Interplay between Topology and Edge Weights in Real-World Graphs

Fanchen Bu, Shinhwan Kang, and Kijung Shin

{boqvezen97, shinhwan.kang, kijungs}@kaist.ac.kr

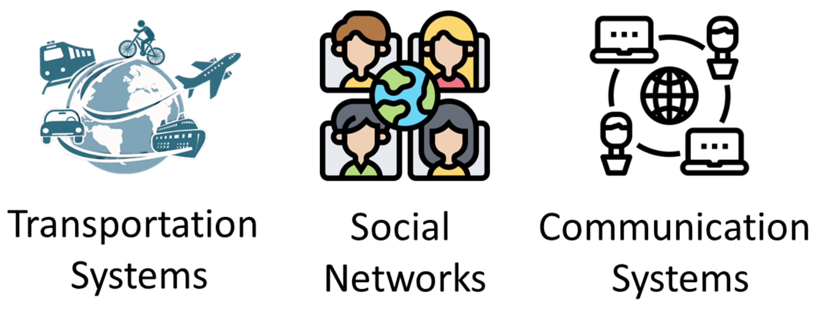
Code and Data: *bit.ly/edge_weight_code (or scan the QR code)*

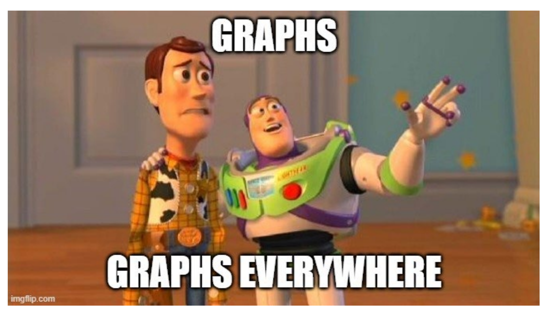
Summary

- **Novel concepts:** Useful tools and concepts for analyzing weighted graphs
- **Various patterns:** Insights into the interplay between topology and edge weights
- **Practical algorithm:** Integrating all the patterns to generate realistic edge weights
- **Extensive evaluation:** Showing the effectiveness of the proposed algorithm

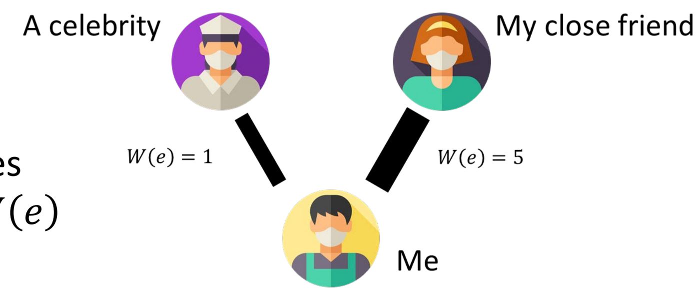
Background: Interplay between topology and edge weights

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

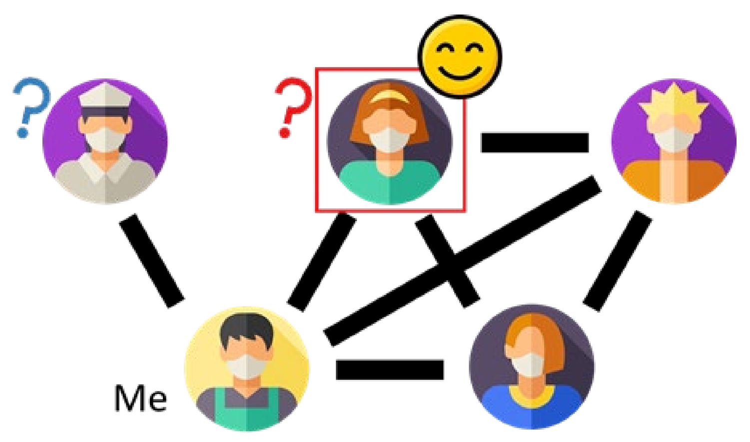




- Even in the same graph, edges are **not all the same**
- We can use **edge weights** to describe the heterogeneity of edges
  - Each edge e has its edge weight, which is a **positive integer** $W(e)$
- **Example:** Online social networks

