

# GRAPH NEURAL NETWORKS FOR GRAPH PROXIMITY COMPUTATION

---

**Yizhou Sun**

Department of Computer Science


University of California, Los Angeles

[yzsun@cs.ucla.edu](mailto:yzsun@cs.ucla.edu)

August 5, 2019

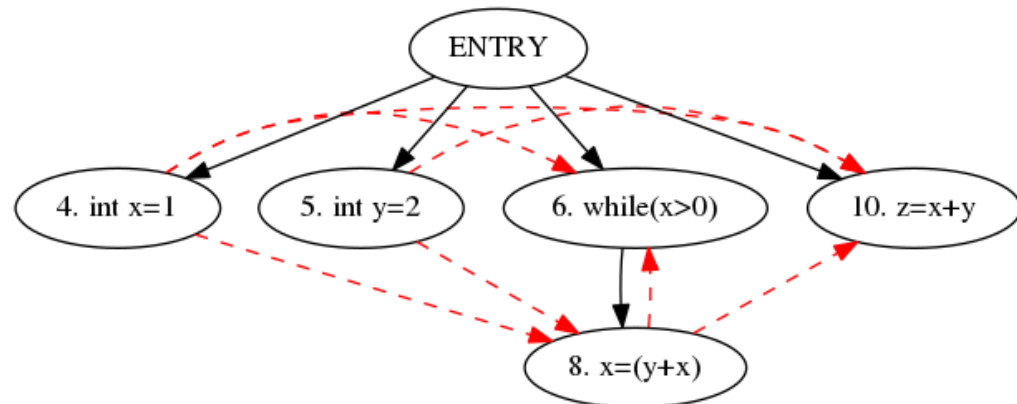
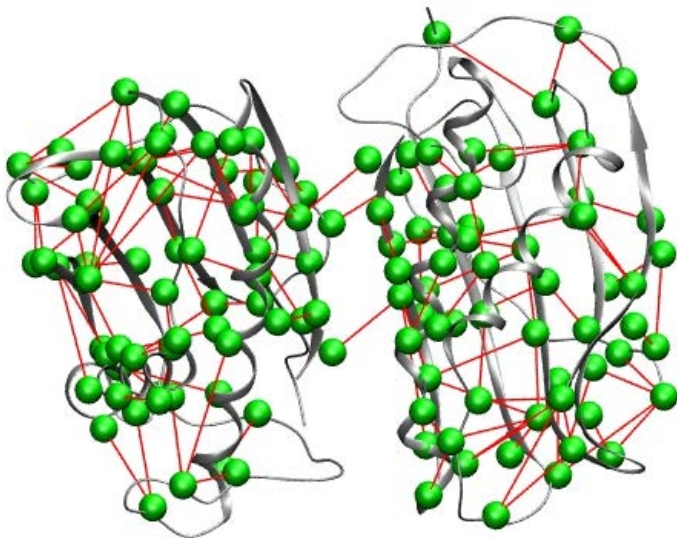
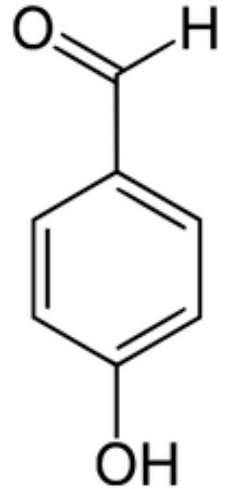
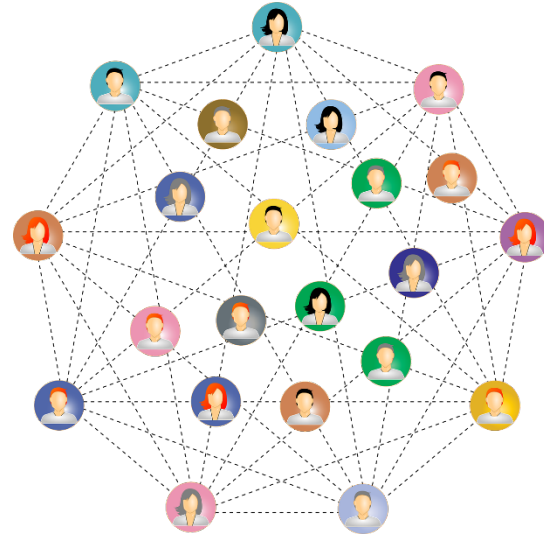
# Outline

---

- Introduction 
- Starting from Graph Edit Distance Computation
- Graph Similarity Computation as Set Matching
- Summary

# Graph Analysis

- Graphs are ubiquitous
  - Social networks
  - Proteins
  - Chemical compounds
  - Program dependence graph
  - ...



# Typical Graph Functions

---

- **Node level**
  - Similarity search
  - Link prediction
  - Classification
  - Community detection
  - Ranking
- **Graph level**
  - Similarity search
  - Classification
  - Clustering

