Connectivity-Guided Sparsification of 2-FWL GNNs: Preserving Full Expressivity with Improved Efficiency

Rongqin Chen[1], Fan Mo[2, 3], Pak Lon Ip[1], Shenghui Zhang[1], Dan Wu[4], Ye Li[4, 5], Leong Hou U[1]*

[1]University of Macau
[2]ZOZO Research
[3]Waseda University
[4]Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences
[5]Faculty of Computer Science and Control Engineering, Shenzhen University of Advanced Technology
chen.rongqin@connect.um.edu.mo; bakubonn@toki.waseda.jp; paklonip@um.edu.mo; zhang.shenghui@connect.um.edu.mo; dan.wu@siat.ac.cn; ye.li@siat.ac.cn; ryanlhu@um.edu.mo

Abstract

Higher-order Graph Neural Networks (HOGNNs) based on the 2-FWL test achieve superior expressivity by modeling 2-node and 3-node interactions, but incur cubic computational cost. Existing efficiency methods typically reduce this burden at the expense of expressivity. We propose Co-Sparsify, a connectivity-aware sparsification framework that eliminates provably redundant computations while preserving full 2-FWL expressive power. Our key insight is that 3-node interactions are expressively necessary only within biconnected components, namely, maximal subgraphs where every node pair lies on a cycle. Outside these components, structural relationships are fully captured via 2-node message passing and graph readouts, rendering higher-order modeling unnecessary. Co-Sparsify restricts 2-node message passing to connected components and 3-node interactions to biconnected components, eliminating redundant computation without approximation or sampling. We prove that Co-Sparsified GNNs match the expressivity of the 2-FWL test. Empirically, when applied to PPGN, Co-Sparsify matches or exceeds accuracy on synthetic substructure counting tasks and achieves state-of-the-art performance on real-world benchmarks (ZINC, QM9 and TUD). This study demonstrates that high expressivity and scalability are not mutually exclusive: principled, topology-guided sparsification enables powerful, efficient GNNs with theoretical guarantees.

Code — https://github.com/RongqinChen/Co-Sparsify
Extended version — https://arxiv.org/abs/2511.12838

1 Introduction

Graph Neural Networks (GNNs) are the predominant framework for learning on graph-structured data. A central challenge is enhancing their *expressive power*—the ability to distinguish non-isomorphic graphs and detect fine-grained structural patterns. Standard message-passing GNNs, such as GCN (Kipf and Welling 2017), GIN (Xu et al. 2019), and GAT (Velickovic et al. 2018), are limited by the expressivity of the 1-WL test. Higher-order GNNs (HOGNNs), inspired by the k-dimensional Weisfeiler-Leman (k-WL) and k-Folklore WL (k-FWL) hierarchies (Morris et al. 2019; Maron et al. 2019), achieve greater expressivity by performing message passing over k-tuples of nodes. These models strictly subsume classical GNNs and even recent architectures like Graph Transformers (Rampásek et al. 2022) in expressive power.

However, HOGNNs suffer from combinatorial complexity: time and memory scale exponentially with order k, rendering higher orders impractical. Among these, 2-FWL GNNs offer a practical trade-off, matching 3-WL expressivity while remaining tractable (Maron et al. 2019). Yet they still require $\mathcal{O}(n^3)$ memory for an n-node graph. Even opti-

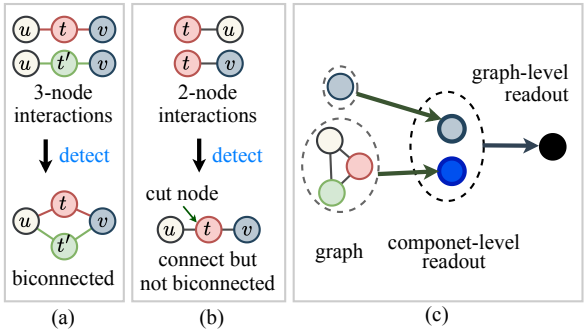

Figure 1: **Principle of connectivity-aware sparsification.** (a) *Biconnected case*: 3-node interactions are necessary to distinguish paths differing in intermediate nodes. (b) *Cut-node case*: When all u-v paths pass through a cut node t, they decompose into t-u and t-v subpaths. Since 2-FWL already captures all 2-node interactions, the 3-node interaction (u, t, v) provides no additional expressivity. (c) *Disconnected case*: Component-level structural properties (e.g., component size) are captured by component-level readout, while global properties (e.g., component count) are captured by graph-level readout, rendering explicit 2- or 3-node modeling unnecessary.

