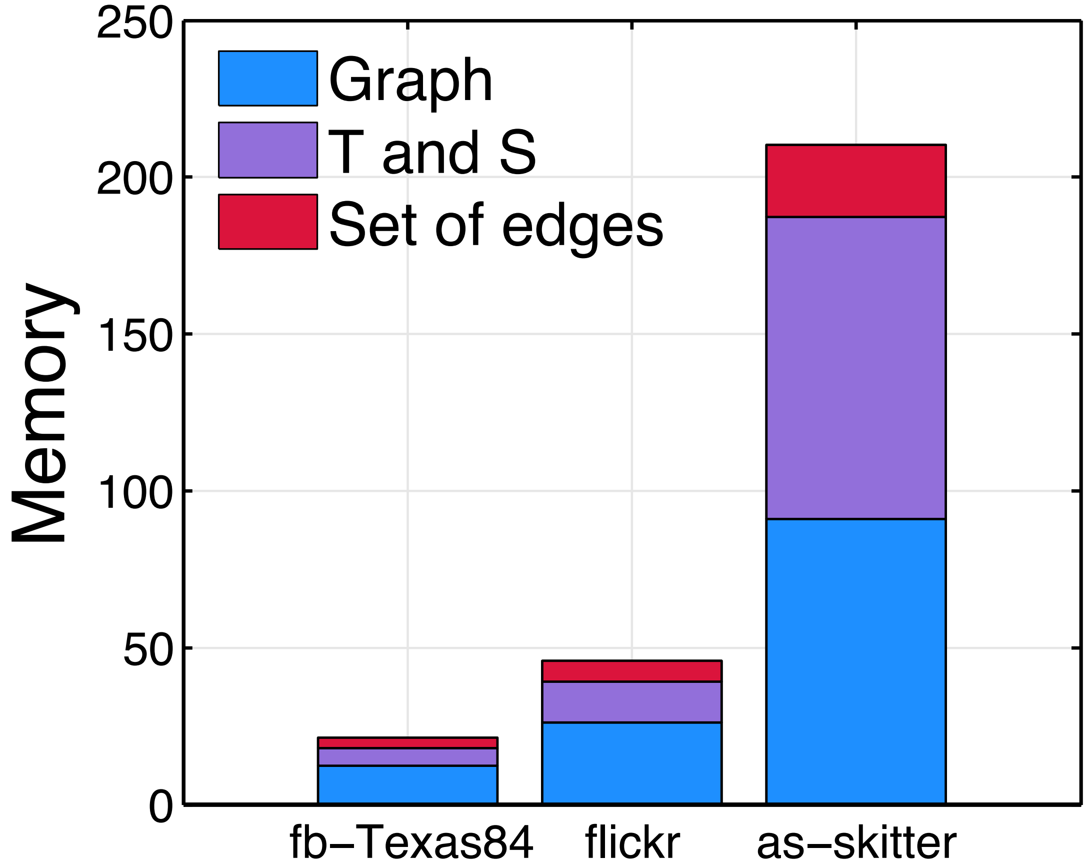
Varying the edge ordering

$$\texttt{vol}(e_k) = \texttt{vol}(u, v) = \sum_{w \in \Gamma(u,v)} d_w$$

	Descending		Reverse order	
graph	**d**	vol	$\mathbf{d}^{-1}$	$\texttt{vol}^{-1}$
socfb-Texas84	263.3x	**284.1x**	23.5x	10.8x

