Interplay between Topology and Edge Weights in Real-World Graphs: Concepts, Patterns, and an Algorithm



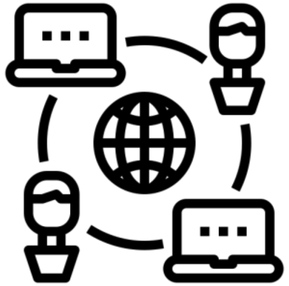
Fanchen Bu   Shinhwan Kang   Kijung Shin

Graphs

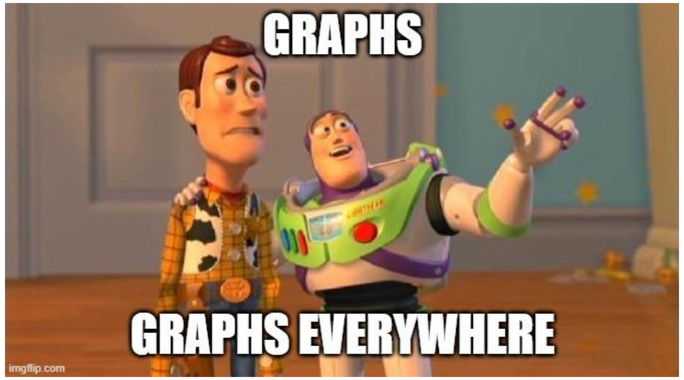
- A **graph** $G = (V, E)$ consists of a node set V and an edge set E
  - Each **edge** joins a pair of nodes
- Graphs naturally represent **relations between real-world objects**

Transportation Systems

Social Networks

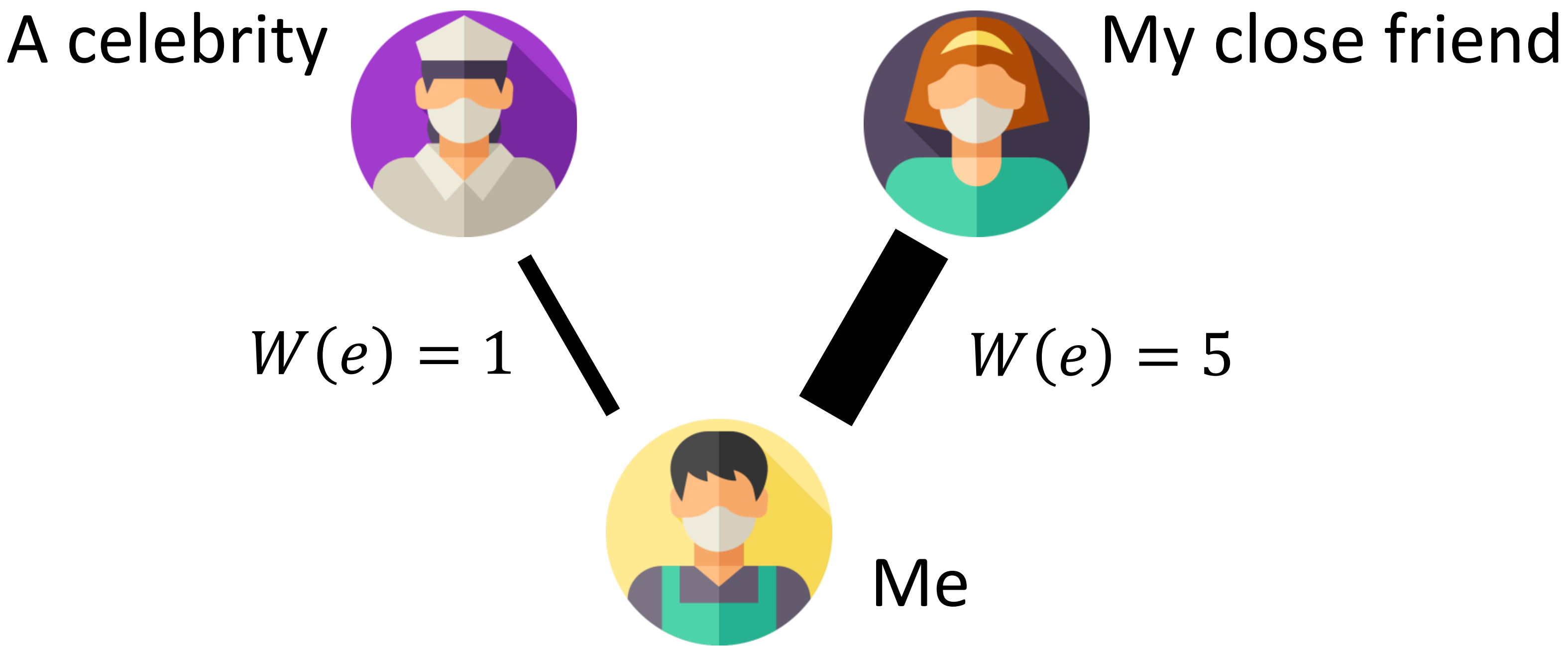


Communication Systems



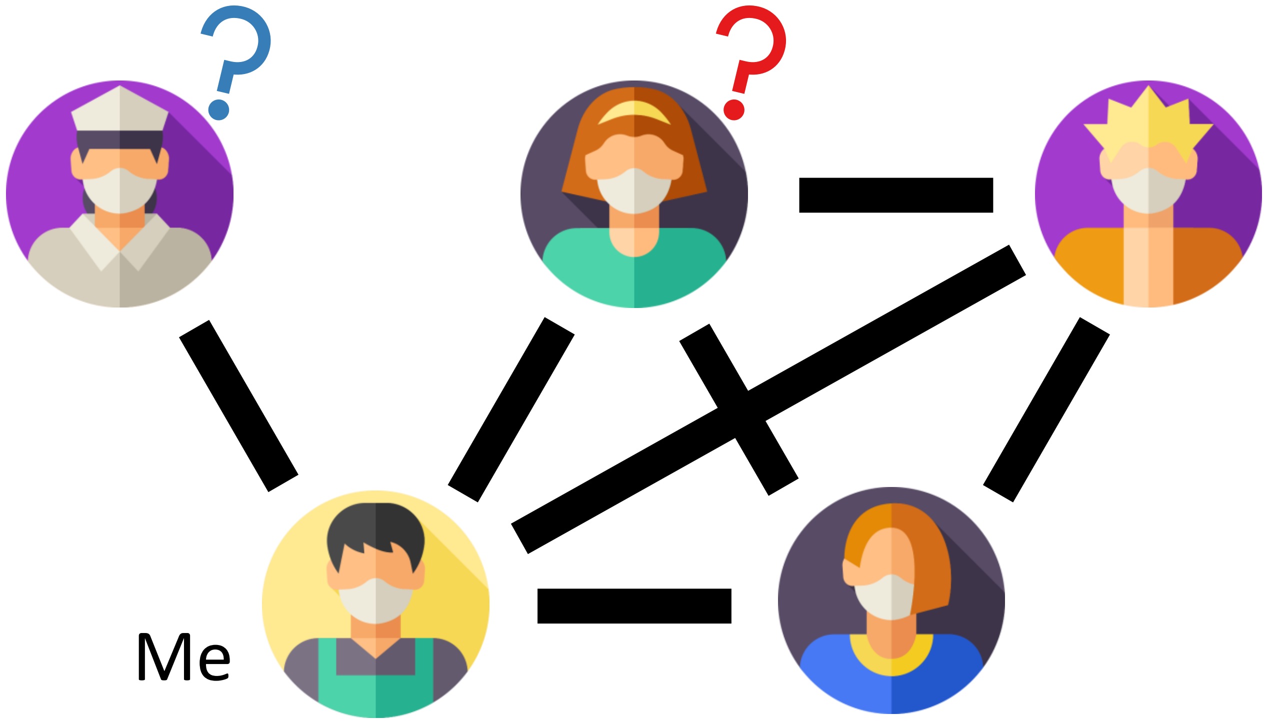
Edges Weights

- Even in the same graph, edges are not all the same
- We can use **edge weights** to describe the heterogeneity of edges
  - Each edge e has its edge weight, which is a **positive integer** $W(e)$
- **Example:** Online social networks



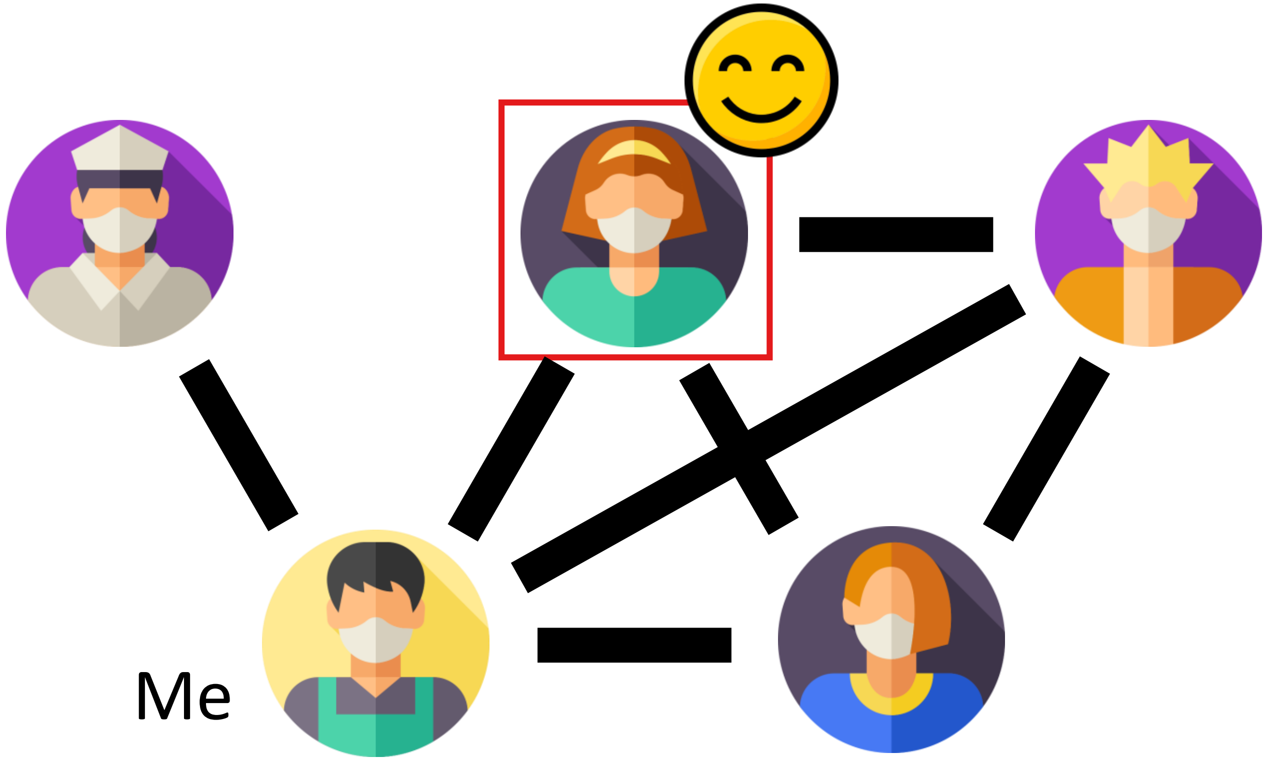
Topology and Edge Weights are Entangled

- In many cases, we can infer edge weights from topology (connections)
- **Example:** Online social networks
  - **Q:** Which person is more likely my close friend with strong connection?



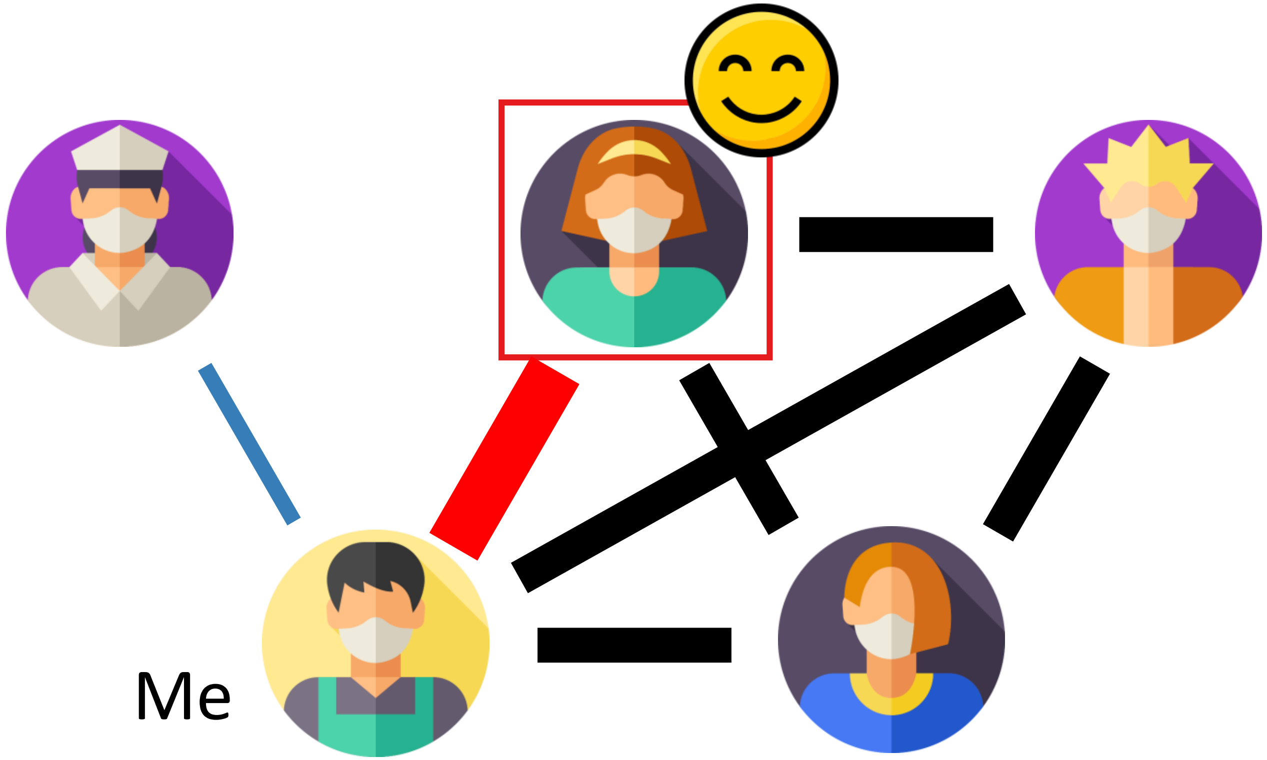
Topology and Edge Weights are Entangled

- In many cases, we can infer edge weights from topology (connections)
- **Example:** Online social networks
  - **Q:** Which person is more likely my close friend with strong connection?



