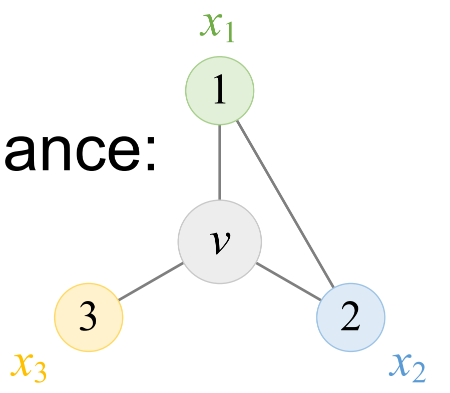
Going Deeper into Permutation-Sensitive Graph Neural Networks

Zhongyu Huang[1,2], Yingheng Wang[3,4], Chaozhuo Li[5], Huiguang He[1,2]

1 中国科学院自动化研究所 INSTITUTE OF AUTOMATION CHINESE ACADEMY OF SCIENCES
2 中国科学院大学 University of Chinese Academy of Sciences
3 清華大学 Tsinghua University
4 Johns Hopkins University
5 Microsoft

Introduction of Permutation-Invariance

Permutation-invariance: permutation of the nodes of the input graph does not affect the output.

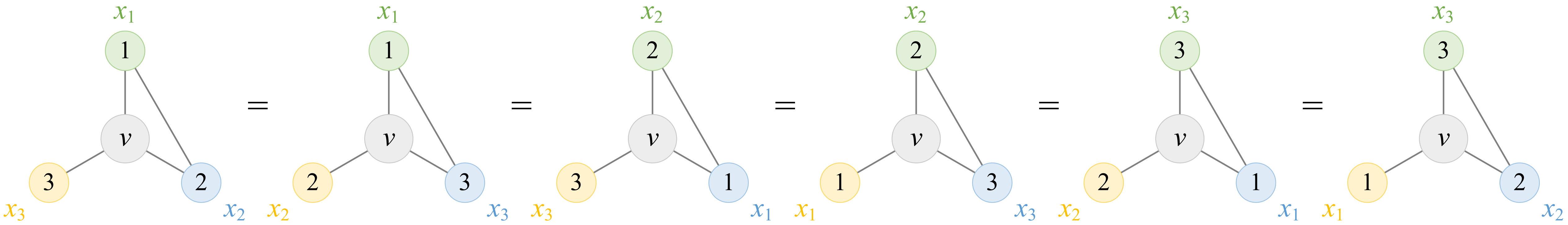


$a_v = f(x_1, x_2, x_3) = x_1 + x_2 + x_3$

For invariant aggregators such as SUM, we have ordering-invariance:

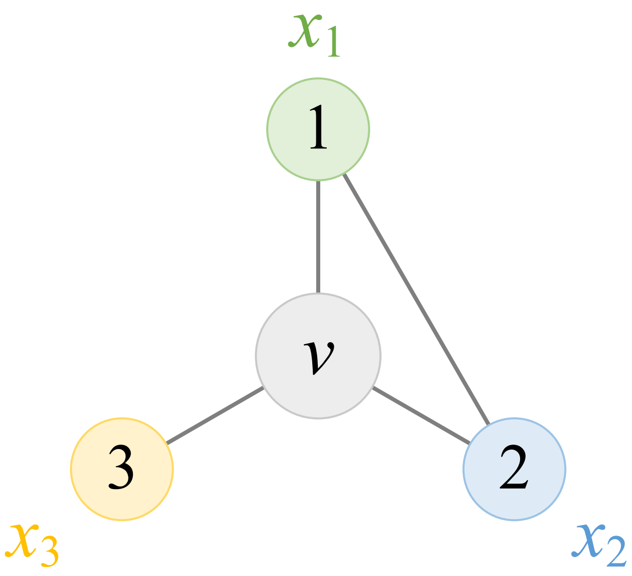
$$f(x_1, x_2, x_3) = f(x_1, x_3, x_2) = f(x_2, x_1, x_3) = f(x_2, x_3, x_1) = f(x_3, x_1, x_2) = f(x_3, x_2, x_1) = x_1 + x_2 + x_3$$

and label-invariance:

