VIRGINIA TECH

UNIVERSITY OF MINNESOTA
Driven to Discover℠

Interpretable Graph Similarity Computation via Differentiable Optimal Alignment of Node Embeddings

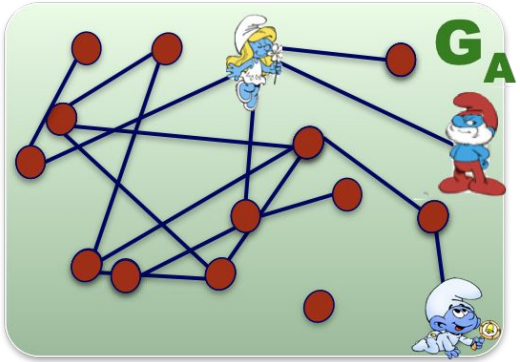
Khoa D. Doan, Saurav Manchanda, Suchismit Mahapatra, Chandan K. Reddy

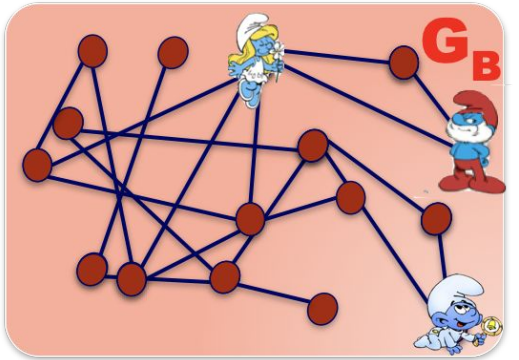
How similar are two graphs?

$$s = 1: \quad G_A = G_B$$

$$s = 0: \quad G_A <> G_B$$

Applications in intrusion detection, drug design, matching behavioral patterns, similarities among software function-call graphs for malware detection etc.



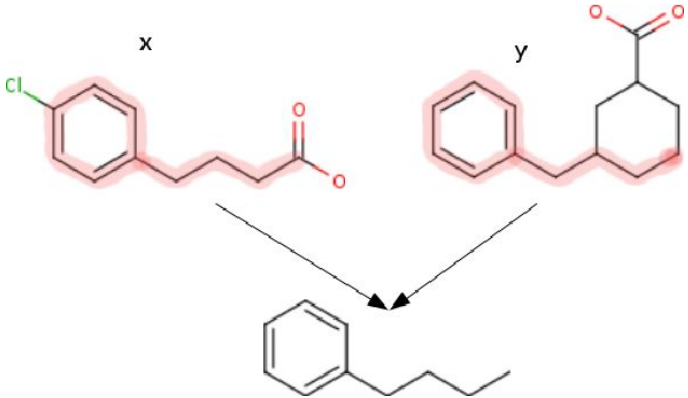


How similar are two graphs?

Intrusion detection

Behavioral patterns

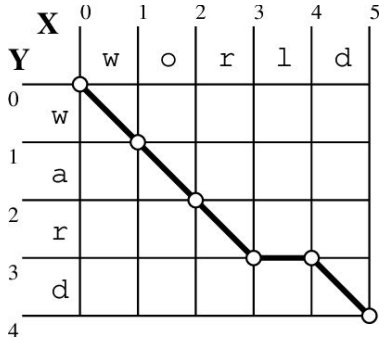
Molecule similarity



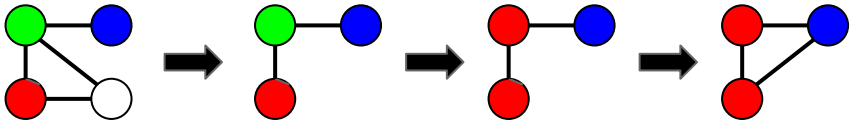
How similar are two graphs?

- Application-agnostic measures such as **GED** or **MCS** are important.
  - MCS is a special case of GED

- Explicitly explainable
  - In case of comparing molecules, GED tells the sequence of atoms/bonds to insert/delete or substitute that transforms once molecule to the other.



String edit distance is based on the sequence of edits (character insertion, deletion or substitution) to transform one string to the other.