*Corresponding author.

mized variants like PPGN (Maron et al. 2019), which leverage batched matrix operations, incur $\mathcal{O}(\eta^2)$ per-graph memory due to padding to the largest graph size η in a batch. This limits scalability on large or structurally diverse graphs.

Prior efficiency methods—such as subgraph sampling (e.g., ESAN (Zhao et al. 2022)), set-based reduction (e.g., KCSetGNN (Zhao, Shah, and Akoglu 2022)), or localized aggregation (e.g., 1-2-3-GNN (Morris et al. 2019))—trade expressivity for efficiency through approximation, sampling, or restricted neighborhoods. We ask instead: **Can we improve efficiency by eliminating only computations that are provably redundant for expressivity, without sacrificing expressive power?**

We focus on 2-FWL GNNs, which update each node pair (u, v) by aggregating messages from coupled pairs $((u, t), (t, v))$ over all $t \in V$. When u, t, and v are distinct, this forms a *3-node interaction*—critical for capturing structures like two internally disjoint paths[1], enabling cycle and biconnectivity detection. When $u = t$ or $t = v$, this interaction reduces to a *2-node interaction*, capturing only whether paths exist between u and v.

Our key insight is that **3-node interactions are expressively necessary only within biconnected components**. By Menger's theorem (Göring 2000), any two nodes in the same biconnected component are connected by at least two internally disjoint paths (Figure 1(a)). Only in this case does modeling triplets (u, t, v) provide additional expressive power. Outside biconnected components, 3-node interactions are redundant. First, all u-v paths must pass through a *cut node* t, decomposing the connection into u-t and t-v subpaths (Figure 1(b)). Since 2-FWL already captures all pairwise interactions, aggregating over (u, t, v) yields no expressivity gain. Second, for *disconnected* node pairs, structural context—such as component count or size—can be encoded by readout functions, rendering explicit 2- or 3-node modeling unnecessary (Figure 1(c)).

We propose **Co-Sparsify**, a connectivity-aware sparsification scheme that preserves full 2-FWL expressivity by eliminating only provably redundant computations. The method restricts 2-node interactions to pairs within the same connected component and 3-node interactions to triples within the same biconnected component. These structural constraints ensure that all removed operations are expressively redundant. No sampling, approximation, or heuristic pruning is used—sparsification is guided purely by exact graph topology.

Co-Sparsify incurs negligible preprocessing overhead: connected components are identified via breadth-first search, and biconnected components via block-cut tree decomposition (Hopcroft and Tarjan 1973), both running in $O(n + m)$ time.

We prove that 2-FWL GNNs implemented with Co-Sparsify (Co-Sparsified GNNs) achieve expressivity equivalent to 2-FWL under standard assumptions: injective aggregation and consistent initialization. The framework generalizes to any 2-FWL architecture, including PPGN (Maron et al. 2019) and TGT (Hussain, Zaki, and Subramanian 2024). When applied to PPGN, Co-Sparsify accurately counts paths and cycles, empirically confirming the preservation of expressivity. On both synthetic tasks and real-world benchmarks such as ZINC and QM9, it achieves performance on par with or superior to the original model.

Our contributions include:

- We identify that *3-node interactions* in 2-FWL GNNs are expressively useful *only within biconnected components*, and *2-node interactions* matter *only within connected components*.
- We propose a *connectivity-aware message-passing scheme*—**Co-Sparsify**—that restricts 3-node interactions to biconnected components and 2-node interactions to connected components. Our method eliminates provably redundant computations without approximation, sampling, or heuristic pruning.
- We prove that Co-Sparsified GNNs are *as powerful as the 2-FWL test* in distinguishing non-isomorphic graphs—making it the first sparsification method for HOGNNs with *guaranteed expressivity preservation*.
- We demonstrate that Co-Sparsify achieves preserved or improved accuracy across synthetic substructure counting tasks and real-world graph benchmarks.

Our work shows that high expressivity need not require uniform, dense computation over all node tuples. Instead, by aligning message passing with *structural criticality*—here, connected and biconnected components—we achieve efficiency through *insight*, not approximation. This opens a principled path toward scalable, theoretically grounded GNNs that preserve expressive power by design.

