Boosting Graph Structure Learning with Dummy Nodes

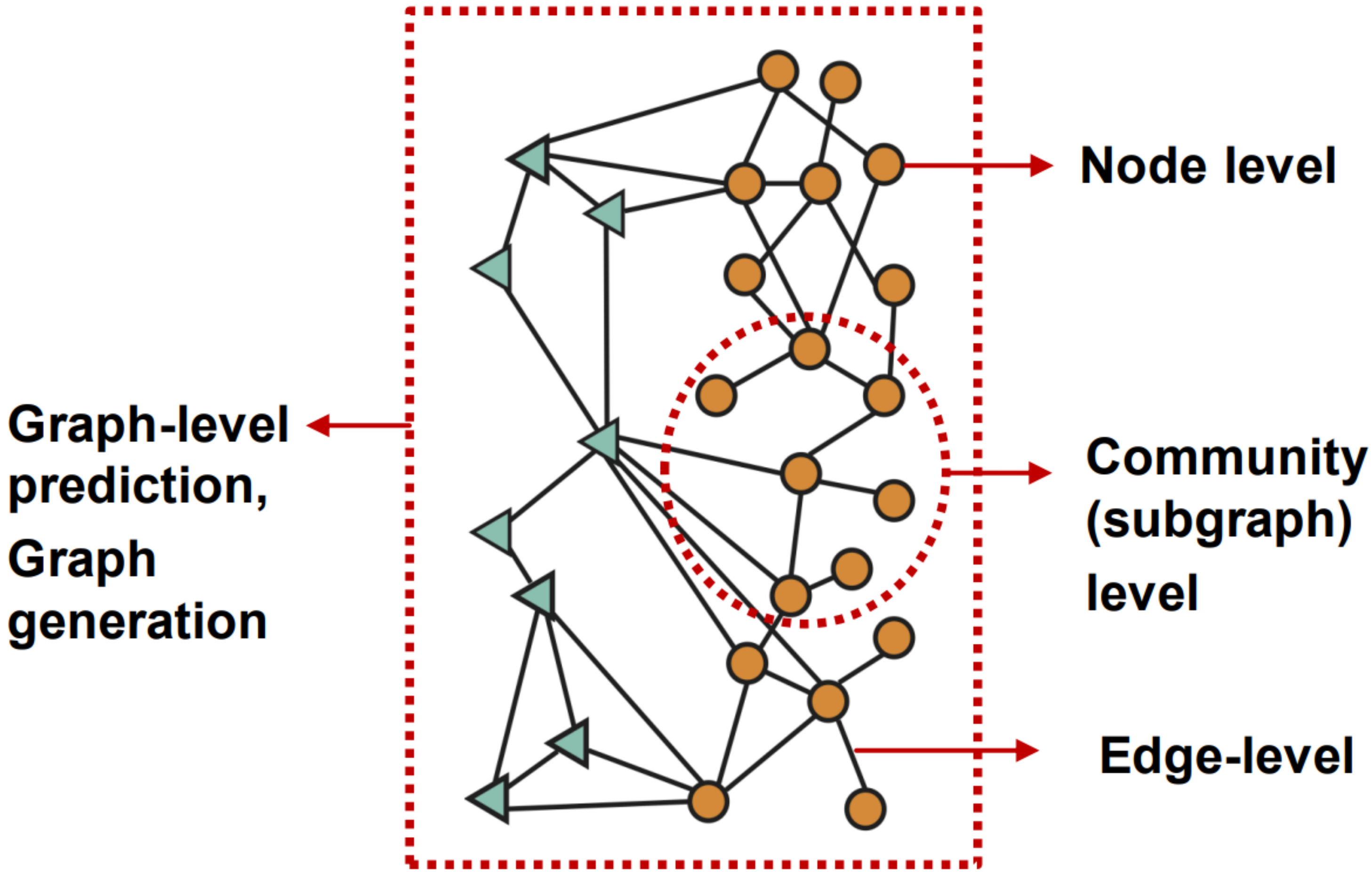
Xin Liu[1], Jiayang Cheng[1], Yangqiu Song[1], Xin Jiang[2]

[1]The Hong Kong University of Science and Technology

[2]Huawei Noah's Ark Lab

香 港 科 技 大 學
THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

Graph Learning



Graph Structure Learning Algorithms

- Graph Kernels
  - Shortest-path Kernel (SP) (Borgwardt & Kriegel, 2005)
  - Graphlet Kernel (GR) (Shervashidze et al., 2009)
  - Weisfeiler-Lehman Subtree Kernel (k-WL) (Shervashidze et al., 2011)
  - δ-k-dimensional Local WL algorithm (δ-LWL) (Morris et al., 2020)

Pro: expressively powerful

Cons: non-inductive, vertex-centric, and computationally expensive

Graph Structure Learning Algorithms

- Graph Kernels
  - Shortest-path Kernel (SP) (Borgwardt & Kriegel, 2005)
  - Graphlet Kernel (GR) (Shervashidze et al., 2009)
  - Weisfeiler-Lehman Subtree Kernel (k-WL) (Shervashidze et al., 2011)
  - δ-k-dimensional Local WL algorithm (δ-LWL) (Morris et al., 2020)

  Pro: expressively powerful

  Cons: non-inductive, vertex-centric, and computationally expensive

- Graph Neural Networks
  - Graph Isomorphism Network (GIN) (Xu et al., 2019) and RGIN (Liu et al., 2020)
  - DiffPool (Ying et al., 2018)
  - DMPNN (Liu et al., 2022)
  - EASN (Bevilacqua et al., 2021)

  Pro: high-efficient in parallel

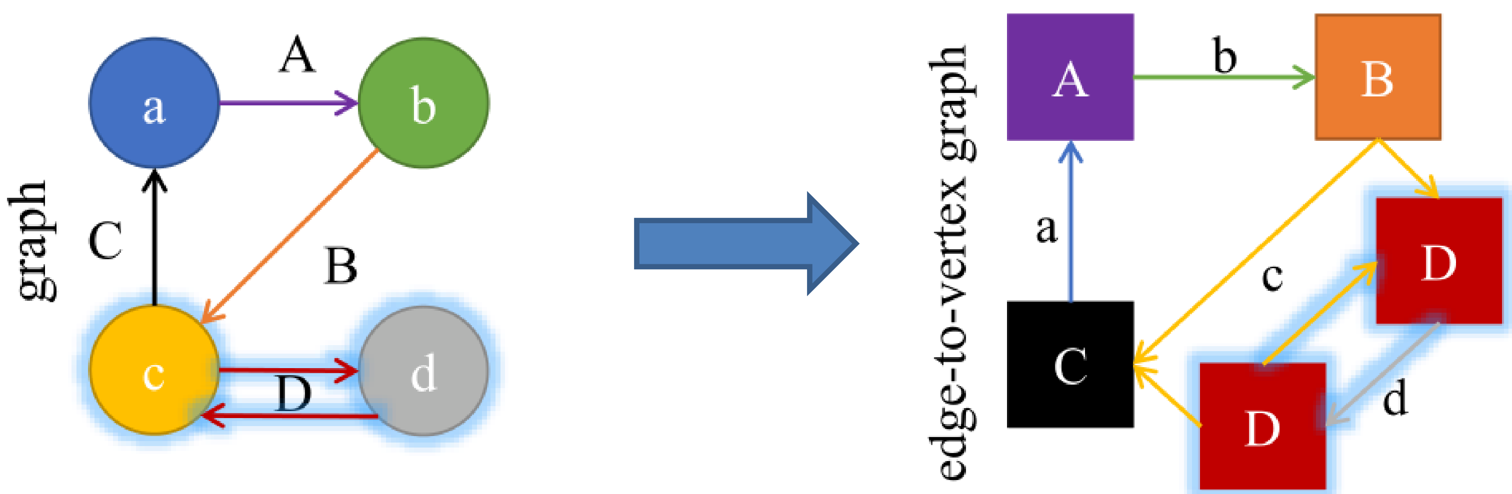
  Cons: over-parameterization and over-smoothing

Preliminary

- Directed Connected Heterogeneous Graphs
- Similar Structures → Similar Edge Structures