Ordering strategy significantly impacts performance

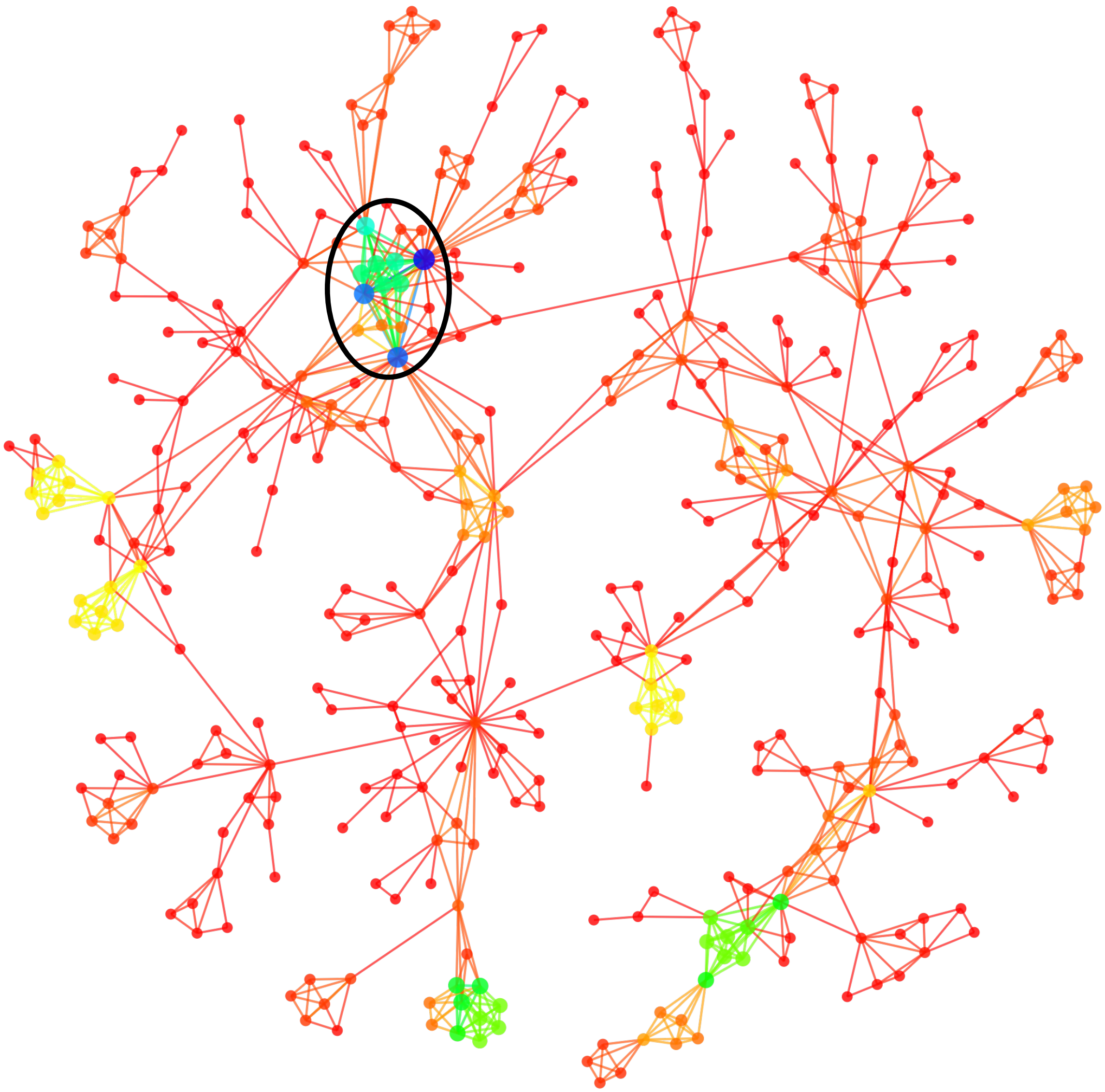