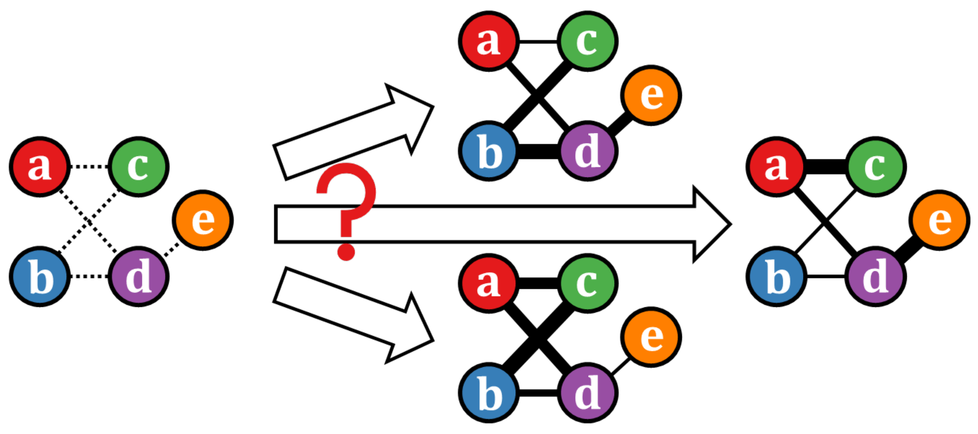


- In many cases, we can **infer edge weights from topology** (connections)
- **Q:** Which person is more likely my close friend with strong connection?
- **A:** The person with more common friends (indirect connections)

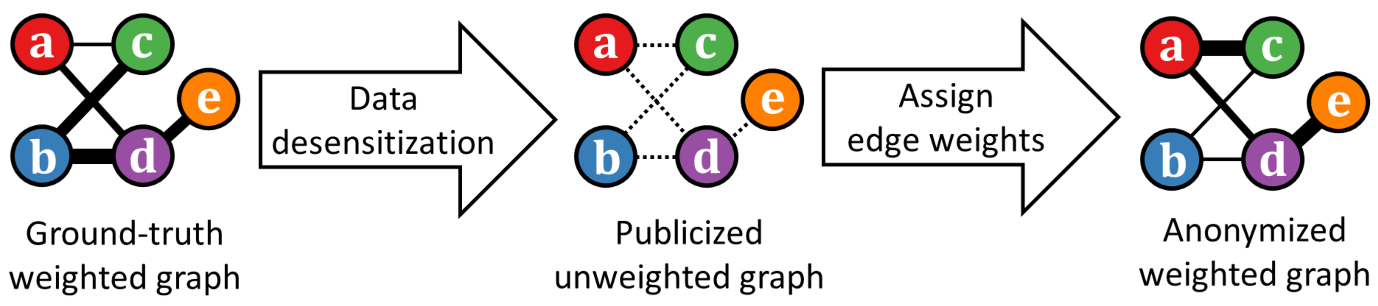


Research questions

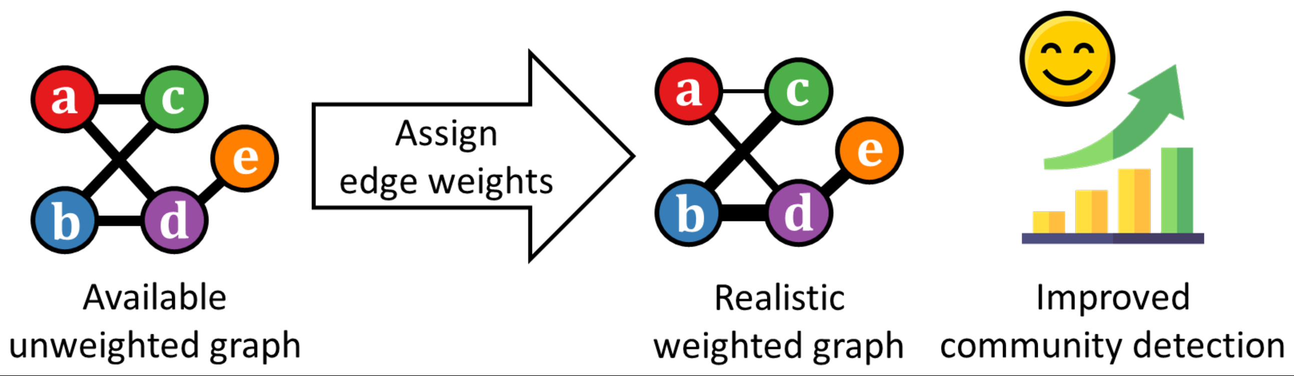
- **Q: What are some realistic properties** on the interplay between topology and edge weights?
- **Q: How can we assign realistic edge weights** to given graph topology based on realistic properties?