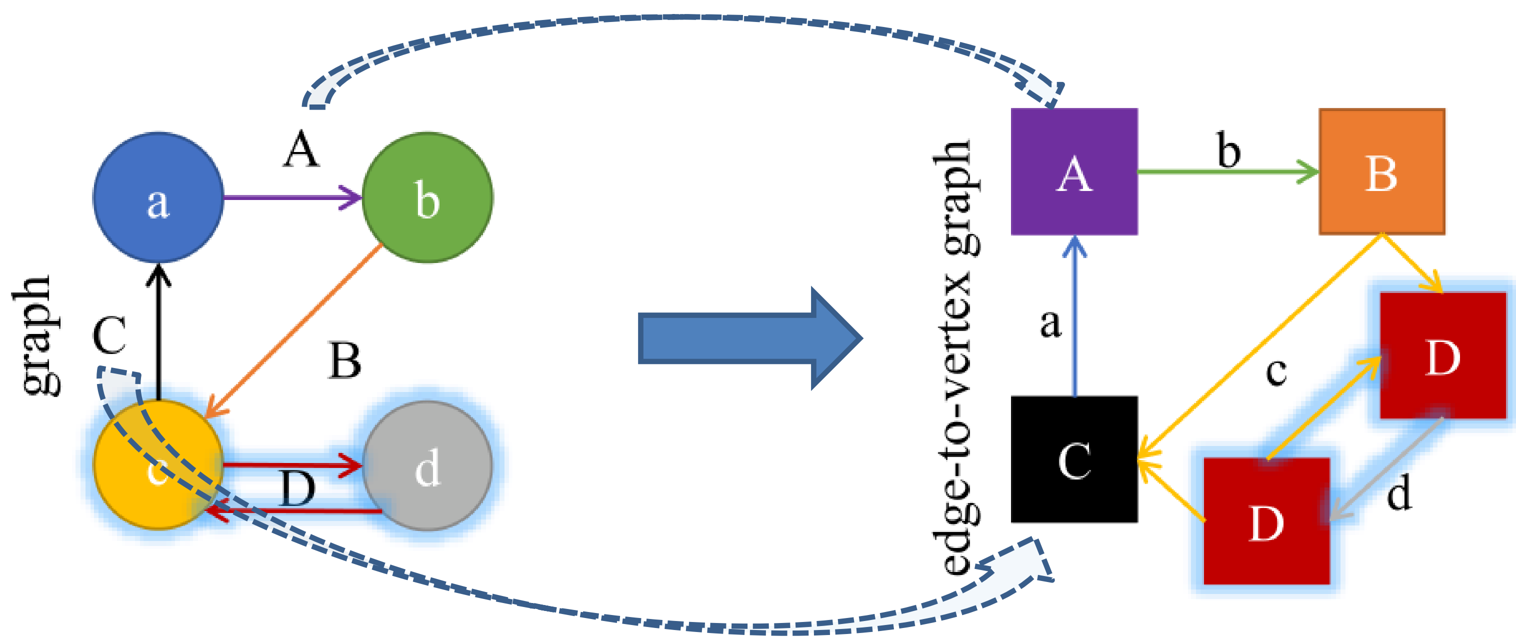
Preliminary

- Directed Connected Heterogeneous Graphs
- Similar Structures → Similar Edge Structures
- Edge-to-vertex Transform L
  - $\mathcal{G} \rightarrow \mathcal{H}$: vertices/edges in the line graph $\mathcal{H}$ correspond to edges/vertices in the original graph $\mathcal{G}$



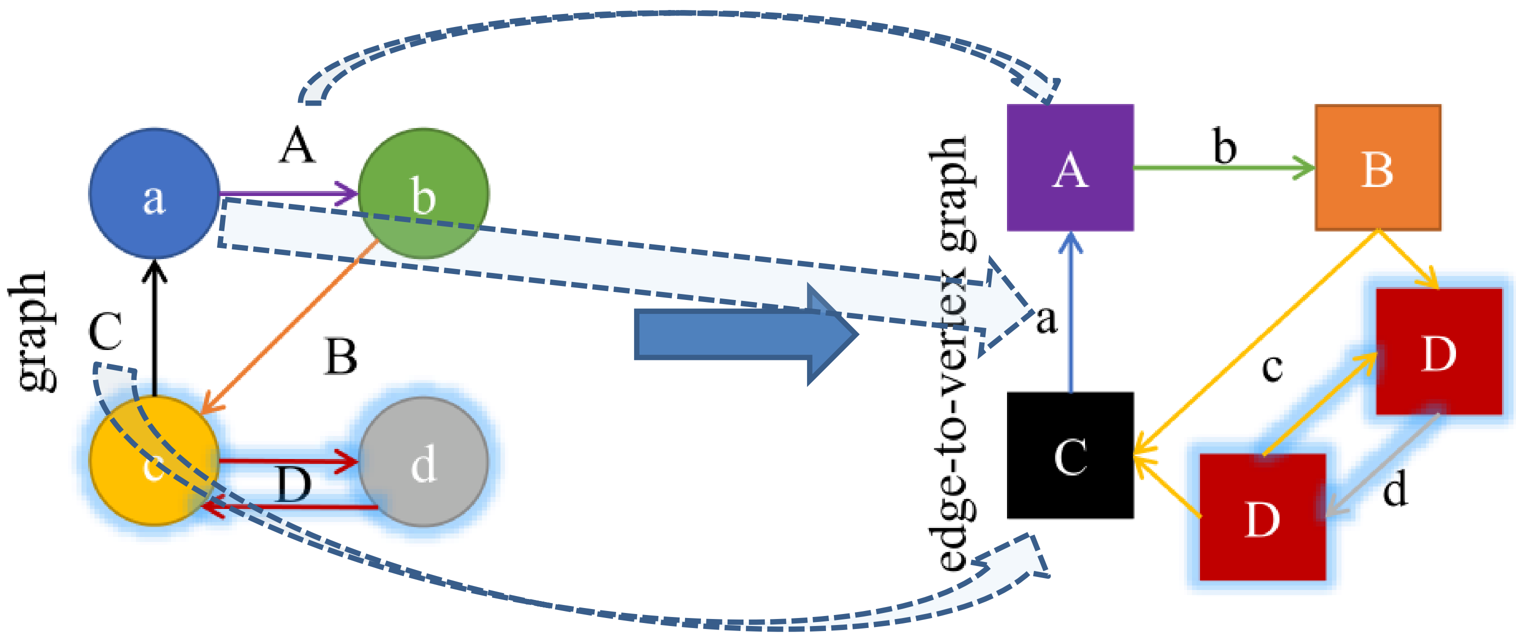
Preliminary

- Directed Connected Heterogeneous Graphs
- Similar Structures → Similar Edge Structures
- Edge-to-vertex Transform L
  - $\mathcal{G} \rightarrow \mathcal{H}$: vertices/edges in the line graph $\mathcal{H}$ correspond to edges/vertices in the original graph $\mathcal{G}$