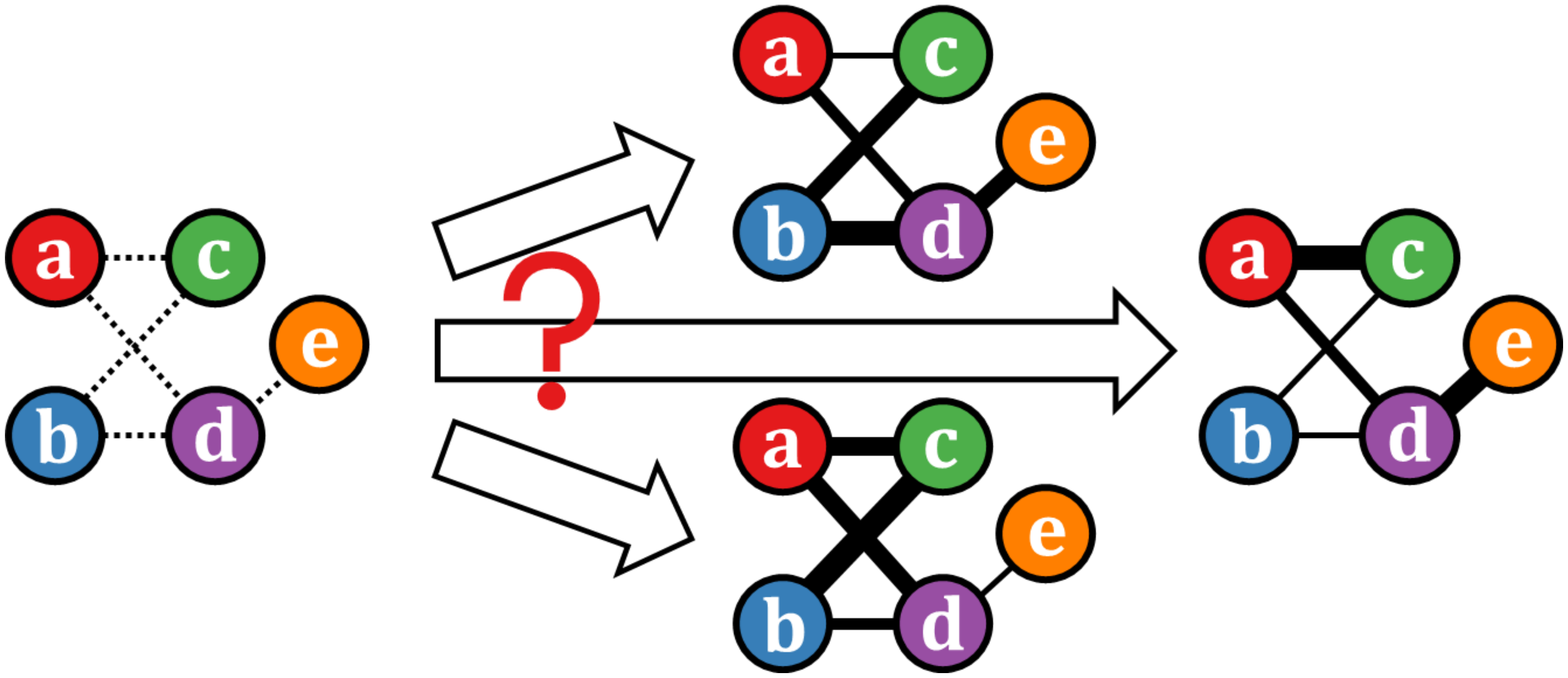
Topology and Edge Weights are Entangled

- In many cases, we can infer edge weights from topology (connections)
- **Example:** Online social networks
  - **Q:** Which person is more likely my close friend with strong connection?



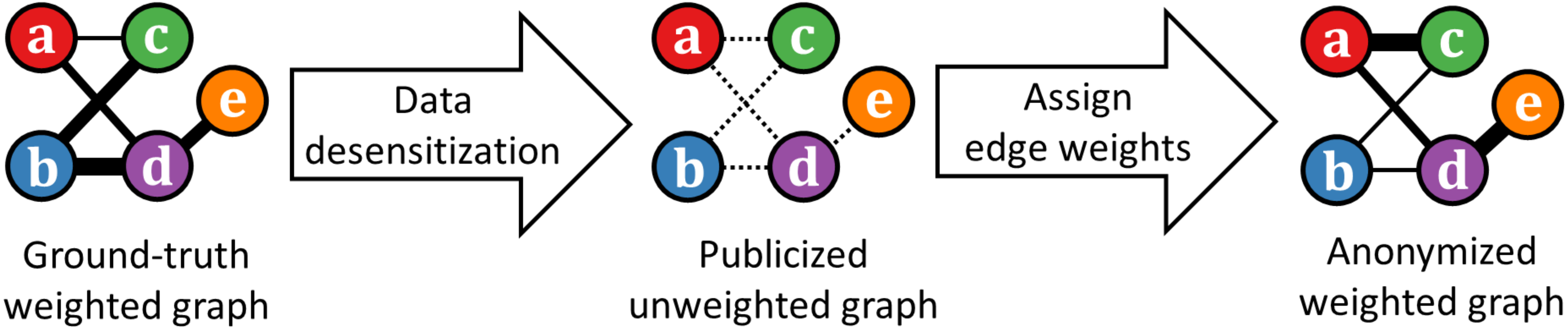
Research Questions

- **Q:** What are some realistic **properties** on the **interplay** between topology and edge weights that realistic edge weights should satisfy?
- **Q:** How can we assign **realistic** edge weights to given graph topology based on realistic properties?



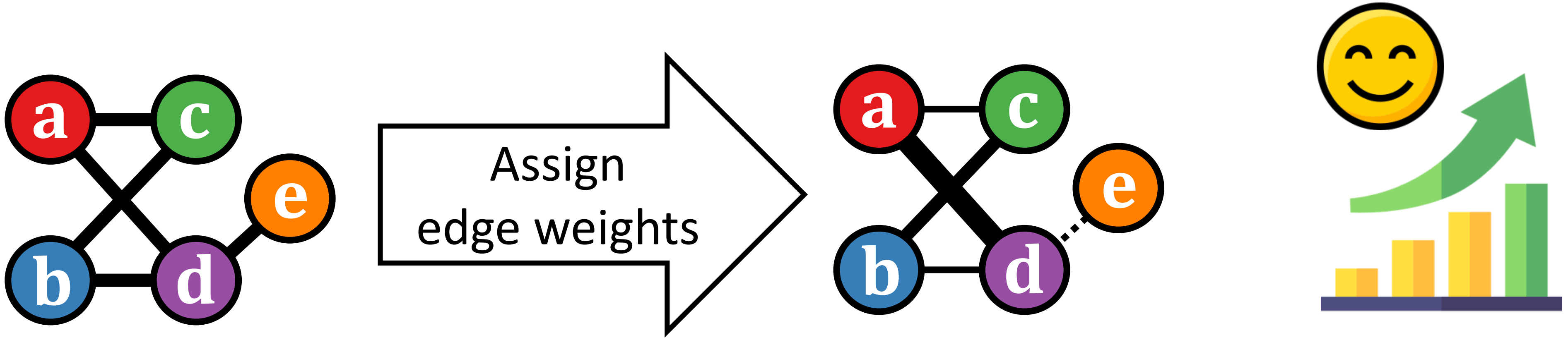
Real-World Applications

- Such a problem has several **real-world applications**
- **Edge weight anonymization:** Sometimes, due to privacy issues, connection information can be publicized but edge weights cannot
  - Assign **fake yet realistic** edge weights to **generate weighted benchmarks** without revealing the ground-truth edge weights



Real-World Applications

- Such a problem has several **real-world applications**
- **Community detection:** Edge weights provide additional information about the connections and thus are helpful for community detection
  - Assign **realistic** edge weights to unweighted graphs to **enhance the performance of community detection**



Available unweighted graph

Realistic weighted graph

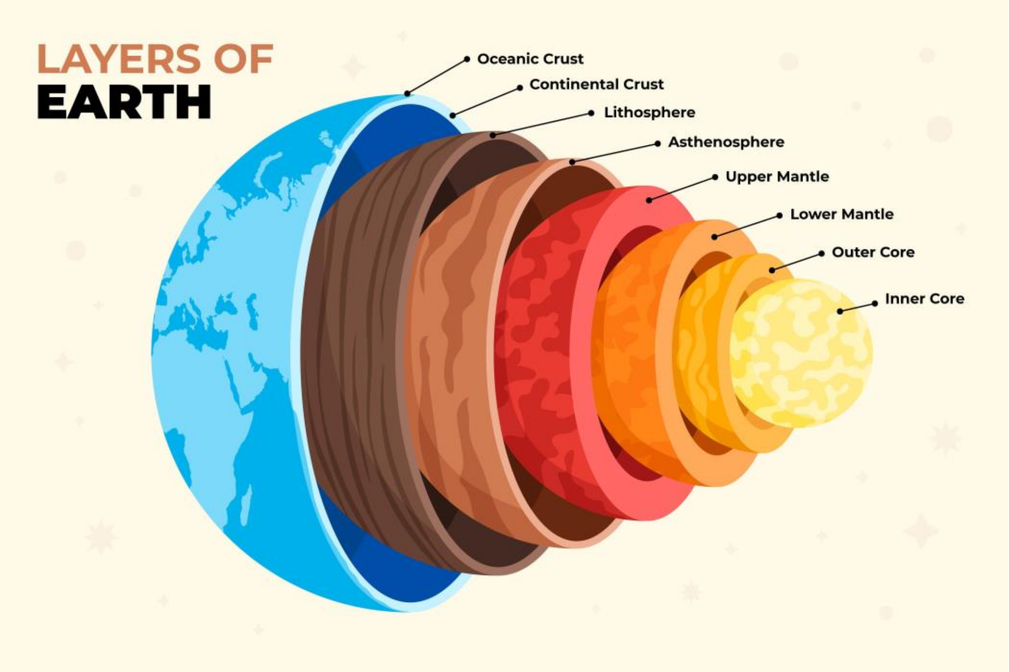
Improved community detection

Roadmap

- **Concepts <<**
- Patterns
- Algorithm & Experiments
- Conclusions

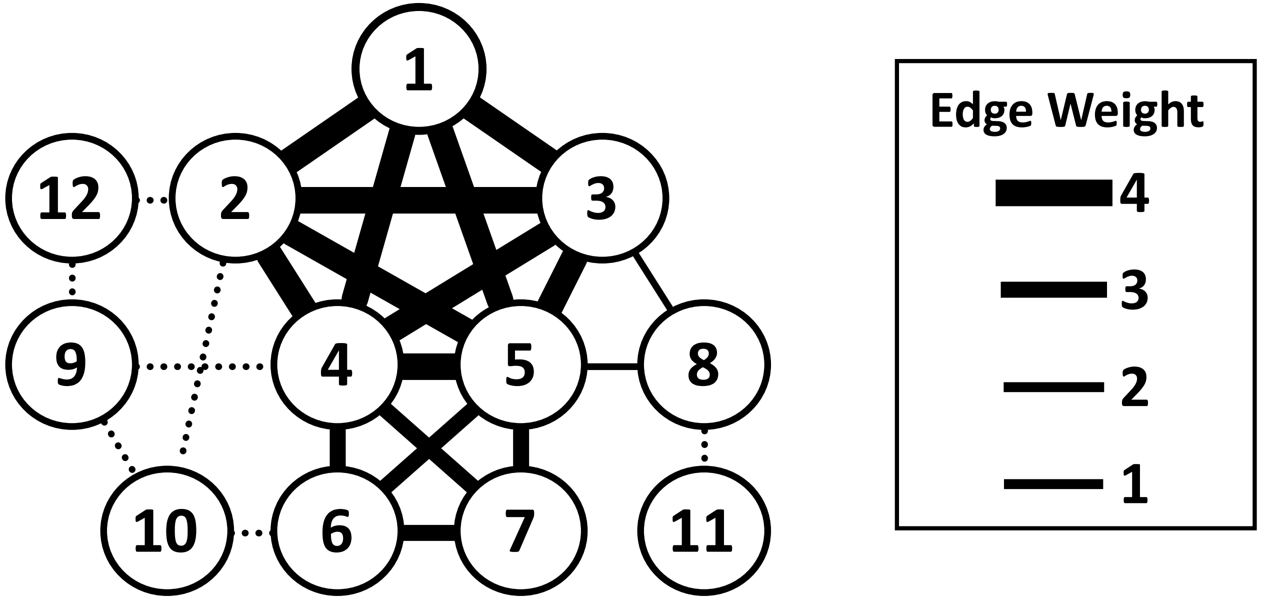
Layers

- The **layer-$\boldsymbol{i}$** of a graph G consists of the edges with edge weights $\geq i$
  - From layer-i, we can obtain layer-$(i+1)$ by removing the edges with weight i



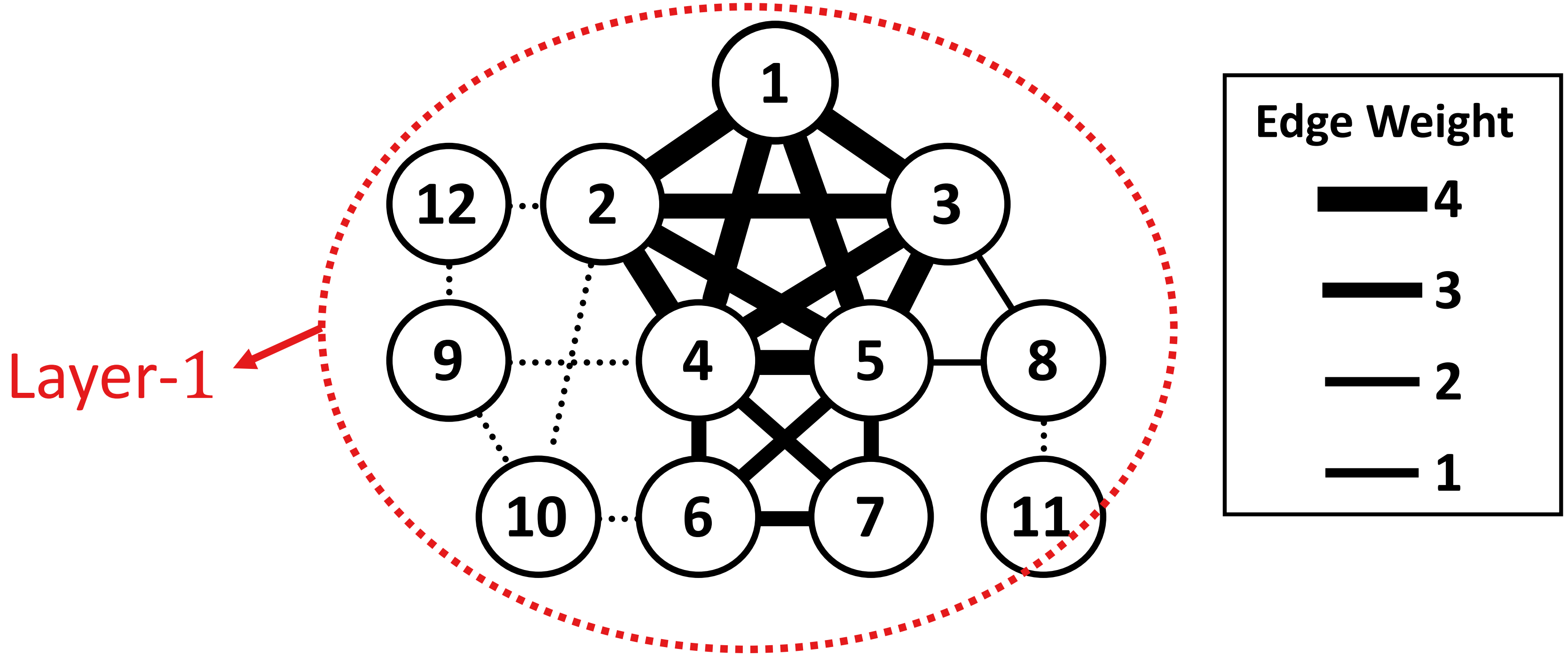
Layers: Example

- The **layer-$\boldsymbol{i}$** of a graph G consists of the edges with edge weights $\geq i$
  - The **layer-$\boldsymbol{i}$** of G is denoted by $G_i = (V_i, E_i, W_i)$