2 Preliminaries

This section introduces notation and background concepts used throughout the paper.

Notation. We denote an undirected graph as $G = (V, E)$, where V is a set of n nodes and $E \subseteq V \times V$ is a set of m edges. The adjacency matrix $A \in \{0, 1\}^{n \times n}$ satisfies $A[u, v] = 1$ if $(u, v) \in E$, and 0 otherwise. The degree matrix D is diagonal, with $D[v, v] = \sum_u A[v, u]$. The random walk probability matrix is computed as $\widehat{A} = D^{-1}A$. We represent multisets using $\{\!\{\cdot\}\!\}$.

Graph Connectivity. A *connected component* is a maximal subgraph in which all node pairs are mutually reachable. A *biconnected component* is a maximal subgraph (on at least three nodes) that remains connected after removing any single node. *Cut nodes*—whose removal increases the number of connected components—are shared across biconnected components and bridge edges. The *block-cut tree* decomposition identifies these components efficiently in $O(n + m)$ time (Hopcroft and Tarjan 1973). This hierarchical structure underpins our sparsification strategy.

Expressive Power. Two graphs G and H are *isomorphic*, denoted $G \simeq H$, if there exists a bijection $\pi : V_G \to V_H$ such that $(u, v) \in E_G$ if and only if $(\pi(u), \pi(v)) \in E_H$. A GNN *distinguishes* two non-isomorphic graphs if it produces different graph-level representations for them. It *captures* a substructure Q if its representations reliably differ

[1] u-v paths are *internally disjoint* if they share no intermediate nodes—e.g., $u \to v$, $u \to t_1 \to v$, and $u \to t_2 \to v$.

between graphs that contain Q and those that do not. The *expressive power* of a GNN is defined by its ability to distinguish non-isomorphic graphs and detect such substructures.

2-FWL GNNs. These models update node pair representations following the 2-FWL test. Let $\boldsymbol{h}^{(l)}(u,v)$ be the representation of pair (u,v) at layer l. Initial features combine node, edge, and structural information:

$$\boldsymbol{h}^{(0)}(u,v) = (\boldsymbol{x}(u), \boldsymbol{x}(v), \boldsymbol{e}(u,v), \boldsymbol{p}(u,v)), \tag{1}$$

where $\boldsymbol{x}(\cdot)$ are node features, $\boldsymbol{e}(u,v)$ is the edge feature (zero if no edge), and $\boldsymbol{p}(u,v)$ is a *structural encoding* (SE). A common SE is relative random walk probability (RRWP):

$$\boldsymbol{P}[u,v,:] = \left[I, \widehat{A}, \widehat{A}^2, \dots\right][u,v,:].$$

It encodes self-pair status, direct connectivity, and multi-hop reachability.

At layer l, each pair (u,v) aggregates messages over intermediate nodes $t \in V$:

$$\begin{aligned}\boldsymbol{t}^{(l)}(u,v) &= \left\{\!\!\left\{ \phi^{(l)}\left(\boldsymbol{h}^{(l-1)}(u,t), \boldsymbol{h}^{(l-1)}(t,v)\right) \mid t \in V \right\}\!\!\right\},\\ \boldsymbol{h}^{(l)}(u,v) &= \Phi^{(l)}\left(\boldsymbol{h}^{(l-1)}(u,v),\ \text{AGG}\left(\boldsymbol{t}^{(l)}(u,v)\right)\right),\end{aligned} \tag{2}$$

where $\Phi^{(l)}$ is learnable and AGG is injective (e.g., sum or MLP-based). Full aggregation over all t leads to $\mathcal{O}(n^3)$ memory and computation per layer.