- **Real-world application:** Edge weight anonymization
  - Due to privacy issues, connection can be publicized but **NOT edge weights**
  - Assign **fake yet realistic** edge weights to generate **weighted benchmarks**

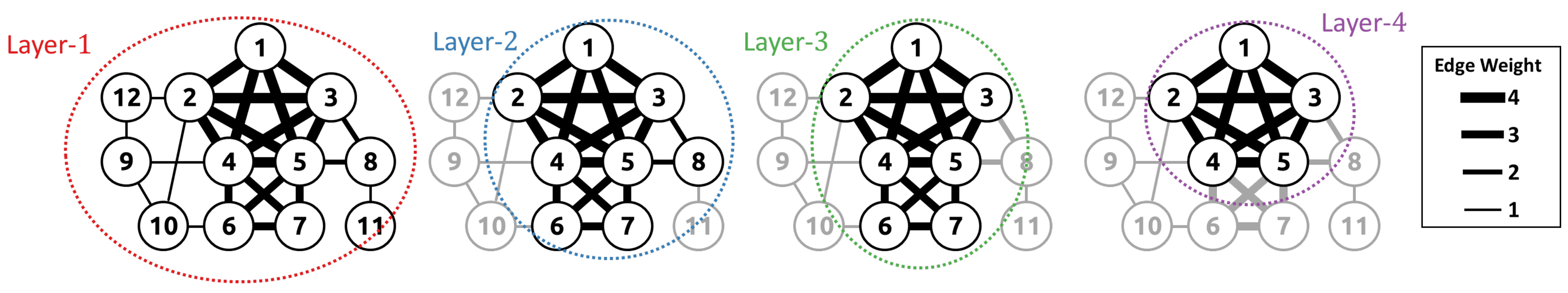


- **Real-world application:** Community detection
  - Edge weights provide **additional information** that are helpful for community detection
  - Assign **realistic** edge weights to **enhance the performance of community detection**

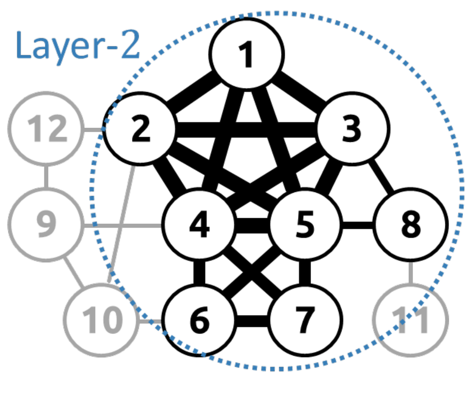


Proposed concepts: Layers and weighty edges

- The **layer-i** of a graph $G = (V, E, W)$ consists of the edges with edge weights $\geq i$
  - The layer-i of G is denoted by $G_i = (V_i, E_i, W_i)$
  - From layer-i, we can obtain layer-$(i + 1)$ by removing the edges with weight i



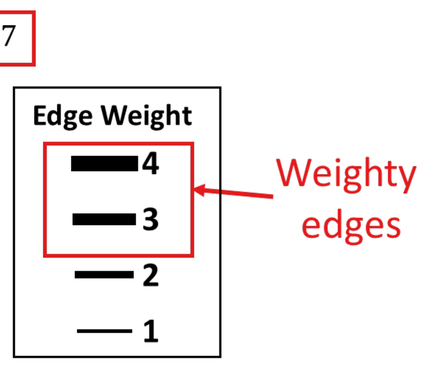
- The **weighty edges** in the layer-i are those with weight $> i$ (i.e., E_{i+1})
- The **overall fraction** of weighty edges in the layer-i is $f_{overall;i}(G) = |E_{i+1}|/|E_i|$
- Let $E_{c;i} \subseteq E_i$ be the set of the edges in E_i whose two endpoints share exactly c common neighbors
- The (local) **fraction of weighty edges (FoWE)** is $f_{c;i} = |E_{c;i} \cap E_{i+1}|/|E_{i+1}|$
- 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 $R_{c;i}$