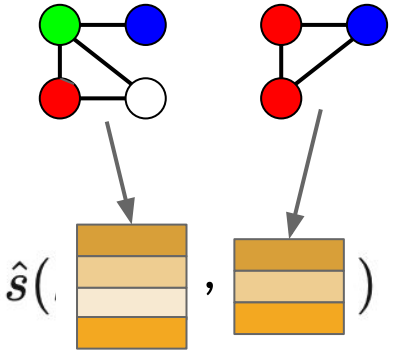
Graph Edit Distance is based on the sequence of edits (node/edge insertion, deletion or substitution) to transform one graph to the other.

Classical GED Computation

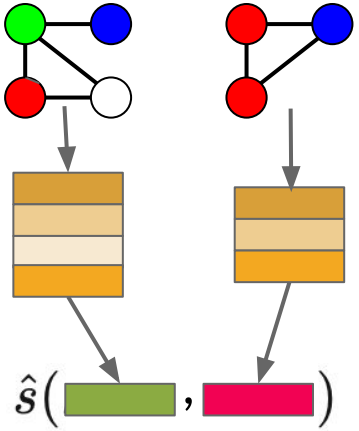
- Exact GED computation is NP-hard
  - Combinatorial Search: A*
  - Impractical for graphs with more than a few tens of nodes.
- Approximate GED computation
  - Bipartite Matching: HUNGARIAN
  - Limited to GED computation
  - Poor approximation quality (because of heuristic greedy approach)

Learning-based GED Computation

- Based on graph neural network such as GCN to first learn node representations
- Then approximate the graph similarity via



Function of node embeddings (ordering-invariant)