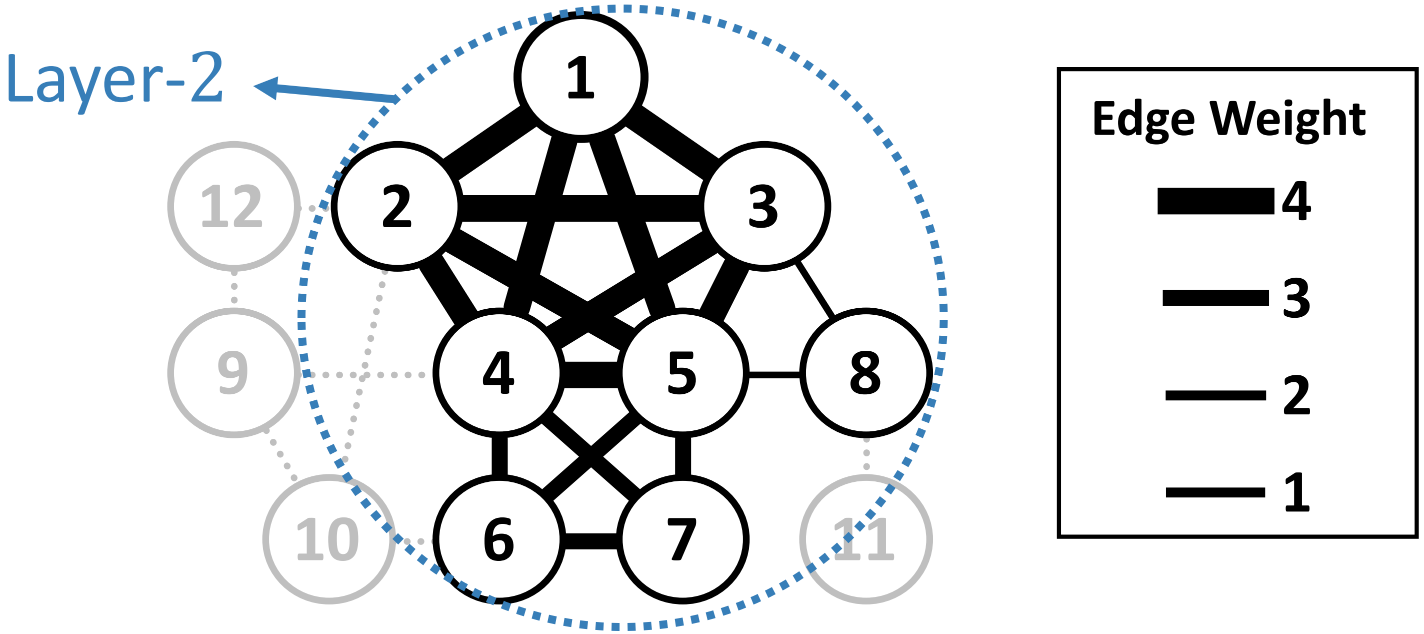
Layers: Example

- The **layer-i** of a graph G consists of the edges with edge weights $\geq i$
  - **Layer-1** is just the whole graph G



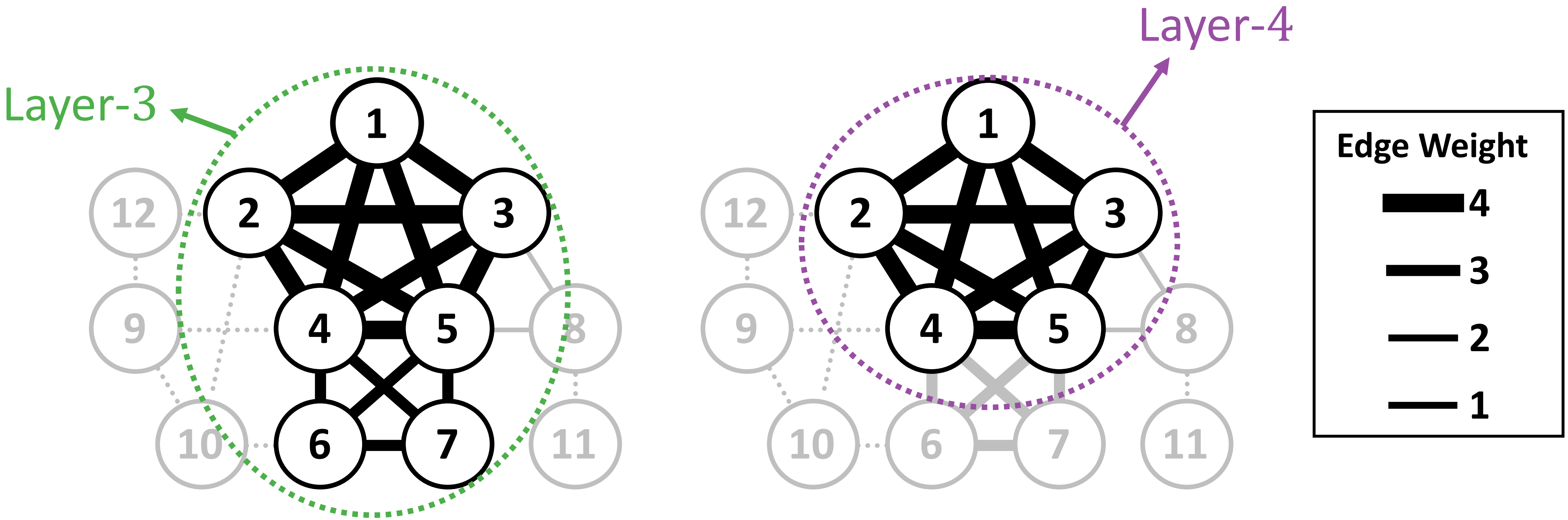
Layers: Example

- The **layer-i** of a graph G consists of the edges with edge weights $\geq i$
  - **Layer-2** can be obtained by removing the edges with weight 1



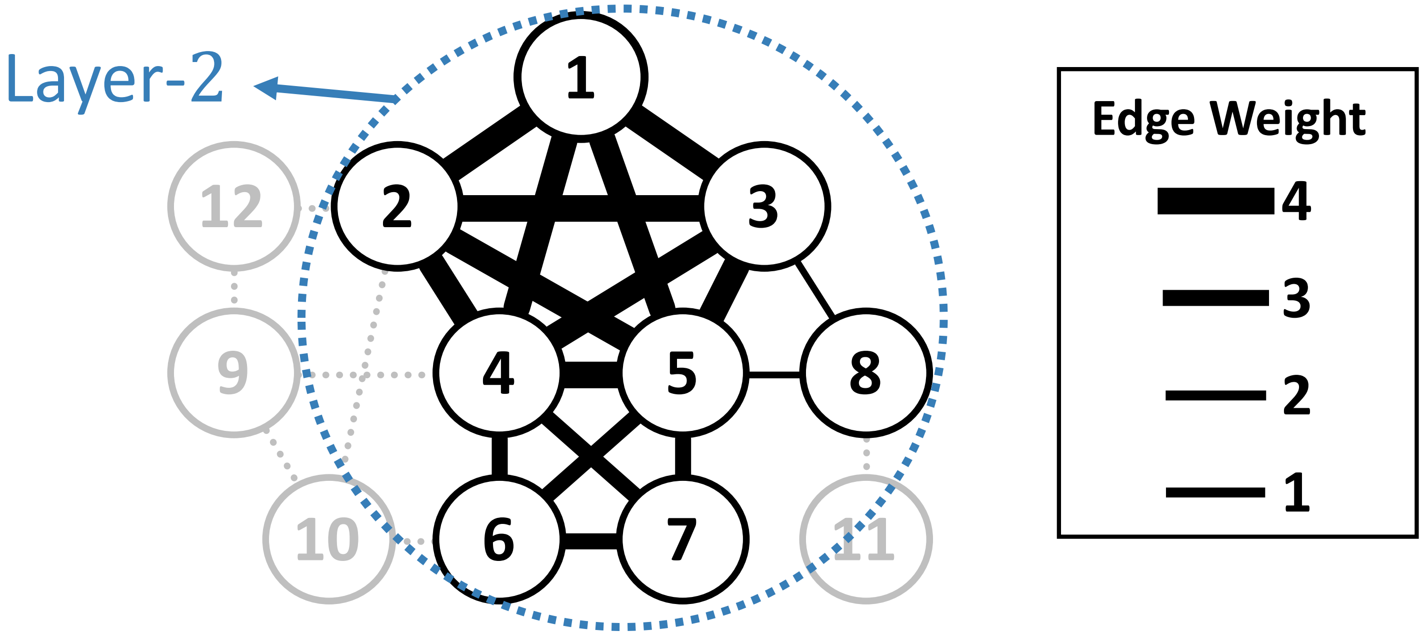
Layers: Example

- The **layer-$\boldsymbol{i}$** of a graph G consists of the edges with edge weights $\geq i$
  - Similarly, we can obtain **layer-3 and layer-4**



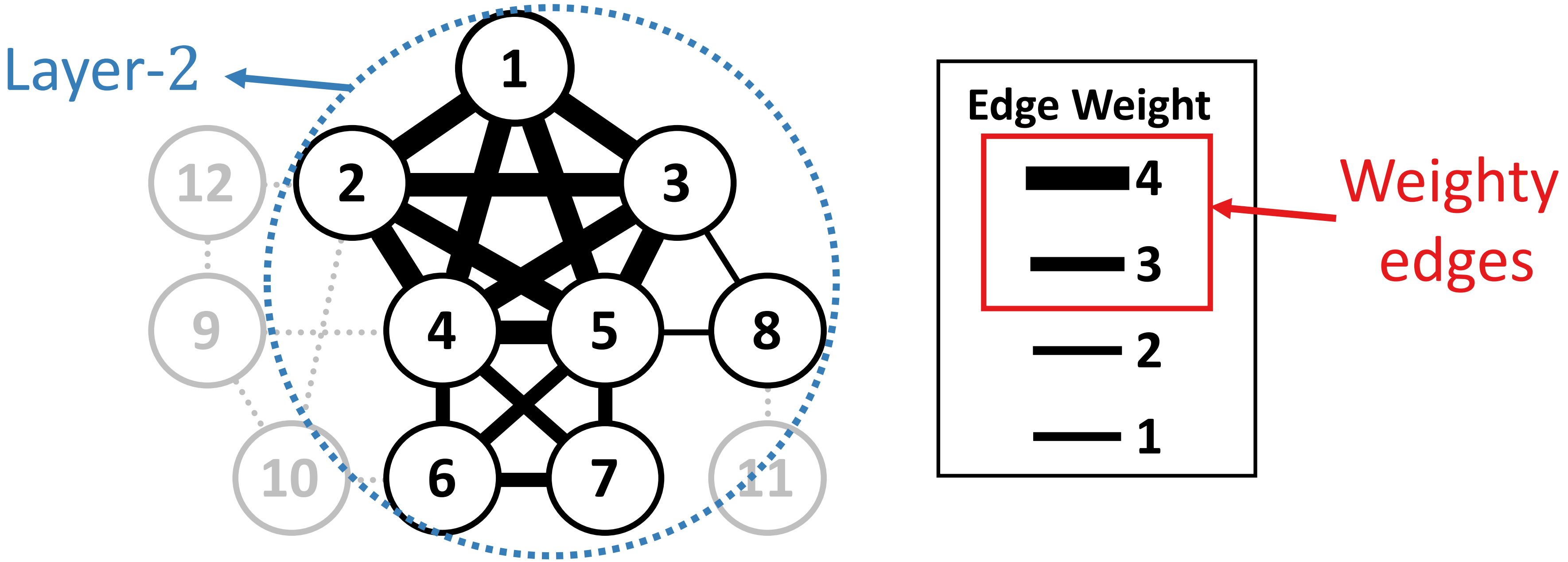
Weighty Edges

- The **layer-$\boldsymbol{i}$** of a graph G consists of the edges with edge weights $\geq i$
- The **weighty edges** in the layer-i are those with weight $> i$ (i.e., E_{i+1})



Weighty Edges

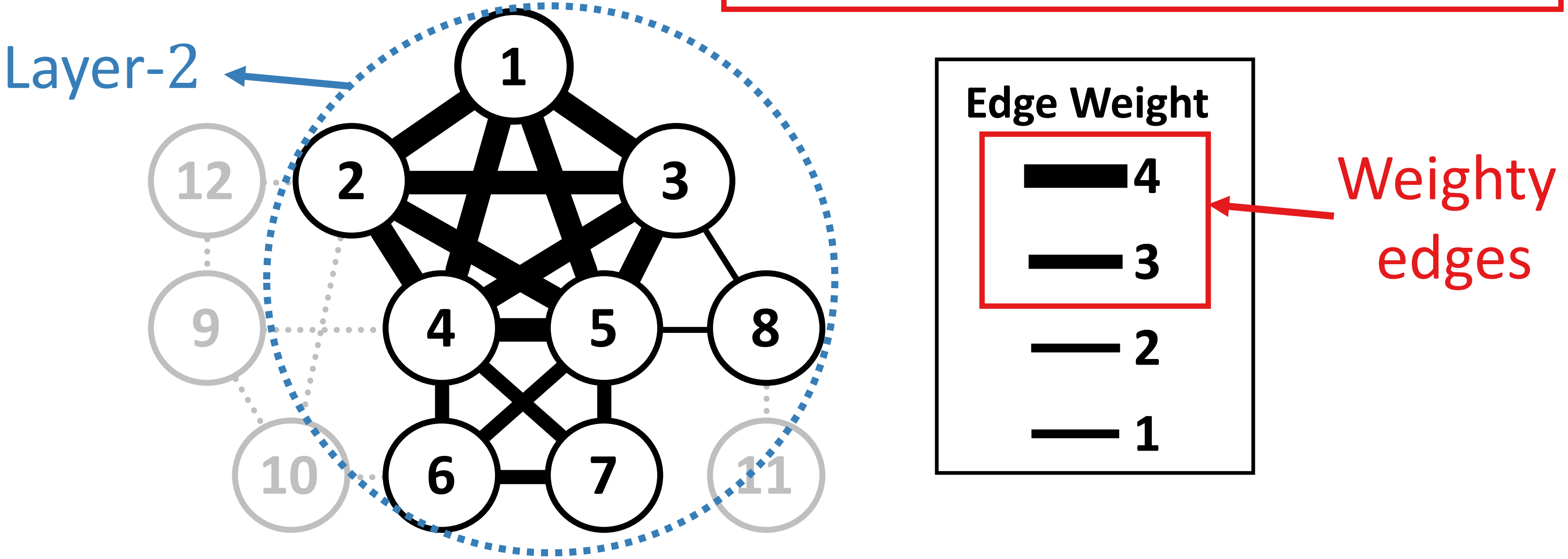
- The **layer-i** of a graph G consists of the edges with edge weights $\geq i$
- The **weighty edges** in the layer-i are those with weight $> i$ (i.e., E_{i+1})



Overall Fractions of Weighty Edges

- We also define the **overall fraction** of weighty edges in the layer-i
  - $f_{overall;i}(G) = |E_{i+1}|/|E_i|$

$$f_{overall;2}(G) = |E_3|/|E_2| = 15/17$$



Roadmap

- Concepts
- **Patterns <<**
- Algorithm & Experiments
- Conclusions

Real-World Weighted Graph Datasets

- **Datasets:** 11 real-world weighted graphs from 5 different domains
  - # nodes: 897 – 2.6M
  - # edges: 15,645 – 28.2M
- **Source:** https://toreopsahl.com/tnet/; https://www.cs.cornell.edu/~arb/data; https://snap.stanford.edu/data/

dataset	$\lvert V\rvert$	$\lvert E\rvert = \lvert E_1\rvert$	$\lvert E_2\rvert$	$\lvert E_3\rvert$	$\lvert E_4\rvert$
OF	897	71,380	47,266 (66.2%)	35,456 (49.7%)	28,546 (40.0%)
FL	2,905	15,645	4,608 (29.5%)	1,507 (9.6%)	564 (3.6%)
th-UB	82,075	182,648	7,297 (4.0%)	2,090 (1.1%)	965 (0.5%)
th-MA	152,702	1,088,735	128,400 (11.8%)	48,605 (4.5%)	26,121 (2.4%)
th-SO	2,301,070	20,989,078	1,168,210 (5.6%)	350,871 (1.7%)	170,618 (0.8%)
sx-UB	152,599	453,221	135,948 (30.0%)	56,115 (12.4%)	28,029 (6.2%)
sx-MA	24,668	187,939	74,493 (39.6%)	36,604 (19.5%)	21,364 (11.4%)
sx-SO	2,572,345	28,177,464	9,871,784 (35.0%)	4,137,454 (14.7%)	2,055,034 (7.3%)
sx-SU	189,191	712,870	216,296 (30.3%)	82,475 (11.6%)	37,655 (5.3%)
co-DB	1,654,109	7,713,116	2,269,679 (29.4%)	1,085,489 (14.1%)	654,182 (8.5%)
co-GE	898,648	4,891,112	1,055,077 (21.6%)	446,833 (9.1%)	246,944 (5.1%)