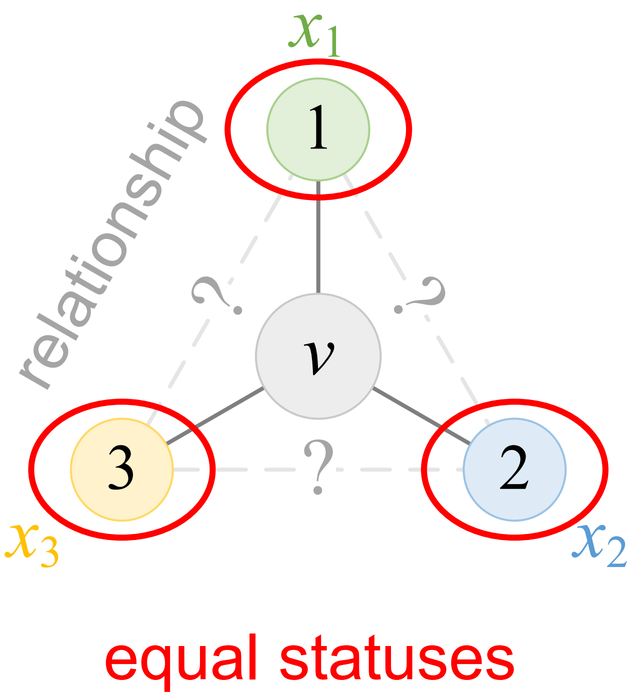


02 Limitation of Permutation-Invariance

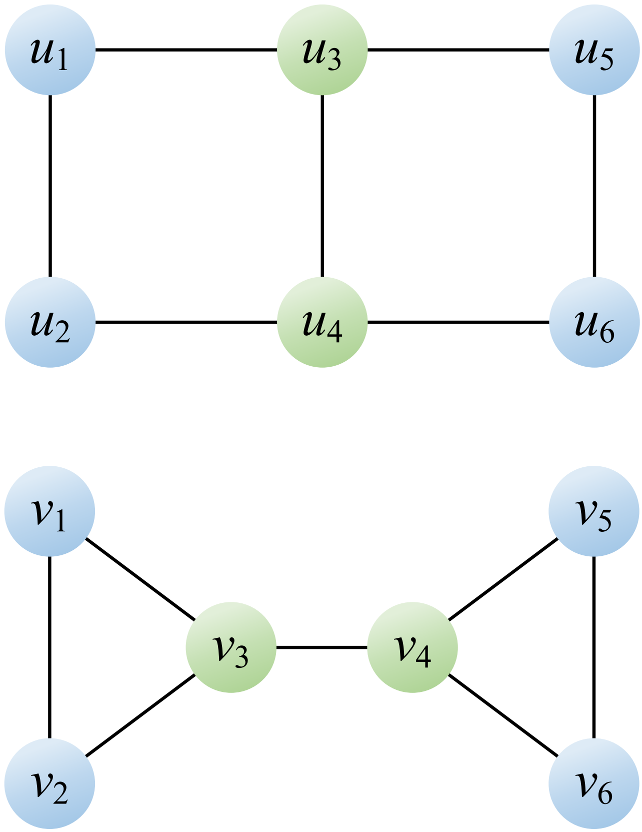
The real graph structure:



What permutation-invariant aggregators can see:



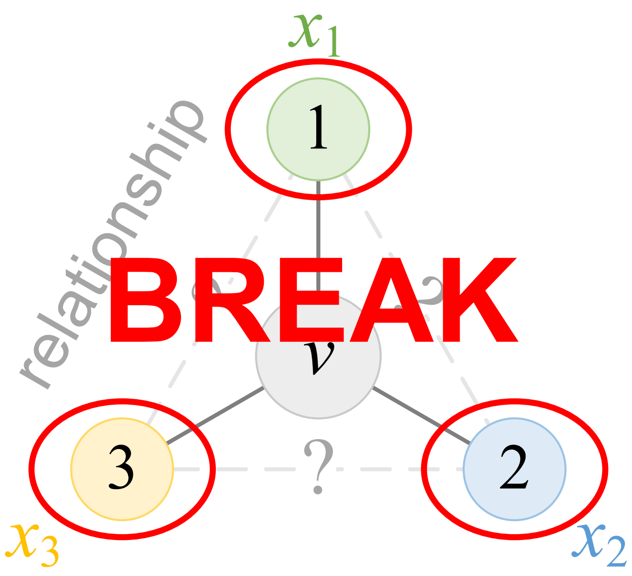
Thus fail to distinguish:



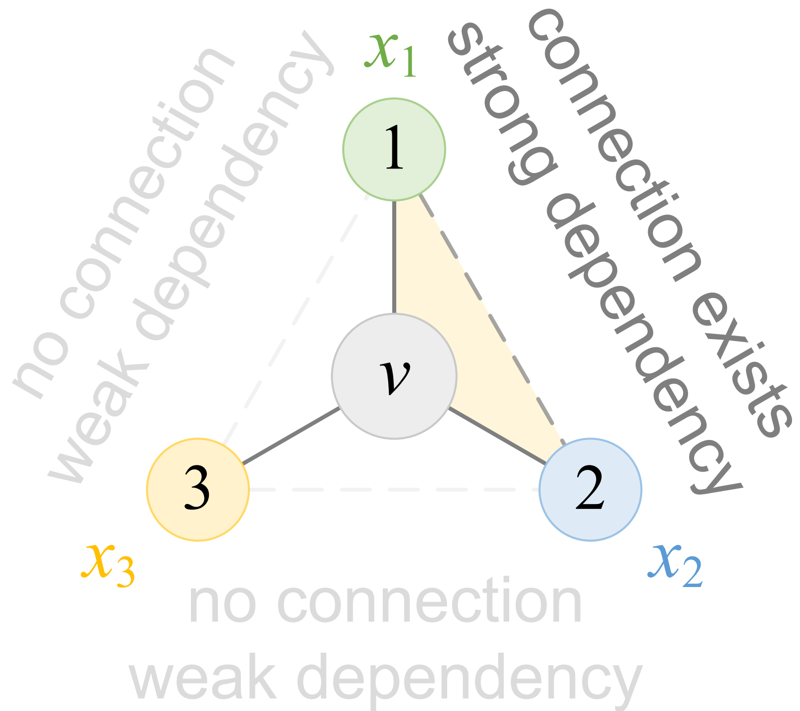
They ***ignore*** the relationships among neighboring nodes.