To improve efficiency, PPGN (Maron et al. 2019) uses batched matrix operations. Let $\boldsymbol{H}^{(l)} \in \mathbb{R}^{b\times d\times \eta\times \eta}$ be the batched pair representations, where b is batch size, d feature dimension, and η the maximum graph size after padding. PPGN updates via:

$$\boldsymbol{H}^{(l)} = \Phi^{(l)}\left(\boldsymbol{H}^{(l-1)},\ \boldsymbol{H}^{(l-1)} \circledast \boldsymbol{H}^{(l-1)}\right), \tag{3}$$

where $\circledast$ denotes matrix multiplication along node dimensions. For graph i, this computes:

$$\boldsymbol{H}^{(l)}[i,:,u,v] = \Phi^{(l)}\left(\boldsymbol{H}^{(l-1)}[i,:,u,v], \sum_{t\in V} \boldsymbol{H}^{(l)}[i,:,u,t] \otimes \boldsymbol{H}^{(l)}[i,:,t,v]\right) \tag{4}$$

where $\otimes$ denotes element-wise multiplication.

Despite avoiding explicit loops, PPGN still requires $\mathcal{O}(\eta^2)$ memory per graph due to padding—limiting scalability on large or irregular graphs.

Problem Setup. We aim to design sparsified 2-FWL GNNs that remove only computations provably redundant for expressivity. Our goal is to preserve full 2-FWL expressive power while improving efficiency. We prove that the resulting model matches 2-FWL expressivity and evaluate its performance on substructure counting and graph-level prediction tasks.

3 Related Work

Expressive GNNs and the WL hierarchy. The expressive power of GNNs is often bounded by the 1-WL test, limiting their ability to distinguish non-isomorphic graphs (Xu et al. 2019). Standard models like GCN (Kipf and Welling 2017), GAT (Velickovic et al. 2018), and Gated GCN (Li et al. 2016) fall into this class. To surpass these limits, HOGNNs based on the k-WL and k-Folklore WL (k-FWL) hierarchies operate on k-tuples of nodes, achieving greater expressivity at $\mathcal{O}(n^k)$ cost (Morris et al. 2019; Maron et al. 2019). Recent works align GNNs with 2-FWL via low-rank attention (Puny, Ben-Hamu, and Lipman 2020), often introducing architectural complexity to balance expressivity and efficiency.

Efficient HOGNNs. To reduce the cost of HOmodels, several directions have emerged. Set-based approaches reformulate k-tuple learning as k-set learning, reducing the number of representations by exploiting permutation invariance (Zhao, Shah, and Akoglu 2022), while local message-passing methods restrict aggregation to neighborhoods (Morris, Rattan, and Mutzel 2020). Subgraph-based methods enhance expressivity by lifting graphs into higher-dimensional spaces (Bodnar et al. 2021; Bouritsas et al. 2023), though often without addressing scalability. Simplified spectral models achieve linear complexity (Wu et al. 2019), but may sacrifice structural fidelity through approximation.

Graph Sparsification. Sparsification reduces computational load by pruning non-essential edges or computations. Learning-based methods train edge masks to improve generalization (Zheng et al. 2020; Rathee et al. 2021) or use optimization frameworks like ADMM (Li et al. 2022). Others sparsify model weights via pruning or sparse training (Peng et al. 2022). While effective, these approaches require additional training and offer no guarantees on expressivity preservation. In contrast, our method, *Co-Sparsify*, uses the graph's connectivity structure—connected and biconnected components—to deterministically remove only expressivity-redundant interactions, with provable 2-FWL equivalence.

Hierarchical GNNs. Hierarchical models capture multiscale structure via pooling or auxiliary layers. DiffPool learns soft cluster assignments for hierarchical representation (Ying et al. 2018), while HGNet organizes graphs into layers for logarithmic message passing (Rampášek and Wolf 2021). These methods enhance expressivity but add architectural complexity. Our approach differs by simplifying computation—leveraging structural hierarchies (e.g., block-cut trees) to sparsify message passing without added parameters or layers.

Our Position in the Literature. While prior work has made significant progress in improving the efficiency of HOGNNs, most approaches either reduce expressivity or still incur high computational costs. In contrast, we propose a novel sparsification strategy for 2-FWL GNNs that preserves full expressive power while reducing runtime. Our method leverages insights into graph connectivity to identify and retain only the most structurally informative interactions, achieving both theoretical guarantees and strong empirical performance.