- **OF:** communication in a blog post
- **FL:** flights between airports
- **th:** interactions within threads
- **sx:** Q&A interactions
- **co:** co-authorship

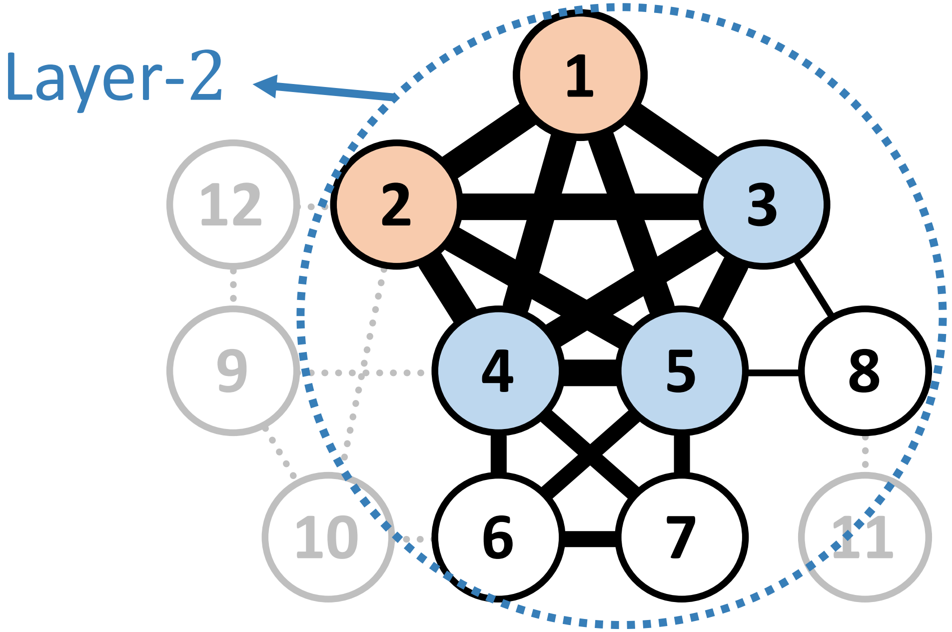
Common Neighbors are Simple and Indicative

- The number of common neighbors is **simplest** and it has the **strongest correlation** with edge weightiness overall!
  - **Edge weightiness:** Specifically, it is 1 if edge weight > 1, 0 otherwise

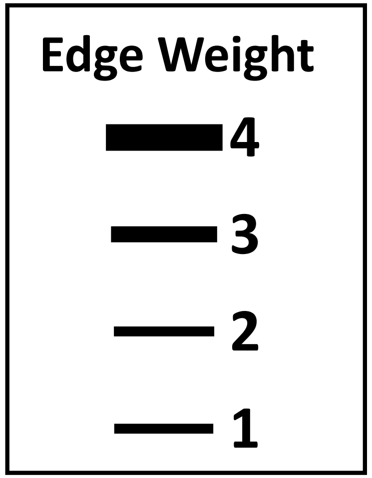
dataset	NC	SA	JC	HP	HD	SI	LI	AA	RA	PA	FM	DL	EC	LP
OF	0.33	0.20	0.20	-0.02	0.21	0.21	-0.13	**0.34**	**0.35**	0.33	0.11	0.32	0.26	0.33
FL	0.32	0.26	0.26	0.19	0.24	0.26	-0.06	**0.35**	**0.35**	0.21	0.08	0.18	0.17	0.31
th-UB	**0.48**	0.02	0.00	0.03	0.00	0.01	-0.05	0.47	0.40	0.33	0.26	0.21	0.37	**0.48**
th-MA	**0.45**	0.22	0.15	0.09	0.15	0.18	-0.05	0.44	0.35	0.33	0.40	0.25	0.38	**0.46**
th-SO	**0.38**	0.11	0.08	0.06	0.08	0.09	-0.03	**0.39**	0.33	0.22	0.33	0.18	0.26	0.37
sx-UB	**0.15**	0.11	0.08	0.09	0.07	0.08	-0.00	0.13	0.10	0.09	0.12	0.09	0.14	**0.15**
sx-MA	**0.25**	0.24	0.21	0.12	0.19	0.21	-0.02	**0.25**	0.22	0.19	0.19	0.16	0.20	0.25
sx-SO	0.10	**0.11**	0.08	0.08	0.07	0.08	0.00	0.10	0.07	0.05	0.10	0.07	0.07	**0.10**
sx-SU	**0.14**	0.11	0.08	0.09	0.07	0.08	-0.00	0.12	0.08	0.08	0.11	0.08	0.13	**0.15**
co-DB	**0.20**	-0.08	-0.09	-0.05	-0.08	-0.08	-0.16	**0.22**	0.20	0.03	0.06	0.07	0.14	0.20
co-GE	**0.30**	-0.07	-0.08	-0.08	-0.07	-0.06	-0.16	**0.32**	0.26	0.16	0.19	0.19	0.22	0.30
avg.	**0.28**	0.11	0.09	0.05	0.09	0.10	-0.06	**0.28**	0.24	0.18	0.18	0.16	0.21	0.28
avg. rank	**2.2**	7.5	10.5	10.6	10.8	8.9	14.0	2.5	5.5	8.2	7.1	8.5	6.5	**2.4**

(Local) Fractions of Weighty Edges

- The overall fraction of weighty edges is $f_{overall;i}(G) = |E_{i+1}|/|E_i|$
- Let $E_{c;i} \subseteq E_i$ be the set of the edges in E_i whose two endpoints share exactly c common neighbors
  - A disjoint partition of E_i

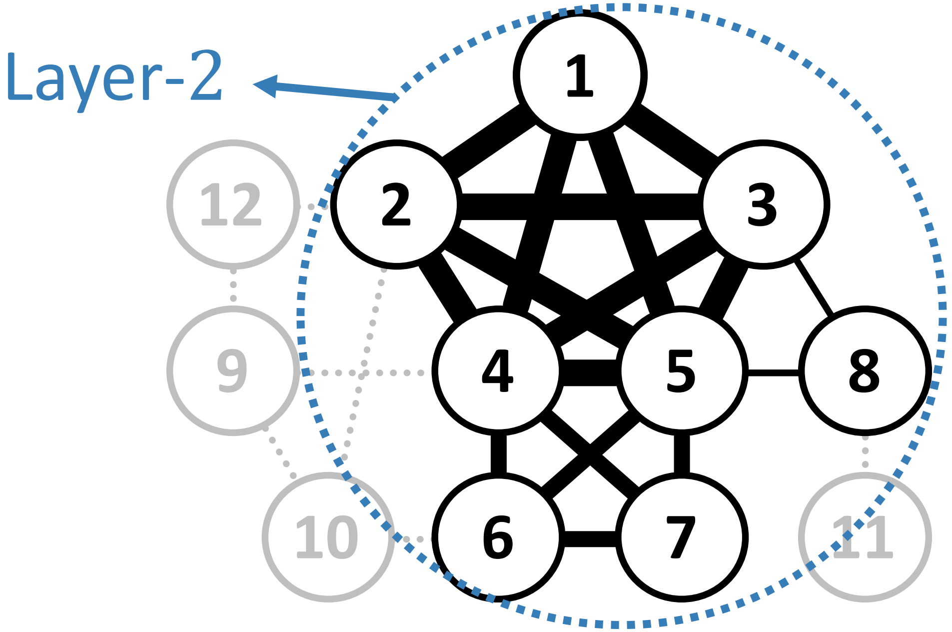


Nodes 1 and 2:
Share 3 common neighbors
(nodes 3, 4, and 5)
So the edge $(1,2) \in E_{3;2}$

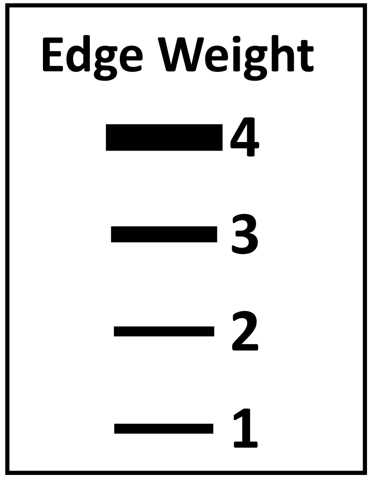


(Local) Fractions of Weighty Edges

- The overall fraction of weighty edges is $f_{overall;i}(G) = |E_{i+1}|/|E_i|$
- Let $E_{c;i} \subseteq E_i$ be the set of the edges in E_i whose two endpoints share exactly c common neighbors
  - The **fraction of weighty edges (FoWE)** is $f_{c;i} = \left|E_{c;i} \cap E_{i+1}\right|/|E_{c;i}|$

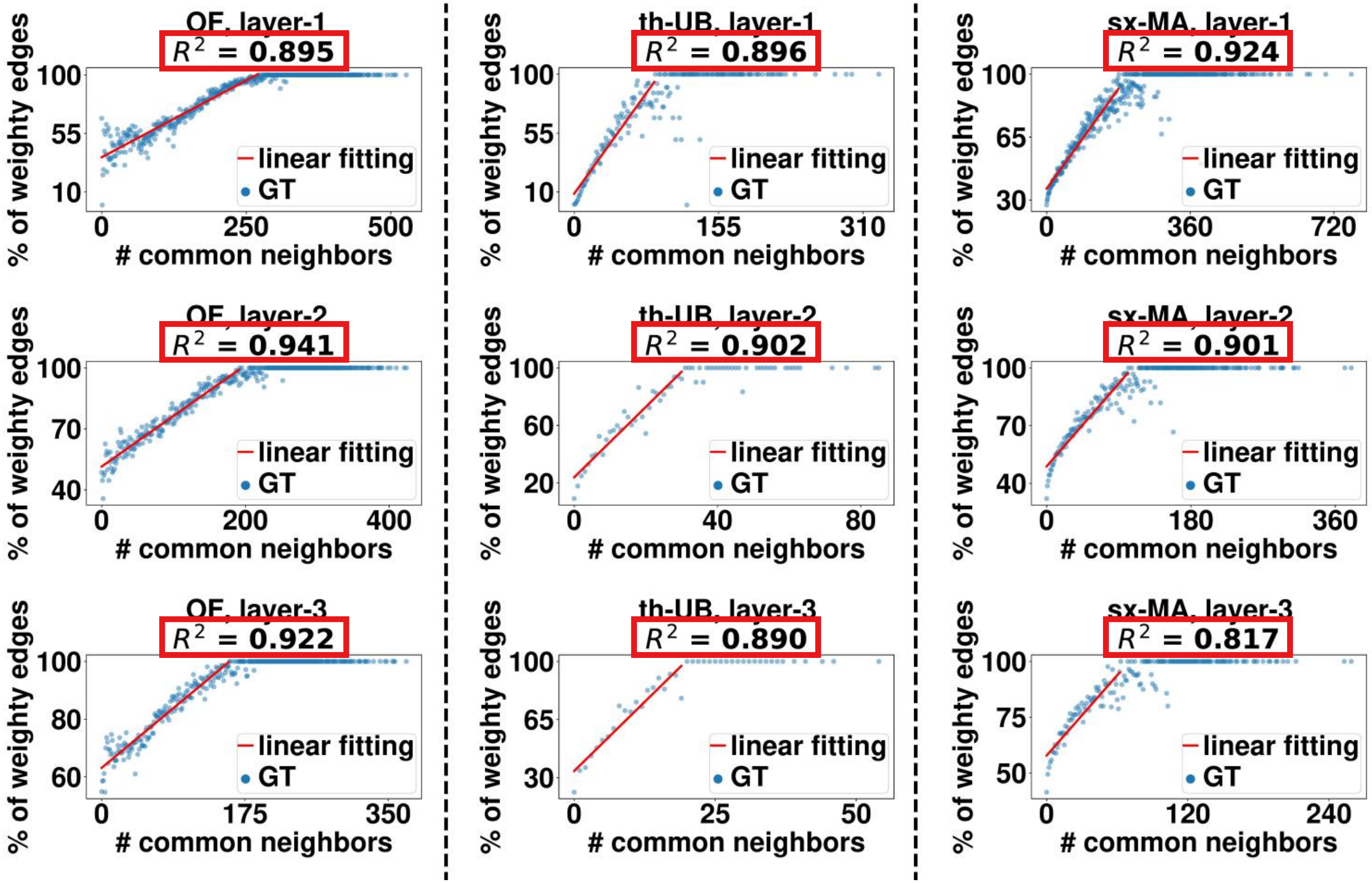


$E_{2;2}$ consists of $\{(4,6),(4,7),(5,7),(6,7)\}$
All are weighty edges!
Therefore, $f_{2;2} = 1$



Pattern 1: Linear Growth of FoWE

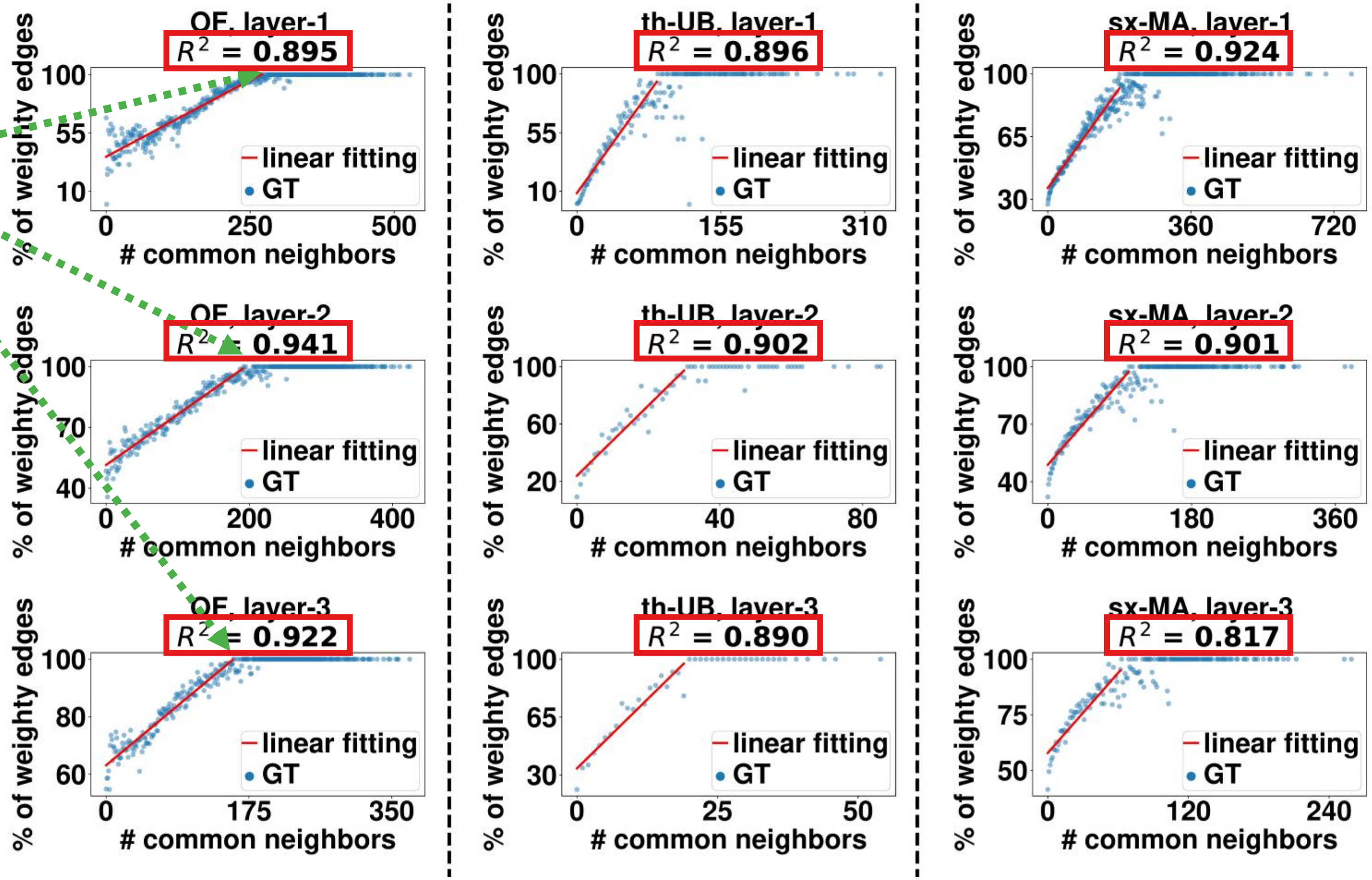
- **Pattern:** In each layer-i, the fraction of weighty edge (FoWE) $f_{c;i}$ **grows nearly linearly** with c, and **saturates** after some point



