Are Graph Transformers Necessary? Efficient Long-Range Message Passing with Fractal Nodes in MPNNs

Jeongwhan Choi[1], Seungjun Park[1], Sumin Park[1], Sung-Bae Cho[2], Noseong Park[1]

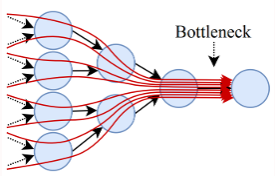
[1]KAIST [2]Yonsei University

AAAI 2026

Problem: Over-squashing Limits Long-Range in MPNNs

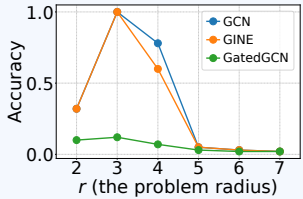
- **Graph Transformers** address over-squashing[1] and long-range dependency, but overlook locality of MPNNs[2] struggle to scale to large graphs.

Over-squashing



- Information from many nodes → **compressed** into fixed-size vectors

Limited Long-Range



- MPNNs **struggle** when $r > 3$ in TreeNeighbourMatch

[1] Uri Alon and Eran Yahav. "On the Bottleneck of Graph Neural Networks and its Practical Implications". In: *International Conference on Learning Representations*. 2021.

[2] Yujie Xing et al. "Less is More: on the Over-Globalizing Problem in Graph Transformers". In: *International Conference on Machine Learning*. 2024.

Motivation: Can MPNNs Achieve Long-Range Interaction?

Question: Can we enable long-range interaction within MPNNs, without Graph Transformers?

- **Key Insight:** Graph partitioning induces **fractal-like properties**
  - Subgraphs share similar structural characteristics with the original graph
  - This self-similarity appears across scales[a][b]

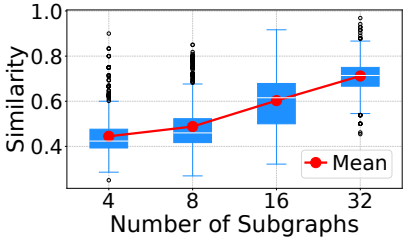


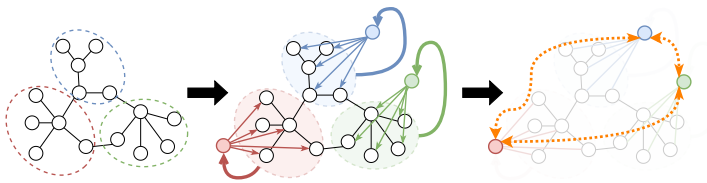
Figure: Structural similarity (Betweenness centrality distribution) between subgraphs and original graph.

Our Idea: Leverage this fractal structure → Introduce **fractal nodes** at subgraph level

[a] Chaoming Song et al. "Self-similarity of complex networks". In: *Nature* 433.7024 (2005), pp. 392–395.

[b] Jin Seop Kim et al. "Fractality and self-similarity in scale-free networks". In: *New Journal of Physics* 9.6 (2007), p. 177.

Our Solution: Fractal Nodes



Fractal nodes **coexist** with original nodes → Create **shortcuts** for long-range interaction

- Partition graph into C subgraphs (e.g., METIS)
- Create fractal node (🔴, 🔵, 🟢) per subgraph
- Each layer ℓ:
  1. **Node MP:** $\tilde{h}_{v,c}^{(\ell+1)} = \text{MPNN}(h_{v,c}^{(\ell)}, \{h_{u,c}^{(\ell)}\}_{u \in \mathcal{N}_v})$
  2. **Fractal node update:** $f_c^{(\ell+1)} = \text{LPF}(\{\tilde{h}_{v,c}^{(\ell+1)}\}) + \omega_c^{(\ell)} \cdot \text{HPF}(\{\tilde{h}_{v,c}^{(\ell+1)}\})$
  3. **Combine:** $h_{v,c}^{(\ell+1)} = \tilde{h}_{v,c}^{(\ell+1)} + f_c^{(\ell+1)}$
- FN_M: + MLP-Mixer at final layer for inter-subgraph