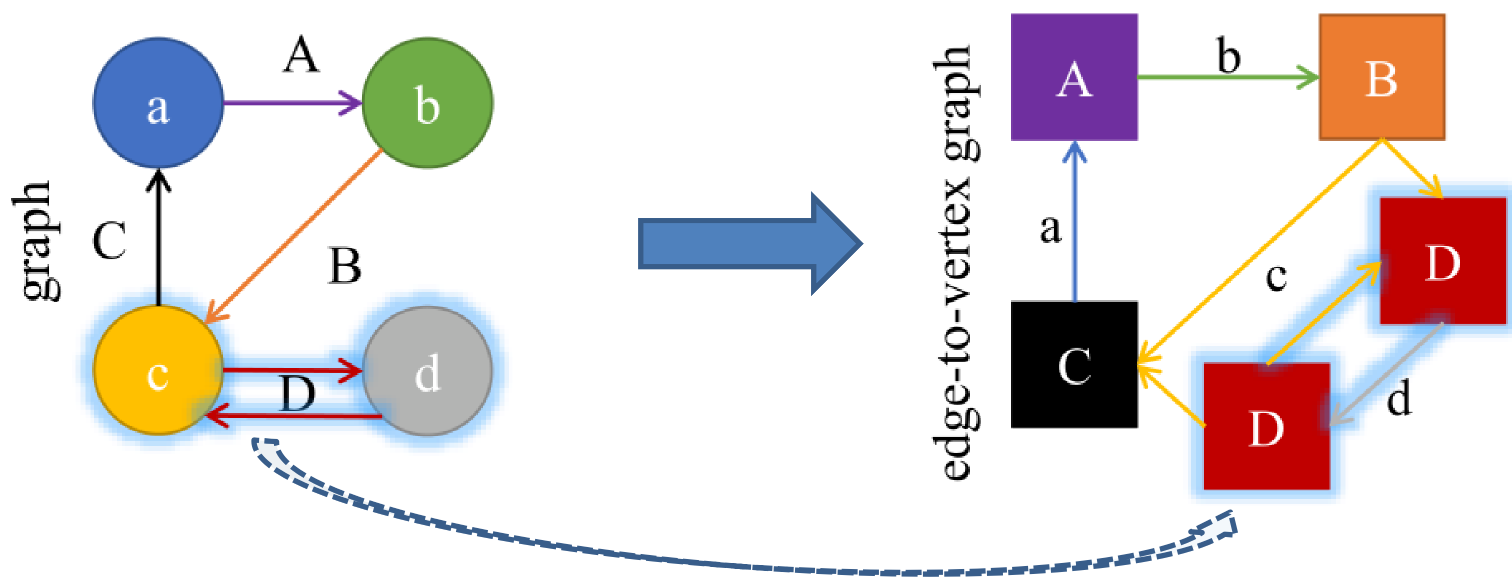
Preliminary

- Directed Connected Heterogeneous Graphs
- Similar Structures → Similar Edge Structures
- Edge-to-vertex Transform L
  - $\mathcal{G} \rightarrow \mathcal{H}$: vertices/edges in the line graph $\mathcal{H}$ correspond to edges/vertices in the original graph $\mathcal{G}$



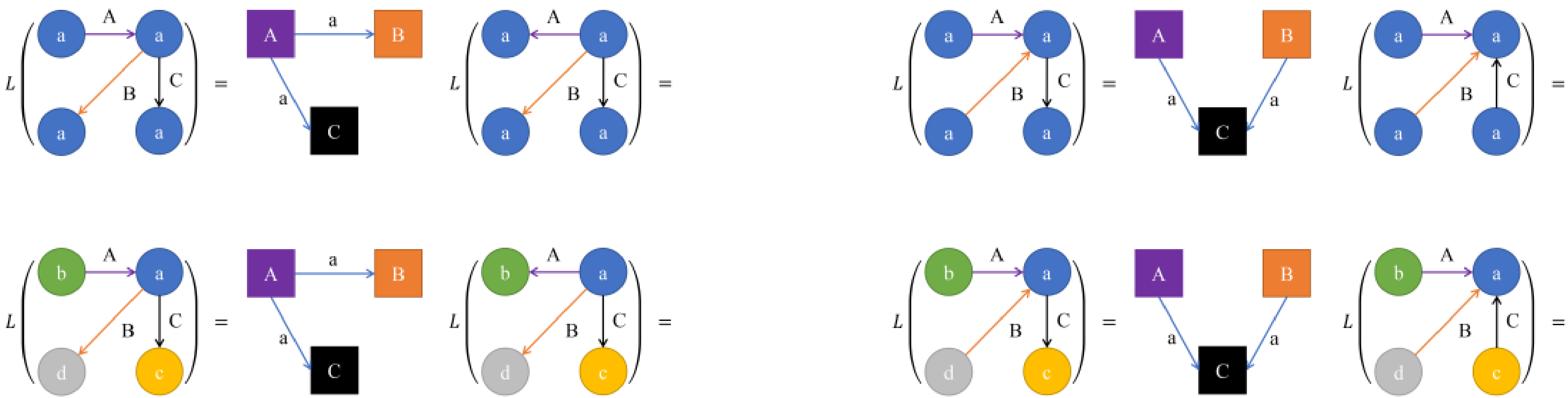
Preliminary

- Directed Connected Heterogeneous Graphs
- Similar Structures → Similar Edge Structures
- Edge-to-vertex Transform L
  - $\mathcal{G} \rightarrow \mathcal{H}$: vertices/edges in the line graph $\mathcal{H}$ correspond to edges/vertices in the original graph $\mathcal{G}$



Non-injective Edge-to-vertex Transforms

- Information Lossless in Edge-to-vertex Transforms



(a) one vertex with 0 indegree, two vertices with 0 outdegree

(b) one vertex with 0 indegree, three vertices with 0 outdegree

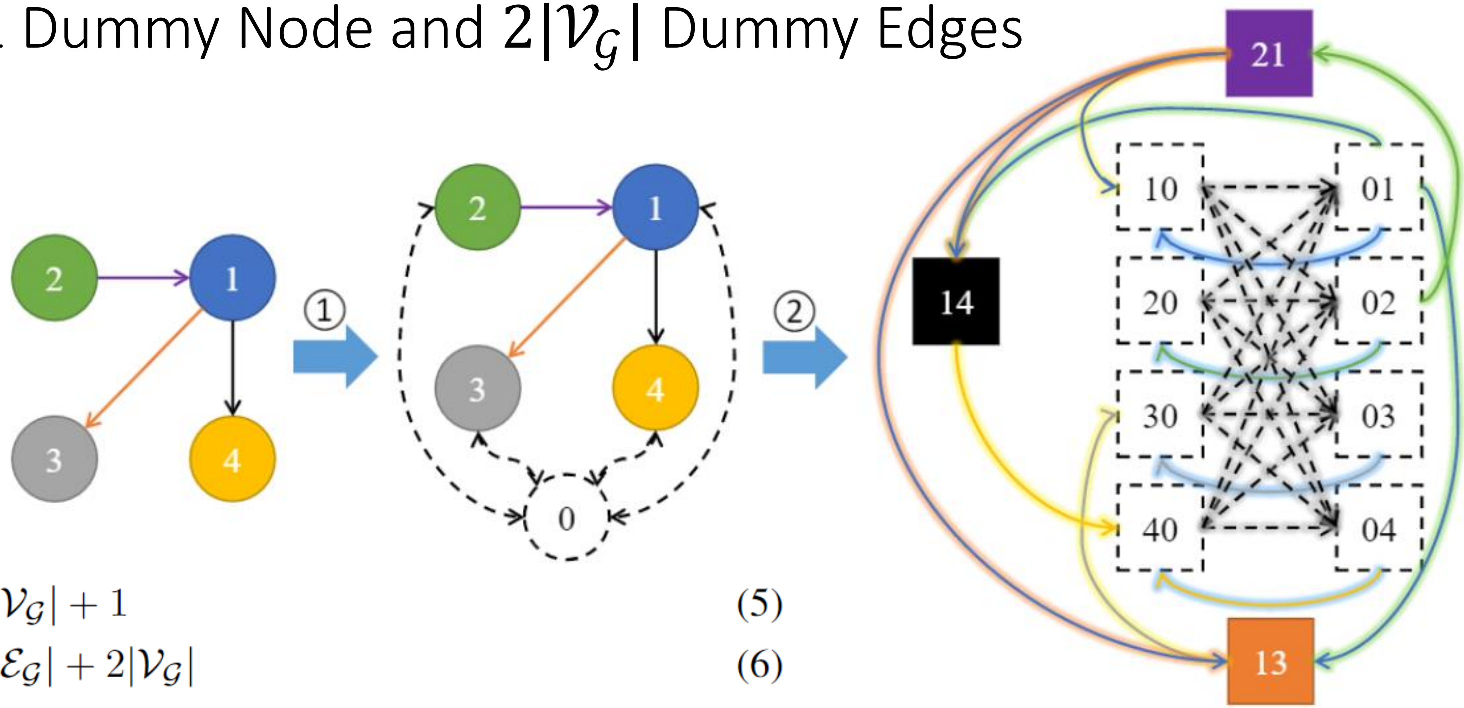
(c) two vertices with 0 indegree, one vertex with 0 outdegree