Pattern 1: Linear Growth of FoWE

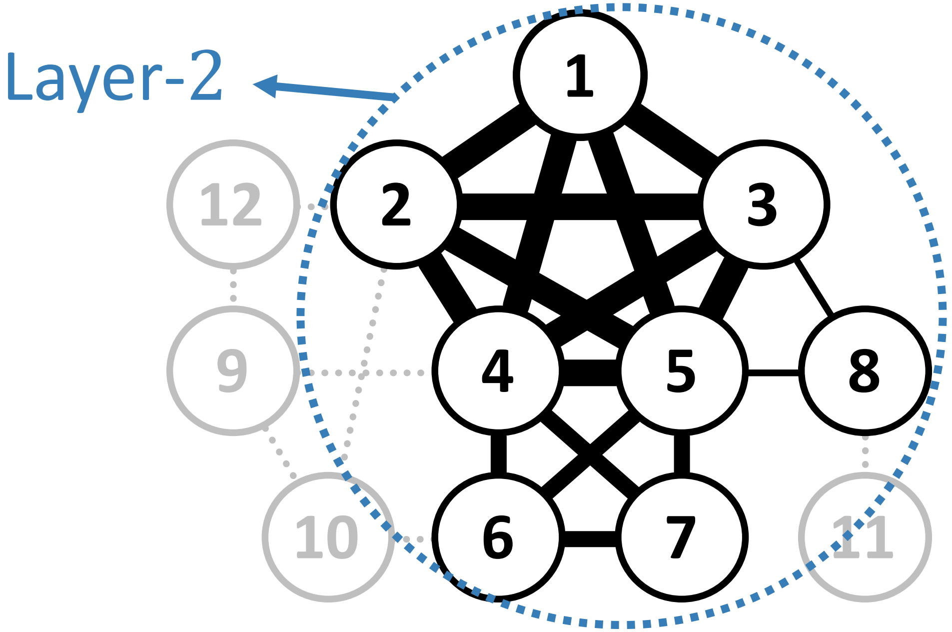
- **Pattern:** In each layer-i, the fraction of weighty edge (FoWE) $f_{c;i}$ **grows nearly linearly** with c, and **saturates** after some point

Saturation points

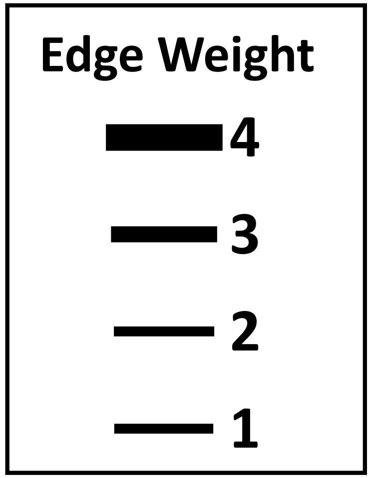


Pattern 2: Fraction of Adjacent Pairs

- **Recall:** The fraction of weighty edges $f_{c;i} = |E_{c;i} \cap E_{i+1}| / |E_{c;i}|$
- The **fraction of adjacent pairs** $\tilde{f}_{c;i} = |E_{c;i}| / |R_{c;i}|$, where $R_{c;i}$ is the set of **pairs** sharing exactly c common neighbors in the layer-i
  - Both adjacent and non-adjacent pairs are counted in $R_{c;i}$

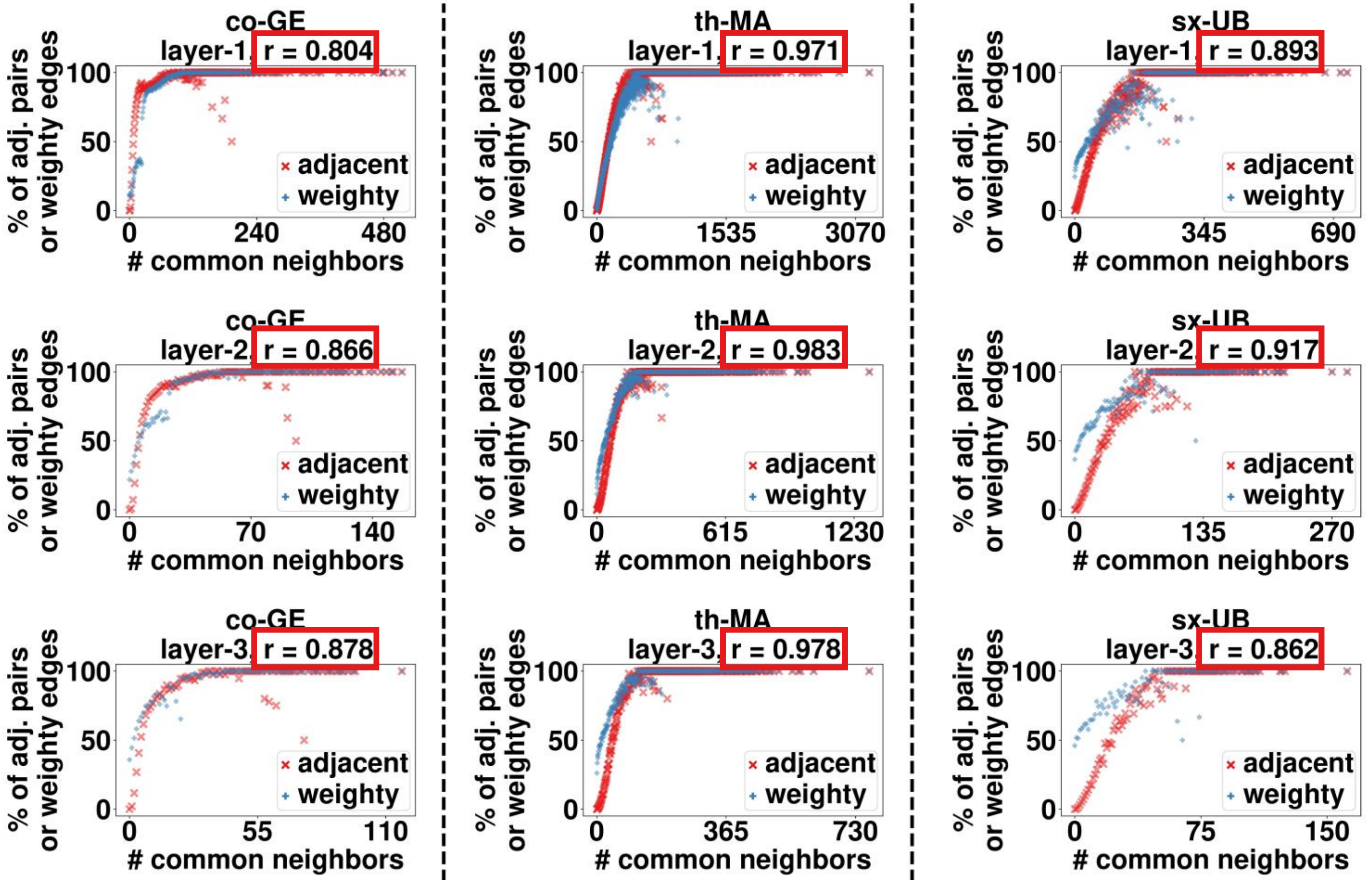


$R_{1;2}$ consists of
$\{(1,8), (3,8), (5,8), (6,8), (7,8)\}$
$E_{1;2}$ consists of
$\{(3,8), (5,8)\}$
$\Rightarrow \tilde{f}_{1;2} = |E_{1;2}| / |R_{1;2}| = 2/5$



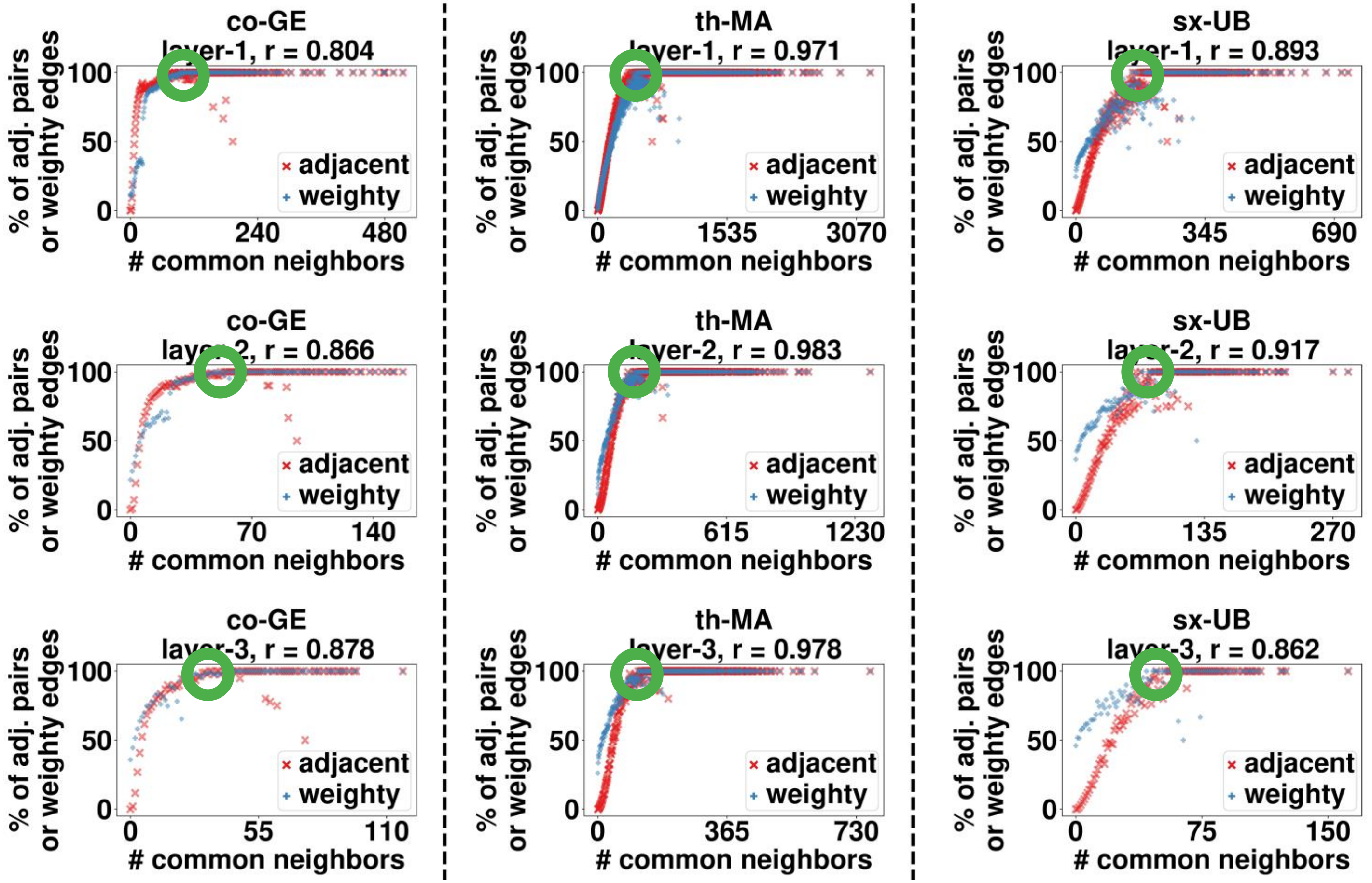
Pattern 2: Similarity between Adjacency and Weightiness

- **Pattern:** In each layer-i, the fraction of weighty edge $f_{c;i}$ and the fraction of adjacent pairs $\tilde{f}_{c;i}$ have **high correlations (similar trends)**



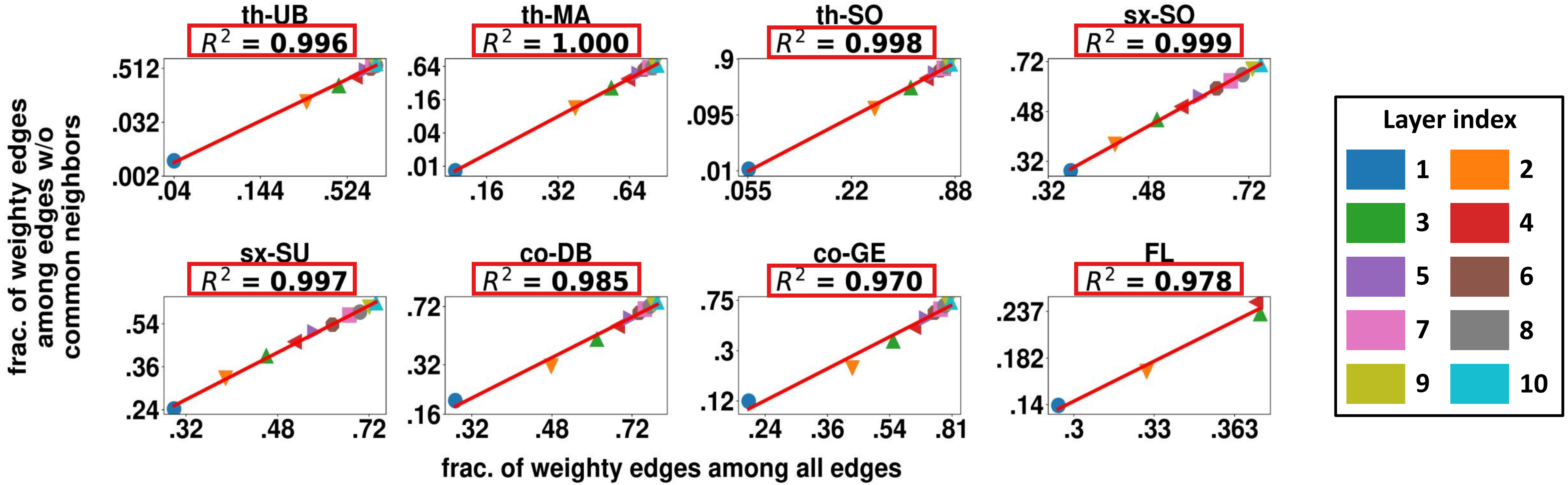
Pattern 2: Similarity between Adjacency and Weightiness

- **Pattern:** In each layer-i, the fraction of weighty edge $f_{c;i}$ and the fraction of adjacent pairs $\tilde{f}_{c;i}$ also have **similar saturation points**