# Neural Networks Have Reshaped Graph Analysis

---

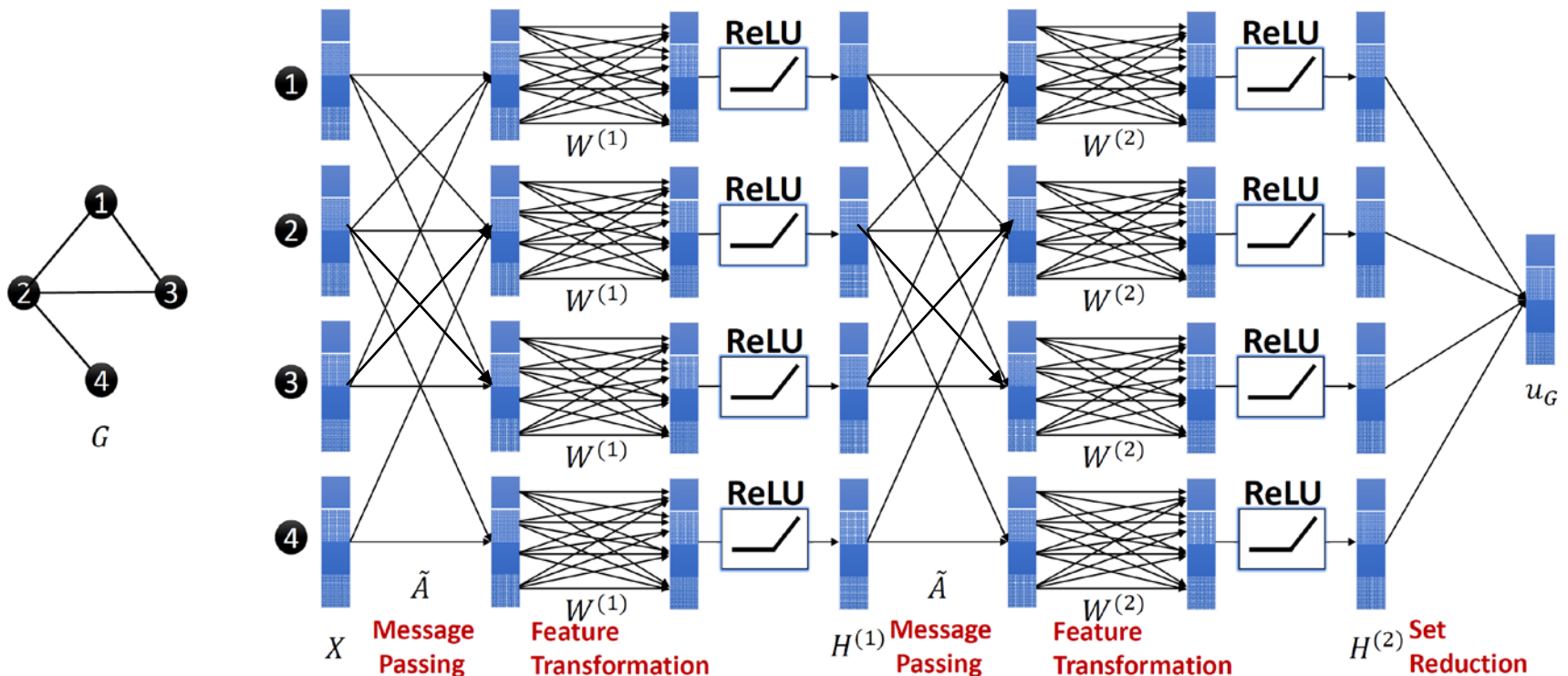
- **Node-level representation learning**
  - Shallow architecture: DeepWalk, LINE, Node2Vec, NetMF, ...
  - Deep architecture: SDNE, GCN, GraphSAGE ...
- **Graph-level representation learning**
  - GCN, GraphSAGE, Graph2Vec, Differential Pooling, GIN, ...

# More on Graph Convolutional Network (GCN)

- Kipf and Welling, ICLR'17

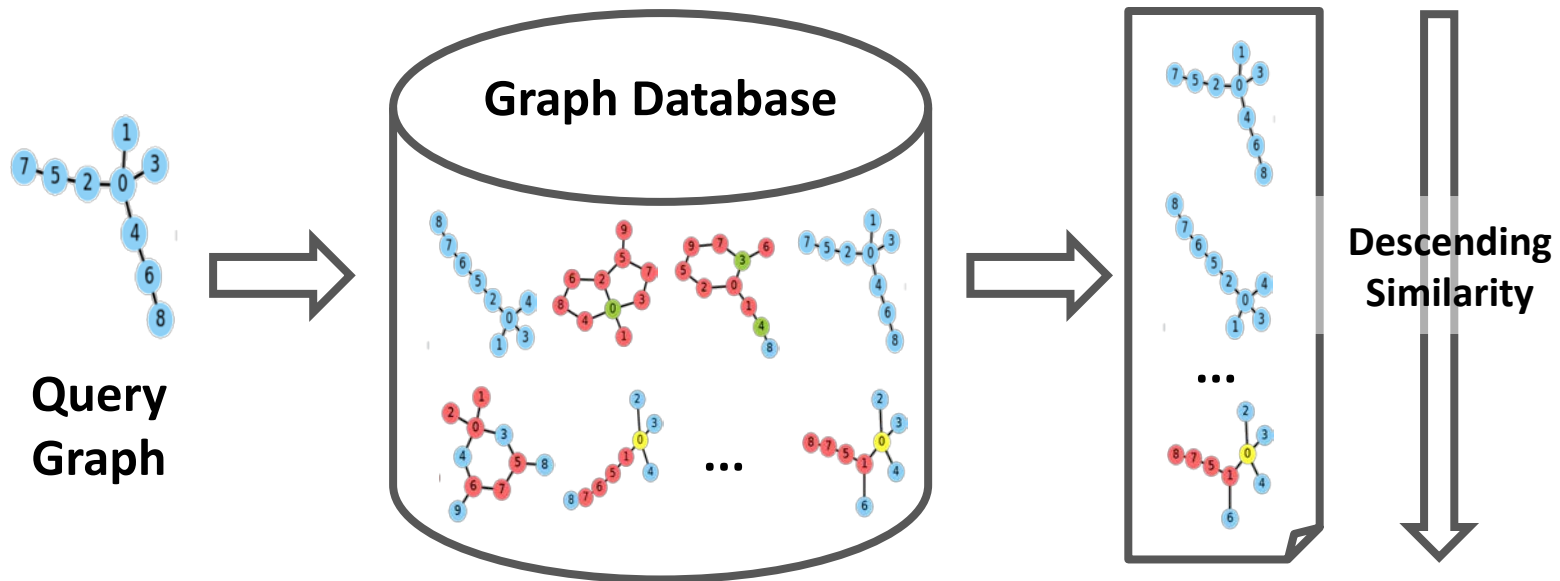
$$f(H^{(l)}, A) = \sigma \left( \hat{D}^{-\frac{1}{2}} \hat{A} \hat{D}^{-\frac{1}{2}} H^{(l)} W^{(l)} \right)$$

- A toy example of 2-layer GCN on a 4-node graph



# What Operators Are Needed for Graph Proximity Search?

- Consider graph-level similarity search




- Two types of graph-level operators will be helpful
  - Graph similarity computation operator:  $\phi: \mathcal{G} \times \mathcal{G} \rightarrow \mathbb{R}^+$
  - Graph-level representation operator:  $\phi: \mathcal{G} \rightarrow \mathbb{R}^d$