(d) three vertices with 0 indegree, one vertex with 0 outdegree

non-injection and information lossless

Our Solution

- Add 1 Dummy Node and $2|\mathcal{V}_{\mathcal{G}}|$ Dummy Edges



$$|\mathcal{V}_{\mathcal{G}'}| = |\mathcal{V}_{\mathcal{G}}| + 1 \quad (5)$$

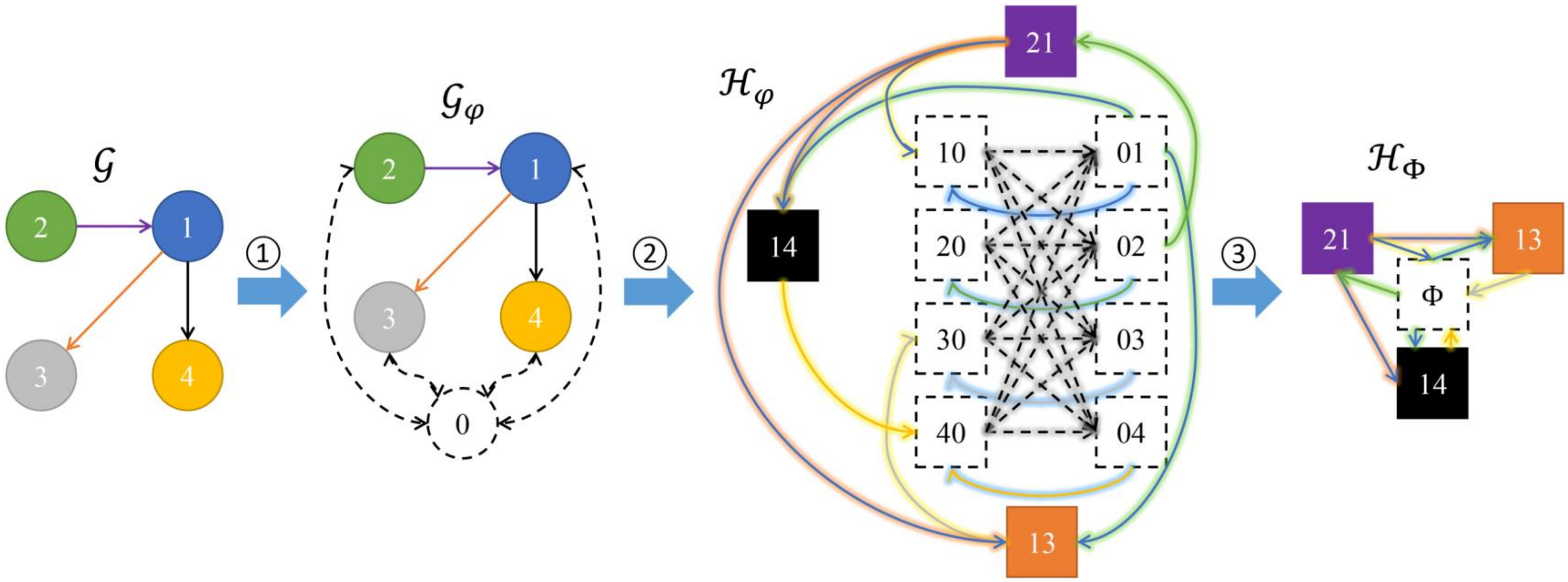
$$|\mathcal{E}_{\mathcal{G}'}| = |\mathcal{E}_{\mathcal{G}}| + 2|\mathcal{V}_{\mathcal{G}}| \quad (6)$$

$$\begin{aligned} |\mathcal{V}_{\mathcal{H}'}| &= |\mathcal{E}_{\mathcal{G}'}| \\ &= |\mathcal{E}_{\mathcal{G}}| + 2|\mathcal{V}_{\mathcal{G}}| = |\mathcal{V}_{\mathcal{H}}| + {\color{red}2|\mathcal{V}_{\mathcal{G}}|} \quad (7) \end{aligned}$$

$$\begin{aligned} |\mathcal{E}_{\mathcal{H}'}| &= |\mathcal{V}_{\mathcal{G}}|^2 + \sum_{v \in \mathcal{V}_{\mathcal{G}}} (d_v^- + 1) \cdot (d_v^+ + 1) \\ &= |\mathcal{V}_{\mathcal{G}}|^2 + \sum_{v \in \mathcal{V}_{\mathcal{G}}} d_v^- \cdot d_v^+ + \sum_{v \in \mathcal{V}_{\mathcal{G}}} (d_v^- + d_v^+) + |\mathcal{V}_{\mathcal{G}}| \\ &= |\mathcal{E}_{\mathcal{H}}| + {\color{red}|\mathcal{V}_{\mathcal{G}}|^2} + {\color{red}|\mathcal{V}_{\mathcal{G}}|} + {\color{red}2|\mathcal{V}_{\mathcal{E}}|} \quad (8) \end{aligned}$$

