A Practical, Progressively-Expressive GNN

Lingxiao Zhao, Louis Härtel, Neil Shah, Leman Akoglu

Is Expressivity Really Necessary?

- GNN with higher expressivity =>
  - Closer to universal function approximator
  - Higher computational cost
  - Potentially worse generalization

- How to study the impact of expressivity?
  - We need a model that is
    - **Practical, implementable**
    - **With tunable, progressive expressivity**

Improving Expressivity of GNN

- Random Node Initialization
  - Problem: generalization is not clear, randomness
- Subgraph Enhanced GNNs
  - Problem: expressivity is limited by 3-WL [Frasca et al. 22]
- Higher-Order GNNs
  - Linear Invariant Graph Network (k-IGN)
  - k-WL Inspired GNNs
  - Problem: <u>Not practical with k>3</u>

How to improve higher-order GNNs to have deserved properties?

k-WL

- Working on k-tuples $\overrightarrow{\boldsymbol{v}} = (v_1, v_2, \ldots, v_k)$ with color $\boldsymbol{wl}_k^{(t)}(\overrightarrow{\boldsymbol{v}})$
- Initial color (t=0): atomic type
  - $\boldsymbol{wl}_k^{(0)}(\overrightarrow{\boldsymbol{v}}) = \boldsymbol{wl}_k^{(0)}(\overrightarrow{\boldsymbol{u}})$ Iff $\overrightarrow{\boldsymbol{v}} \mapsto \overrightarrow{\boldsymbol{u}}$ is isomorphism of $G[\overrightarrow{\boldsymbol{v}}]$ & $H[\overrightarrow{\boldsymbol{u}}]$
- t-th iteration:
  - Let $\overrightarrow{\boldsymbol{v}}[x/i] := (v_1, \ldots, v_{i-1}, x, v_{i+1}, \ldots, v_k)$
  - $\boldsymbol{wl}_k^{(t+1)}(G, \overrightarrow{\boldsymbol{v}}) = \mathrm{HASH}\Big(\boldsymbol{wl}_k^{(t)}(G, \overrightarrow{\boldsymbol{v}}),$

11	12	13	14	15	16
21	22	23	24	25	26
31	32	33	34	35	36
41	42	43	44	45	46
51	52	53	54	55	56
61	62	63	64	65	66

$$\{\!\!\{\boldsymbol{wl}_k^{(t)}(G, \overrightarrow{\boldsymbol{v}}[x/1]) \,\big|\, x \in V(G)\}\!\!\}, \ldots,$$

$$\{\!\!\{\boldsymbol{wl}_k^{(t)}(G, \overrightarrow{\boldsymbol{v}}[x/k]) \,\big|\, x \in V(G)\}\!\!\}\Big)$$

Computational Bottleneck

- k-tuples [super-nodes]
  - n^k
- Connections among k-tuples [super-edges]
  - n*k for each k-tuple

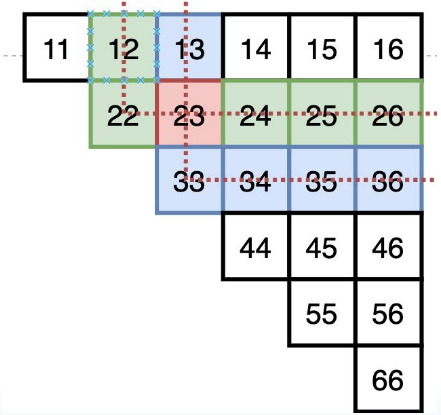
- **Can we reduce both parts?**

1 - Tuples to MultiSets (↓super-nodes)

- Removing ordering information

$$\vec{v} = (v_1, v_2, \ldots, v_k) \longrightarrow \tilde{\boldsymbol{v}} = \{\!\{v_1, v_2, \ldots, v_k\}\!\}$$

- **k-MultisetWL**
  - Initial color: isomorphism type
  - t-th iteration color updating:



$$\boldsymbol{mwl}_k^{(t+1)}(G, \tilde{\boldsymbol{v}}) = \mathrm{HASH}\Big(\boldsymbol{mwl}_k^{(t)}(G, \tilde{\boldsymbol{v}}), \Big\{\!\!\Big\{ \{\!\{\boldsymbol{mwl}_k^{(t)}(G, \tilde{\boldsymbol{v}}[x/1]) \big| x \in V(G)\}\!\}, \ldots, \{\!\{\boldsymbol{mwl}_k^{(t)}(G, \tilde{\boldsymbol{v}}[x/k]) \big| x \in V(G)\}\!\} \Big\}\!\!\Big\}\Big)$$