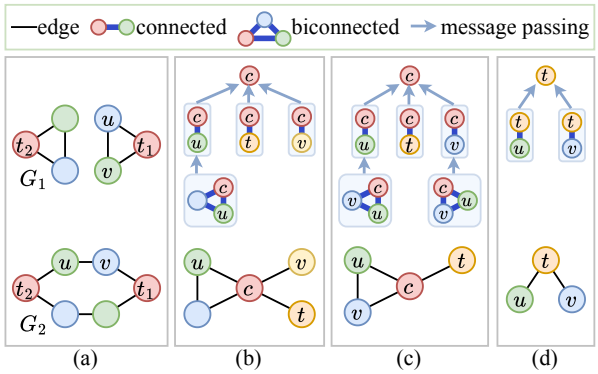

Figure 2: **Justification for connectivity-aware sparsification.** (a) Distinguishing graphs with different path multiplicities (e.g., $u \to v$ vs. $u \to t \to v$) requires 3-node interactions. (b) Cut node c separates u, t, and v into distinct components; pairwise interactions (u, c), (t, c), (v, c) suffice to determine the structure. (c) Cut node c separates t from $\{u, v\}$, with u and v connected; structure is captured by (u, c), (t, c), and (v, c). (d) Cut node t separates u and v; interactions (u, t) and (v, t) fully determine the graph.

4 Method

We present **Co-Sparsify**, a connectivity-aware sparsification framework for HOGNNs. It eliminates only expressivity-redundant computations while preserving full 2-FWL expressive power, yielding **Co-Sparsified GNNs**. Leveraging connected component and block-cut tree decompositions, Co-Sparsify restricts 3-node interactions to biconnected components and 2-node modeling to connected components—reducing computation without compromising substructure detection.

4.1 Connectivity-Guided Sparsification Principle

The expressive power of 2-FWL GNNs stems from modeling 2-node and 3-node interactions. However, we show that such interactions are only expressively necessary within connected and biconnected components, respectively. Outside these regions, higher-order interactions contribute no additional discriminative power and can be provably removed without loss of expressivity. This principle is formalized in lemmas.

Lemma 1 (Necessity of 3-Node Interactions in Biconnected Components)**.** *Let u and v be distinct nodes in a graph G. If u and v lie in the same biconnected component, then there are at least two internally disjoint paths between them. Detecting such configurations requires 3-node interactions and cannot be captured by 2-node interactions alone.*

Proof. By Menger's theorem, biconnectedness implies multiple internally disjoint paths. A 3-node interaction (u, t, v) indicates whether t lies on a path from u to v, and aggregating over t allows the model to detect path multiplicity—essential for detecting biconnectedness. □

As shown in Figure 2(a), the node pairs (u, v) in G_1 and G_2 are connected by multiple internally disjoint paths. While 1-WL cannot distinguish these graphs or pairs, 2-FWL can, illustrating the necessity of 3-node interactions.

Lemma 2 (Redundancy of 3-Node Interactions Across Cut Nodes)**.** *Let u, t, and v be distinct nodes in a connected graph G. If every path from u to v through t passes through a cut node separating t from $\{u, v\}$, then the 3-node interaction (u, t, v) is expressivity-redundant: its contribution to $\boldsymbol{h}(u, v)$ is fully captured by 2-node interactions (u, t) and (t, v), and removing it preserves 2-FWL expressivity.*

Proof. Let c be a cut node separating t from $\{u, v\}$. After removing c, either: (i) u, t, v are in three components: then $\boldsymbol{h}(u, c)$, $\boldsymbol{h}(t, c)$, $\boldsymbol{h}(v, c)$ encode all structural roles; or (ii) u and v remain connected: then u, v, c lie in one biconnected component, and 3-node interactions within it (e.g., (u, c, v)) capture local structure, while (t, c) is updated separately. If t is the cut node, $\boldsymbol{h}(u, t)$ and $\boldsymbol{h}(t, v)$ are computed independently across blocks. In all cases, the interaction (u, t, v) only combines precomputed, separable pairwise information and contributes no new expressive power. □

Lemma 3 (Irrelevance of Interactions Between Disconnected Components)**.** *Let u and v be nodes in different connected components. Then no path exists between them, and their structural independence is implicitly encoded. Explicit modeling of 2-node or 3-node interactions involving both components is unnecessary for substructure detection, provided that component-level readouts capture macro-topological features.*

Proof. Disconnected pairs have no shared paths. The absence of interaction itself encodes disconnection. Component-level readouts aggregate over intra-component pairs, capturing local structure (e.g., cycles), while graph-level aggregation detects global properties (e.g., number of triangles). No inter-component interaction is needed to detect any connected subgraph. □

4.2 Co-Sparsified Neighborhood Construction

To implement our sparsification principle, we define a *co-sparsified neighbor set* $\mathcal{N}_{\text{sp}}(u, v)$ for each node pair (u, v), based on the graph's connectivity structure. We first decompose G into connected components $\mathcal{C}$ and biconnected components (blocks of at least three nodes) $\mathcal{B}$ using Tarjan's algorithm (Hopcroft and Tarjan 1973), which runs in $O(n + m)$ time.