03 More Powerful Permutation-Sensitive Aggregators

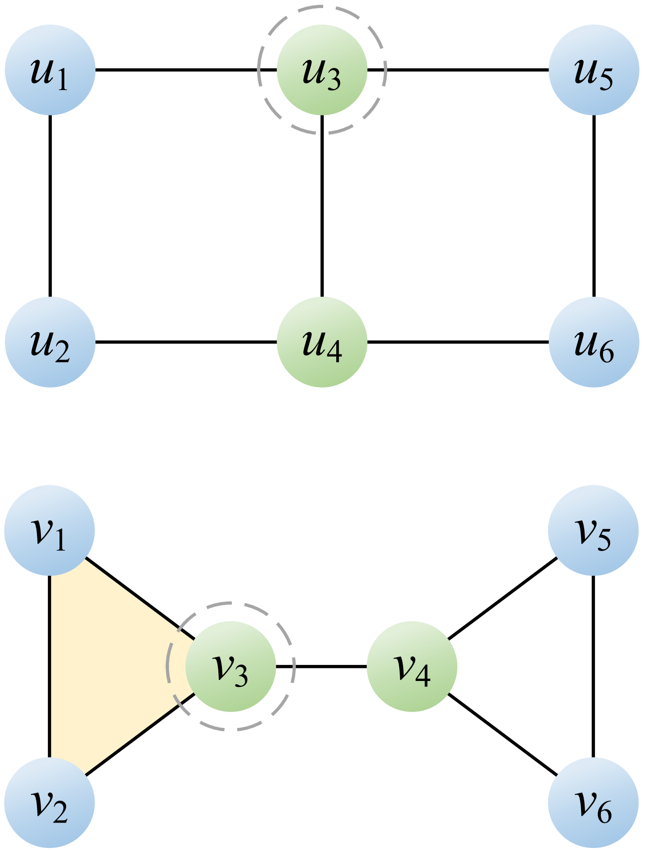
Breaking the symmetry of invariant aggregators:



What permutation-sensitive aggregators can see:



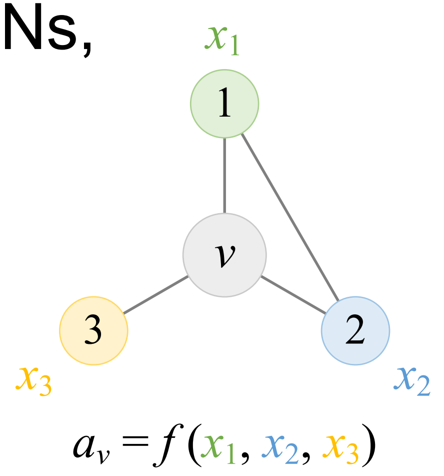
Thus can distinguish:



They ***can count*** the graph substructures such as triangles.

04 Limitation of Permutation-Sensitivity

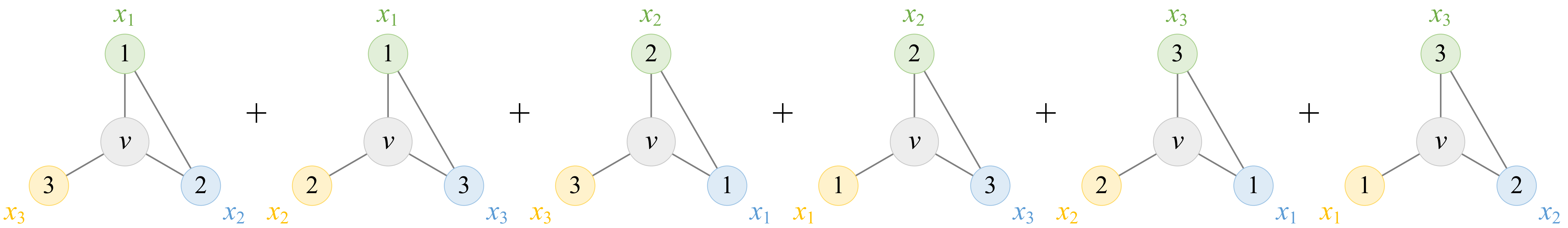
For sensitive aggregators, they need to cover all $n!$ possible permutations (node orderings) to guarantee the permutation-invariance of GNNs,



such as ordering-invariance:

$$f(x_1, x_2, x_3) + f(x_1, x_3, x_2) + f(x_2, x_1, x_3) + f(x_2, x_3, x_1) + f(x_3, x_1, x_2) + f(x_3, x_2, x_1) \rightarrow \text{overall invariant to } \{x_1, x_2, x_3\}$$

and label-invariance:



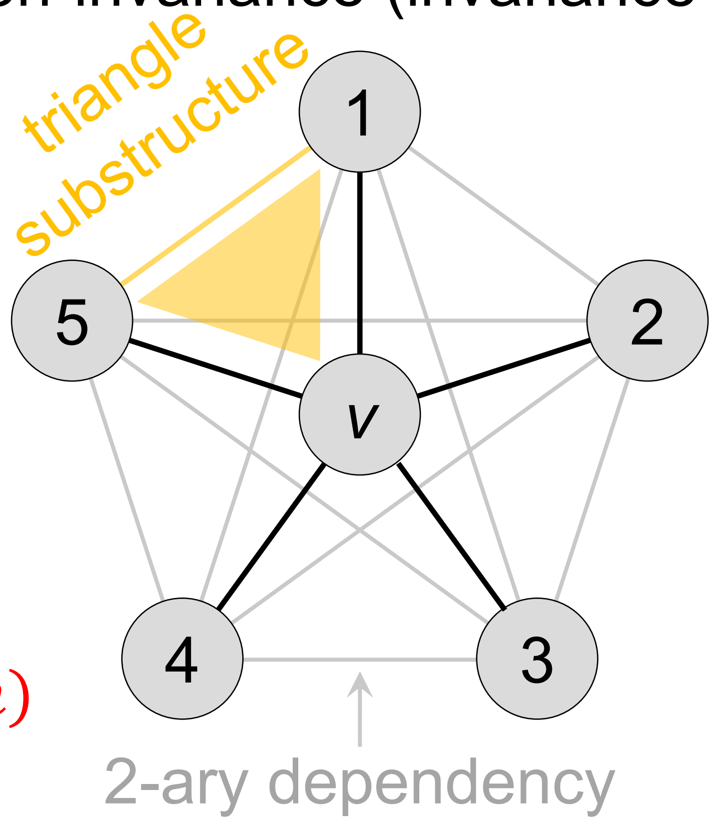
Core Ideas of Reducing Complexity

Approximate the permutation-invariance: Avoid $\mathcal{O}(n!)$

Model all 2-ary dependencies (pairwise correlations) to ensure the invariance to 2-ary dependencies and thus approximate the permutation-invariance (invariance to n-ary dependencies): From $\mathcal{O}(n!)$ to $\mathcal{O}(n^2)$

- Full 2-ary dependencies can also capture whether any two neighbors are connected, helping count substructures and improve the expressive power.

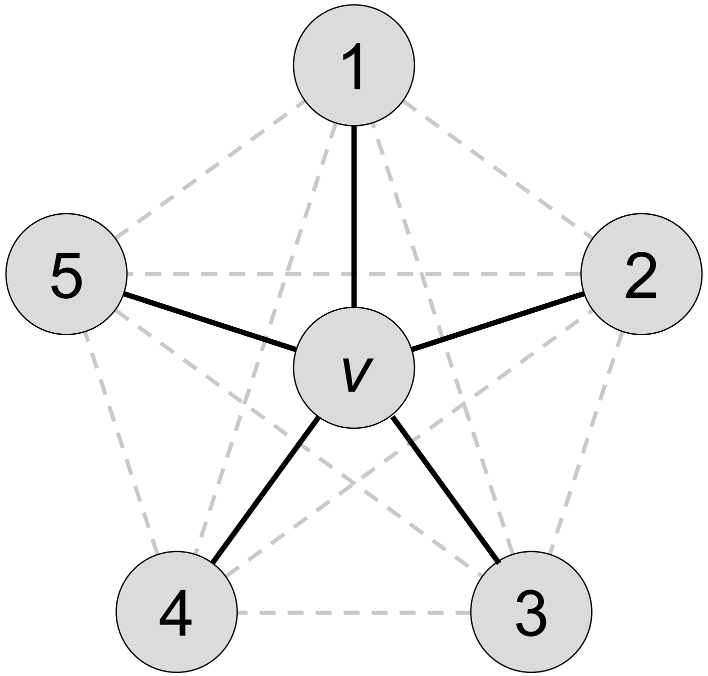
Devise a permutation sampling strategy to minimize the complexity of covering all 2-ary deps: From $\mathcal{O}(n^2)$ to $\mathcal{O}(n)$



06

Permutation Sampling Strategy

Graph topology:



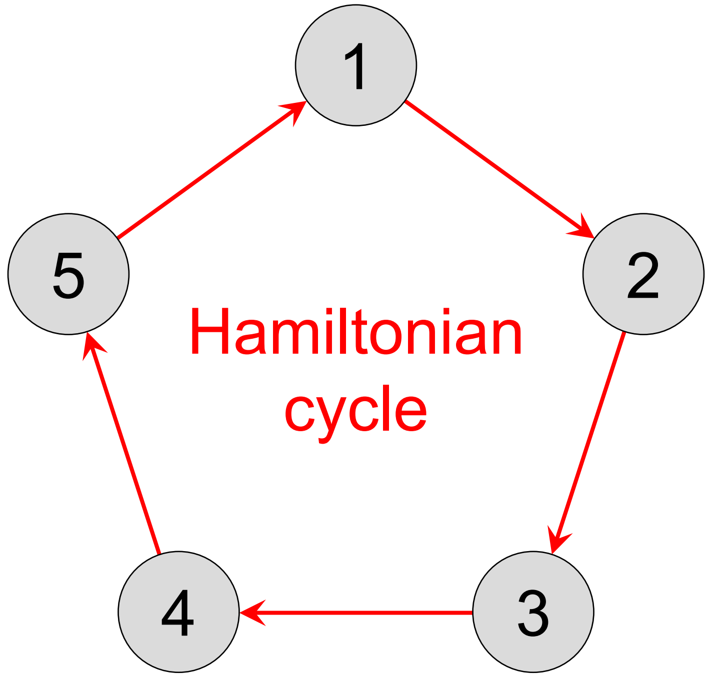
unknown relationships
(to be modeled)