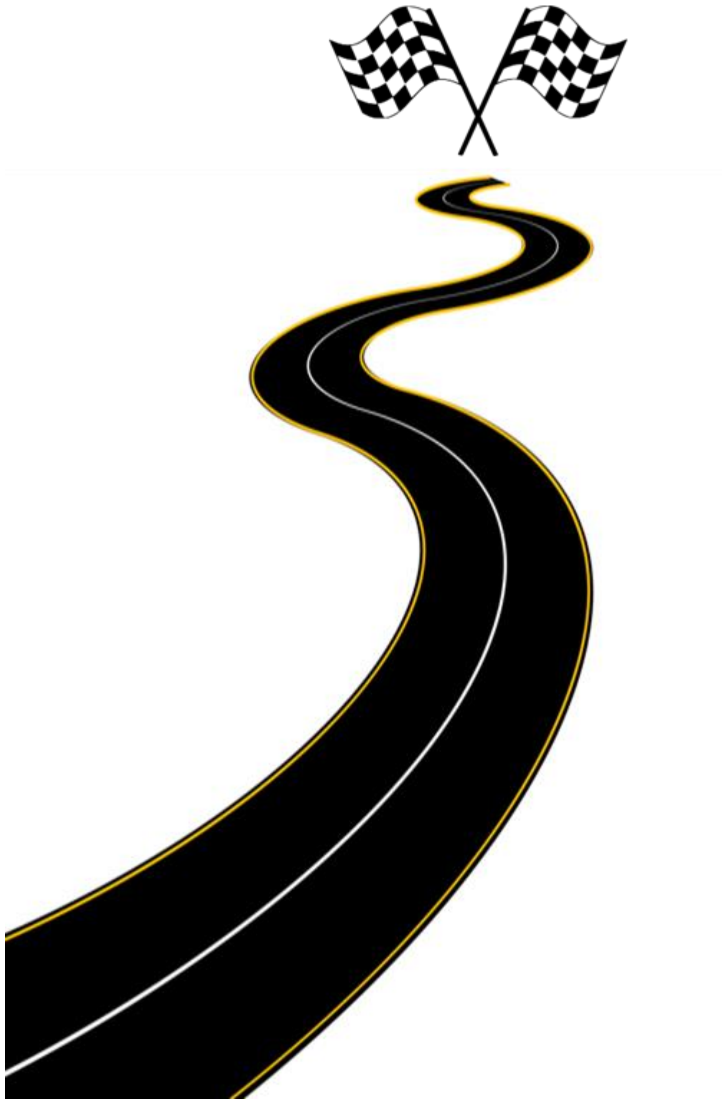
Pattern 3: A Power Law across Layers

- **Q:** Are there also some patterns across different layers?
- **Pattern:** **Across layers**, the overall fractions of weighty edges $f_{overall;i} = |E_{i+1}|/|E_i|$ and the $f_{0;i}$'s exhibit a **strong power law**
  - $f_{0;i}$: the fraction of weighty edges among those without common neighbors



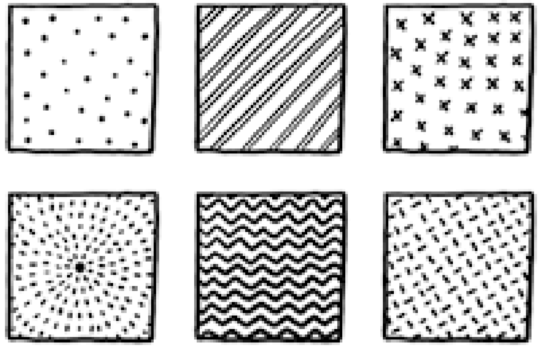
Roadmap

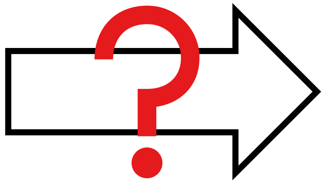
- Concepts
- Patterns
- **Algorithm & Experiments <<**
- Conclusions

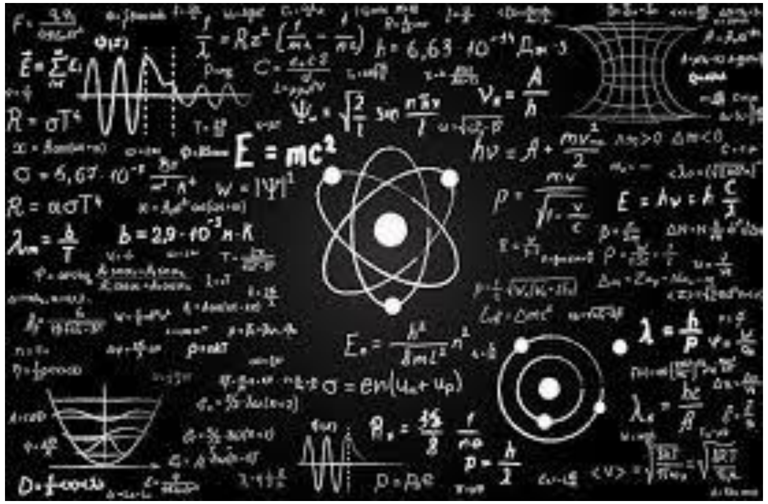


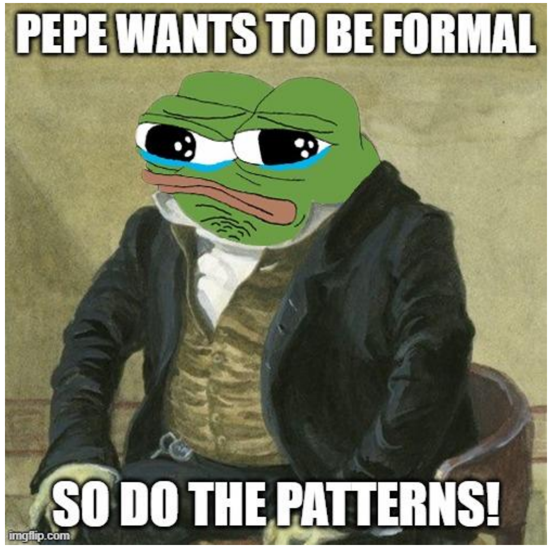
Formalize the Patterns

- **Q:** How can we propose an algorithm to assign **realistic** edge weights based on the patterns we observed?
- Let us **mathematically formalize** the patterns!



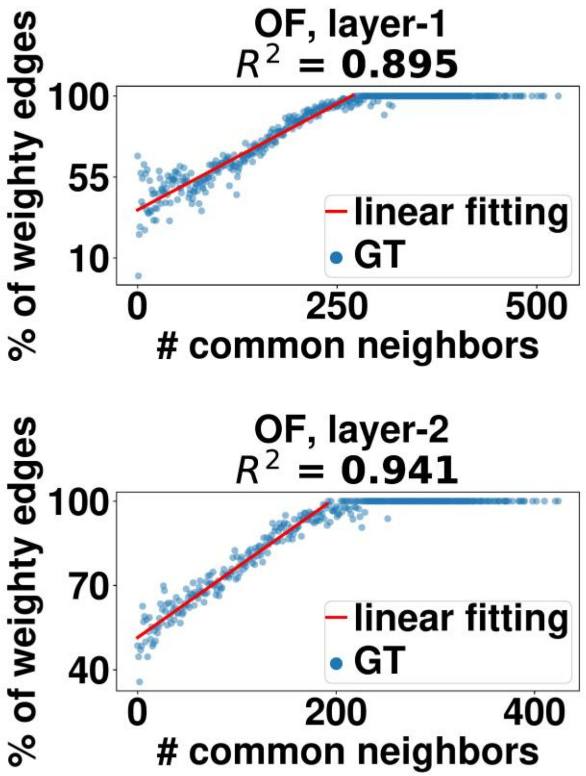


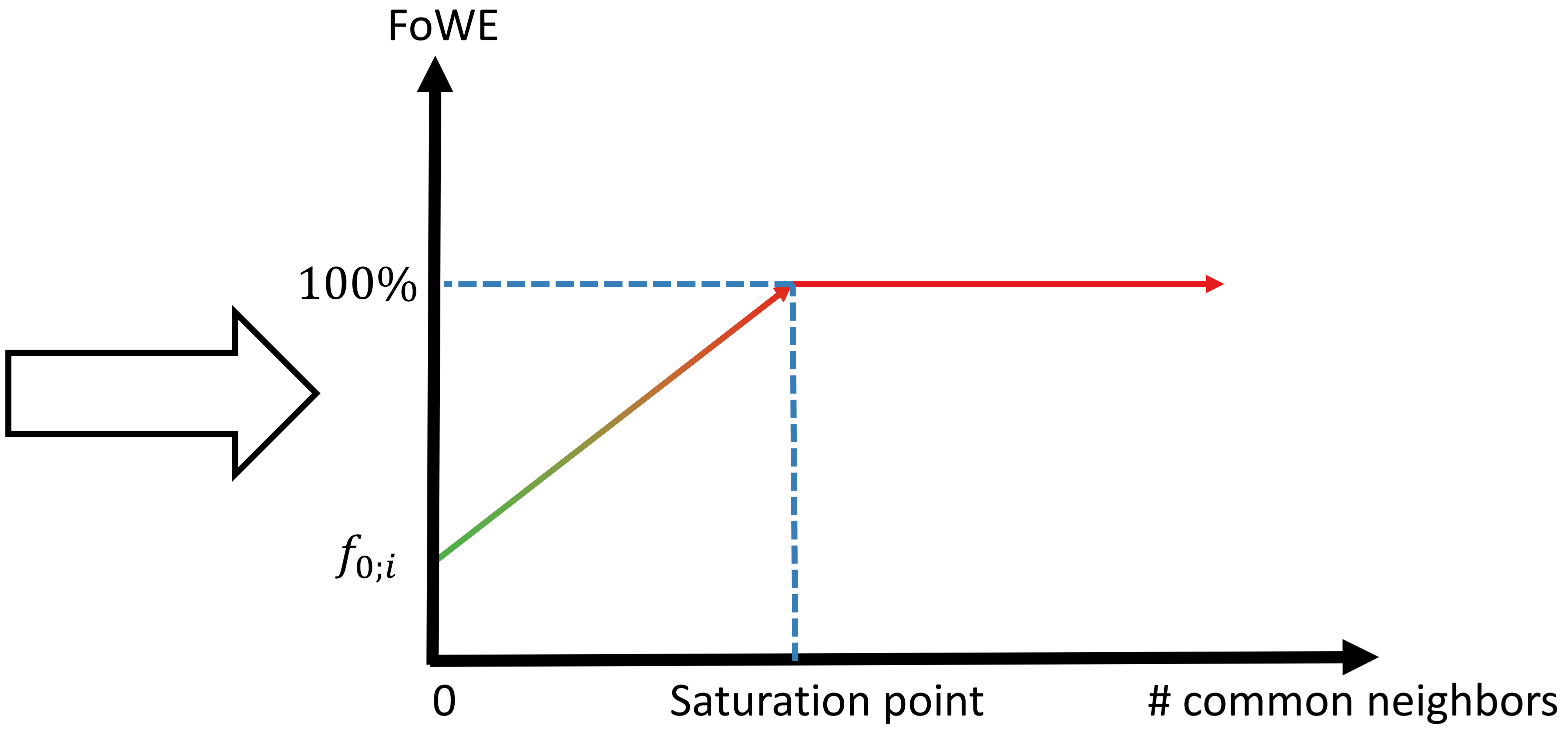




Formalize the Patterns: Pattern 1

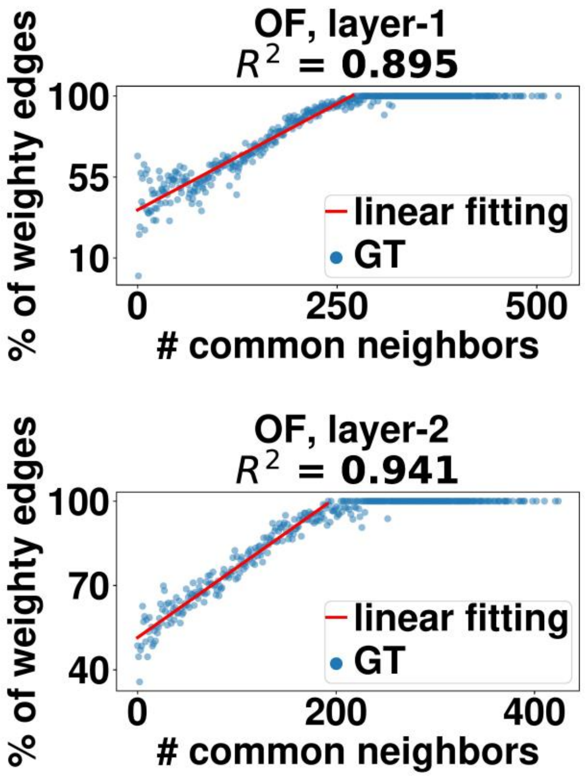
- **Pattern:** In each layer-i, the fraction of weighty edge (FoWE) $f_{c;i}$ **grows nearly linearly** with c, and saturates after some point
- **Idealization:** Perfect linearity