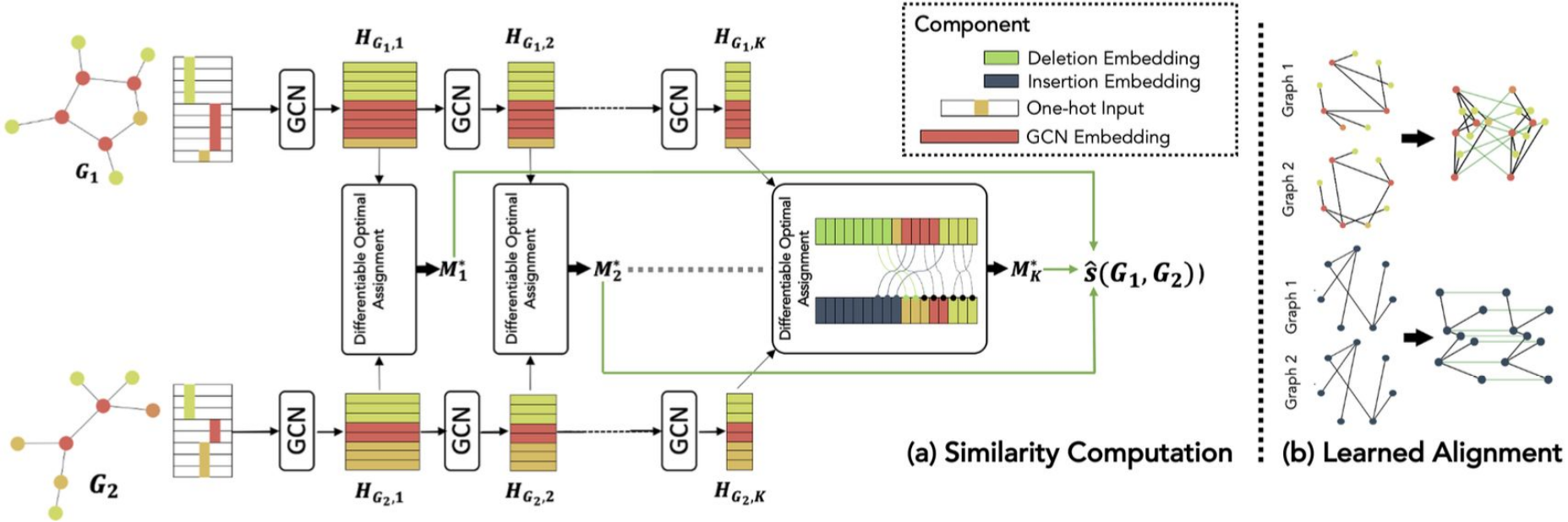
Function of graph embeddings

Special case of the first approach

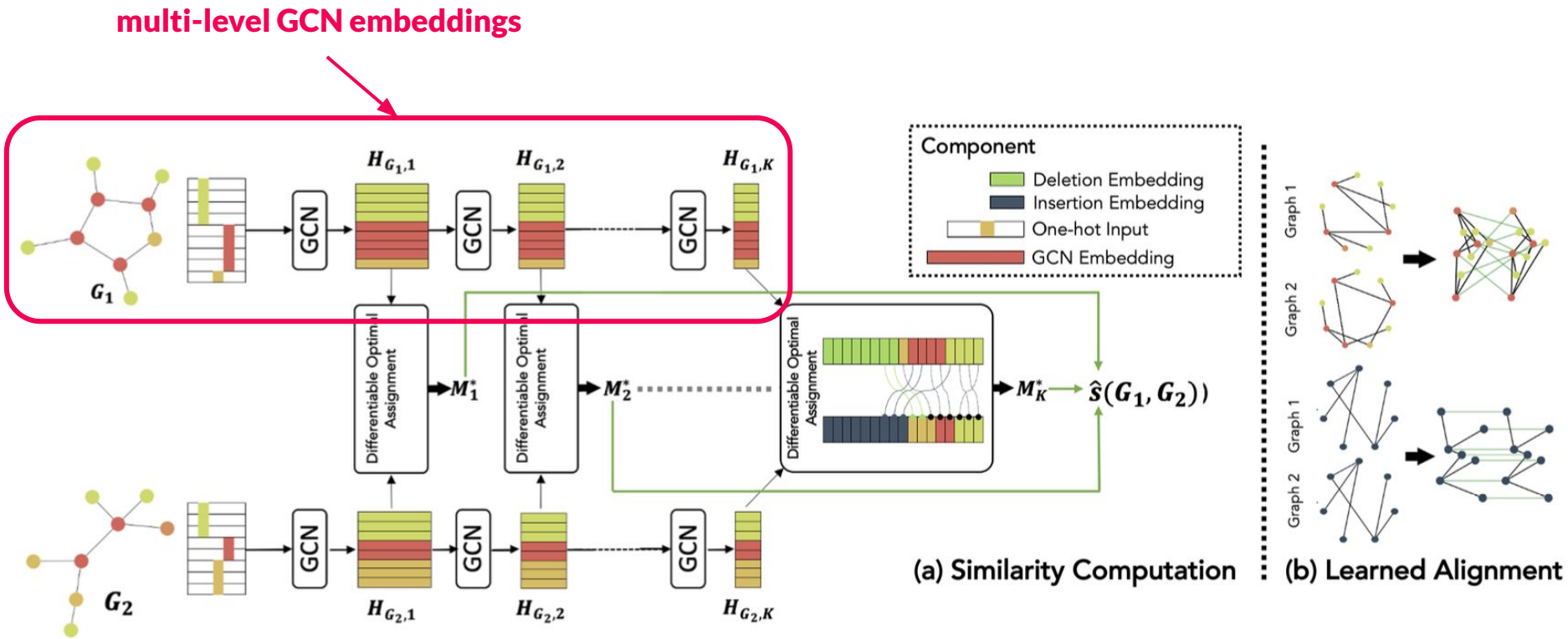
Limitations of existing learning-based methods

- Existing methods (especially the SOTAs) are based on heuristic assumptions
  - Does not capture, thus cannot explain the edit operations of the GED.
  - Sensitive to ordering of nodes' embeddings (i.e. heuristics violate the ordering-invariant)
- Graph-matching approaches do not provide suitable explanation for the GED
  - Do not learn bijective mapping between nodes without labeled node-node correspondence.