1 - Tuples to Multisets (↓super-nodes)

- Expressivity of k-MultisetWL
  - Thm. 1: Upper-bounded by k-WL
  - Thm. 2: No less powerful than (k-1)-WL
  - Thm. 3:
    Same expressivity as *doubly bijective k-pebble game*
    (k-WL ⇔ bijective k-pebble game)
  - Conjecture: (hard to find failure case)
    k-WL ⇔ k-MultisetWL

2 - Multisets to Sets (↓super-nodes &-edges)

- Removing repeated elements
  - $\tilde{\boldsymbol{v}} = \{\!\{v_1, v_2, ..., v_k\}\!\} \longrightarrow \hat{\boldsymbol{v}} = \{\hat{v}_1, ..., \hat{v}_m\}$
  - Set $\hat{\boldsymbol{v}}$ can has less elements, $1 \leq m \leq k$
- **k(≤)-SetWL**

$$\boldsymbol{swl}_k^{(t+1)}(G, \hat{\boldsymbol{v}}) = \text{HASH}\Big(\boldsymbol{swl}_k^{(t)}(G, \hat{\boldsymbol{v}}), \{\!\{\boldsymbol{swl}_k^{(t)}(G, \hat{\boldsymbol{v}} \cup \{x\}) \mid x \in V(G) \setminus \hat{\boldsymbol{v}}\}\!\}, \{\!\{\boldsymbol{swl}_k^{(t)}(G, \hat{\boldsymbol{v}} \setminus x) \mid x \in \hat{\boldsymbol{v}}\}\!\},$$

$$\Big\{\!\!\Big\{ \{\!\{\boldsymbol{swl}_k^{(t)}(G, \hat{\boldsymbol{v}}[x/o_G^{-1}(\hat{\boldsymbol{v}}, 1)]) \mid x \in V(G) \setminus \hat{\boldsymbol{v}}\}\!\}, ..., \{\!\{\boldsymbol{swl}_k^{(t)}(G, \hat{\boldsymbol{v}}[x/o_G^{-1}(\hat{\boldsymbol{v}}, m)]) \mid x \in V(G) \setminus \hat{\boldsymbol{v}}\}\!\} \Big\}\!\!\Big\}\Big)$$

  - Expressivity:
    Thm. 4: Upper-bounded by k-MultisetWL

3 - To Sets with Connectivity (↓super-nodes &-edges)

- Further reduce super-nodes
  - Only consider $\hat{v}$ with subgraph $G[\hat{v}]$ having **≤ c** connected components
  - **(k, c)(≤)-SetWL**
  - Expressivity: Thm. 5
    - (k,c)(≤)-SetWL has less expressivity than (k+1,c)(≤)-SetWL
    - (k,c)(≤)-SetWL has less expressivity than (k,c+1)(≤)-SetWL
    - (k,k)(≤)-SetWL ⇔ k(≤)-SetWL
  - Fine-grained, progressively expressive

Note: [SpeqNets, Morris et al. 22] also used the same idea of restricting connected components, concurrently.

4 - K-bipartite Connection (↓super-edges)

- Nearby super-nodes of a single m-set $\hat{\boldsymbol{v}}$ in k(≤)-SetWL
  - (m-1)-sets : $\hat{\boldsymbol{v}} \setminus x, \text{for } x \in \hat{\boldsymbol{v}}$ Define as $\mathcal{N}^G_{\text{left}}(\hat{\boldsymbol{v}})$
  - (m+1)-sets: $\hat{\boldsymbol{v}} \cup x, \text{for } x \in V(G) \setminus \hat{\boldsymbol{v}}$ Define as $\mathcal{N}^G_{\text{right}}(\hat{\boldsymbol{v}})$
  - m-sets: $\hat{\boldsymbol{v}} \cup x \setminus y, \text{ for } x \in V(G) \setminus \hat{\boldsymbol{v}}, y \in \hat{\boldsymbol{v}}$
- Connections to m-sets can be safely <u>removed</u>!