Scan me
for a full demo

06

Permutation Sampling Strategy

Initial permutation:

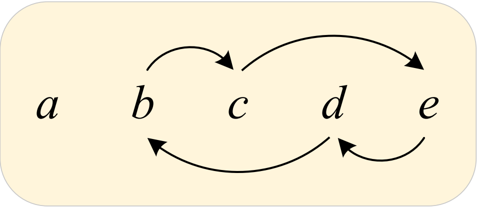


$$\text{RNN}\ (\underbrace{u_1 \text{ — } u_2}_{\text{2-ary dependency}} \text{ — } u_3 \text{ — } u_4 \text{ — } u_5 \text{ — } u_1)$$

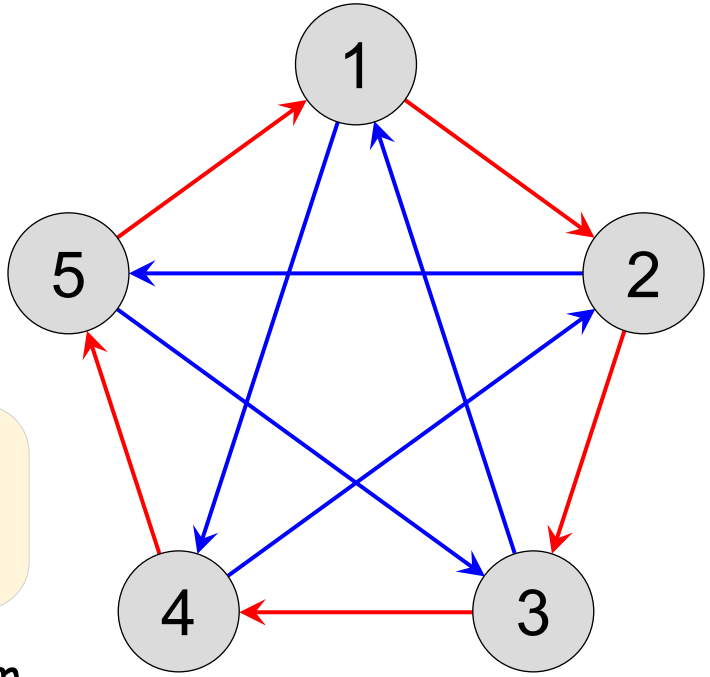
Scan me
for a full demo

06 Permutation Sampling Strategy

Generate a new permutation:



Permutation diagram



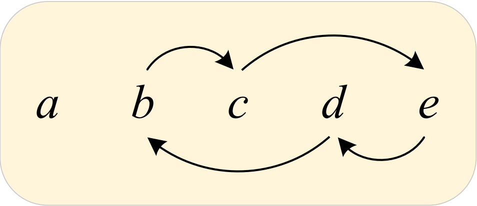
Scan me
for a full demo

$$\text{RNN}\,(u_1 - u_2 - u_3 - u_4 - u_5 - u_1)$$

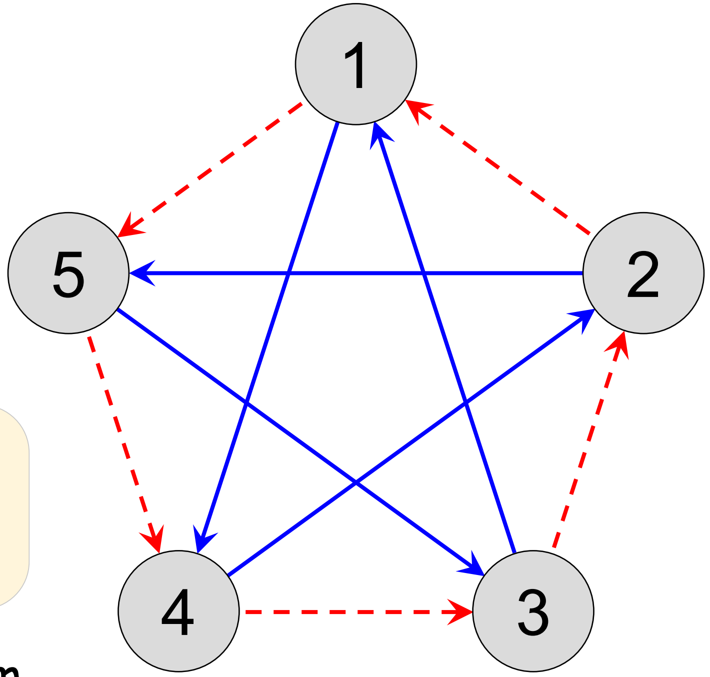
$$+\ \text{RNN}\,(u_1 - u_4 - u_2 - u_5 - u_3 - u_1)$$