**Q: Can existing operators solve the task?**

# Outline

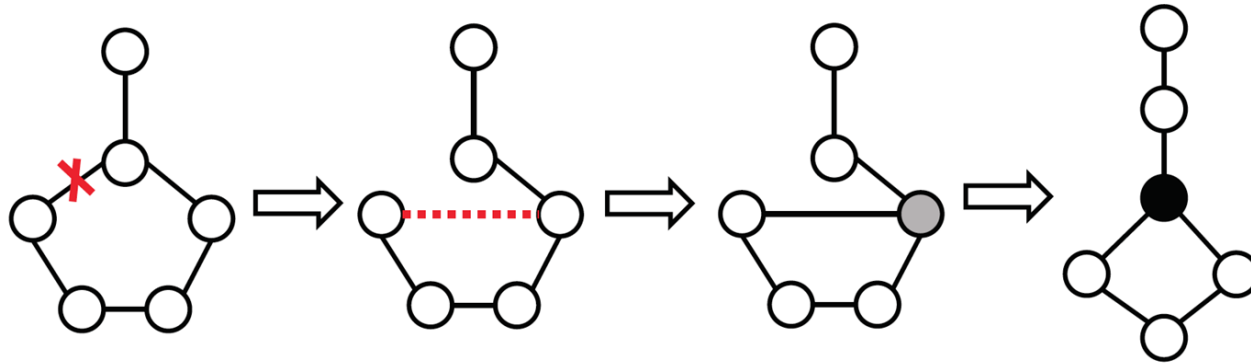
---

- Introduction
- Starting from Graph Edit Distance  Computation
- Graph Similarity Computation as Set Matching
- Summary



# Graph Edit Distance

- Measure the distance between two graphs



- NP-hard problem

- No currently available algorithm can reliably compute the exact GED within reasonable time between graphs with more than 16 nodes.

# Traditional Solution

---

- Combinatorial optimization approaches
  - domain knowledge and heuristics;
  - difficult to design and implement

Method	Time Complexity
A* [19]	$O(N_1^{N_2})$
Beam [11]	subexponential
Hungarian [13]	$O((N_1 + N_2)^3)$
VJ [12]	$O((N_1 + N_2)^3)$
HED [14]	$O((N_1 + N_2)^2)$

$N_1$  and  $N_2$  refer to the node size of the two graphs

# Existing Graph-Level Representation?

---

- Feature engineering
  - E.g., Graphlets
- Neural network-based approach
  - E.g., GCN, GraphSAGE, Graph2Vec, Differential Pooling, GAN, GIN, ...



But why do they happen to be related to **GED**?

# Our Goal: A Learnable Graph Similarity Operator

---

- $\phi: \mathcal{G} \times \mathcal{G} \rightarrow \mathbb{R}^+$
- Goal:
  - For a given pair of graphs  $G_i$  and  $G_j$
  - Require:  $\phi(G_i, G_j) \approx GED(G_i, G_j)$ 
    - In practice, we transform GED into similarity between 0 and 1 using 1-1 mapping
- Properties of  $\phi$ 
  - Permutation-invariant: handle graph isomorphism
  - Inductive: can apply to unseen graphs
  - Learnable: flexible to different similarity metric

# Solution: SimGNN [Bai et al., WSDM'19]

---

- **Strategy 1**
  - Learn a graph-level embedding that reflect the specified proximity
- **Strategy 2**
  - Directly learn a score function based on two sets of node-level embedding vectors

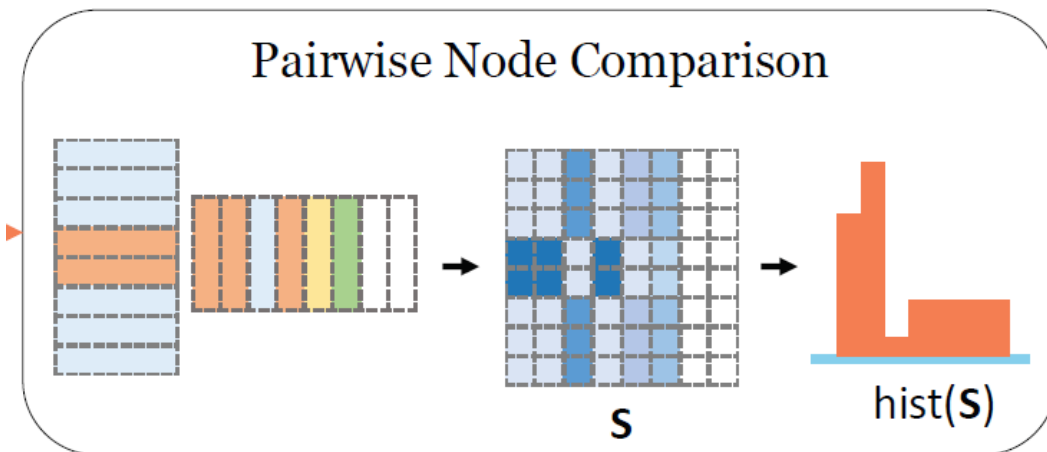
# Strategy I: Operation on Graph-Level Representations

---

- How to get a proximity-aware graph-level embedding?
  - Use learnable **attention** to get graph-level embedding
  - Use learnable neural tensor network to compute **proximity score**
  - Guided by proximity-based loss function