Space-efficient & comm. avoidance



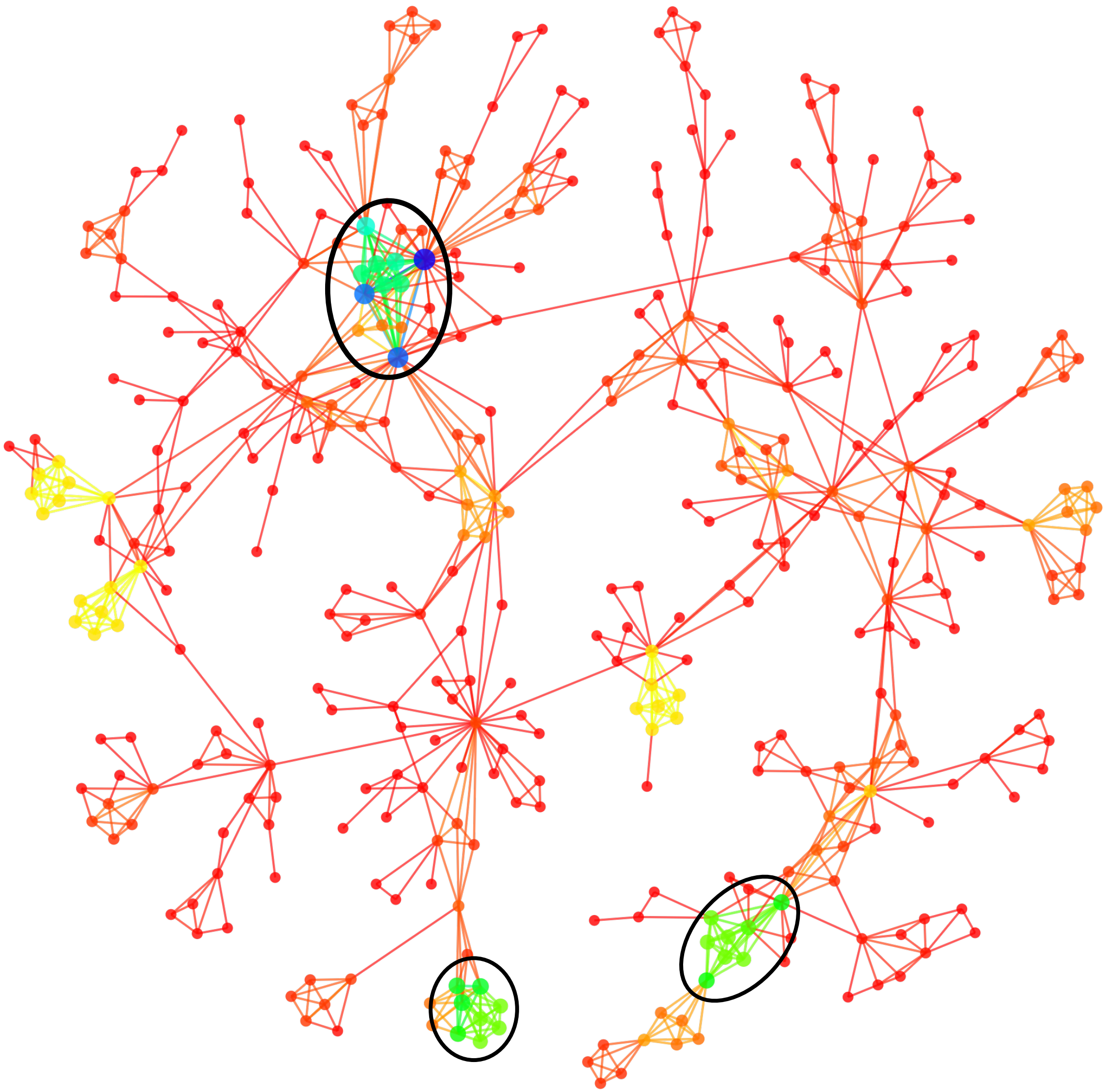
Average memory (MB) per GPU for three networks.