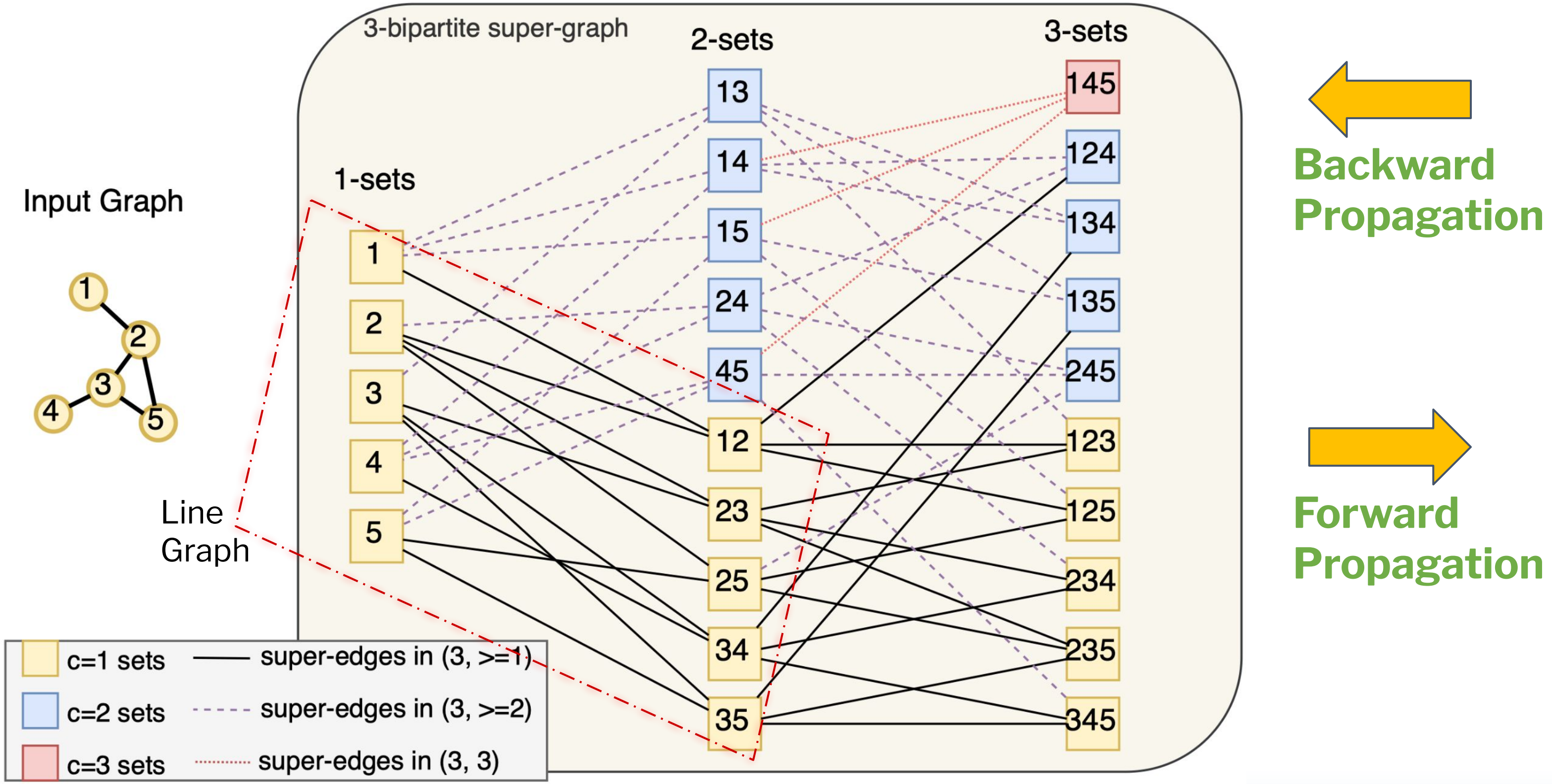
$$\boldsymbol{swl}^{(t+\frac{1}{2})}_{k,c}(G, \hat{\boldsymbol{v}}) = \text{HASH}\{\!\{\boldsymbol{swl}^{(t)}_{k,c}(G, \hat{\boldsymbol{u}}) \mid \hat{\boldsymbol{u}} \in \mathcal{N}^G_{\text{right}}(\hat{\boldsymbol{v}})\}\!\}$$