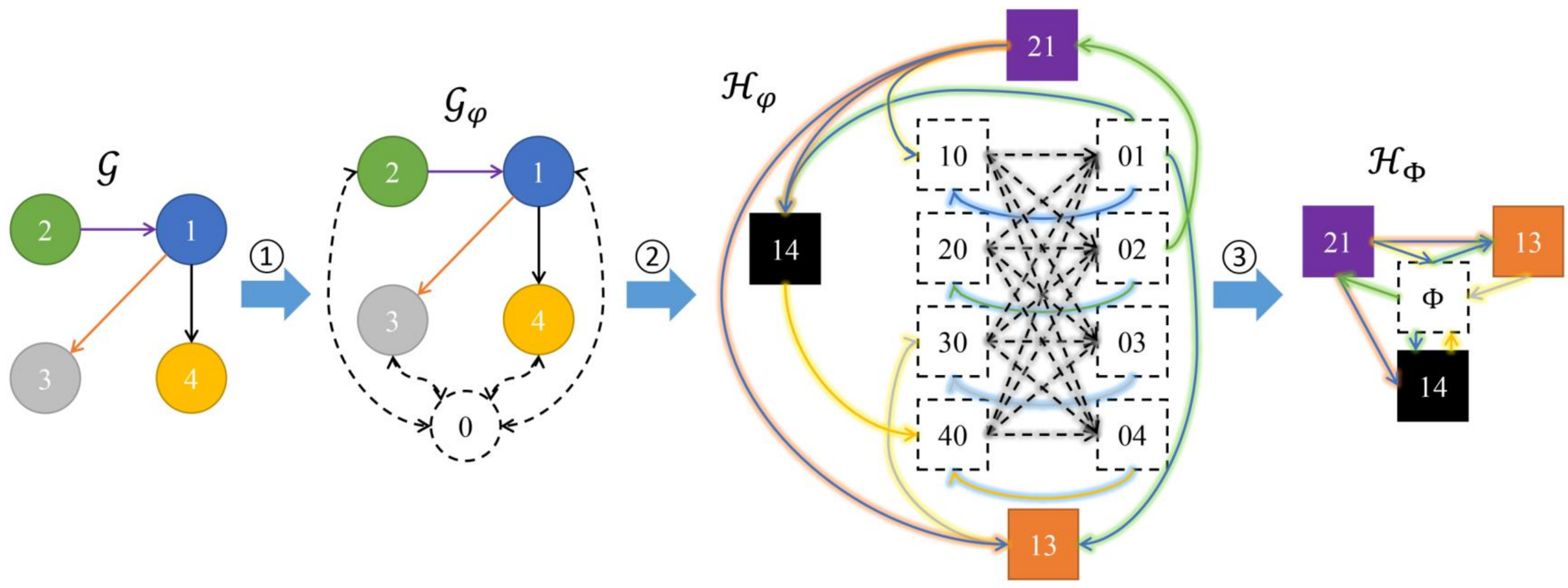
computation disaster

New Edge-to-vertex Transform



(a) edge-to-vertex transform L_Φ

New Edge-to-vertex Transform



(a) edge-to-vertex transform L_Φ

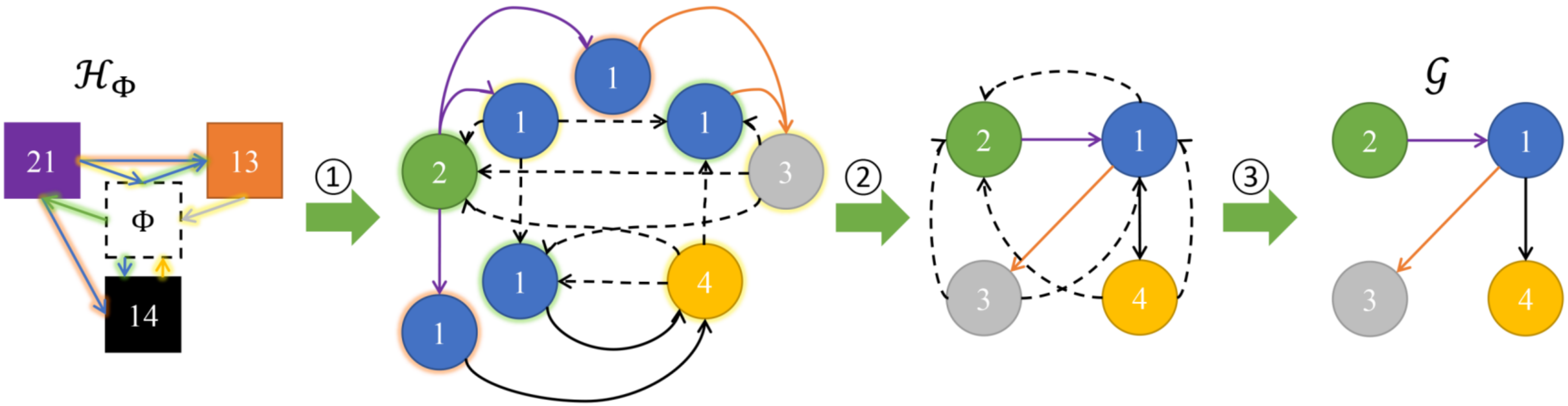
$$
\begin{aligned}
|\mathcal{V}_{\mathcal{H}'}| &= |\mathcal{E}_{\mathcal{G}'}| \\
&= |\mathcal{E}_{\mathcal{G}}| + 2|\mathcal{V}_{\mathcal{G}}| = |\mathcal{V}_{\mathcal{H}}| + 2|\mathcal{V}_{\mathcal{G}}| \qquad (7)
\end{aligned}
$$

$$
\begin{aligned}
|\mathcal{E}_{\mathcal{H}'}| &= |\mathcal{V}_{\mathcal{G}}|^2 + \sum_{v \in \mathcal{V}_{\mathcal{G}}} (d_v^- + 1) \cdot (d_v^+ + 1) \\
&= |\mathcal{V}_{\mathcal{G}}|^2 + \sum_{v \in \mathcal{V}_{\mathcal{G}}} d_v^- \cdot d_v^+ + \sum_{v \in \mathcal{V}_{\mathcal{G}}} (d_v^- + d_v^+) + |\mathcal{V}_{\mathcal{G}}| \\
&= |\mathcal{E}_{\mathcal{H}}| + |\mathcal{V}_{\mathcal{G}}|^2 + |\mathcal{V}_{\mathcal{G}}| + 2|\mathcal{V}_{\mathcal{E}}| \qquad (8)
\end{aligned}
$$

$$
\begin{aligned}
|\mathcal{V}_{\mathcal{H}'}| &= |\mathcal{V}_{\mathcal{H}}| + 1 \qquad (10) \\
|\mathcal{E}_{\mathcal{H}'}| &= |\mathcal{E}_{\mathcal{H}}| + 2|\mathcal{V}_{\mathcal{E}}| \qquad (11)
\end{aligned}
$$

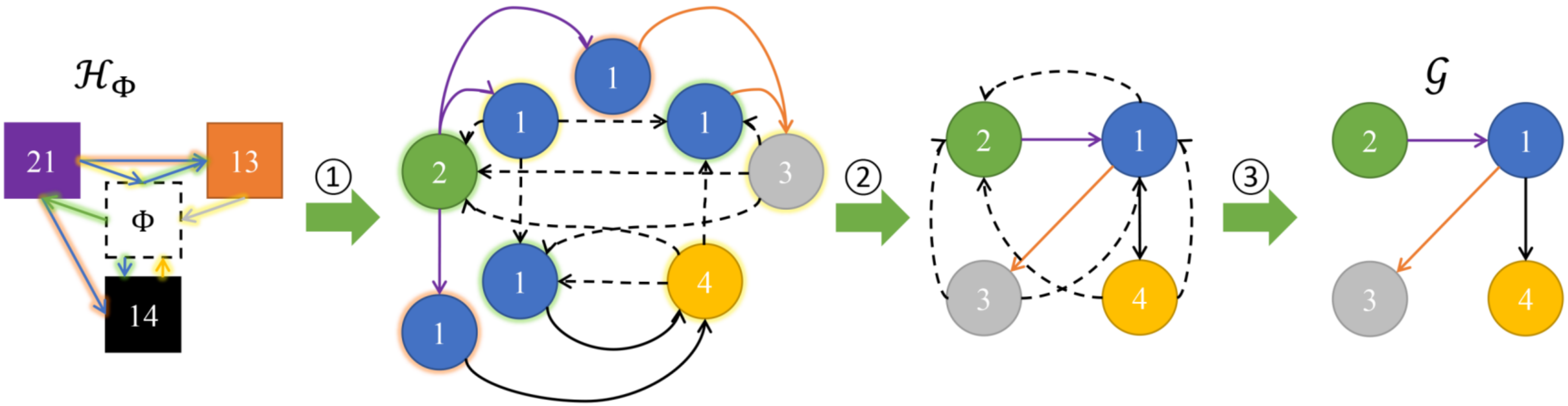
efficient and injective

Inverse of New Edge-to-vertex Transform



(b) inverse edge-to-vertex transform L_Φ^{-1}

Inverse of New Edge-to-vertex Transform

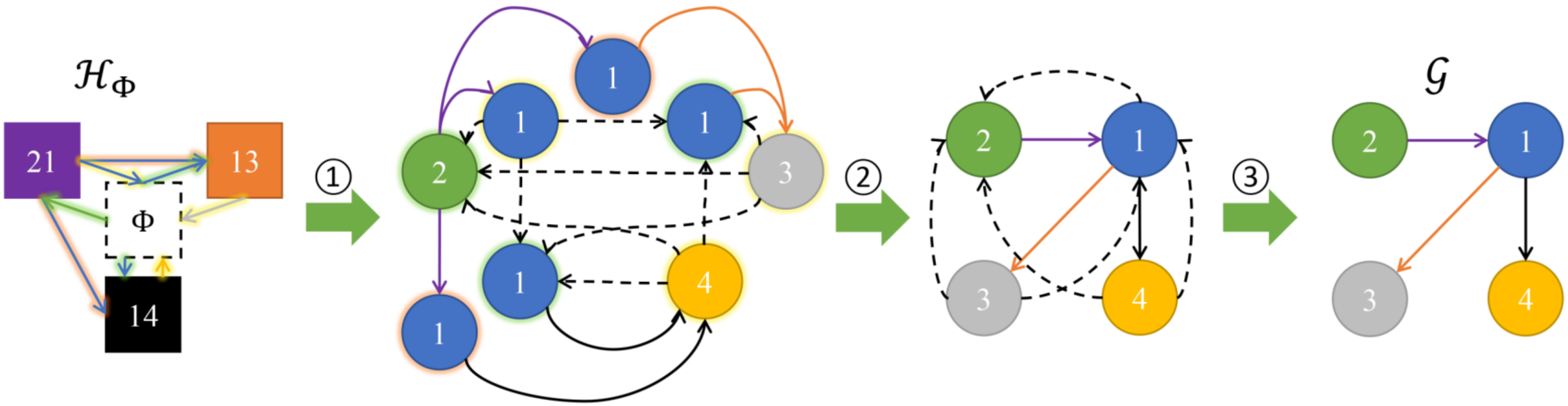


(b) inverse edge-to-vertex transform L_Φ^{-1}

Theorem 3.4. For any $\mathcal{H}_\Phi$ transformed by L_Φ such that $\mathcal{H}_\Phi = L_\Phi(\mathcal{G})$, L_Φ^{-1} can always transform $\mathcal{H}_\Phi$ back to $\mathcal{G}$, i.e., $L_\Phi^{-1}\big(L_\Phi(\mathcal{G})\big) = \mathcal{G}$

surjective and elegant

Inverse of New Edge-to-vertex Transform



(b) inverse edge-to-vertex transform L_Φ^{-1}

Theorem 3.4. For any $\mathcal{H}_\Phi$ transformed by L_Φ such that $\mathcal{H}_\Phi = L_\Phi(\mathcal{G})$, L_Φ^{-1} can always transform $\mathcal{H}_\Phi$ back to $\mathcal{G}$, i.e., $L_\Phi^{-1}\left(L_\Phi(\mathcal{G})\right) = \mathcal{G}$

surjective and elegant

Proposition 3.6. L_Φ is a monomorphism.