Applications

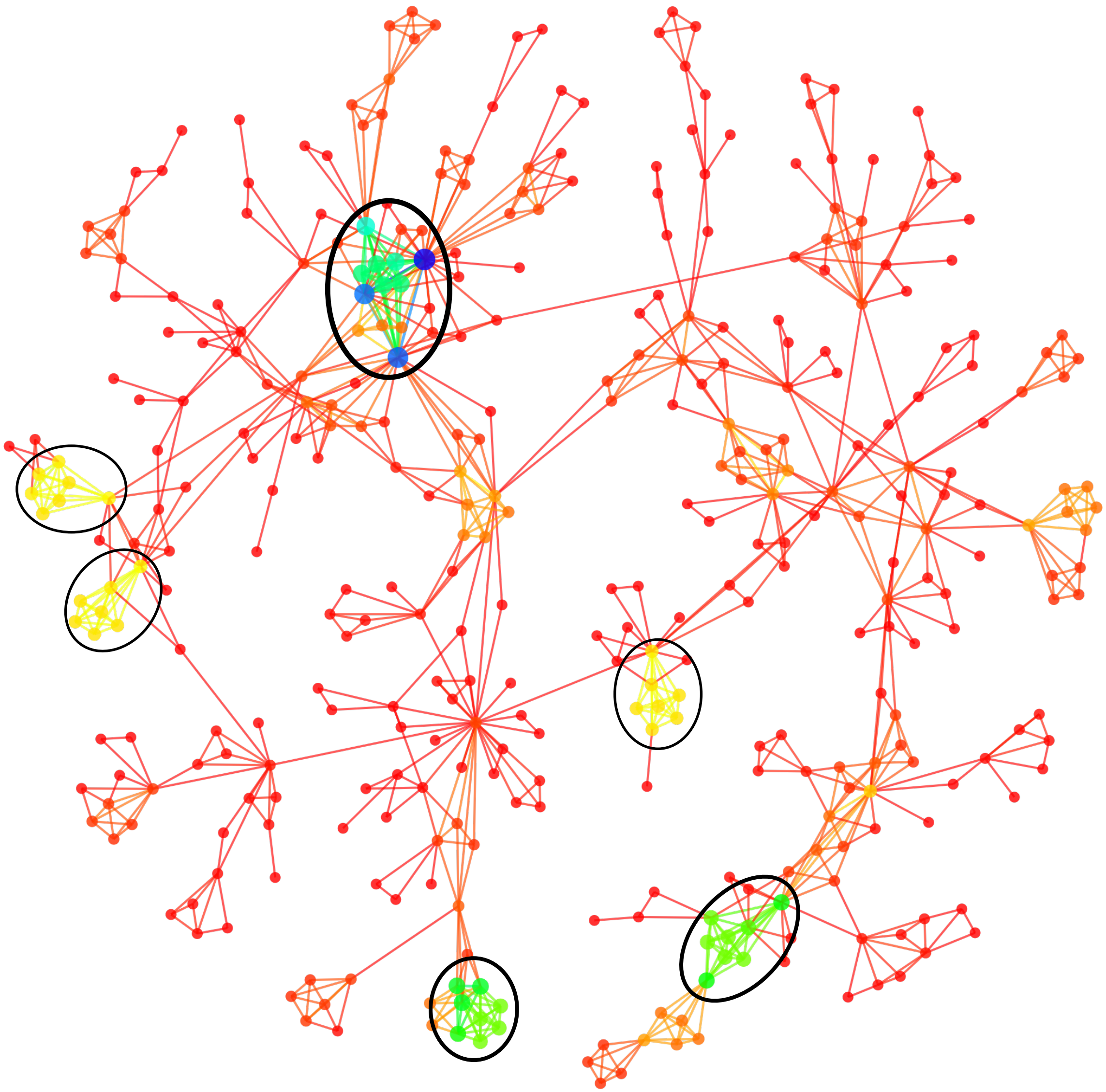
Ranking/spotting **Large Cliques** via Graphlets