Backward Propagation

$$\boldsymbol{swl}^{(t+1)}_{k,c}(G, \hat{\boldsymbol{v}}) = \text{HASH}\Big(\boldsymbol{swl}^{(t)}_{k,c}(G, \hat{\boldsymbol{v}}),\ \boldsymbol{swl}^{(t+\frac{1}{2})}_{k,c}(G, \hat{\boldsymbol{v}}), \{\!\{\boldsymbol{swl}^{(t)}_{k,c}(G, \hat{\boldsymbol{u}}) \mid \hat{\boldsymbol{u}} \in \mathcal{N}^G_{\text{left}}(\hat{\boldsymbol{v}})\}\!\}, \{\!\{\boldsymbol{swl}^{(t+\frac{1}{2})}_{k,c}(G, \hat{\boldsymbol{u}}) \mid \hat{\boldsymbol{u}} \in \mathcal{N}^G_{\text{left}}(\hat{\boldsymbol{v}})\}\!\}\Big)$$

Forward Propagation

Visualizing K-bipartite Super-graph



(k,c)(≤)-SetWL to (k,c)(≤)-SetGNN

- "Color" Initialization
  - Each m-set should be initialized with the isomorphism type of its induced subgraph.
  - Use a base 1-WL GNN to encode isomorphism type

$$h^{(0)}(\hat{\boldsymbol{v}}) = \mathrm{BaseGNN}(G[\hat{\boldsymbol{v}}])$$

- Message passing among k-bipartite super-graph

$$h^{(t+\frac{1}{2})}(\hat{\boldsymbol{v}}) = \sum_{\hat{\boldsymbol{u}} \in \mathcal{N}^G_{\mathrm{right}}(\hat{\boldsymbol{v}})} \mathrm{MLP}^{(t+\frac{1}{2})}(h^{(t)}(\hat{\boldsymbol{u}}))$$ **Backward Propagation**

$$h^{(t+1)}(\hat{\boldsymbol{v}}) = \mathrm{MLP}^{(t)}\Big(h^{(t)}(\hat{\boldsymbol{v}}), h^{(t+\frac{1}{2})}(\hat{\boldsymbol{v}}), \sum_{\hat{\boldsymbol{u}} \in \mathcal{N}^G_{\mathrm{left}}(\hat{\boldsymbol{v}})} \mathrm{MLP}^{(t)}_A(h^{(t)}(\hat{\boldsymbol{u}})), \sum_{\hat{\boldsymbol{u}} \in \mathcal{N}^G_{\mathrm{left}}(\hat{\boldsymbol{v}})} \mathrm{MLP}^{(t)}_B(h^{(t+\frac{1}{2})}(\hat{\boldsymbol{u}}))\Big)$$

Forward Propagation

(k,c)(≤)-SetGNN*

- Bidirectional Sequential Message Passing

$$m = k-1 \text{ to } 1, \forall m\text{-set } \hat{\boldsymbol{v}}, h^{(t+\frac{1}{2})}(\hat{\boldsymbol{v}}) = \mathrm{MLP}^{(t)}_{m,1}\Big(h^{(t)}(\hat{\boldsymbol{v}}), \sum_{\hat{\boldsymbol{u}} \in \mathcal{N}^{G}_{\text{right}}(\hat{\boldsymbol{v}})} \mathrm{MLP}^{(t)}_{m,2}(h^{(t+\frac{1}{2})}(\hat{\boldsymbol{u}}))\Big)$$ **Backward**