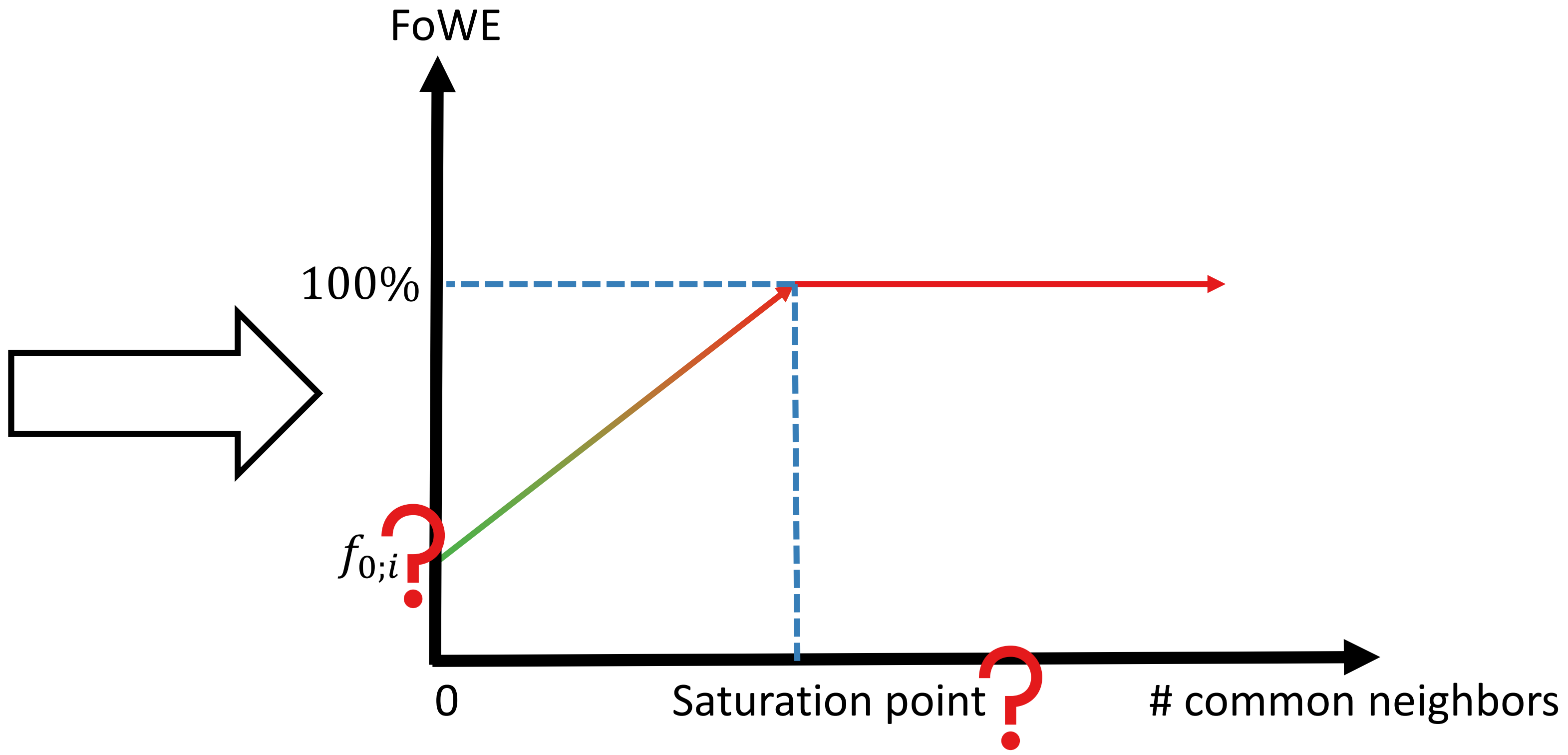




Formalize the Patterns: Pattern 1

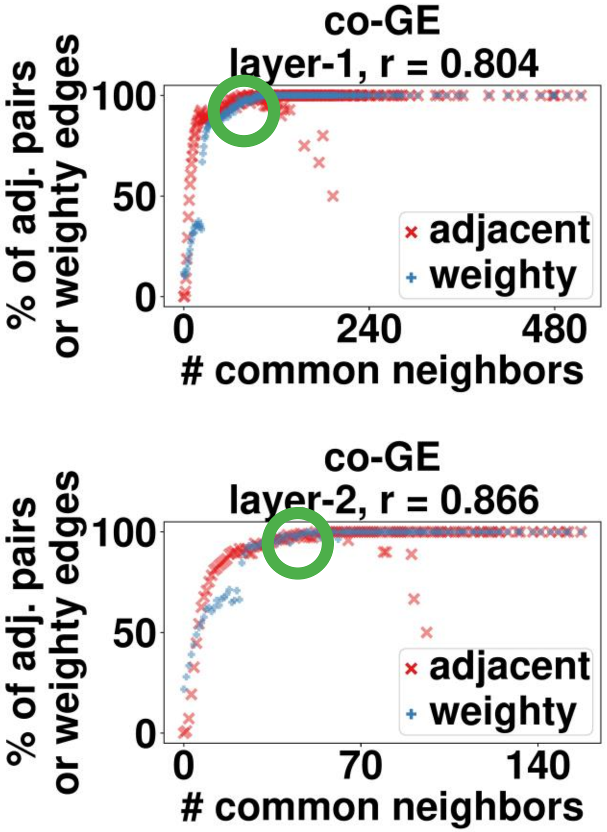
- **Pattern:** In each layer-i, the fraction of weighty edge (FoWE) $f_{c;i}$ **grows nearly linearly** with c, and **saturates** after some point
- **Idealization:** Perfect linearity

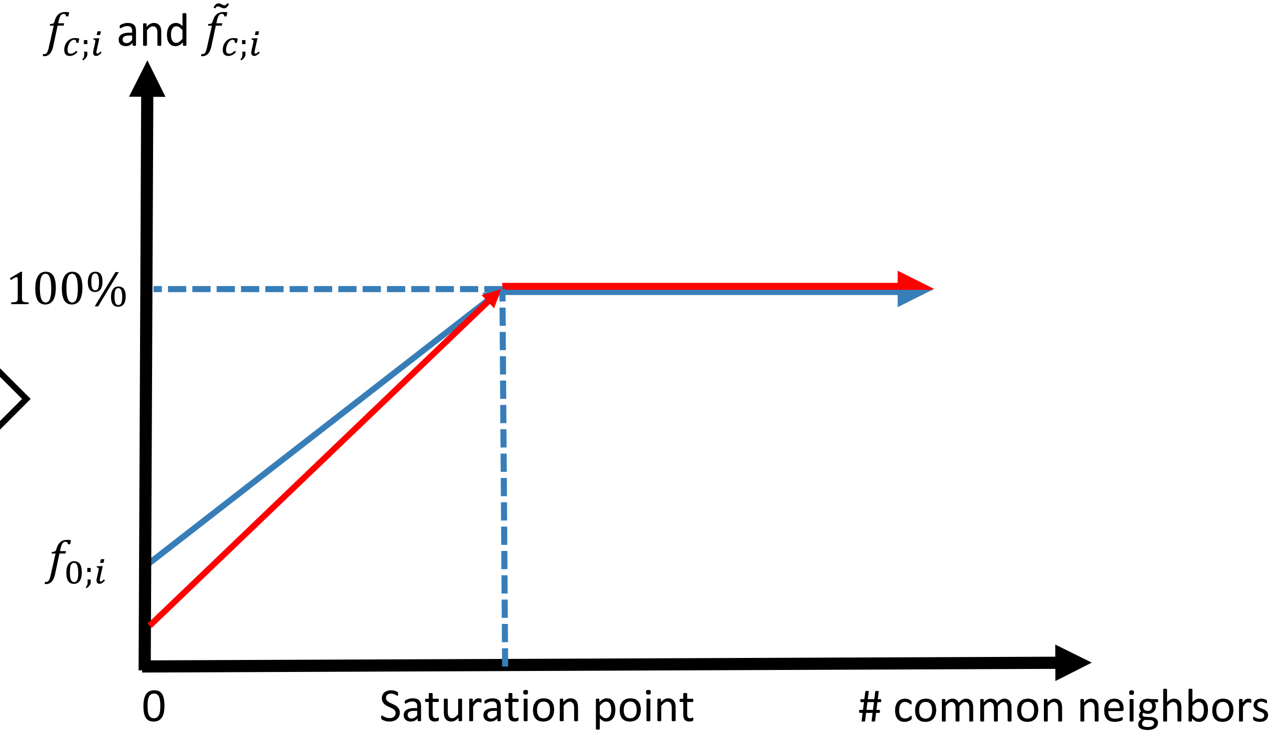




Formalize the Patterns: Pattern 2

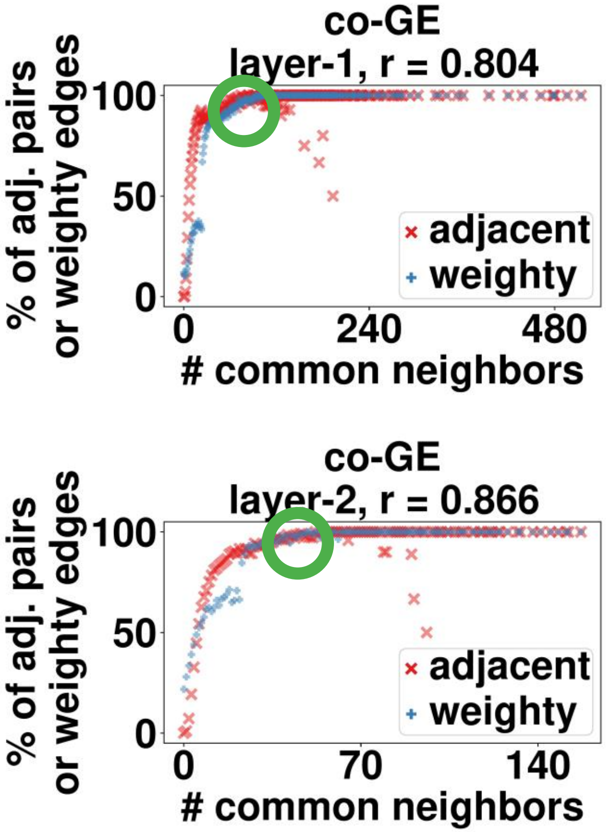
- **Pattern:** In each layer-i, the fraction of weighty edge $f_{c;i}$ and the fraction of adjacent pairs $\tilde{f}_{c;i}$ also have **similar saturation points**
- **Idealization:** Exactly the same saturation point

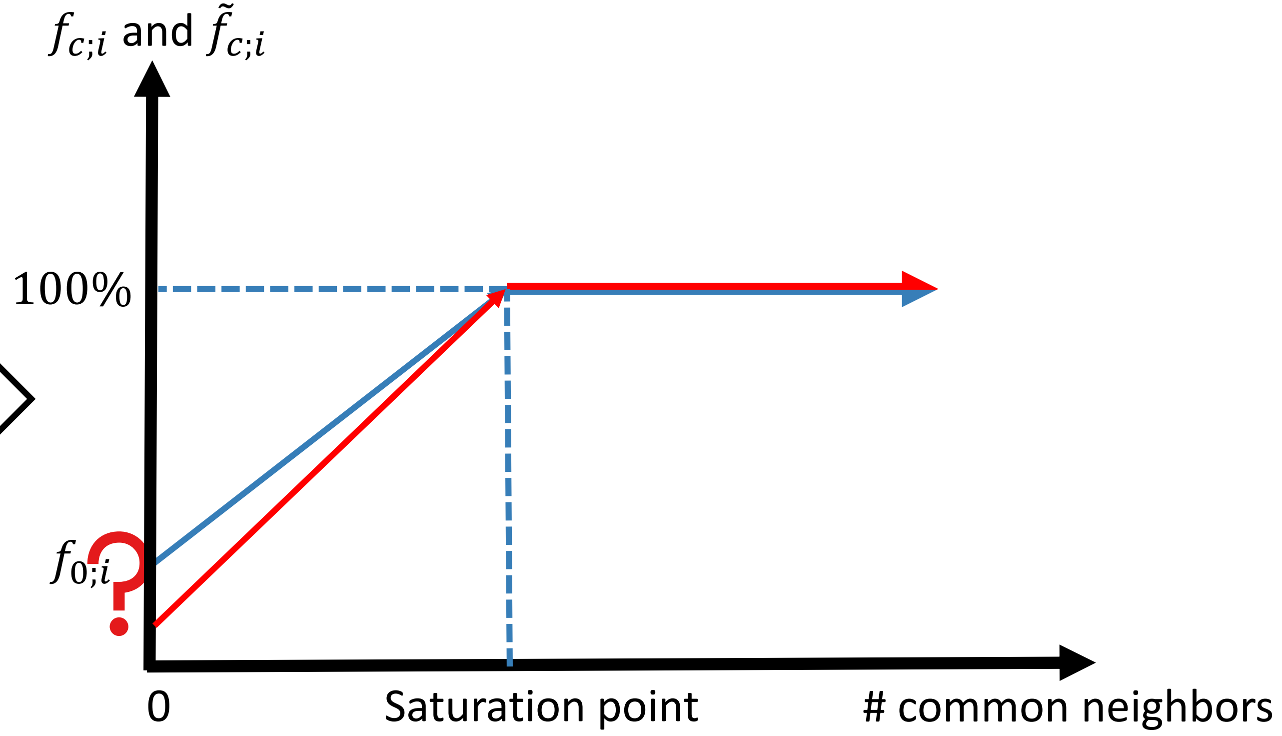




Formalize the Patterns: Pattern 2

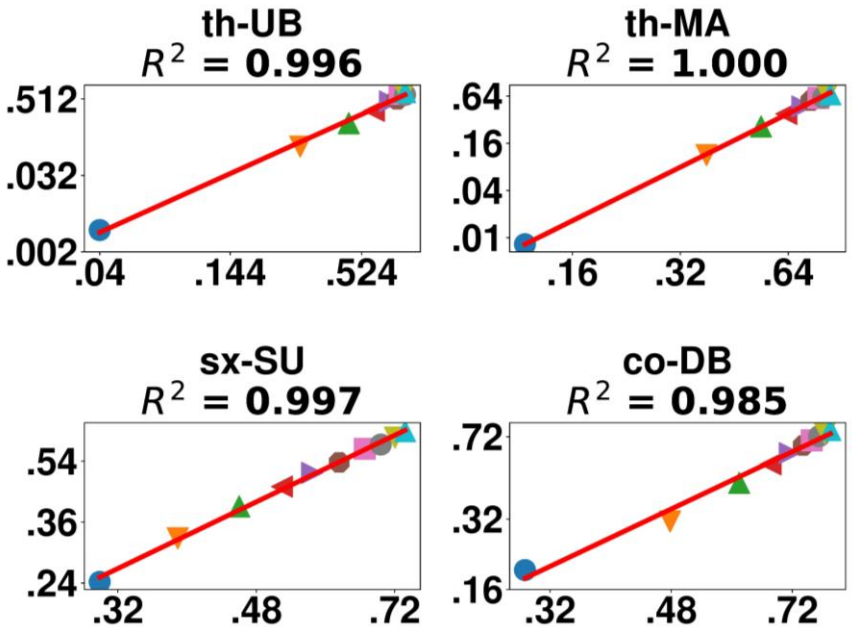
- **Pattern:** In each layer-i, the fraction of weighty edge $f_{c;i}$ and the fraction of adjacent pairs $\tilde{f}_{c;i}$ also have **similar saturation points**
- **Idealization:** Exactly the same saturation point

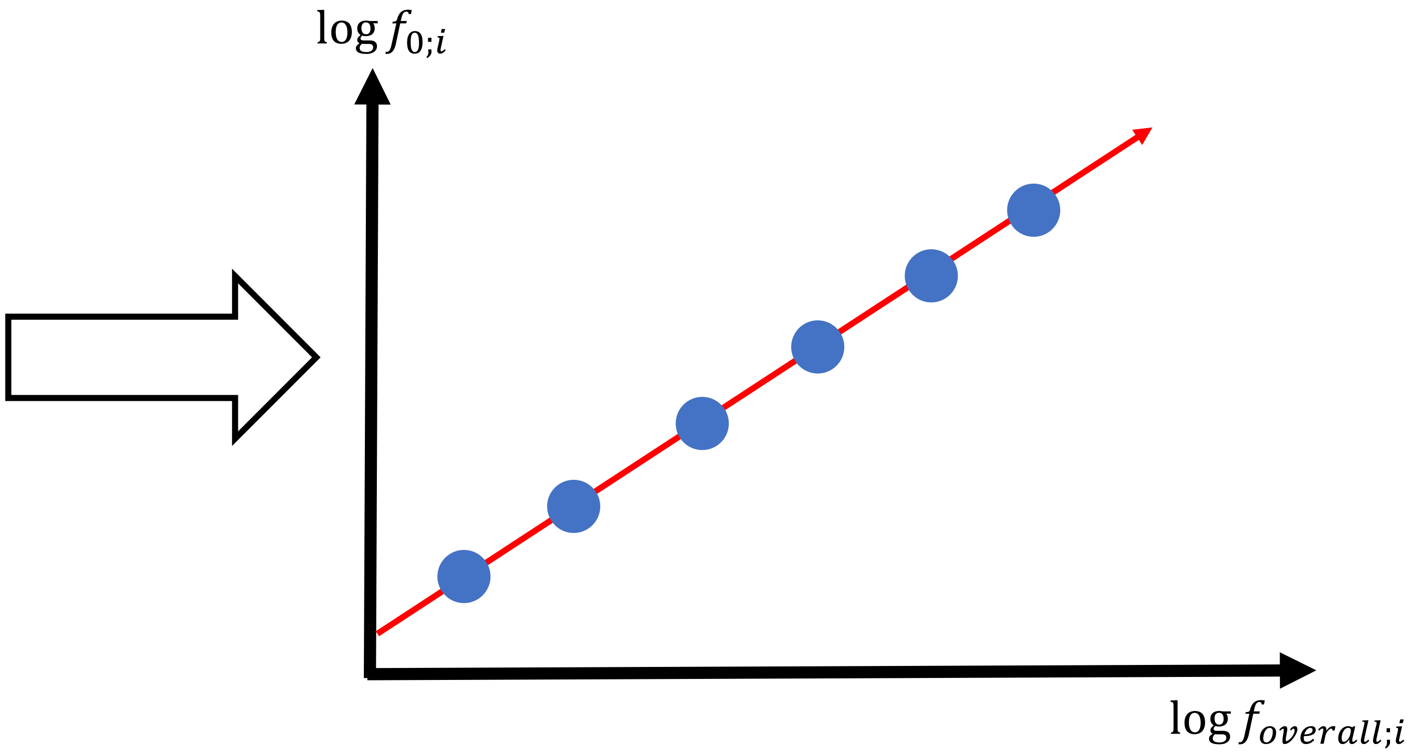




Formalize the Patterns: Pattern 3

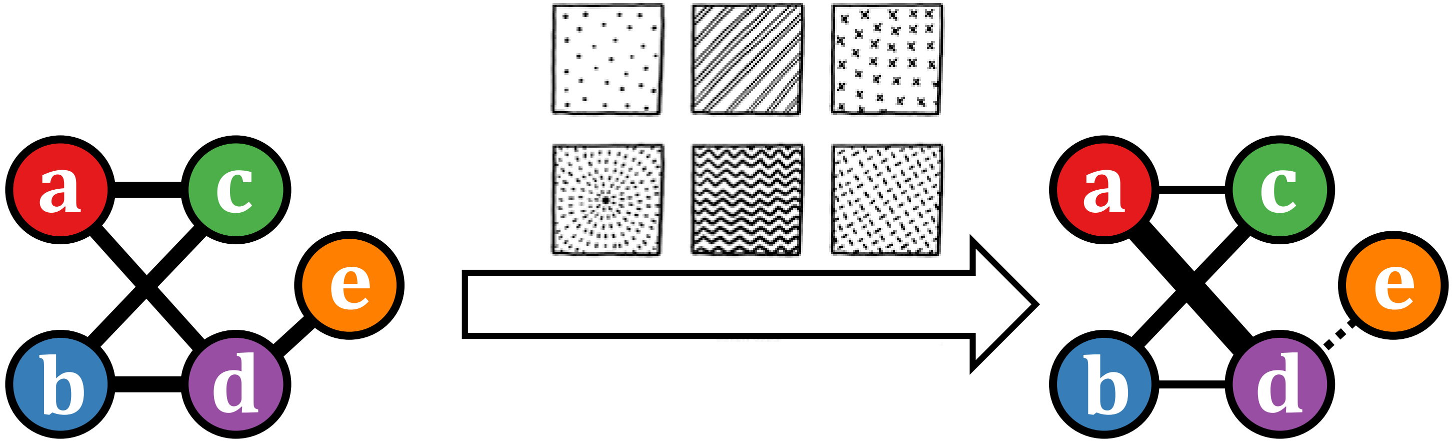
- **Pattern:** **Across layers**, the overall fractions of weighty edges $f_{overall;i} = |E_{i+1}|/|E_i|$ and the $f_{0;i}$'s exhibit a **strong power law**
- **Idealization:** Perfect power law