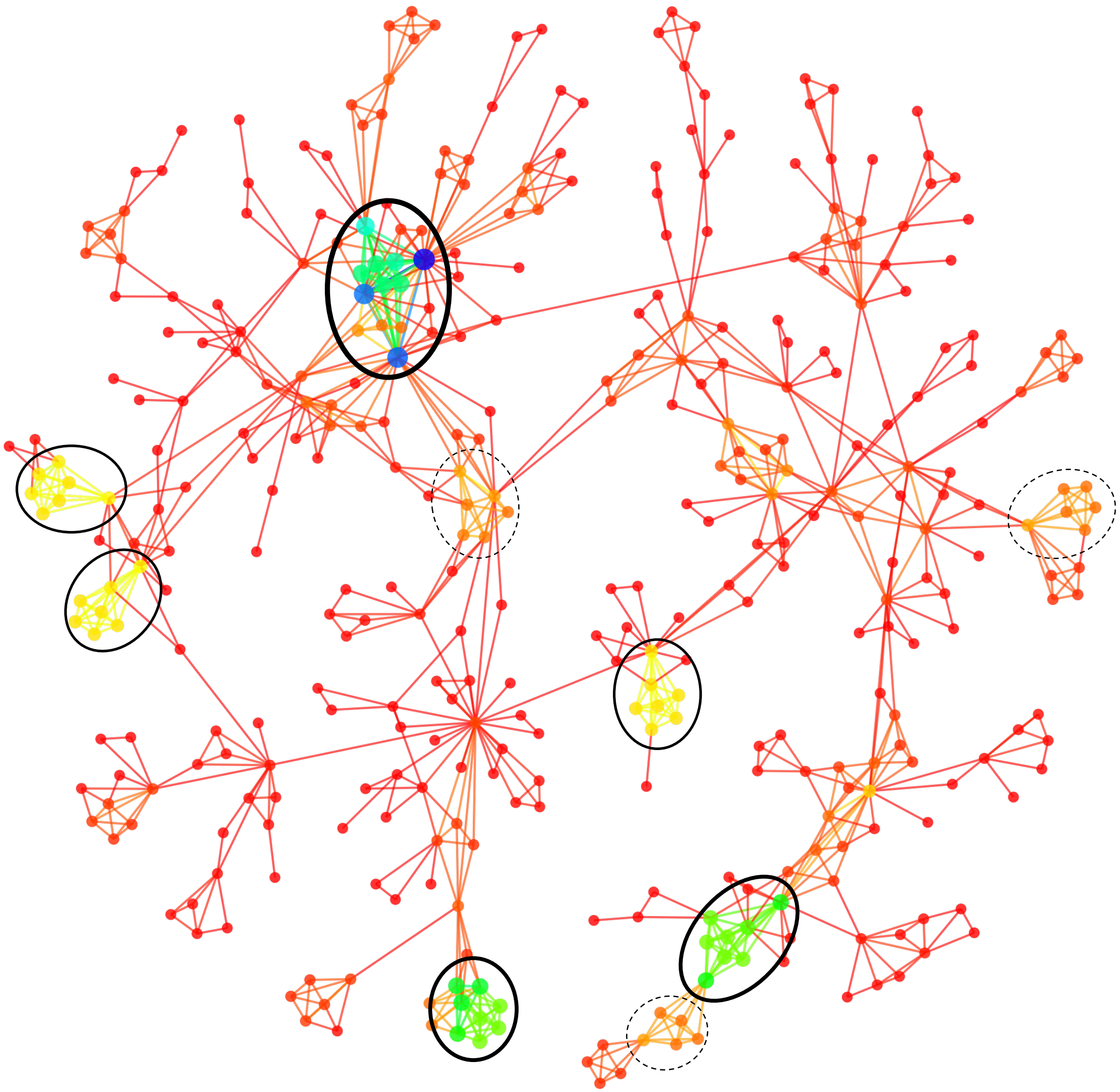
Ranking/spotting **Large Cliques** via Graphlets