06 Permutation Sampling Strategy

Reverse the permutation:



Permutation diagram



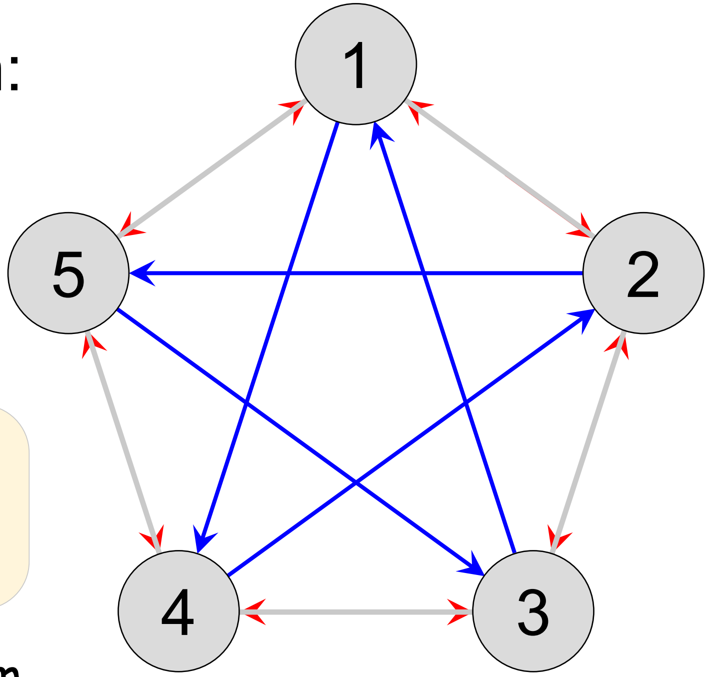
$$\text{RNN}\ (u_1 - u_2 - u_3 - u_4 - u_5 - u_1)$$

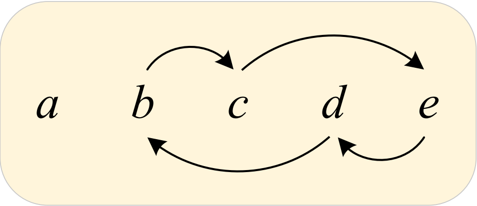
$$+\ \text{RNN}\ (u_1 - u_4 - u_2 - u_5 - u_3 - u_1)$$

$$+\ \text{RNN}\ (u_1 - u_5 - u_4 - u_3 - u_2 - u_1)$$

Permutation Sampling Strategy

Bi-directional transformation:





Permutation diagram

Scan me
for a full demo

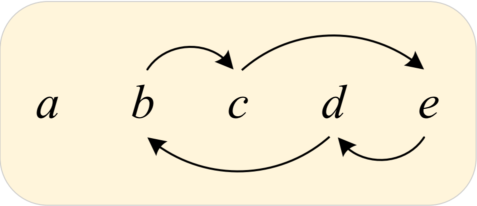
$$\text{RNN}\ (u_1 - u_2 - u_3 - u_4 - u_5 - u_1)$$

$$+\ \text{RNN}\ (u_1 - u_4 - u_2 - u_5 - u_3 - u_1)$$

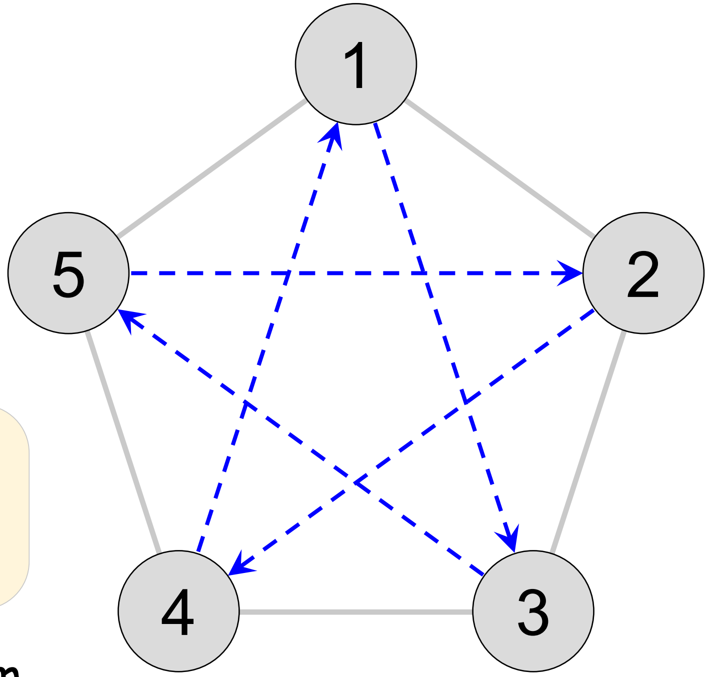
$$+\ \text{RNN}\ (u_1 \dashrightarrow u_5 \dashrightarrow u_4 \dashrightarrow u_3 \dashrightarrow u_2 \dashrightarrow u_1)$$