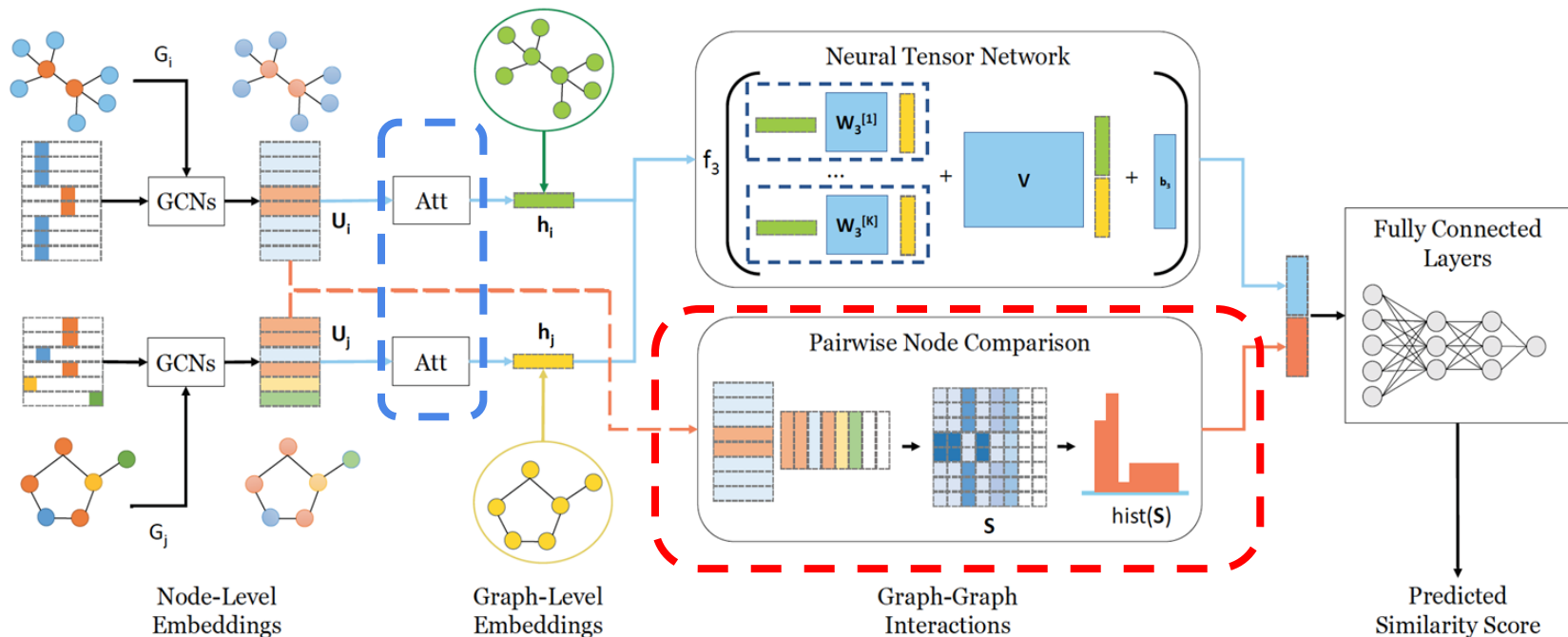
# Strategy II: Operation on Two Sets of Node-Level Representations

- Challenges:
  - Nodes are not ordered
- Solution:
  - Form node-node similarity matrix
  - Extract **histogram** to represent overall distribution



# Putting Together

- The SimGNN architecture



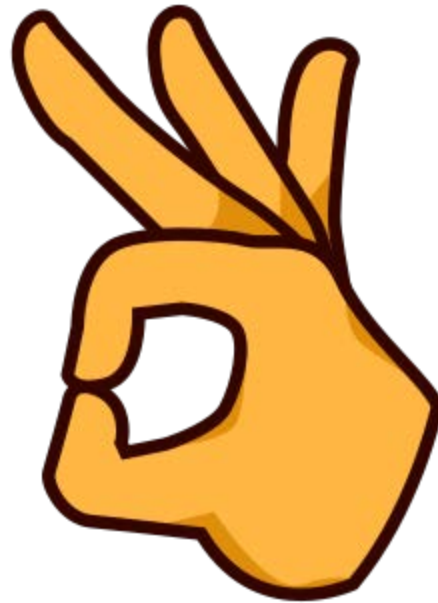
1. Attention-based graph-level embedding
2. Histogram features from pairwise node similarities



# Check Properties of SimGNN

---

- Permutation-invariant
- Inductive
- Learnable



# Experiments

---

- **Datasets**

<b>Dataset</b>	<b>Graph Meaning</b>	<b>#Graphs</b>	<b>#Pairs</b>
<b>AIDS</b>	Chemical Compounds	700	490K
<b>LINUX</b>	Program Dependency Graphs	1000	1M
<b>IMDB</b>	Actor/Actress Ego-Networks	1500	2.25M

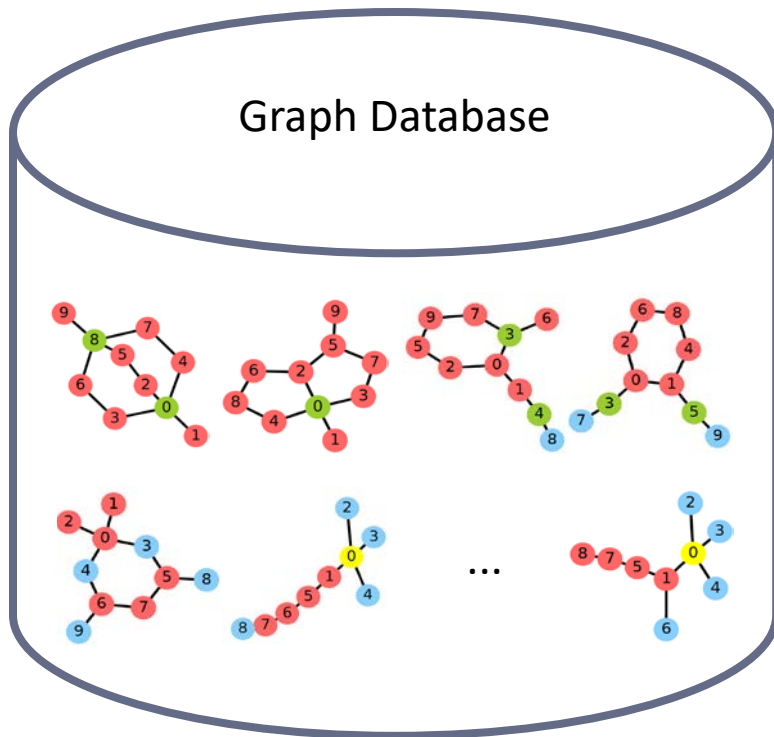
- **Metric**

- Effectiveness
- Efficiency

- **Baselines**

- Traditional GED computation methods
- Simple NN Graph operators

# Training Preparation



Training Validation Test

1.00	0.71	0.53	0.48	0.41	0.10	...	0.14
0.71	1.00	0.73	0.71	0.39	0.12	...	0.15
0.53	0.73	1.00	0.88	0.45	0.23	...	0.27
0.48	0.71	0.88	1.00	0.65	0.23	...	0.28
0.41	0.39	0.45	0.65	1.00	0.81	...	0.86
0.10	0.12	0.23	0.23	0.81	1.00	...	0.91
...	...	...	...	...	...	...	...
0.14	0.15	0.27	0.28	0.86	0.91	...	1.00