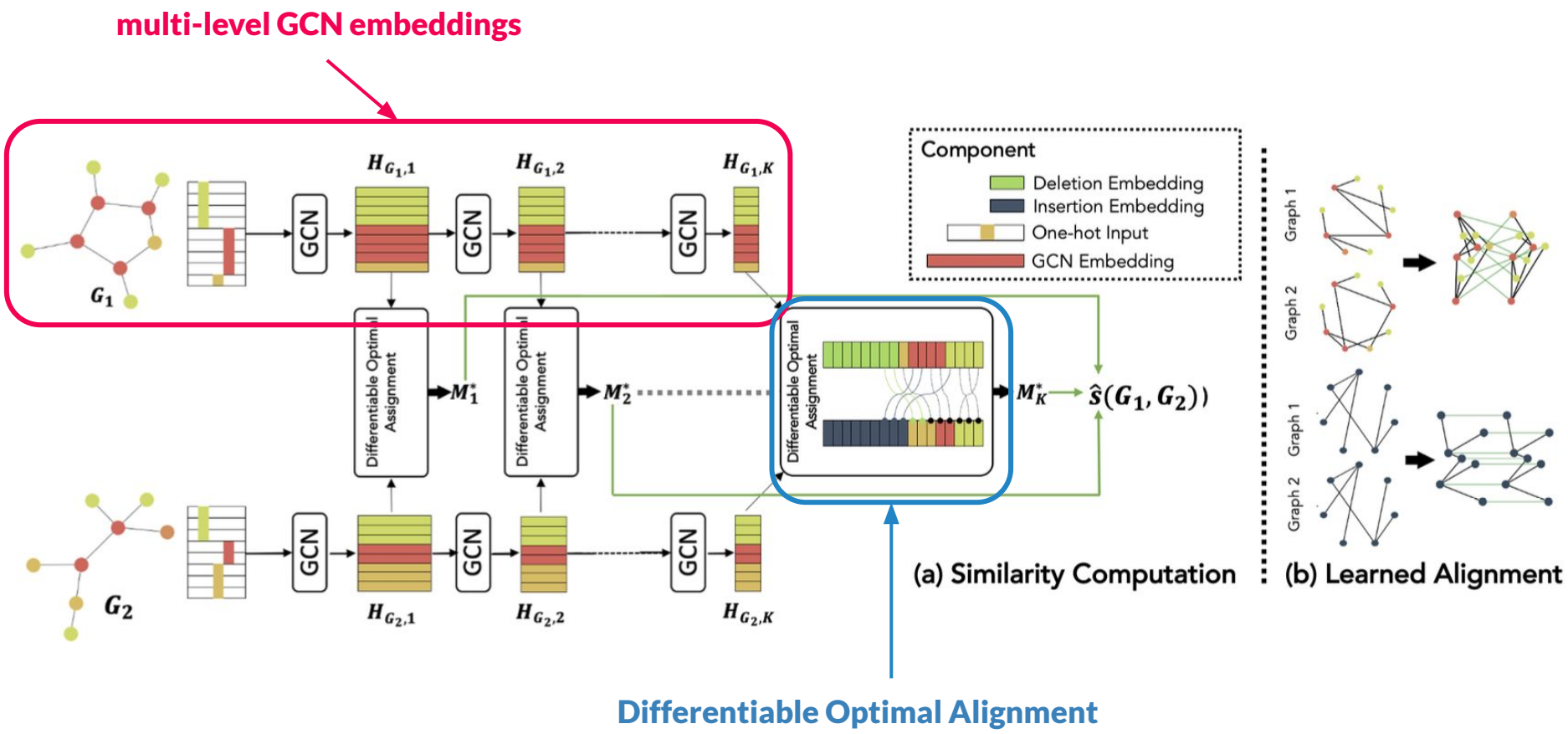
Graph Optimal Transport Similarity Computation



Graph Optimal Transport Similarity Computation



Graph Optimal Transport Similarity Computation



Similarity via optimal assignment computation

Given the transformation cost matrix

$$\mathbf{C}_{G_1,G_2,k} = \begin{bmatrix} \mathbf{c}_{k,1,1} & \cdots & \mathbf{c}_{k,1,N} \\ \vdots & \ddots & \vdots \\ \mathbf{c}_{k,N,1} & \cdots & \mathbf{c}_{k,N,N} \end{bmatrix}$$

e.g. cosine distance

Optimal transformation cost is defined as:

$$\mathbf{L}_{G_1,G_2,k} = \min_{\mathbf{M}} \mathbf{M} \circ \mathbf{C}_{G_1,G_2,k} \Rightarrow \frac{\partial \mathbf{L}_{G_1,G_2,k}}{\partial \theta} = \frac{\partial \mathbf{M}_k^*}{\partial \theta} \mathbf{C}_{G_1,G_2,k} + \mathbf{M}_k^* \frac{\partial \mathbf{C}_{G_1,G_2,k}}{\partial \theta}$$

Similarity via optimal assignment computation

Given the similarity matrix

$$\mathbf{C}_{G_1,G_2,k} = \begin{bmatrix} \mathbf{c}_{k,1,1} & \cdots & \mathbf{c}_{k,1,N} \\ \vdots & \ddots & \vdots \\ \mathbf{c}_{k,N,1} & \cdots & \mathbf{c}_{k,N,N} \end{bmatrix}$$

Problem:

(1) cost matrix only accounts for transformation of one node to another node

(2) Unstable solution (*M is doubly stochastic matrix*)

Optimal transformation cost is defined as:

$$\mathbf{L}_{G_1,G_2,k} = \min_{\mathbf{M}} \mathbf{M} \circ \mathbf{C}_{G_1,G_2,k} \Rightarrow \frac{\partial \mathbf{L}_{G_1,G_2,k}}{\partial \theta} = \frac{\partial \mathbf{M}_k^*}{\partial \theta} \mathbf{C}_{G_1,G_2,k} + \mathbf{M}_k^* \frac{\partial \mathbf{C}_{G_1,G_2,k}}{\partial \theta}$$

Augmented similarity matrix

node assignment

node deletion

$$\mathbf{C}_{G_1,G_2,k} = \left[\begin{array}{ccc|ccc} \mathbf{c}_{k,1,1} & \cdots & \mathbf{c}_{k,1,N_2} & \mathbf{d}_{k,1} & \cdots & \infty \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{c}_{k,N_1,1} & \cdots & \mathbf{c}_{k,N_1,N_2} & \infty & \cdots & \mathbf{d}_{k,N_1} \\ \hline \mathbf{a}_{k,1} & \cdots & \infty & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \infty & \cdots & \mathbf{a}_{k,N_2} & 0 & \cdots & 0 \end{array}\right]$$

node addition

$$\mathbf{c}_{k,i,j} = \mathbf{c}(\mathbf{h}_{G_1,k,i}, \mathbf{h}_{G_2,k,j})$$

$$\mathbf{a}_{k,i} = c(\mathbf{h}_{G_2,k,i}, \mathbf{h}_{k,a})$$

$$\mathbf{d}_{k,i} = c(\mathbf{h}_{G_1,k,i}, \mathbf{h}_{k,d})$$

Chosen as global embedding vectors

Augmented similarity matrix

node assignment

node deletion

$$\mathbf{C}_{G_1,G_2,k} = \left[\begin{array}{ccc|ccc} \mathbf{c}_{k,1,1} & \cdots & \mathbf{c}_{k,1,N_2} & \mathbf{d}_{k,1} & \cdots & \infty \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{c}_{k,N_1,1} & \cdots & \mathbf{c}_{k,N_1,N_2} & \infty & \cdots & \mathbf{d}_{k,N_1} \\ \hline \mathbf{a}_{k,1} & \cdots & \infty & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \infty & \cdots & \mathbf{a}_{k,N_2} & 0 & \cdots & 0 \end{array}\right]$$

node addition

$$\mathbf{c}_{k,i,j} = \mathbf{c}(\mathbf{h}_{G_1,k,i}, \mathbf{h}_{G_2,k,j})$$

$$\mathbf{a}_{k,i} = c(\mathbf{h}_{G_2,k,i}, \mathbf{h}_{k,a})$$

$$\mathbf{d}_{k,i} = c(\mathbf{h}_{G_1,k,i}, \mathbf{h}_{k,d})$$

Chosen as global embedding vectors

M is now a permutation matrix

Simpler and more stable derivative

$$\frac{\partial \mathbf{L}_{G_1,G_2,k}}{\partial \theta} = \mathbf{M}_k^* \frac{\partial \mathbf{C}_{G_1,G_2,k}}{\partial \theta}$$

Experimental Setup

- Multiple tasks
  - Similarity approximation: GED, MCS.
  - Similarity search/ranking.
- Datasets from different domains
  - Chemical compound graphs: AIDS (~100k pairs), PTC (~1M pairs).
  - LINUX kernel dependency Graphs (~0.5M pairs).
  - Ego-networks of actors IMDB (~2.25M pairs).
- Evaluation metrics
  - Approximation: MSE, MAE
  - Ranking: Spearman's Rank Correlation, Kendall's Rank Correlation Coefficient, Precision at k

Compare with several representative baselines

- Classical Approximate GED approaches
  - Hungarian (Bipartite), Hausdorff
- Learning-based graph similarity approaches
  - Graph-embedding approaches: GCNMean, GCNMax, GCNGated and SimGNN
  - Node-embedding approaches: GraphSim (SOTA)
- Graph matching approach: GMN