06 Permutation Sampling Strategy

Reverse the permutation:



Permutation diagram



Scan me
for a full demo

$$\text{RNN}\ (u_1 — u_2 — u_3 — u_4 — u_5 — u_1)$$

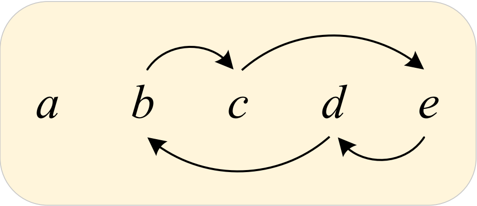
$$+\ \text{RNN}\ (u_1 — u_4 — u_2 — u_5 — u_3 — u_1)$$

$$+\ \text{RNN}\ (u_1 \text{-}\text{-}\text{-} u_5 \text{-}\text{-}\text{-} u_4 \text{-}\text{-}\text{-} u_3 \text{-}\text{-}\text{-} u_2 \text{-}\text{-}\text{-} u_1)$$

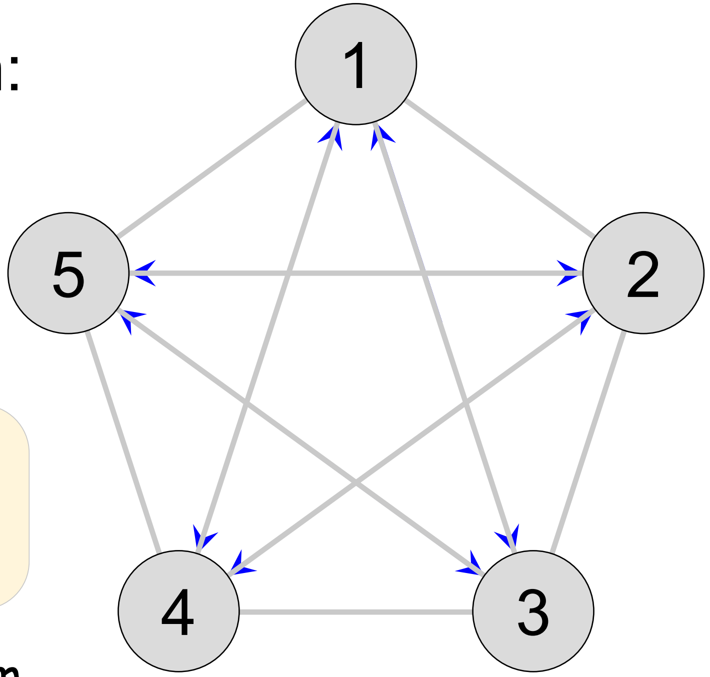
$$+\ \text{RNN}\ (u_1 \text{-}\text{-}\text{-} u_3 \text{-}\text{-}\text{-} u_5 \text{-}\text{-}\text{-} u_2 \text{-}\text{-}\text{-} u_4 \text{-}\text{-}\text{-} u_1)$$

06 Permutation Sampling Strategy

Bi-directional transformation:



Permutation diagram



Scan me
for a full demo

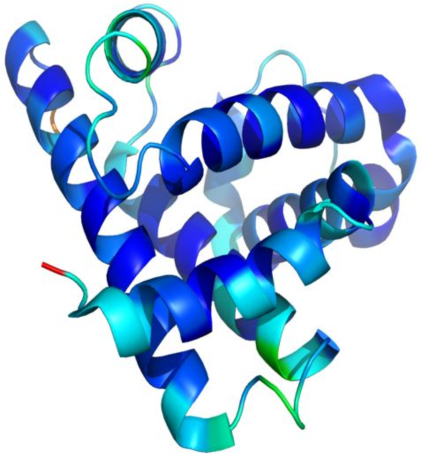
$$\mathrm{RNN}\,(u_1 - u_2 - u_3 - u_4 - u_5 - u_1)$$

$$+\ \mathrm{RNN}\,(u_1 - u_4 - u_2 - u_5 - u_3 - u_1)$$

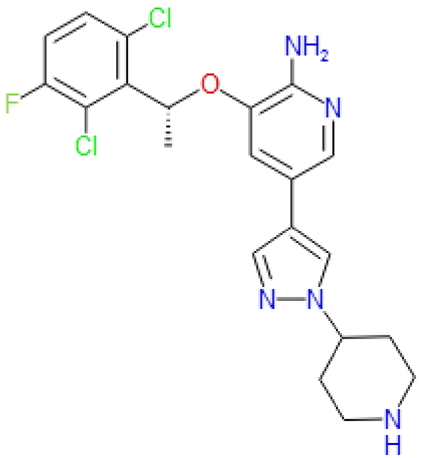
$$+\ \mathrm{RNN}\,(u_1 \cdots u_5 \cdots u_4 \cdots u_3 \cdots u_2 \cdots u_1)$$

$$+\ \mathrm{RNN}\,(u_1 \cdots u_3 \cdots u_5 \cdots u_2 \cdots u_4 \cdots u_1)$$