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

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

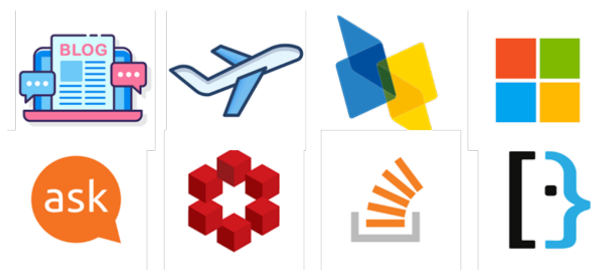
$R_{1;2}$ consists of $\{(1,8), (3,8), (5,8), (6,8), (7,8)\}$
$E_{1;2}$ consists of $\{(3,8), (5,8)\}$
$\Rightarrow \tilde{f}_{1;2} = |E_{1;2}|/|R_{1;2}| = 2/5$



Real-world weighted graph datasets

- **Datasets: 11 real-world weighted graphs from 5 different domains**
  - **# nodes:** 897 – 2.6M
  - **# edges:** 15,645 – 28.2M

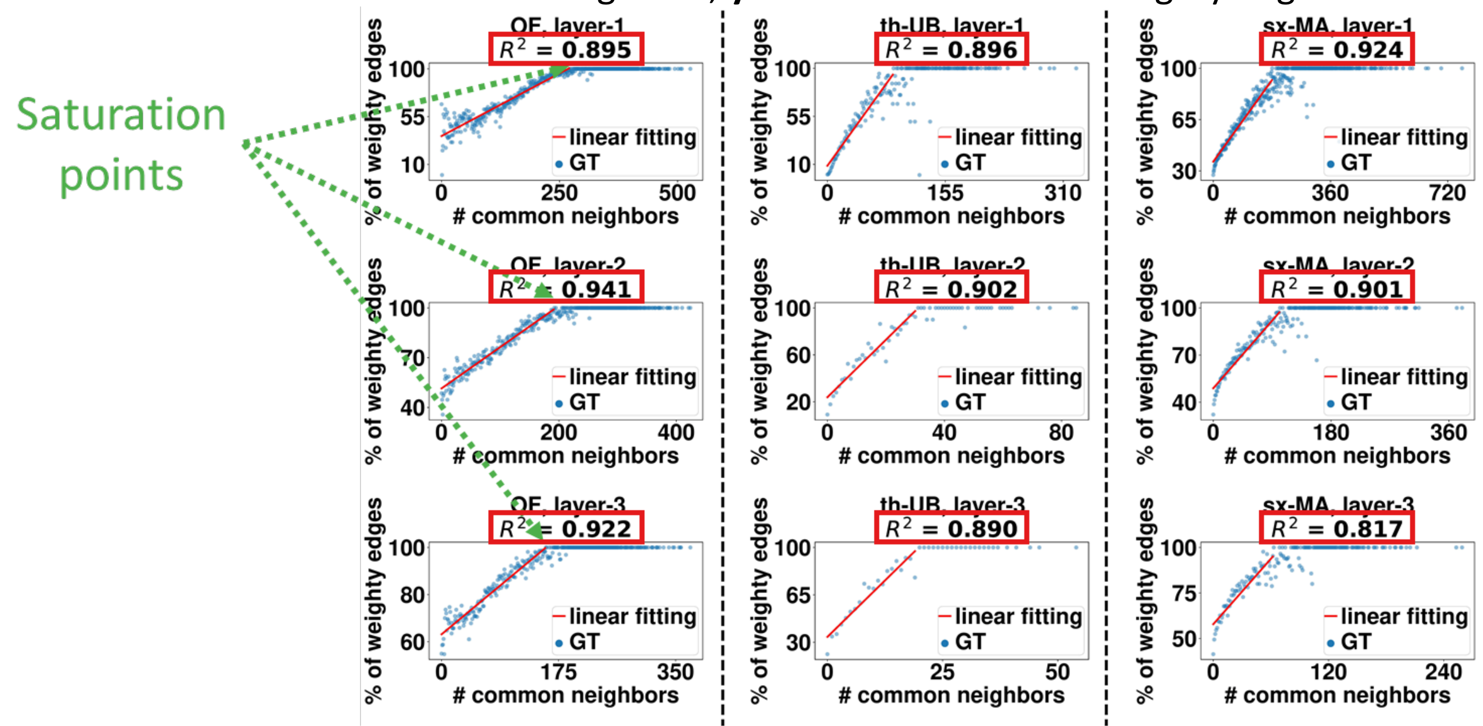
dataset	$\|V\|$	$\|E\| = \|E_1\|$	$\|E_2\|$	$\|E_3\|$	$\|E_4\|$
OF	897	71,380	47,266 (66.2%)	35,456 (49.7%)	28,546 (40.0%)
FL	2,905	15,645	4,608 (29.5%)	1,507 (9.6%)	564 (3.6%)
th-UB	82,075	182,648	7,297 (4.0%)	2,090 (1.1%)	965 (0.5%)
th-MA	152,702	1,088,735	128,400 (11.8%)	48,605 (4.5%)	26,121 (2.4%)
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