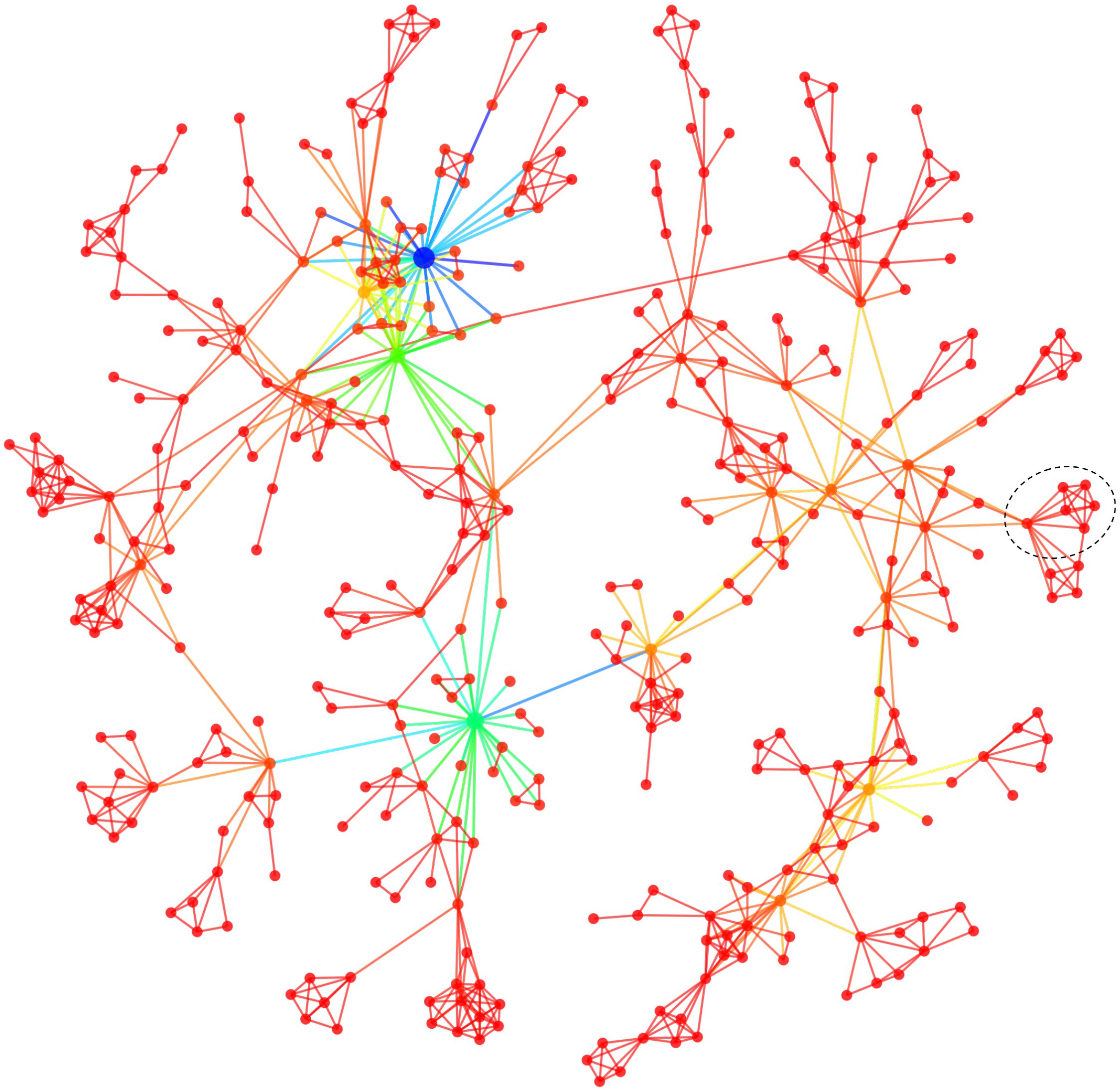
Ranking/spotting **Large Cliques** via Graphlets