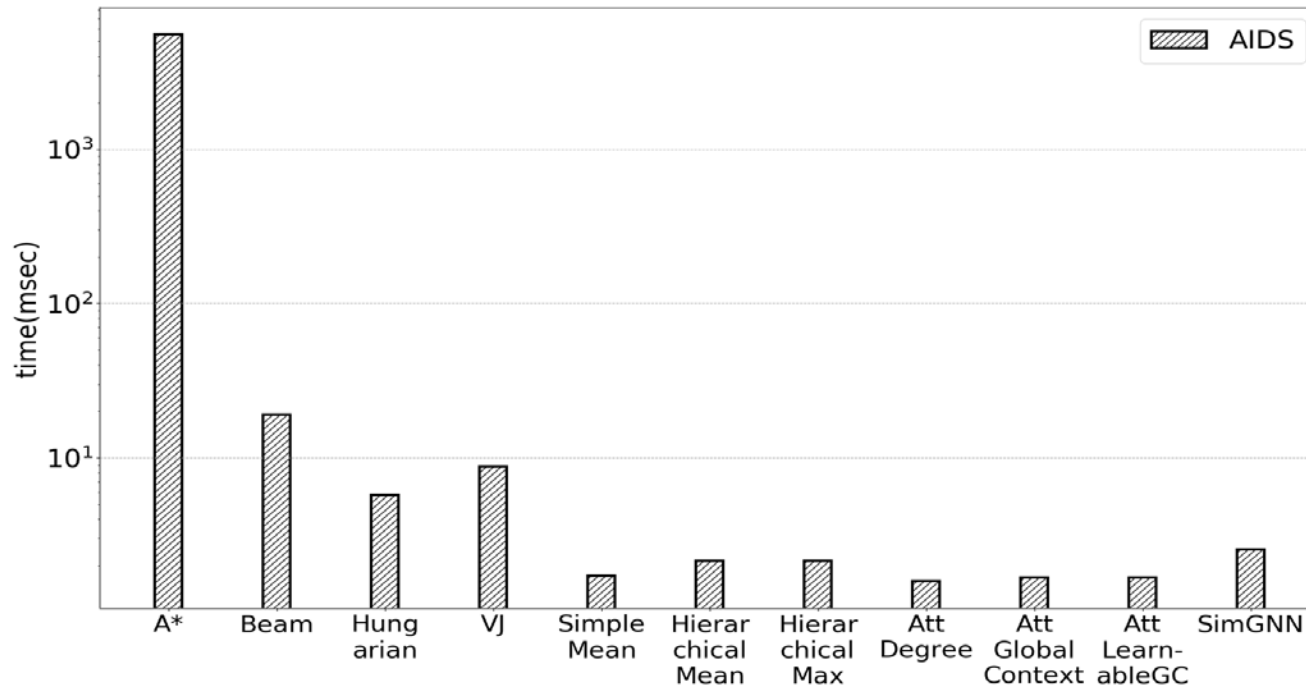
# Effectiveness

---

Method	mse( $10^{-3}$ )	$\rho$	$\tau$	p@10	p@20
Beam	12.090	0.609	0.463	<b>0.481</b>	0.493
Hungarian	25.296	0.510	0.378	0.360	0.392
VJ	29.157	0.517	0.383	0.310	0.345
SimpleMean	3.115	0.633	0.480	0.269	0.279
HierarchicalMean	3.046	0.681	0.629	0.246	0.340
HierarchicalMax	3.396	0.655	0.505	0.222	0.295
AttDegree	3.338	0.628	0.478	0.209	0.279
AttGlobalContext	1.472	0.813	0.653	0.376	0.473
AttLearnableGC	1.340	0.825	0.667	0.400	0.488
SimGNN	<b>1.189</b>	<b>0.843</b>	<b>0.690</b>	<b>0.421</b>	<b>0.514</b>

More accurate than *most* the existing approximate GED algorithms and simple NN methods.

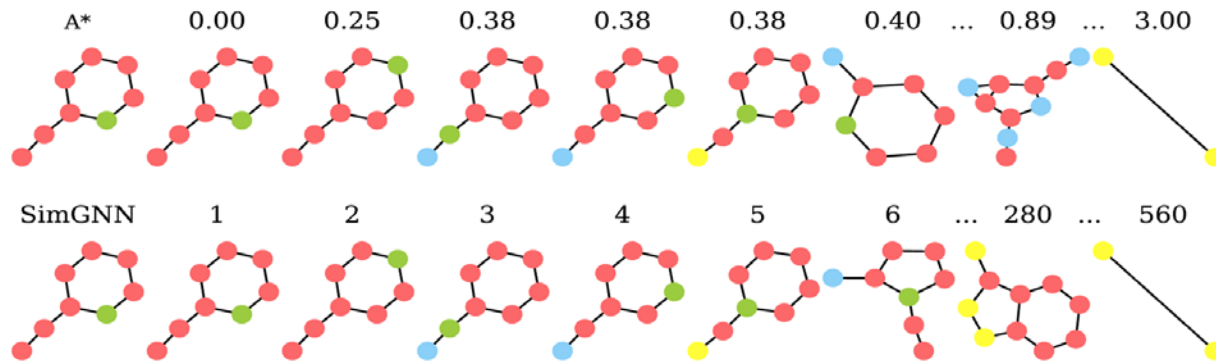
# Efficiency



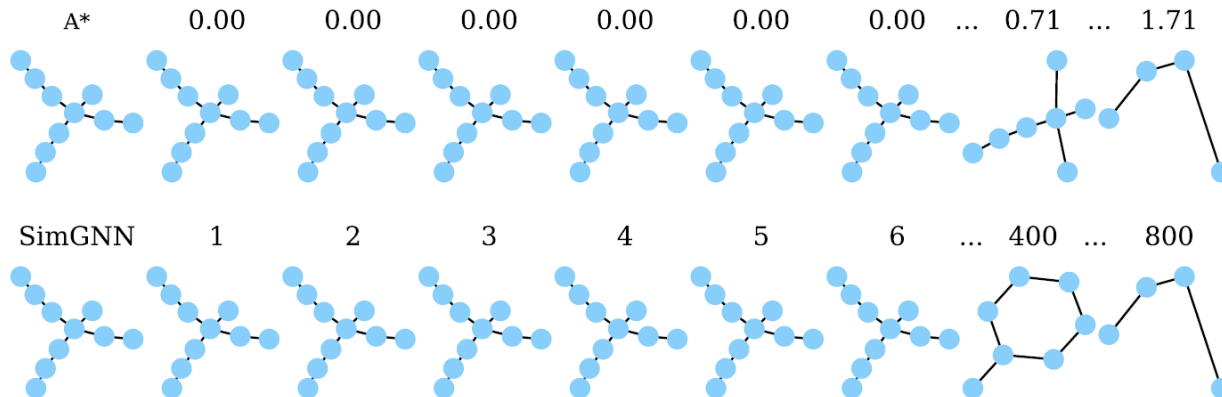
- Other NN approaches are with similar running time but less accurate than SimGNN.
- Beam is better, but is much slower (log-scale).