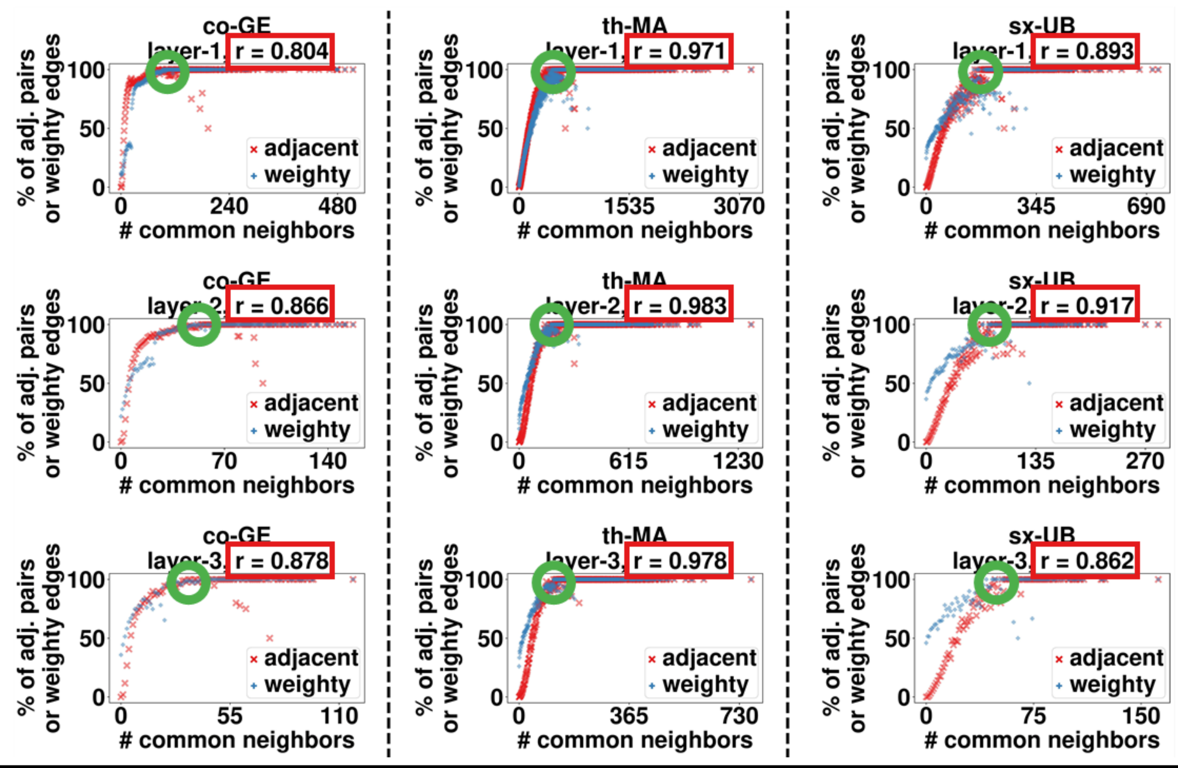
Pattern 1: Linear growth of FoWE

- **Pattern:** In each layer-i, the fraction of weighty edge (FoWE) $f_{c;i}$ **grows nearly linearly** with c (the number of common neighbors), and **saturates** after some point
- We plot the fraction of weighty edges in each layer
- **x-Axis:** The number of common neighbors; **y-Axis:** The fraction of weighty edges