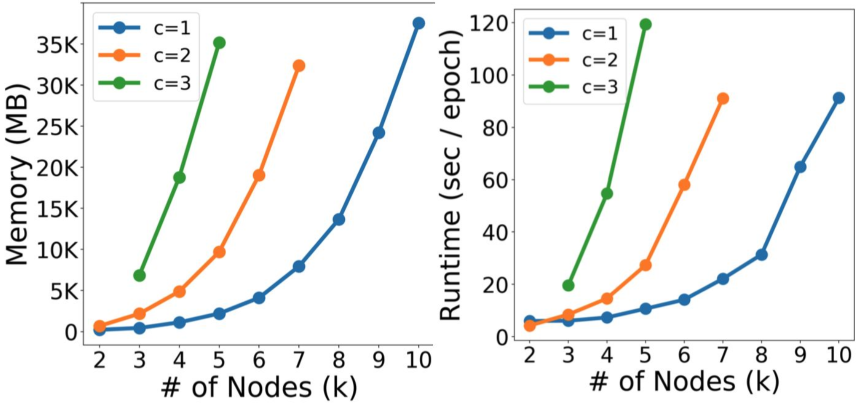
$$m = 2 \text{ to } k, \forall m\text{-set } \hat{\boldsymbol{v}}, h^{(t+1)}(\hat{\boldsymbol{v}}) = \mathrm{MLP}^{(t+\frac{1}{2})}_{m,1}\Big(h^{(t+\frac{1}{2})}(\hat{\boldsymbol{v}}), \sum_{\hat{\boldsymbol{u}} \in \mathcal{N}^{G}_{\text{left}}(\hat{\boldsymbol{v}})} \mathrm{MLP}^{(t+\frac{1}{2})}_{m,2}(h^{(t+1)}(\hat{\boldsymbol{u}}))\Big)$$ **Forward**

- Expressivity:
  - Thm. 6: (k,c)(≤)-SetGNN ⇔ (k,c)(≤)-SetWL
  - Thm. 7:
    t-layer (k,c)(≤)-SetGNN* is more expressive than
    t-layer (k,c)(≤)-SetGNN

Experimental Results

Table 4: $(k, c)(\leq)$-SETGNN* performances on ZINC-12K by varying (k,c). **Test MAE** at lowest Val. MAE, and lowest <u>Test MAE</u>.

k	c	Train loss	Val. MAE	Test MAE
2	1	0.1381 ± 0.0240	0.2429 ± 0.0071	0.2345 ± 0.0131
3	1	0.1172 ± 0.0063	0.2298 ± 0.0060	0.2252 ± 0.0030
4	1	0.0693 ± 0.0111	0.1645 ± 0.0052	0.1636 ± 0.0052
5	1	0.0643 ± 0.0019	0.1593 ± 0.0051	0.1447 ± 0.0013
6	1	0.0519 ± 0.0064	0.0994 ± 0.0093	0.0843 ± 0.0048
7	1	0.0543 ± 0.0048	0.0965 ± 0.0061	0.0747 ± 0.0022
8	1	0.0564 ± 0.0152	0.0961 ± 0.0043	<u>0.0732 ± 0.0037</u>
9	1	0.0817 ± 0.0274	0.0909 ± 0.0094	0.0824 ± 0.0056
10	1	0.0894 ± 0.0266	0.1060 ± 0.0157	0.0950 ± 0.0102
2	2	0.1783 ± 0.0602	0.2913 ± 0.0102	0.2948 ± 0.0210
3	2	0.0640 ± 0.0072	0.1668 ± 0.0078	0.1391 ± 0.0102
4	2	0.0499 ± 0.0043	0.1029 ± 0.0033	0.0836 ± 0.0010
5	2	0.0483 ± 0.0017	0.0899 ± 0.0056	**0.0750 ± 0.0027**
6	2	0.0530 ± 0.0064	0.0927 ± 0.0050	0.0737 ± 0.0006
7	2	0.0547 ± 0.0036	0.0984 ± 0.0047	0.0784 ± 0.0043
3	3	0.0798 ± 0.0062	0.1881 ± 0.0076	0.1722 ± 0.0086
4	3	0.0565 ± 0.0059	0.1121 ± 0.0066	0.0869 ± 0.0026
5	3	0.0671 ± 0.0156	0.1091 ± 0.0097	0.0920 ± 0.0054



(a) Memory usage (b) Training time

Figure 2: $(k, c)(\leq)$-SETGNN*'s footprint scales practically with both k and c in memory (a) and running time (b) – results on ZINC-12K.

Summary

- (k,c)(≤)-SetGNN(*): a practical and progressively expressive GNN improved from k-WL.
- Code: https://github.com/LingxiaoShawn/KCSetGNN

Thank you!