# Case Studies

- Query on AIDS

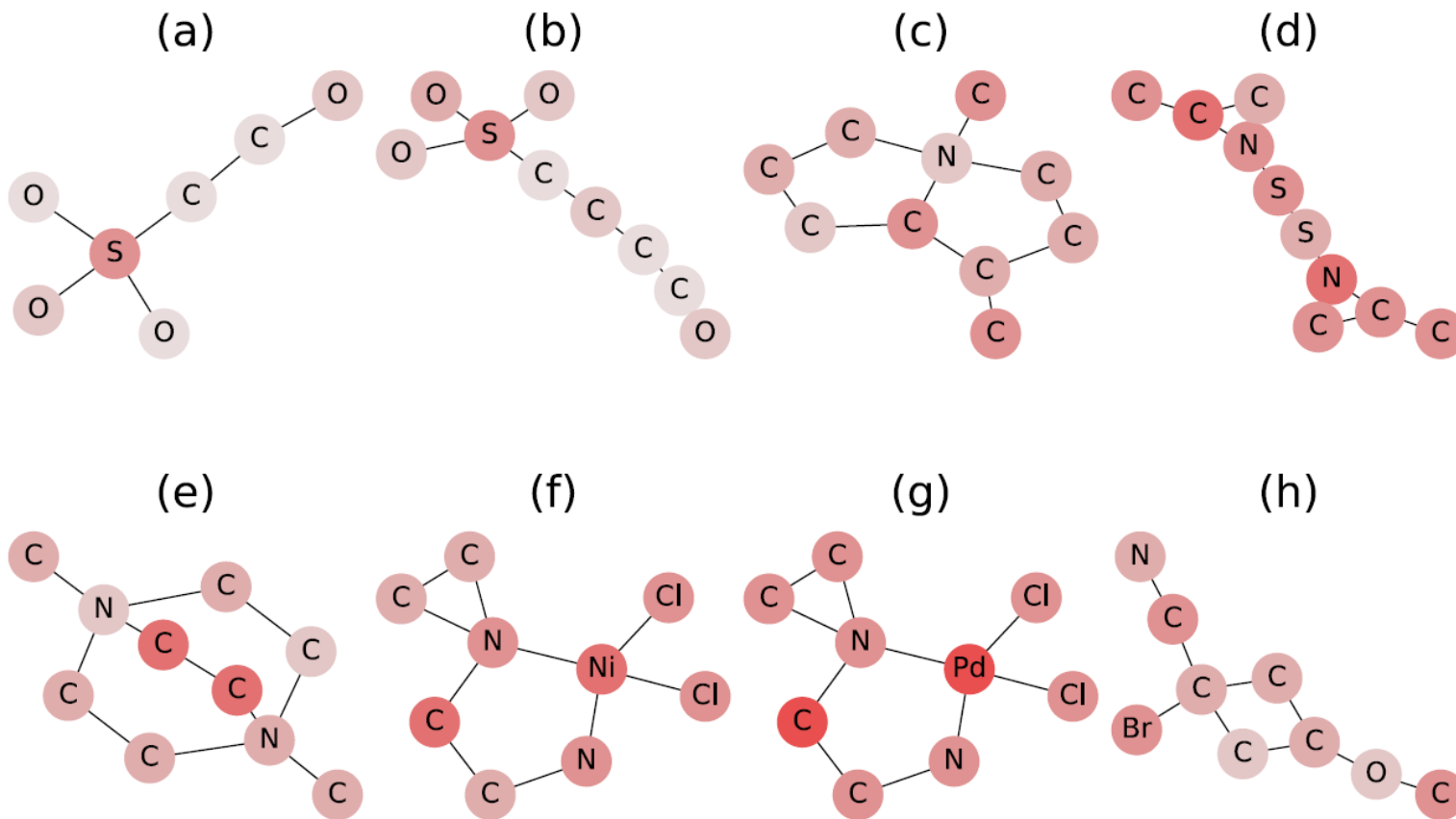


- Query on LINUX




# Where do we put attention?

- AIDS Dataset



# Outline

---

- Introduction
- Starting from Graph Edit Distance Computation
- Graph Similarity Computation as Set Matching 
- Summary



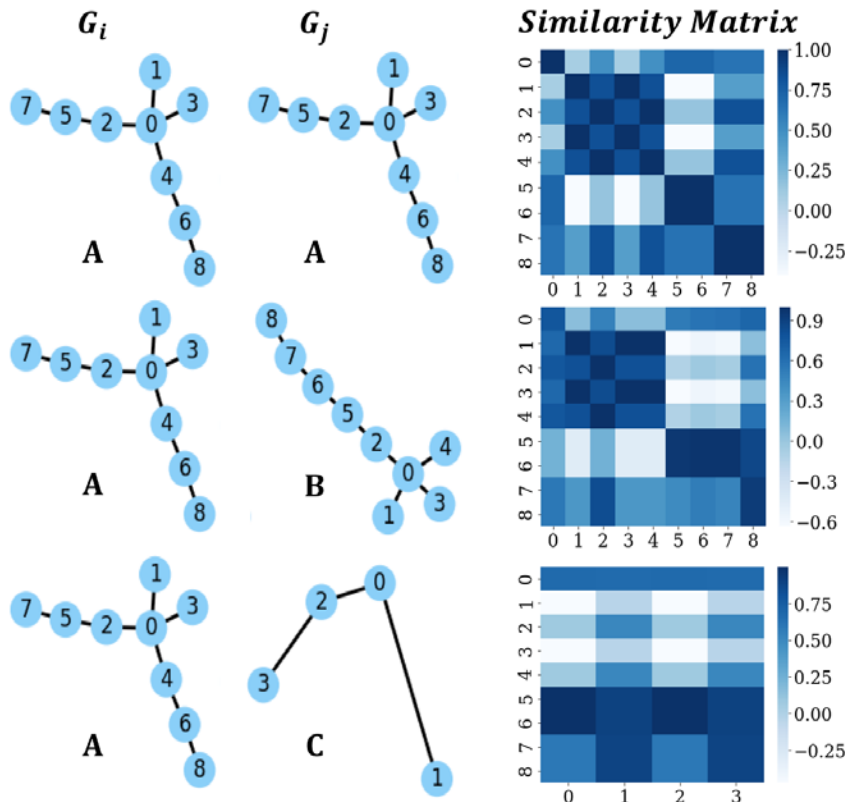
# Limitation of SimGNN

---

- Graph-Graph interaction modeling is oversimplified
  - Histogram function in Strategy II provides a coarse summary
- Not a fully end-to-end framework
  - Histogram function in Strategy II is not differentiable

# Can We Do Better? [Bai et al., NeurIPS'18 Workshop]

- Directly use the original set of node vectors, and turn it into a set matching problem



- Graph similarity  $\rightarrow$  Image pattern recognition
- Similarity matrix encodes node alignment information.