Corollary 3.7. Isomorphisms hold after L_Φ.

Corollary 3.8. If a function h is permutation-invariant, then $h \circ L_\Phi$ is also permutation-invariant.

Extensions of Graph Kernel Functions

- Conventional Graph Kernels

$$k(\mathcal{G}_1, \mathcal{G}_2) \;=\; \langle h(\mathcal{G}_1), h(\mathcal{G}_2) \rangle$$

where k is a kernel function to measure graph similarity, and h is the permutation-invariant function from graph space to Hilbert space.

Extensions of Graph Kernel Functions

- Conventional Graph Kernels

$$k(\mathcal{G}_1, \mathcal{G}_2) \;=\; \langle h(\mathcal{G}_1), h(\mathcal{G}_2) \rangle$$

where k is a kernel function to measure graph similarity, and h is the permutation-invariant function from graph space to Hilbert space.

- Graph Kernels with Dummy Nodes and Edges

$$\begin{aligned} k_{\varphi}(\mathcal{G}_1, \mathcal{G}_2) &= k(\mathcal{G}_1, \mathcal{G}_2) + k(\mathcal{G}_{\varphi_1}, \mathcal{G}_{\varphi_2}) \\ &= \langle h(\mathcal{G}_1), h(\mathcal{G}_2) \rangle + \langle h(\mathcal{G}_{\varphi_1}), h(\mathcal{G}_{\varphi_2}) \rangle, \end{aligned}$$

- Graph Kernels with Edge-to-vertex Transforms

$$\begin{aligned} k_{\Phi}(\mathcal{G}_1, \mathcal{G}_2) &= k(\mathcal{G}_1, \mathcal{G}_2) + k(\mathcal{H}_{\Phi_1}, \mathcal{H}_{\Phi_2}) \\ &= \langle h(\mathcal{G}_1), h(\mathcal{G}_2) \rangle + \langle h(L_{\Phi}(\mathcal{G}_1)), h(L_{\Phi}(\mathcal{G}_2)) \rangle, \end{aligned}$$

enforce the kernel functions to pay more attention to original structures

Extensions of Graph Neural Networks

- Conventional Message Passing Framework

$$\mathbf{\Delta}_v^{(t+1)} = Aggregate(\{Message(\boldsymbol{x}_v^{(t)}, \boldsymbol{x}_u^{(t)}, \boldsymbol{y}_{(u,v)})|u \in \mathcal{N}_v\}),$$
$$\boldsymbol{x}_v^{(t+1)} = Update(\boldsymbol{x}_v, \mathbf{\Delta}_v^{(t+1)}),$$

where $\boldsymbol{x}_v^{(t)}$ is the hidden state of vertex v at the t-th layer network, $\mathcal{N}_v$ is v's neighbor collection, $\boldsymbol{y}_{(u,v)}^{(t)}$ is the edge tensor for (u, v), $\mathbf{\Delta}_v^{(t+1)}$ is the aggregated message from neighbors.

Extensions of Graph Neural Networks

- Conventional Message Passing Framework

$$\mathbf{\Delta}_v^{(t+1)} = Aggregate(\{Message(\boldsymbol{x}_v^{(t)}, \boldsymbol{x}_u^{(t)}, \boldsymbol{y}_{(u,v)}) | u \in \mathcal{N}_v\}),$$
$$\boldsymbol{x}_v^{(t+1)} = Update(\boldsymbol{x}_v, \mathbf{\Delta}_v^{(t+1)}),$$

where $\boldsymbol{x}_v^{(t)}$ is the hidden state of vertex v at the t-th layer network, $\mathcal{N}_v$ is v's neighbor collection, $\boldsymbol{y}_{(u,v)}^{(t)}$ is the edge tensor for (u, v), $\mathbf{\Delta}_v^{(t+1)}$ is the aggregated message from neighbors.

- Graph Neural Networks with Dummy Nodes and Edges

simply replacing the input graph $\mathcal{G}$ with $\mathcal{G}_\varphi$, i.e., introducing a new x_φ and each neighbor collection except $\mathcal{N}_\varphi$ adds the dummy node φ.

- Graph Neural Networks with Edge-to-vertex Transforms

simply replacing the input graph $\mathcal{G}$ with $\mathcal{H}_\Phi$, i.e., performing edge-centric message passing in the line graph.

Experiments on Graph Classification

Classification error estimation: accuracy

Models		PROTEINS $\mathcal{G}$	PROTEINS $\mathcal{G}_\varphi$	PROTEINS $\mathcal{H}_\Phi$	D&D $\mathcal{G}$	D&D $\mathcal{G}_\varphi$	D&D $\mathcal{H}_\Phi$	NCI109 $\mathcal{G}$	NCI109 $\mathcal{G}_\varphi$	NCI109 $\mathcal{H}_\Phi$	NCI1 $\mathcal{G}$	NCI1 $\mathcal{G}_\varphi$	NCI1 $\mathcal{H}_\Phi$
Kernel	SP	73.48±3.9	**74.20±3.23**	73.39±3.04	0.50±3.6	79.58±3.91	**81.51±3.91**	73.65±2.34	73.84±2.07	74.11±2.22	74.18±1.67	74.70±1.74	74.40±1.74
	GR	70.45±6.5	74.20±4.44	73.66±4.00	8.82±3.8	79.66±5.18	78.82±3.87	66.45±2.14	72.46±2.51	71.81±2.69	65.16±2.30	73.04±1.81	71.07±1.47
	WLOA	72.59±2.4	73.84±3.29	74.02±3.47	9.24±3.6	79.24±3.81	78.57±3.59	85.43±1.51	84.61±1.52	84.81±1.11	85.96±1.82	86.33±1.77	**86.37±1.75**
	1-WL	71.79±4.5	73.30±4.14	73.48±5.02	0.50±4.4	81.26±4.08	80.42±3.85	**85.54±1.34**	83.74±0.94	84.37±1.02	85.13±1.69	84.87±1.77	85.38±1.21
	2-WL	74.11±5.1	75.27±4.67	OOM	OOM	OOM	OOM	68.09±1.55	68.38±1.21	72.24±1.85	67.71±1.33	67.49±1.45	69.00±2.34
	δ-2-WL	74.20±4.9	74.82±4.16	OOM	OOM	OOM	OOM	68.00±1.94	68.26±1.59	70.34±1.87	67.32±1.34	67.37±1.40	69.20±2.18
	δ-2-LWL	73.66±5.1	74.37±3.34	74.11±3.72	7.06±5.9	77.31±5.98	79.41±5.28	84.20±1.44	83.12±1.34	83.82±1.06	85.40±1.28	84.06±1.54	85.40±1.51
	δ-2-LWL$^+$	78.12±4.7	83.48±4.34	**84.55±3.62**	7.14±6.0	77.56±6.30	**79.58±6.24**	88.79±0.94	**89.42±1.37**	88.57±0.97	91.92±1.93	**93.67±0.84**	91.65±1.96
Network	GraphSAGE	73.48±5.6	73.93±5.68	-	7.73±4.6	78.91±4.59	-	73.38±2.68	74.13±2.30	-	73.82±2.17	74.31±2.27	-
	GCN	72.95±3.8	74.02±3.82	-	2.77±4.6	**80.76±5.37**	-	50.34±2.69	51.67±5.52	-	61.75±11.1	68.95±10.8	-
	GIN	73.84±4.4	74.11±4.12	-	6.97±3.8	77.65±3.46	-	72.61±2.37	73.82±2.50	-	73.50±1.80	75.16±1.49	-
	RGCN	73.30±4.9	74.98±4.50	**75.09±4.03**	9.16±9.9	69.24±10.0	78.47±5.24	50.29±2.08	51.52±4.37	71.71±7.59	52.75±4.75	57.27±9.49	74.04±1.15
	RGIN	68.75±6.5	70.54±5.03	74.20±2.93	7.65±4.6	78.15±4.60	77.73±4.42	64.20±2.85	64.52±2.58	**75.43±3.50**	66.11±1.77	66.11±1.69	**76.18±2.03**
	DiffPool	75.62±5.1	**75.98±3.89**	-	1.41±5.1	80.25±4.69	-	75.29±1.85	**75.44±1.90**	-	76.62±1.93	**77.08±1.33**	-
	HGP-SL	71.25±7.1	74.46±3.77	-	4.62±3.1	**82.07±2.11**	-	74.78±2.37	74.32±1.84	-	74.94±0.88	76.08±1.94	-
Average		*73.13±2.1*	*74.77±2.60*	*75.31±3.53*	*7.20±3.2*	*78.59±3.05*	*79.31±1.13*	*72.07±11.20*	*72.62±10.53*	*77.72±6.51*	*73.48±10.1*	*75.10±8.98*	*78.27±7.77*