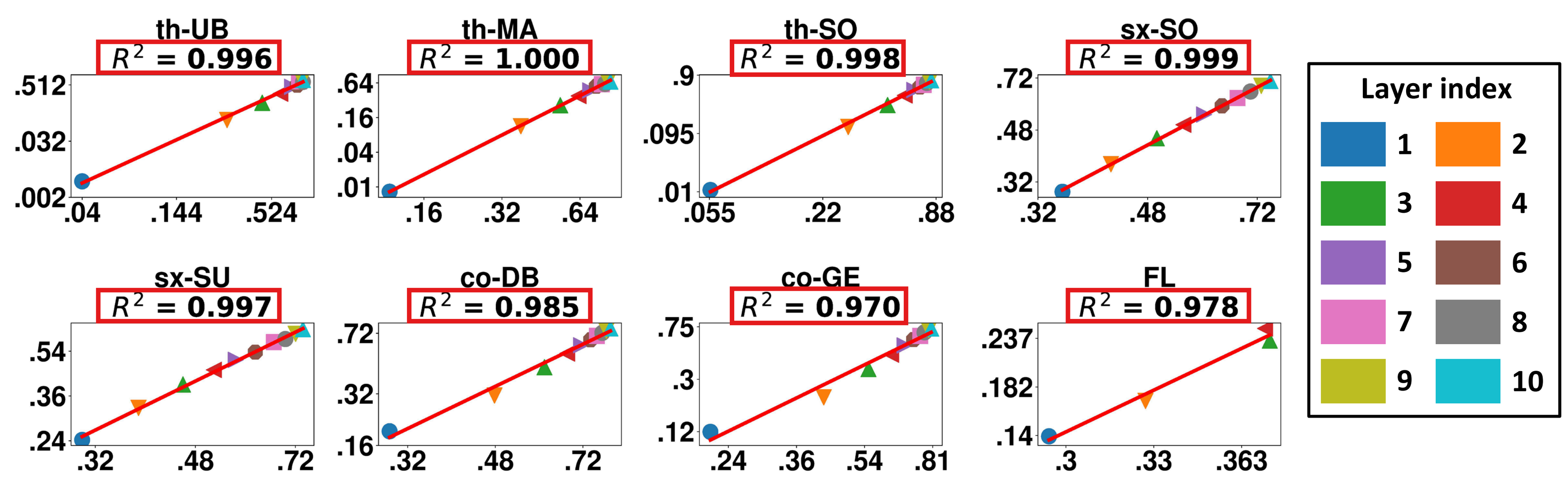
Pattern 2: Similarity between adjacency and weightiness

- **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)**, and have **similar saturation points**
- We plot the two fractions in each layer
  - **x-Axis:** The number of common neighbors; **y-Axis:** The two types of fractions



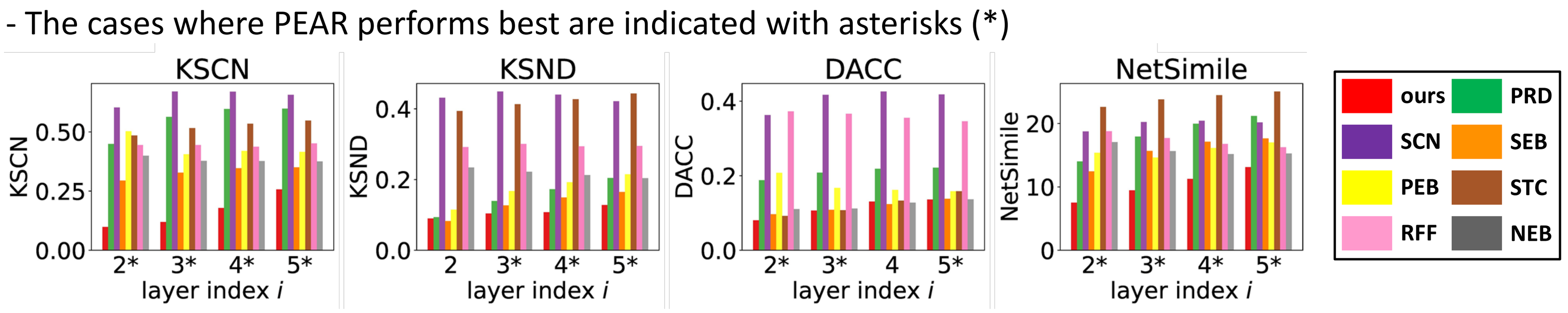
Pattern 3: A power law across layers

- **Pattern:** Across layers, the $f_{overall;i}$'s and the $f_{0;i}$'s exhibit a **strong power law**
  - $f_{0;i}$: the fraction of weighty edges among those **without common neighbors**
- For each dataset, we plot the two quantities in different layers in a log-log scale
- **x-Axis:** $f_{overall;i}$; **y-Axis:** $f_{0;i}$; Each point represents a different layer-i



Algorithm and experimental results

- **Algorithm:** The proposed algorithm **PEAR** integrates the three patterns
- Only **two parameters** are used in the algorithm
- Compared with **baseline methods** with **additional information**
- **Results:** With **fewest parameters** and **least information**, the proposed algorithm PAER performs best
- The cases where PEAR performs best are indicated with asterisks (*)



- **Detailed distributions** of degrees and the number of common neighbors