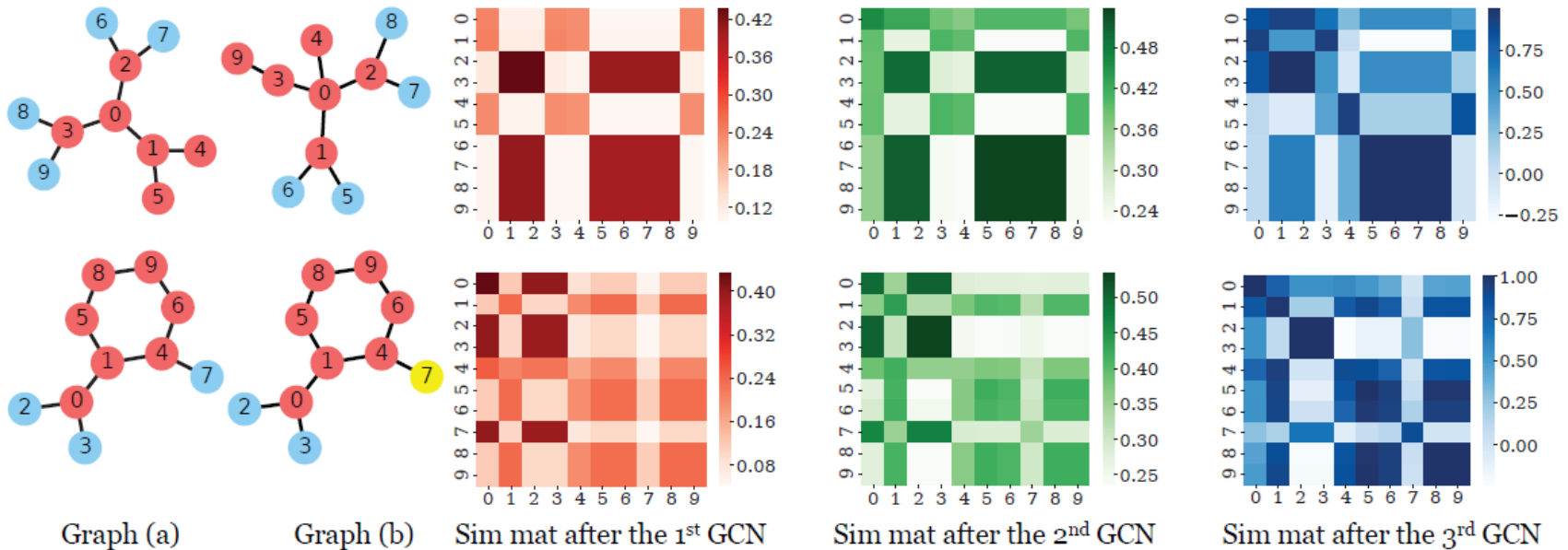
# Issues and Solutions to Similarity Matrix

---

- **Not permutation-invariant**
  - Different similarity matrices under different node orders
  - Solution: Fix a order using BFS starting from highest-degree node
- **Graph pairs are with different sizes**
  - Solution: Padding zeros to get squared similarity matrix
- **Different sizes over different graph pairs**
  - Solution: Resize via interpolation

# Extend from one scale to multiple scales

- Compare two graphs at different scales



- Multiscale image regression problem
  - Feed Similarity Matrices into CNN (or other NN architecture)

# Experiments

---

- **Two Tasks**
  - GED (Graph Edit Distance)
  - MCS (Maximum Common Subgraph)
- **Metric**
  - Effectiveness
  - Efficiency

# Summary of Baselines

Method	GED	MCS	Exact	NN	Worst-Case Time
A* [17]	✓	×	✓	×	$O(N_1 N_2)$
BEAM [33]	✓	×	×	×	$O(N_1 N_2)$
HUNGARIAN [40]	✓	×	×	×	$O((N_1 + N_2)^3)$
VJ [12]	✓	×	×	×	$O((N_1 + N_2)^3)$
HED [13]	✓	×	×	×	$O((N_1 + N_2)^2)$
MCSPLIT [32]	×	✓	✓	×	$O(N_1 N_2)$
SMPNN [39]	✓	✓	×	✓	$O(\max(E_1, E_2, N_1 N_2))$
EMBAVG	✓	✓	×	✓	$O(\max(E_1, E_2))$
GCNMEAN [25]	✓	✓	×	✓	$O(\max(E_1, E_2))$
GCNMAX [25]	✓	✓	×	✓	$O(\max(E_1, E_2))$
SIMGNN [1]	✓	✓	×	✓	$O(\max(N_1, N_2)^2)$
GRAPHSIM	✓	✓	×	✓	$O(\max(N_1, N_2)^2)$

# Effectiveness on GED and MCS

Method	Aids				LINUX				IMDB			
	mse	$\rho$	p@10	p@20	mse	$\rho$	p@10	p@20	mse	$\rho$	p@10	p@20
A*	0.000*	1.000*	1.000*	1.000*	0.000*	1.000*	1.000*	1.000*	–	–	–	–
BEAM	12.090	0.609	0.481	0.493	9.268	0.827	0.973	0.924	2.413*	0.905*	0.803*	0.814*
HUNGARIAN	25.296	0.510	0.360	0.392	29.805	0.638	0.913	0.836	1.845*	0.932*	0.825*	0.824*
VJ	29.157	0.517	0.310	0.345	63.863	0.581	0.287	0.251	1.831*	0.934*	0.815*	0.819*
HED	28.925	0.621	0.386	0.423	19.553	0.897	0.982	0.936	19.400	0.751	0.801	0.808
SMPNN	5.184	0.294	0.032	0.056	11.737	0.036	0.009	0.021	32.596	0.107	0.023	0.029
EMBAVG	3.642	0.601	0.176	0.250	18.274	0.165	0.071	0.142	71.789	0.229	0.233	0.500
GCNMEAN	3.352	0.653	0.186	0.274	8.458	0.419	0.141	0.195	68.823	0.402	0.200	0.208
GCNMAX	3.602	0.628	0.195	0.280	6.403	0.633	0.437	0.454	50.878	0.449	0.425	0.418
SIMGNN	1.189	0.843	0.421	0.514	1.509	0.939	0.942	0.933	1.264	0.878	0.759	0.777
GRAPHSIM	<b>0.787</b>	<b>0.874</b>	<b>0.534</b>	<b>0.599</b>	<b>0.058</b>	<b>0.993</b>	<b>0.992</b>	<b>0.981</b>	<b>0.743</b>	<b>0.926</b>	<b>0.828</b>	<b>0.841</b>