- We observe consistent performance improvement after adding dummy nodes ($\mathcal{G}_\varphi$) for most classifiers.
- We also see the further improvement on average after using $\mathcal{H}_\Phi$.

Experiments on Graph Classification

Classification error estimation: accuracy

	Models	PROTEINS			D&D			NCI109			NCI1		
		$\mathcal{G}$	$\mathcal{G}_\varphi$	$\mathcal{H}_\Phi$	$\mathcal{G}$	$\mathcal{G}_\varphi$	$\mathcal{H}_\Phi$	$\mathcal{G}$	$\mathcal{G}_\varphi$	$\mathcal{H}_\Phi$	$\mathcal{G}$	$\mathcal{G}_\varphi$	$\mathcal{H}_\Phi$
Kernel	SP	73.48±3.93	**74.20±3.23**	73.39±3.04	80.50±3.66	79.58±3.91	**81.51±3.91**	73.65±2.34	73.84±2.07	74.11±2.22	74.18±1.67	74.70±1.74	74.40±1.74
	GR	70.45±6.54	74.20±4.44	73.66±4.00	78.82±3.83	79.66±5.18	78.82±3.87	66.45±2.14	72.46±2.51	71.81±2.69	65.16±2.30	73.04±1.81	71.07±1.47
	WLOA	72.59±2.46	73.84±3.29	74.02±3.47	79.24±3.61	79.24±3.81	78.57±3.59	85.43±1.51	84.61±1.52	84.81±1.11	85.96±1.82	86.33±1.77	**86.37±1.75**
	1-WL	71.79±4.52	73.30±4.14	73.48±5.02	80.50±4.43	81.26±4.08	80.42±3.85	**85.54±1.34**	83.74±0.94	84.37±1.02	85.13±1.69	84.87±1.77	85.38±1.21
	2-WL	74.11±5.19	75.27±4.67	OOM	OOM	OOM	OOM	68.09±1.55	68.38±1.21	72.24±1.85	67.71±1.33	67.49±1.45	69.00±2.34
	δ-2-WL	74.20±4.98	74.82±4.16	OOM	OOM	OOM	OOM	68.00±1.94	68.26±1.59	70.34±1.87	67.32±1.34	67.37±1.40	69.20±2.18
	δ-2-LWL	73.66±5.10	74.37±3.34	74.11±3.72	77.06±5.99	77.31±5.98	79.41±5.28	84.20±1.44	83.12±1.34	83.82±1.06	85.40±1.28	84.06±1.54	85.40±1.51
	δ-2-LWL$^+$	78.12±4.75	83.48±4.34	**84.55±3.62**	77.14±6.05	77.56±6.30	**79.58±6.24**	88.79±0.94	**89.42±1.37**	88.57±0.97	91.92±1.93	**93.67±0.84**	91.65±1.96
Network	GraphSAGE	73.48±5.66	73.93±5.68	-	77.73±4.66	78.91±4.59	-	73.38±2.68	74.13±2.30	-	73.82±2.17	74.31±2.27	-
	GCN	72.95±3.88	74.02±3.82	-	72.77±4.62	**80.76±5.37**	-	50.34±2.69	51.67±5.52	-	61.75±11.1	68.95±10.8	-
	GIN	73.84±4.46	74.11±4.12		76.97±3.87	77.65±3.46		72.61±2.37	73.82±2.50		73.50±1.80	75.16±1.49	
	RGCN	73.30±4.90	74.98±4.50	**75.09±4.03**	69.16±9.97	69.24±10.0	78.47±5.24	50.29±2.08	51.52±4.37	71.71±7.59	52.75±4.75	57.27±9.49	74.04±1.15
	RGIN	68.75±6.59	70.54±5.03	74.20±2.93	77.65±4.62	78.15±4.60	77.73±4.42	64.20±2.85	64.52±2.58	**75.43±3.50**	66.11±1.77	66.11±1.69	**76.18±2.03**
	DiffPool	75.62±3.17	**75.98±3.89**	-	81.41±3.11	80.25±4.69	-	75.29±1.85	**75.44±1.90**	-	76.62±1.95	**77.08±1.33**	-
	HGP-SL	71.25±7.13	74.46±3.77	-	74.62±3.19	**82.07±2.11**	-	74.78±2.37	74.32±1.84	-	74.94±0.88	76.08±1.94	-
	Average	*73.13±2.10*	*74.77±2.60*	*75.31±3.53*	*77.20±3.26*	*78.59±3.05*	*79.31±1.13*	*72.07±11.20*	*72.62±10.53*	*77.72±6.51*	*73.48±10.16*	*75.10±8.98*	*78.27±7.77*

- We observe consistent performance improvement after adding dummy nodes ($\mathcal{G}_\varphi$) for most classifiers.
- We also see the further improvement on average after using $\mathcal{H}_\Phi$.
- Kernels with the 2-order structures nearly surpass the 1-order kernels and 1-WL graph neural networks.
- Accuracies of GNNs get boosted again when the input changes from $\mathcal{G}_\varphi$ to $\mathcal{H}_\Phi$.

Experiments on Subgraph Isomorphisms

counting error estimation: RMSE, MAE   matching error estimation: graph edit distance

Models		Homogeneous						Heterogeneous					
		Erdős-Renyi			Regular			Complex			MUTAG		
		RMSE	MAE	GED	RMSE	MAE	GED	RMSE	MAE	GED	RMSE	MAE	GED
RGCN	$\mathcal{G}$	9.386	5.829	28.963	14.789	9.772	70.746	28.601	9.386	64.122	0.777	0.334	1.441
	$\mathcal{G}_\varphi$	7.764	4.654	24.438	14.077	9.511	71.393	26.389	7.110	55.600	0.534	0.191	1.052
RGIN	$\mathcal{G}$	6.063	3.712	22.155	13.554	8.580	56.353	20.893	4.411	56.263	0.273	0.082	0.329
	$\mathcal{G}_\varphi$	**4.769**	**2.898**	**15.219**	**10.871**	**6.874**	**43.537**	19.436	**3.846**	**41.337**	**0.193**	**0.064**	0.277
HGT	$\mathcal{G}$	24.376	14.630	104.000	26.713	17.482	191.674	34.055	8.336	70.080	1.317	0.526	3.644
	$\mathcal{G}_\varphi$	5.969	3.691	23.401	13.813	8.813	64.926	20.841	4.707	47.409	0.876	0.345	2.973
CompGCN	$\mathcal{G}$	6.706	4.274	25.548	14.174	9.685	64.677	22.287	5.127	57.082	0.300	0.085	0.278
	$\mathcal{G}_\varphi$	4.981	3.019	16.263	11.450	7.443	46.802	20.786	4.048	56.269	0.321	0.089	**0.262**
DMPNN	$\mathcal{G}$	5.330	3.308	23.411	11.980	7.832	56.222	18.974	3.992	56.933	0.232	0.088	0.320
	$\mathcal{G}_\varphi$	5.220	3.130	23.285	11.259	7.136	49.179	**18.885**	3.892	73.161	0.259	0.101	0.623
Deep-LRP	$\mathcal{G}$	0.794	0.436	2.571	1.373	0.788	5.432	27.490	5.850	56.772	0.260	0.094	0.437
	$\mathcal{G}_\varphi$	0.710	0.402	2.218	1.145	0.718	4.611	24.458	5.094	57.398	0.356	0.115	0.849
DMPNN-LRP	$\mathcal{G}$	**0.475**	0.287	1.538	**0.617**	0.422	2.745	20.425	4.173	**32.200**	0.196	0.062	**0.210**
	$\mathcal{G}_\varphi$	0.477	**0.260**	**1.457**	0.633	**0.413**	**2.538**	**18.127**	**4.112**	39.594	**0.186**	**0.057**	0.265

- Overall, RGIN with $\mathcal{G}_\varphi$ outperforms other GNNs except LRP-based models.
- HGT has the most prominent performance boost, indicating that the dummy nodes provide an option to drop all pattern-irrelevant messages.