07 Experimental Datasets



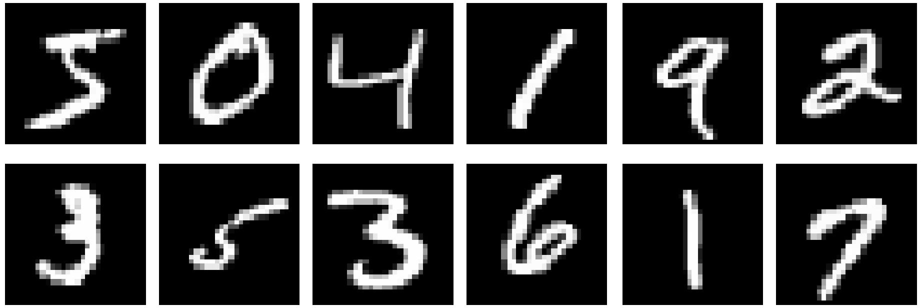
PROTEINS



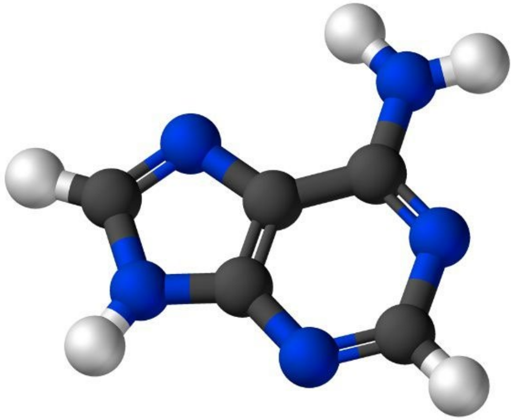
NCI1

IMDB

COLLAB



MINST



ZINC

08

Experimental Results on TUDataset

Model	PROTEINS	NCI1	IMDB-B	IMDB-M	COLLAB
WL	75.0 ± 3.1	**86.0 ± 1.8**	73.8 ± 3.9	50.9 ± 3.8	78.9 ± 1.9
DGCNN	75.5 ± 0.9	74.4 ± 0.5	70.0 ± 0.9	47.8 ± 0.9	73.8 ± 0.5
IGN	76.6 ± 5.5	74.3 ± 2.7	72.0 ± 5.5	48.7 ± 3.4	78.4 ± 2.5
GIN	76.2 ± 2.8	82.7 ± 1.7	75.1 ± 5.1	52.3 ± 2.8	80.2 ± 1.9
PPGN	**77.2 ± 4.7**	83.2 ± 1.1	73.0 ± 5.8	50.5 ± 3.6	80.7 ± 1.7
CLIP	77.1 ± 4.4	N/A	76.0 ± 2.7	52.5 ± 3.0	N/A
WEGL	76.5 ± 4.2	N/A	75.4 ± 5.0	52.3 ± 2.9	80.6 ± 2.0
SIN	76.5 ± 3.4	82.8 ± 2.2	75.6 ± 3.2	52.5 ± 3.0	N/A
CIN	77.0 ± 4.3	83.6 ± 1.4	75.6 ± 3.7	52.7 ± 3.1	N/A
PG-GNN (Ours)	76.8 ± 3.8	82.8 ± 1.3	**76.8 ± 2.6**	**53.2 ± 3.6**	**80.9 ± 0.8**

Experimental Results on Benchmark Dataset

Model	MNIST		ZINC	
	Accuracy ↑	Time / Epoch	MAE ↓	Time / Epoch
GraphSAGE	97.31 ± 0.10	113.12s	0.468 ± 0.003	3.74s
GatedGCN	97.34 ± 0.14	128.79s	0.435 ± 0.011	5.76s
GIN	96.49 ± 0.25	39.22s	0.387 ± 0.015	2.29s
3-WL-GNN	95.08 ± 0.96	1523.20s	0.407 ± 0.028	286.23s
Ring-GNN	91.86 ± 0.45	2575.99s	0.512 ± 0.023	327.65s
PPGN	N/A	N/A	0.256 ± 0.054	334.69s
Deep-LRP	N/A	N/A	**0.223 ± 0.008**	72s
PNA	97.41 ± 0.16	N/A	0.320 ± 0.032	N/A
PG-GNN (Ours)	**97.51 ± 0.07**	82.60s	0.282 ± 0.011	6.92s

10

Conclusions

- Permutation-sensitive GNNs are ***more powerful*** than permutation-invariant ones since they are capable of modeling the relationships among neighboring nodes and thus counting graph substructures.
- A good ***approximation of the permutation-invariance*** (e.g., the invariance to 2-ary dependencies) can significantly reduce the computational complexity with a minimal loss of generalization capability.
- The proposed permutation sampling strategy achieves ***linear permutation sampling complexity*** and is promising to be incorporated into broader design.

Going Deeper into Permutation-Sensitive Graph Neural Networks

Contact: huangzhongyu2020@ia.ac.cn (Zhongyu Huang)

huiguang.he@ia.ac.cn (Huiguang He)

Preprint link: https://arxiv.org/abs/2205.14368

Code link: https://github.com/zhongyu1998/PG-GNN