SOTA Approximation Performance

Exact Ground-truth available

Only approximated ground-truth available

	AIDS		LINUX		PTC		IMDB	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ExactGT	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	–	–	–	–
Bipartite	130.68 ± 9.91	337.28 ± 16.11	266.11 ± 5.60	485.26 ± 6.66	60.68 ± 1.10	177.72 ± 3.36	13.22 ± 0.60	69.23 ± 3.93
Hausdorff	77.48 ± 5.64	255.39 ± 11.58	58.07 ± 1.46	221.84 ± 3.41	171.57 ± 7.94	388.30 ± 9.62	24.98 ± 1.71	90.91 ± 4.39
EmbMean	8.34 ± 0.50	69.60 ± 2.15	15.29 ± 1.92	151.12 ± 4.23	10.98 ± 1.22	79.92 ± 7.07	63.72 ± 2.37	102.63 ± 4.24
EmbMax	9.37 ± 0.24	74.44 ± 0.90	13.11 ± 1.88	140.73 ± 4.27	10.60 ± 1.52	80.18 ± 8.29	54.74 ± 1.94	98.12 ± 3.38
EmbGated	5.92 ± 0.32	65.46 ± 2.35	14.54 ± 2.01	119.01 ± 5.17	5.71 ± 2.21	51.05 ± 16.10	6.28 ± 3.23	99.23 ± 4.87
GMN	5.01 ± 0.25	61.23 ± 6.56	7.23 ± 0.94	55.78 ± 3.38	5.82 ± 1.89	56.7 ± 8.20	74.12 ± 4.22	168.02 ± 9.73
SimGNN	2.70 ± 0.30	38.34 ± 0.30	4.43 ± 0.62	50.41 ± 3.52	1.98 ± 0.43	27.85 ± 3.02	9.05 ± 4.12	78.01 ± 3.01
GraphSim	8.60 ± 0.33	37.06 ± 0.21	4.75 ± 0.78	45.72 ± 5.12	5.85 ± 0.83	55.17 ± 4.79	6.87 ± 4.02	111.39 ± 7.96
GOTSim	**2.36 ± 0.12**	**35.19 ± 0.15**	**4.25 ± 0.60**	**44.27 ± 3.23**	**1.90 ± 0.43**	**26.79 ± 0.43**	**5.92 ± 3.19**	**75.31 ± 3.33**

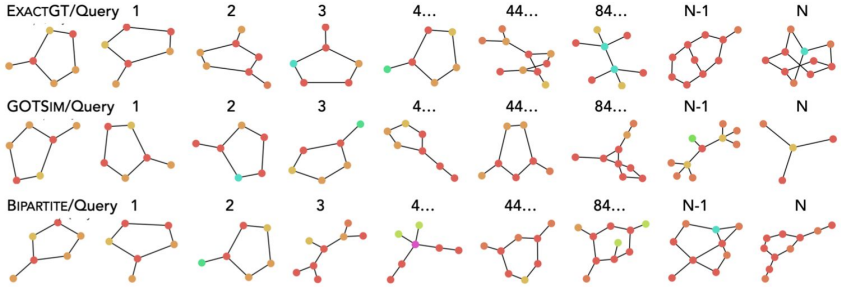
- GOTSim achieves lowest approximation errors
- When exact ground-truth is available, GOTSim achieves even better approximation than classical, approximate GED methods
- Similar observations when approximating MCS.