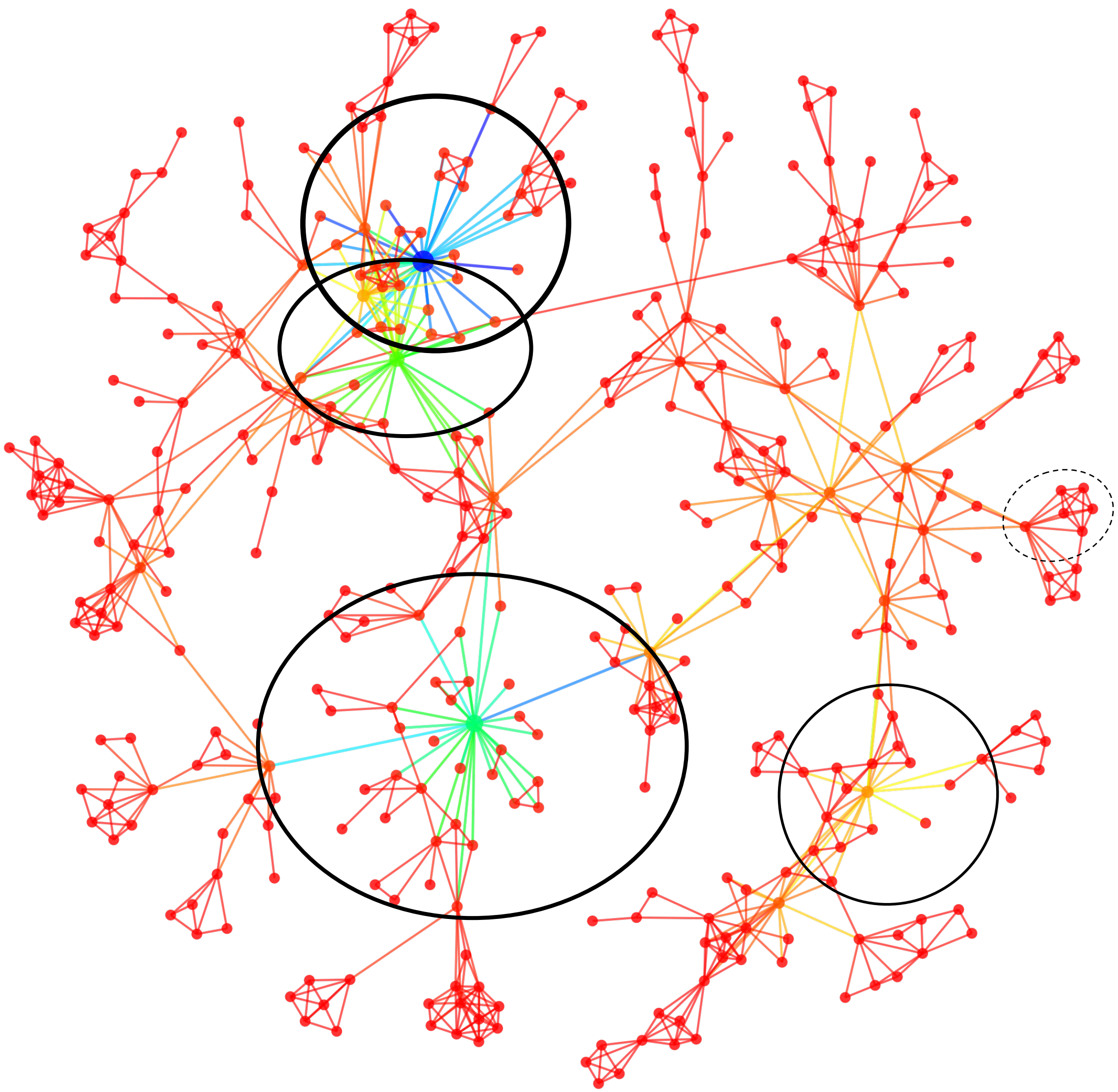
Ranking/spotting **Large Cliques** via Graphlets



Ranking/spotting **Large Stars** via Graphlets



Ranking/spotting **Large Stars** via Graphlets



Summary

Framework & Algorithms

- Introduced hybrid graphlet counting approach that leverages all available CPUs & GPUs
- First **hybrid CPU-GPU approach** for graphlet counting
- On average **126x faster** than current methods
  - Edge-centric computations (only requires access to edge neighborhood)
- Time and space-efficient

Applications

- Visual analytics and **real-time graphlet mining**

Thanks!

Questions?

Data: http://networkrepository.com

Research generously supported by:

xerox

NVIDIA®
Research Gift