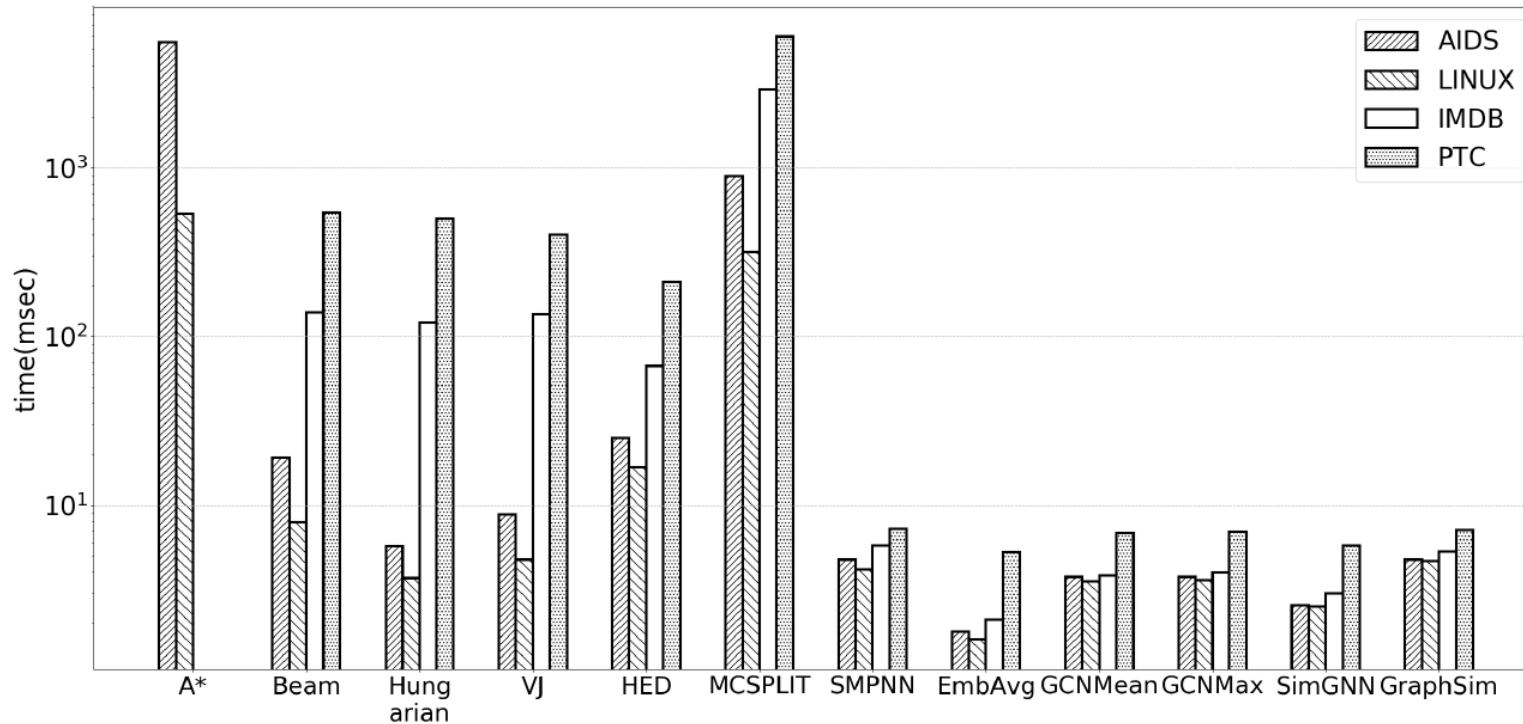
## GED Computation

Method	Aids				LINUX				IMDB			
	mse	$\rho$	p@10	p@20	mse	$\rho$	p@10	p@20	mse	$\rho$	p@10	p@20
MCSPLIT	0.000*	1.000*	1.000*	1.000*	0.000*	1.000*	1.000*	1.000*	0.000*	1.000*	1.000*	1.000*
SMPNN	4.592	0.755	0.348	0.399	3.558	0.126	0.236	0.233	11.018	0.330	0.003	0.193
EMBAVG	6.466	0.701	0.236	0.321	2.663	0.427	0.343	0.350	17.853	0.524	0.166	0.216
GCNMEAN	5.956	0.776	0.316	0.397	2.706	0.439	0.368	0.364	9.316	0.753	0.364	0.387
GCNMAX	5.525	0.782	0.328	0.403	2.466	0.543	0.440	0.394	8.234	0.796	0.435	0.441
SIMGNN	3.433	0.822	0.374	0.434	0.729	0.859	0.850	0.839	1.153	0.938	0.705	0.738
GRAPHSIM	<b>2.402</b>	<b>0.858</b>	<b>0.505</b>	<b>0.566</b>	<b>0.164</b>	<b>0.962</b>	<b>0.951</b>	<b>0.947</b>	<b>0.307</b>	<b>0.976</b>	<b>0.817</b>	<b>0.845</b>

## MCS Computation

# Running Time


- Similar to SimGNN





# Outline

---

- Introduction
- Starting from Graph Edit Distance Computation
- Graph Similarity Computation as Set Matching
- Summary 

# Take Away Messages

---

- **Node-Level Vs. Graph-Level operators**
  - Treat nodes as data points
  - Treat graphs as data points
- **Many graph tasks in addition to classification are waiting us to solve**
  - GED, MCS, graph alignment, subgraph matching, ...
  - Most of them are NP-hard problems
- **Raw node-level info will provide more information in graph proximity computation**

# Q & A

- Thanks to my collaborators:
  - Yunsheng Bai, Ting Chen, Ziniu Hu, Wei Wang