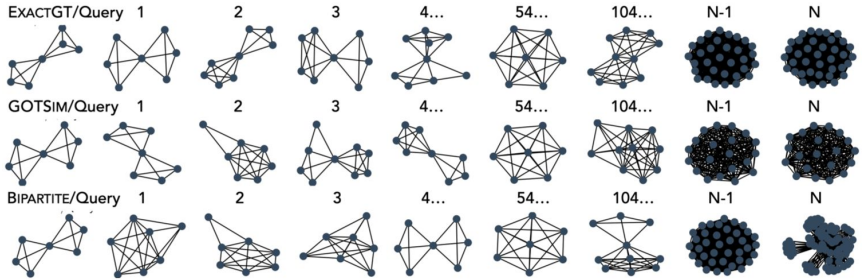
SOTA Retrieval Performance

Exact Ground-truth available

Only approximated ground-truth available

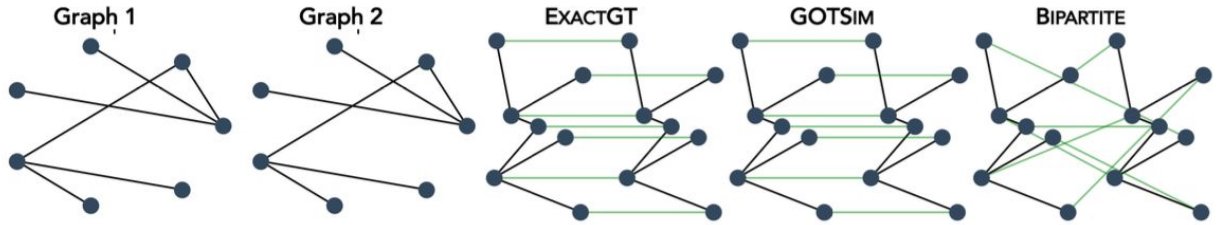
	AIDS			LINUX			PTC		
	τ	ρ	P@10	τ	ρ	P@10	τ	ρ	P@10
ExactGT	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	–	–	–
Bipartite	0.26 ± 0.02	0.37 ± 0.02	0.32 ± 0.04	0.32 ± 0.02	0.43 ± 0.03	0.45 ± 0.03	0.65 ± 0.02	0.82 ± 0.02	0.79 ± 0.03
Hausdorff	0.48 ± 0.02	0.63 ± 0.03	0.54 ± 0.05	0.78 ± 0.01	0.88 ± 0.01	0.78 ± 0.05	0.78 ± 0.01	0.91 ± 0.01	0.90 ± 0.02
EmbMean	0.41 ± 0.04	0.53 ± 0.04	0.60 ± 0.10	0.55 ± 0.02	0.60 ± 0.02	0.51 ± 0.01	0.16 ± 0.04	0.23 ± 0.06	0.41 ± 0.08
EmbMax	0.42 ± 0.02	0.57 ± 0.01	0.61 ± 0.06	0.57 ± 0.03	0.65 ± 0.01	0.71 ± 0.01	0.21 ± 0.03	0.29 ± 0.03	0.49 ± 0.03
EmbGated	0.43 ± 0.02	0.66 ± 0.02	0.70 ± 0.04	0.58 ± 0.03	0.88 ± 0.01	0.81 ± 0.01	0.66 ± 0.10	0.81 ± 0.13	0.84 ± 0.08
GMN	0.47 ± 0.05	0.69 ± 0.02	0.72 ± 0.02	0.78 ± 0.03	0.88 ± 0.01	0.80 ± 0.01	0.40 ± 0.02	0.71 ± 0.03	0.70 ± 0.04
SimGNN	0.67 ± 0.03	0.82 ± 0.02	0.84 ± 0.04	0.80 ± 0.01	0.92 ± 0.01	0.82 ± 0.05	0.80 ± 0.02	0.93 ± 0.01	0.91 ± 0.01
GraphSim	0.68 ± 0.03	0.57 ± 0.03	0.76 ± 0.03	0.83 ± 0.02	0.92 ± 0.02	0.84 ± 0.03	0.79 ± 0.01	0.91 ± 0.01	0.91± 0.02
GOTSim	**0.72 ± 0.02**	**0.86± 0.02**	**0.87± 0.03**	**0.89 ± 0.02**	**0.92± 0.01**	**0.86± 0.02**	**0.82 ± 0.01**	**0.95± 0.01**	**0.94± 0.01**





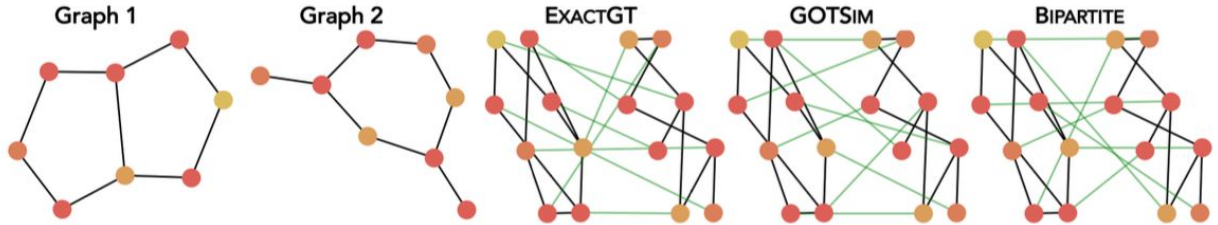
Learned Node Assignment (approximating GED)