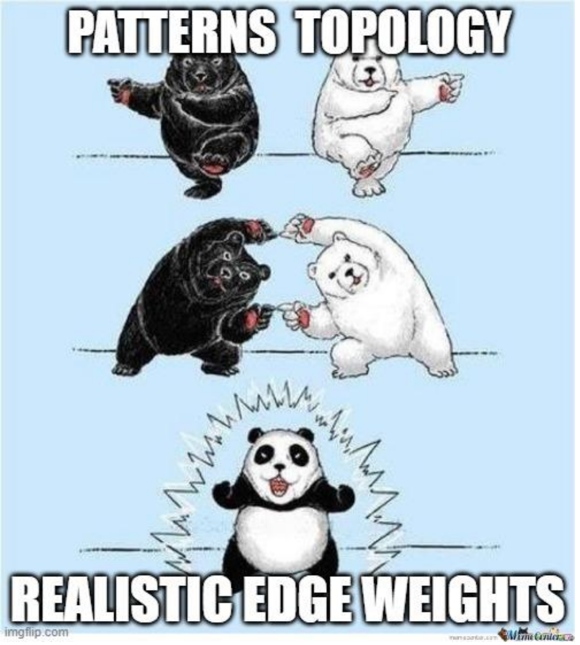




Proposed Algorithm: PEAR

- With all the idealization, we propose an algorithm called **PEAR**, with **only two parameters** a and k in the power law in Pattern 3
  - $f_{0;i} = a(f_{overall;i})^k$





Proposed Algorithm: PEAR

- **Algorithm overview:**
  - Starting from the layer-1, i.e., the given graph topology
  - Combine all the patterns to compute the fractions of weighty edges
  - Sample weighty edges according to the computed fractions
  - Repeat until all the layers are built up

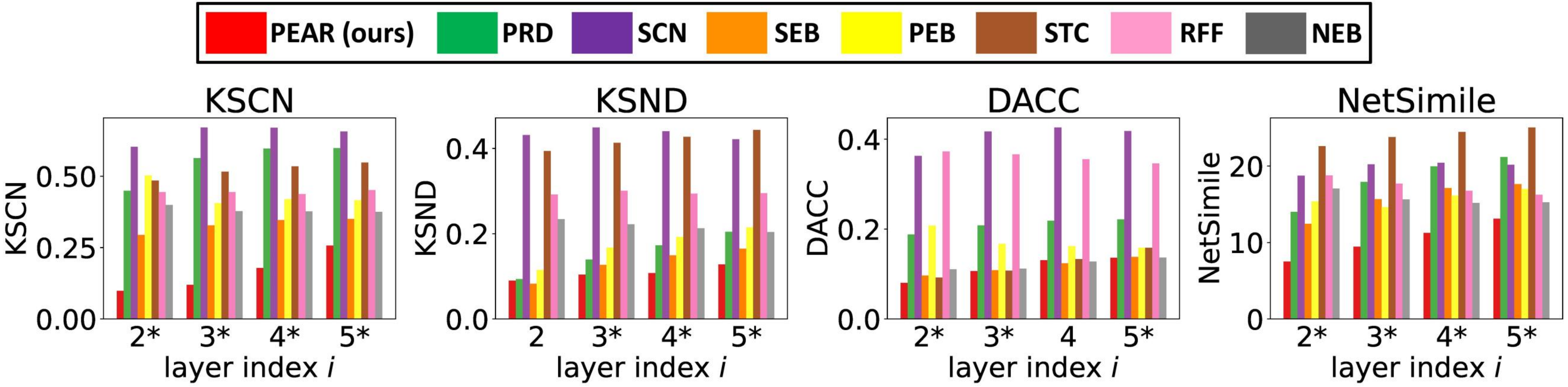


Experimental Settings

- **Baseline methods** with **additional information**
  - Five unsupervised baseline methods are given ground-truth $|E_i|$'s
  - Two supervised baseline methods are given ground-truth edge weights
- **Evaluation metrics:** We compare the generated layers with the original layers, and realistic edge weights are supposed to
  - Preserve graph statistics (KSCN, KSND, DACC)
  - Exhibit overall similarity (NetSimile)
- **Results summary:** With **fewest parameters** and **least information**, PEAR usually achieves the **best performance**

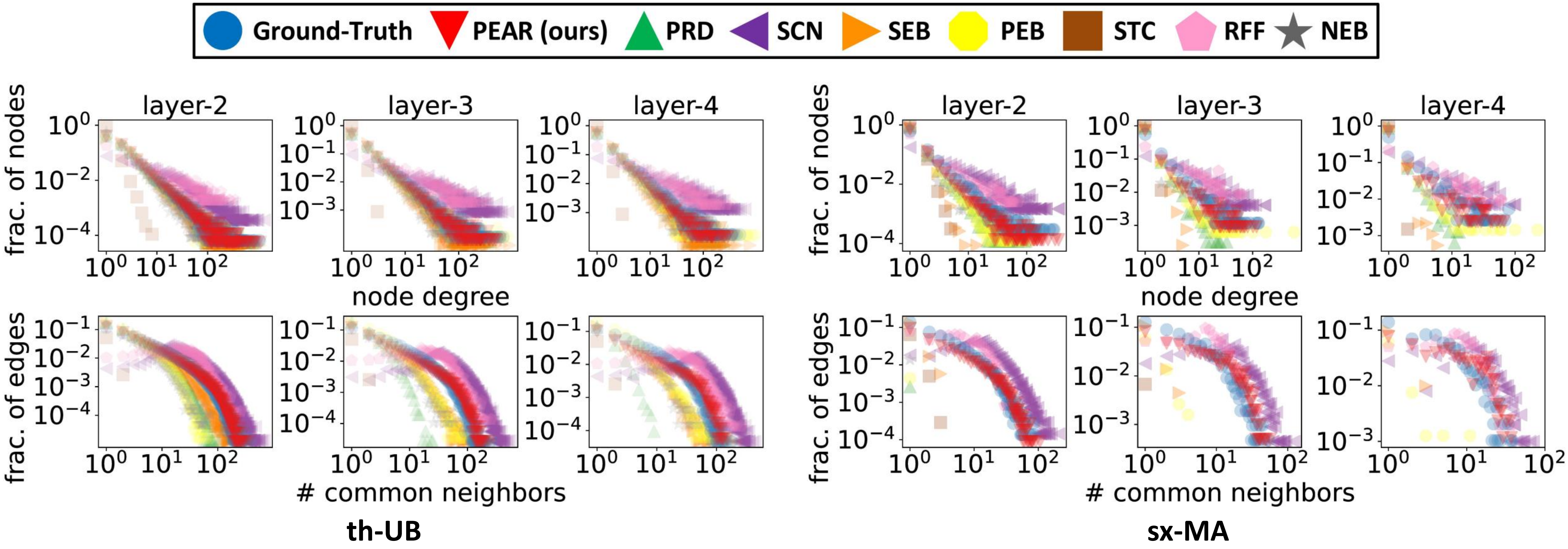
Experimental Results

- **Results summary:** With **fewest parameters** and **least information**, PEAR usually achieves the **best performance (*)**
  - For each considered metric, the smaller the better
  - The cases where PEAR performs best are indicated with asterisks (*)



Experimental Results

- **Detailed distributions** of degrees and # common neighbors



Application: Community Detection

- The performance of the **Louvain method** on the original unweighted graph and the weighted ones with edge weights output by PEAR
  - The predicted edge weights output by PEAR indeed **enhance the community detection performance** of the Louvain method

dataset	ARI (unweighted)	ARI (PEAR)	NMI (unweighted)	NMI (PEAR)
cora	0.2481 ± 0.0189	$\mathbf{0.2507 \pm 0.0151}$	0.4546 ± 0.0069	$\mathbf{0.4570 \pm 0.0055}$
citeseer	0.0937 ± 0.0069	$\mathbf{0.0950 \pm 0.0059}$	0.3287 ± 0.0027	$\mathbf{0.3292 \pm 0.0020}$
pubmed	0.0946 ± 0.0034	$\mathbf{0.0948 \pm 0.0083}$	0.1774 ± 0.0033	$\mathbf{0.1779 \pm 0.0035}$
computer	0.3147 ± 0.0119	$\mathbf{0.3201 \pm 0.0199}$	0.5411 ± 0.0083	$\mathbf{0.5459 \pm 0.0056}$
photo	0.5696 ± 0.0301	$\mathbf{0.5796 \pm 0.0074}$	0.6673 ± 0.0165	$\mathbf{0.6722 \pm 0.0069}$
cornell	0.0230 ± 0.0006	$\mathbf{0.0274 \pm 0.0011}$	0.0956 ± 0.0019	$\mathbf{0.1014 \pm 0.0030}$
texas	0.0513 ± 0.0025	$\mathbf{0.0758 \pm 0.0007}$	0.0698 ± 0.0014	$\mathbf{0.0835 \pm 0.0030}$
wisconsin	0.0230 ± 0.0039	$\mathbf{0.0290 \pm 0.0045}$	0.0911 ± 0.0057	$\mathbf{0.0977 \pm 0.0047}$

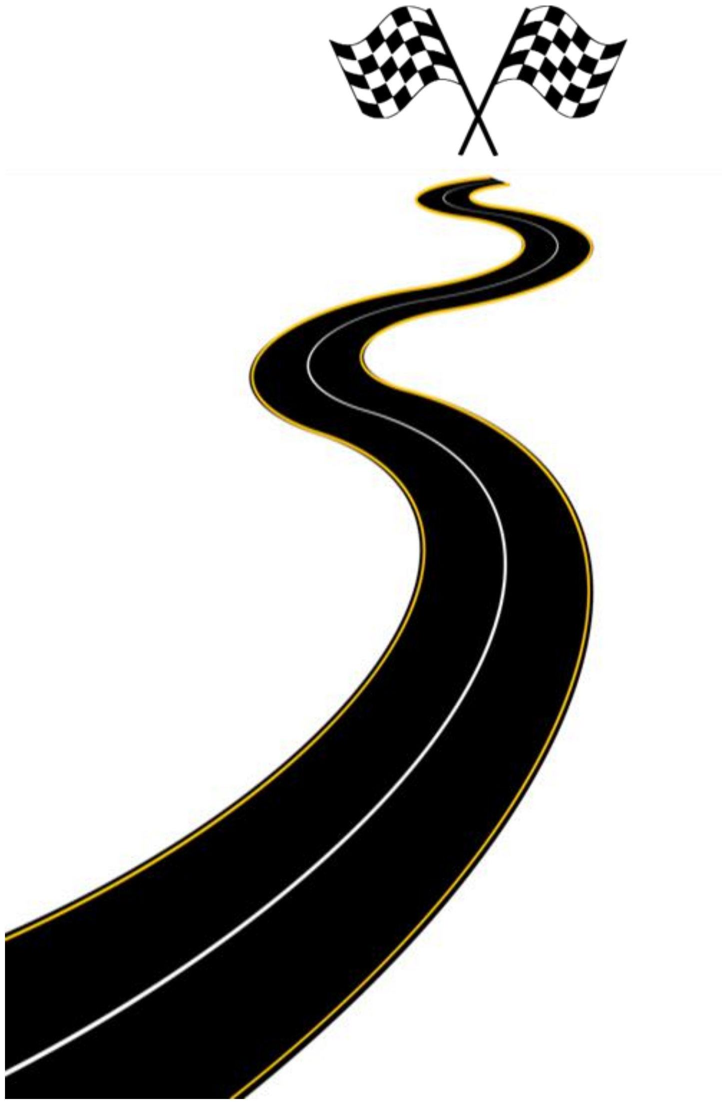
Application: Community Detection

- The performance of the **Louvain method** on the original unweighted graph and the weighted ones with edge weights output by PEAR
  - The predicted edge weights output by PEAR indeed **enhance the community detection performance** of the Louvain method

dataset	ARI (unweighted)	ARI (PEAR)	NMI (unweighted)	NMI (PEAR)
cora	0.2481 ± 0.0189	$\mathbf{0.2507 \pm 0.0151}$	0.4546 ± 0.0069	$\mathbf{0.4570 \pm 0.0055}$
citeseer	0.0937 ± 0.0069	$\mathbf{0.0950 \pm}$ [illegible]	0.3287 ± 0.0027	$\mathbf{0.3292 \pm}$ [illegible]
pubmed	0.0946 ± 0.0034	$\mathbf{0.0948 \pm}$ [illegible]	0.1774 ± 0.0033	$\mathbf{0.1779 \pm}$ [illegible]
computer	0.3147 ± 0.0119	[illegible]	0.5411 ± 0.0083	[illegible]
photo	0.5696 ± 0.0301	[illegible]	0.6673 ± 0.0165	[illegible]
cornell	0.0230 ± 0.0006	[illegible] $\mathbf{\pm 0.0011}$	0.0956 ± 0.0019	[illegible] $\mathbf{\pm 0.0030}$
texas	0.0513 ± 0.0025	[illegible] $\mathbf{\pm 0.0007}$	0.0698 ± 0.0014	[illegible] $\mathbf{\pm 0.0030}$
wisconsin	0.0230 ± 0.0039	$\mathbf{0.0290 \pm 0.0045}$	0.0911 ± 0.0057	$\mathbf{0.0977 \pm 0.0047}$

Roadmap

- Concepts
- Patterns
- Algorithm & Experiments
- **Conclusions <<**



Conclusions

- Our contributions are summarized as follows:

✓**Novel concepts** useful for analyzing weighted graphs

✓**Various patterns** extensively observed in real-world graphs

✓**A practical algorithm** integrating all the observed patterns

✓**Extensive evaluation** showing the effectiveness of the algorithm

Code: bit.ly/edge_weight_code