How to Aggregate: Beyond Mean Pooling

- How should fractal nodes aggregate information from their subgraph?
- Simple approach: Mean pooling (like Virtual Node)

$$\text{Mean} = \frac{1}{|\mathcal{V}_c|} \sum_{v \in \mathcal{V}_c} h_v$$

Problem (Theorem 3.1):
Mean pooling extracts **only the DC component**
$\rightarrow$ **Discards high-frequency** (local details)

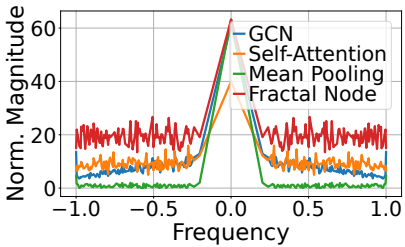


Figure: Mean pooling: only low frequency. Fractal node: both frequencies

- **Our approach:** Combine LPF and HPF

$$f_c^{(\ell+1)} = \underbrace{\text{LPF}(\{h_{v,c}\})}_{\text{Sub-global trend}} + \omega_c \cdot \underbrace{\text{HPF}(\{h_{v,c}\})}_{\text{Local details}}$$

Fractal Node Update: Preserving Both Global and Local Information

$$f_c^{(\ell+1)} = \underbrace{\mathrm{LPF}(\{h_{v,c}\})}_{\textbf{Sub-global trend}} + \omega_c \underbrace{\mathrm{HPF}(\{h_{v,c}\})}_{\textbf{Local details}}$$

$$\mathrm{LPF} := \frac{1}{|\mathcal{V}_c|} \sum_{v \in \mathcal{V}_c} h_v$$

- Subgraph mean (DC component)
- Captures overall trend

$$\mathrm{HPF} := h_v - \mathrm{LPF}$$

- Deviation from mean
- **Preserves local variations**

- ω_c: Learnable $\rightarrow$ optimal balance per layer
- FN_M: + MLP-Mixer at final layer
  $\rightarrow$ Inter-subgraph communication

vs. Virtual Node[a]

- VN = global + LPF only
- FN = **subgraph + LPF + HPF**

[a] Chen Cai et al. "On the connection between mpnn and graph transformer". In: *International Conference on Machine Learning*. PMLR. 2023, pp. 3408–3430.

Why Fractal Nodes Work: Shortcuts Reduce Resistance

- Fractal nodes create shortcuts: k-hop $\rightarrow$ **2-hop** via FN
- We analyze this from two perspectives[3],[4]:

Theorem 4.1 (Resistance Reduction):

$$R_f(u, v) \leq R(u, v)$$

- Effective resistance = difficulty of information flow
- Adding paths via FN $\rightarrow$ **lower resistance**

Theorem 4.2 (Signal Propagation):

$$\|h_u^{(\ell)} - h_v^{(\ell)}\| \leq \exp(-\ell / R_f(u, v)) \|h_u^{(0)} - h_v^{(0)}\|$$

- Smaller $R_f \rightarrow$ **slower decay** $\rightarrow$ better long-range

[3] Mitchell Black et al. "Understanding Oversquashing in GNNs through the Lens of Effective Resistance". In: *International Conference on Machine Learning*. 2023.

[4] Francesco Di Giovanni et al. "On Over-Squashing in Message Passing Neural Networks: The Impact of Width, Depth, and Topology". In: *International Conference on Machine Learning*. 2023.

Empirical Validation: Over-squashing Mitigation

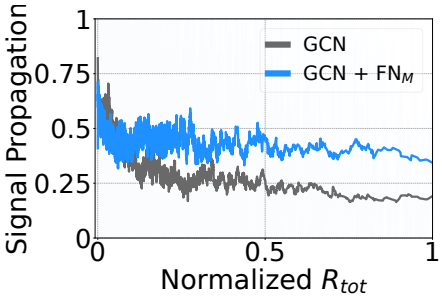


Figure: Signal propagation on PEPTIDES-FUNC

- X-axis: Effective resistance
- GCN: signal drops quickly
- **GCN+**FN$_M$: maintains signal

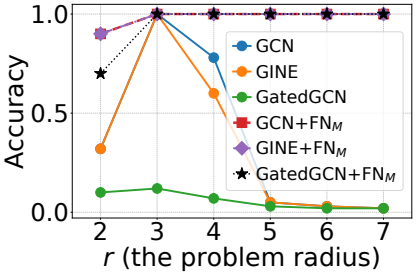


Figure: TREENEIGHBOURMATCH

- Task: propagate info across depth r
- MPNNs: **fail at** $r > 4$
- **With FN**: works up to $r = 7$