LINUX

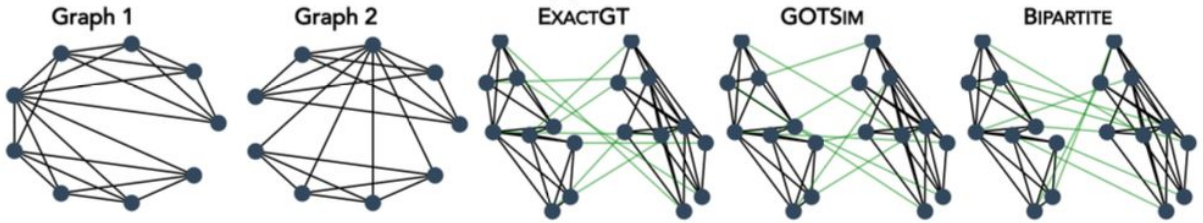


Capture the exact alignment of isomorphic graphs

PTC



IMDB



Capture similar alignments as those of the ground-truth GED computation, for both multi-labeled and unlabeled cases

Conclusions

- Proposed a learning-based, graph-similarity method GOTSIM
  - Graph similarity is approximated via the a differentiable, minimal transformation cost from one graph to another
  - GOTSIM can be used with any existing GNN-based embedding methods
  - GOTSIM achieves SOTA performance on both graph approximation and retrieval/ranking tasks.
  - GOTSIM's results are interpretable and we can visualize the learned node assignments between graphs.
- GOTSIM can have useful applications in domains where explainable alignment or sequence of edits are required along with the similarity score.

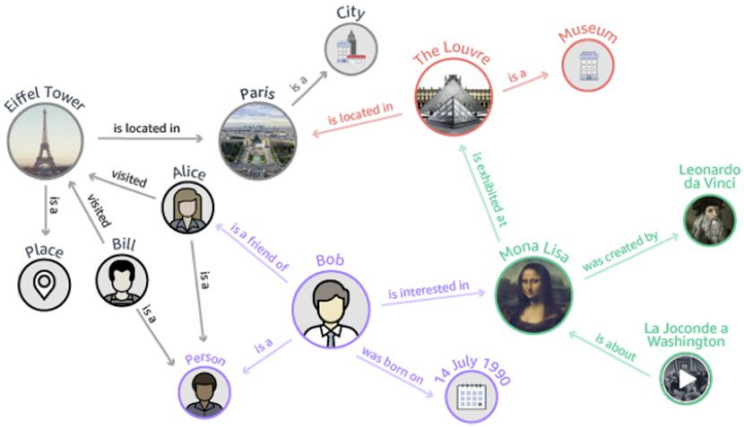
Thank you!

Contact
Khoa D. Doan
khoadoan@vt.edu

Saurav Manchanda
manch043@umn.edu

Code
https://github.com/khoadoan/GraphOTSim

Graphs in the wild



- Graphs in health, database analytics, finance, drug design and recommender systems.
- Tasks on graphs:
  - Graph classification: Protein function prediction
  - Node classification: Unreliable credit card owners
  - Link prediction: Product recommendation, knowledge graph (KG) completion
  - Graph similarity: Drug design, similarities among software function-call graphs for malware detection

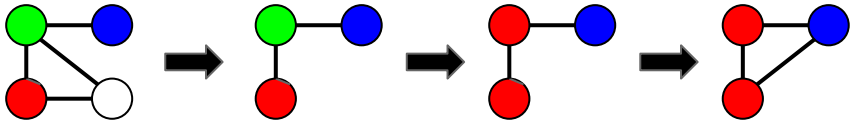
Importance of Graph Similarity Computation

- Computing graph similarity has several applications
  - finding the chemical compounds that are most similar to a query compound
  - finding
- Among these, application-agnostic measures such as GED or MCS are important.

Outline

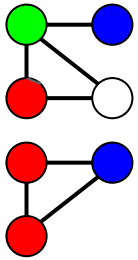
- Introduction
- Existing solutions and limitations
- The proposed approach: GOTSim
- Experimental evaluation
- Conclusions

GED Computation



Learning-based GED Computation

- Based on graph neural network such as GCN to first learn node representations
- Then approximate the graph similarity via
  - Pair of graphs' vector representation (e.g., average of node embeddings)
  - Pair of bag-of-vector representations



Given two graphs $G_1 = \{\mathcal{V}_1, A_1\}$ and $G_2\{\mathcal{V}_2, A_2\}$