Experiments on Subgraph Isomorphisms

counting error estimation: RMSE, MAE　　matching error estimation: graph edit distance

Models		Homogeneous						Heterogeneous					
		Erdős-Renyi			Regular			Complex			MUTAG		
		RMSE	MAE	GED	RMSE	MAE	GED	RMSE	MAE	GED	RMSE	MAE	GED
RGCN	$\mathcal{G}$	9.386	5.829	28.963	14.789	9.772	70.746	28.601	9.386	64.122	0.777	0.334	1.441
	$\mathcal{G}_\varphi$	7.764	4.654	24.438	14.077	9.511	71.393	26.389	7.110	55.600	0.534	0.191	1.052
RGIN	$\mathcal{G}$	6.063	3.712	22.155	13.554	8.580	56.353	20.893	4.411	56.263	0.273	0.082	0.329
	$\mathcal{G}_\varphi$	**4.769**	**2.898**	**15.219**	**10.871**	**6.874**	**43.537**	19.436	**3.846**	**41.337**	**0.193**	**0.064**	0.277
HGT	$\mathcal{G}$	24.376	14.630	104.000	26.713	17.482	191.674	34.055	8.336	70.080	1.317	0.526	3.644
	$\mathcal{G}_\varphi$	5.969	3.691	23.401	13.813	8.813	64.926	20.841	4.707	47.409	0.876	0.345	2.973
CompGCN	$\mathcal{G}$	6.706	4.274	25.548	14.174	9.685	64.677	22.287	5.127	57.082	0.300	0.085	0.278
	$\mathcal{G}_\varphi$	4.981	3.019	16.263	11.450	7.443	46.802	20.786	4.048	56.269	0.321	0.089	**0.262**
DMPNN	$\mathcal{G}$	5.330	3.308	23.411	11.980	7.832	56.222	18.974	3.992	56.933	0.232	0.088	0.320
	$\mathcal{G}_\varphi$	5.220	3.130	23.285	11.259	7.136	49.179	**18.885**	3.892	73.161	0.259	0.101	0.623
Deep-LRP	$\mathcal{G}$	0.794	0.436	2.571	1.373	0.788	5.432	27.490	5.850	56.772	0.260	0.094	0.437
	$\mathcal{G}_\varphi$	0.710	0.402	2.218	1.145	0.718	4.611	24.458	5.094	57.398	0.356	0.115	0.849
DMPNN-LRP	$\mathcal{G}$	**0.475**	0.287	1.538	**0.617**	0.422	2.745	20.425	4.173	**32.200**	0.196	0.062	**0.210**
	$\mathcal{G}_\varphi$	0.477	**0.260**	**1.457**	0.633	**0.413**	**2.538**	**18.127**	**4.112**	39.594	**0.186**	**0.057**	0.265

- Overall, RGIN with $\mathcal{G}_\varphi$ outperforms other GNNs except LRP-based models.
- HGT has the most prominent performance boost, indicating that the dummy nodes provide an option to drop all pattern-irrelevant messages.
- Deep-LRP and DMPNN-LRP benefit from the dummy nodes in counting, but they gets the increase of matching errors due to the star typology around the dummy.

Discussions about Expressive Power

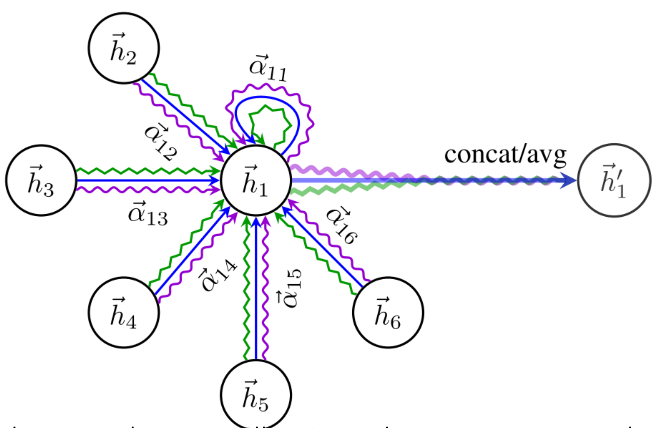
- Pattern Learning Capability

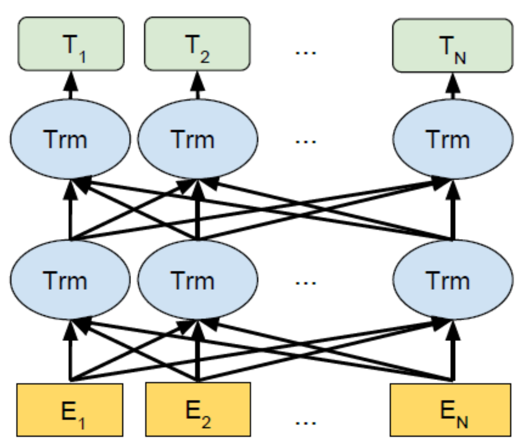
Theorem I: a k-WL graph neural network can count any patterns at most k vertices.

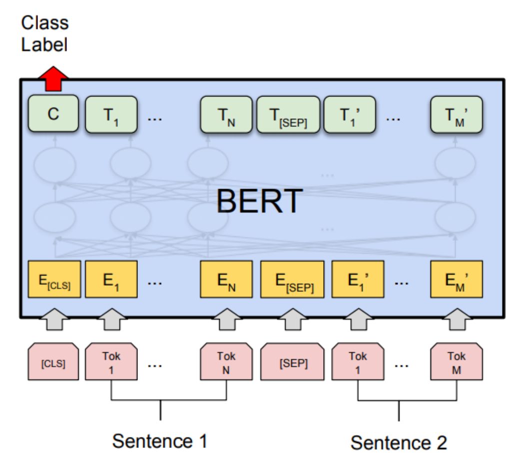
Theorem II: a T-layer k-WL graph neural network cannot count any patterns with equal to or more than $(k+1)2^T$ vertices (Chen et al., 2020).

Lemma III: 1-WL = RGIN = RGAT < 2-WL = RGIN w/ φ = RGAT w/ φ $\leq$ Transformer with a dummy [CLS] token.

Corollary IV: a T-layer Transformer with a dummy [CLS] token can capture any patterns with 2 tokens, and cannot capture patterns with no less than $3 \cdot 2^T$ tokens.







Chen, Z., Chen, L., Villar, S., and Bruna, J. Can graph neural networks count substructures? In NeurIPS, 2020.

Conclusion

- We analyze the role of dummy nodes in the lossless edge-to-vertex transform.
- We further prove that a dummy node with connections to all existing vertices can preserve the graph structure.
- We design an efficient monomorphic edge-to-vertex transform and find its inverse to recover the original graph back.
- We extend graph kernels and graph neural networks with dummy nodes to boost their graph/subgraph learning performance.
- We discuss the capability of MPNNs and Transformers with special dummy elements.

Thanks

Email: xliucr@cse.ust.hk

Paper

https://arxiv.org/abs/2206.08561

Code

github.com/HKUST-KnowComp/DummyNode4GraphLearning