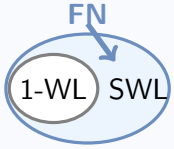
Expressive Power: Beyond 1-WL

Table: Synthetic datasets: Graphs indistinguishable by 1-WL

Method	CSL	SR25	EXP
GCN	10.00	6.67	52.17
GINE	10.00	6.67	51.35
GatedGCN	10.00	6.67	51.25
GCN + FN_M	39.67	**100.0**	86.40
GINE + FN_M	47.33	**100.0**	95.58
GatedGCN + FN_M	**49.67**	**100.0**	**96.50**

Theoretical insight:

- Subgraph-based methods exceed 1-WL[a]
- Fractal nodes achieve power comparable to Subgraph WL[b]



[a] Lingxiao Zhao et al. "From Stars to Subgraphs: Uplifting Any GNN with Local Structure Awareness". In: *International Conference on Learning Representations*. 2022.

[b] Bingxu Zhang et al. "The expressive power of graph neural networks: A survey". In: *arXiv preprint arXiv:2308.08235* (2023).

Main Results: Graph-Level Benchmarks

Table: Performance comparison on 6 benchmark datasets

Method	PEP-FUNC AP ↑	PEP-STRUCT MAE ↓	MNIST Acc ↑	CIFAR10 Acc ↑	MOLHIV ROC ↑	MOLTOX21 ROC ↑
GCN	0.6328	0.2758	0.9269	0.5423	0.7529	0.7525
GINE	0.6405	0.2780	0.9705	0.6131	0.7885	0.7730
GatedGCN	0.6300	0.2778	0.9776	0.6628	0.7874	0.7641
GraphGPS	0.6534	0.2509	0.9805	0.7230	0.7880	0.7570
GRIT	0.6988	0.2460	0.9811	**0.7647**	-	-
Graph-ViT	0.6970	0.2449	0.9846	0.7158	0.7997	0.7910
Exphormer	0.6527	0.2481	0.9841	0.7469	-	-
GECO	0.6975	0.2464	-	-	0.7980	-
GNN-AK+	0.6480	0.2736	-	0.7219	0.7961	-
CRaWl	0.6963	0.2506	0.9794	0.6901	0.7707	-
GCN + FN_M	0.6787	0.2464	0.9455	0.6413	0.7866	0.7882
GINE+FN_M	**0.7018**	**0.2446**	0.9786	0.6672	**0.8127**	**0.7926**
GatedGCN+FN_M	0.6950	0.2453	**0.9848**	0.7526	0.8097	0.7922

Scalability: Million-Node Graphs

Method	ogbn-arxiv (169K nodes)	ogbn-products (**2.4M** nodes)
GraphGPS	70.97	**OOM**
Exphormer	72.44	**OOM**
GCN	71.74	75.64
GCN + FN	**73.03**	81.29
GraphSAGE + FN_M	72.54	**83.11**

Graph Transformers: **Out of Memory**
Fractal Nodes: **Scales to 2.4M nodes**

Method	Time	Memory
GCN	1.27s	16.49GB
GraphGPS	1.32s	**38.91GB**
Exphormer	0.74s	**34.04GB**
GCN + FN	1.27s	16.49GB
GCN + FN_M	1.27s	16.49GB

No overhead!
Same time & memory as GCN

Summary

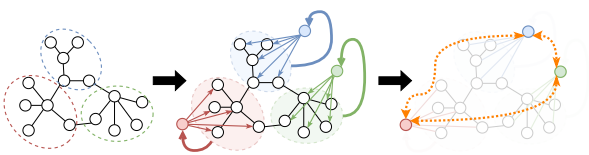
Question:

- Can MPNNs achieve long-range without Transformers?

Key Insight:

- Graph partitioning → fractal-like properties

Our Solution — Fractal Nodes:



Takeaway: Fractal Nodes enable **long-range interaction** in MPNNs while maintaining **efficiency & scalability**

Are Graph Transformers necessary?

For many tasks, **Fractal Nodes offer a compelling alternative.**

Thank You!

Questions?

Homepage: jeongwhanchoi.com
Email: jeongwhan.choi@kaist.ac.kr

Paper

Code