If u and v are in different connected components, we set $\mathcal{N}_{\text{sp}}(u, v) = \emptyset$, as they are structurally independent—this disconnection is captured at the global level through readout. Otherwise, for u, v in the same component $C \in \mathcal{C}$, $\mathcal{N}_{\text{sp}}(u, v)$ includes only interactions that are expressively necessary.

We include 3-node interactions $\big((u, t), (t, v)\big)$ only when u, t, and v are distinct and lie within the same biconnected block $B \in \mathcal{B}$, ensuring aggregation occurs precisely where path multiplicity and cyclic structure must be resolved (Lemma 1). For 2-node interactions, we include $\big((u, u), (u, v)\big)$ and $\big((u, v), (v, v)\big)$ to propagate information from self-pairs (i.e., nodes) to pairs. Additionally, for $u \neq v$,

we include $\big((v,u),(u,v)\big)$ in $\mathcal{N}_{\text{sp}}(v,v)$ to propagate messages from pairs back to nodes. Finally, for self-pairs (u,u), we include $\big((u,u),(u,u)\big)$ to ensure non-empty neighborhoods and stable updates.

This construction preserves all interactions needed for full 2-FWL expressivity, while eliminating only those proven redundant.

4.3 Co-Sparsified Message Passing Scheme

Using $\mathcal{N}_{\text{sp}}(u,v)$, we define the sparse 2-FWL message passing update. Initial representations follow Section 2:

$$\boldsymbol{h}_{\text{sp}}^{(0)}(u,v) = (\boldsymbol{x}(u), \boldsymbol{x}(v), \boldsymbol{e}(u,v), \boldsymbol{p}(u,v)) .$$

At layer l, for (u,v) in the same connected component:

$$\begin{aligned} \boldsymbol{t}_{\text{sp}}^{(l)}(u,v) &= \Big\{\!\!\Big\{ \phi^{(l)} \left(\boldsymbol{h}_{\text{sp}}^{(l-1)}(u,t), \boldsymbol{h}_{\text{sp}}^{(l-1)}(t,v) \right) \\ &\qquad \mid \big((u,t),(t,v)\big) \in \mathcal{N}_{\text{sp}}(u,v) \Big\}\!\!\Big\}, \qquad (5) \\ \boldsymbol{h}_{\text{sp}}^{(l)}(u,v) &= \Phi^{(l)} \left(\boldsymbol{h}_{\text{sp}}^{(l-1)}(u,v),\ \text{AGG} \left(\boldsymbol{t}_{\text{sp}}^{(l)}(u,v) \right) \right) , \end{aligned}$$

where AGG is an injective aggregation (e.g., sum, mean, or MLP-based), and $\Phi^{(l)}$ is a learnable function (e.g., MLP). For disconnected pairs, no update is performed.

4.4 Node-Level and Graph-Level Readouts

For node-level tasks, the representation $\boldsymbol{z}_v$ is computed via readout over incoming pair representations within v's connected component C:

$$\boldsymbol{z}_v = \Psi \left(\text{AGG} \left(\left\{\!\!\left\{ \boldsymbol{h}_{\text{sp}}^{(L)}(u,v) \mid u \in C \right\}\!\!\right\} \right) \right) .$$

For graph-level tasks, we first compute component-level representations. For each connected component C, we apply two parallel readouts over self-pairs and off-diagonal pairs:

$$\begin{aligned} \boldsymbol{z}_C = \Phi \Big(&\text{AGG} \left(\left\{\!\!\left\{ \boldsymbol{h}_{\text{sp}}^{(L)}(u,u) \mid u \in C \right\}\!\!\right\} \right) , \\ &\text{AGG} \left(\left\{\!\!\left\{ \boldsymbol{h}_{\text{sp}}^{(L)}(u,v) \mid u,v \in C, u \neq v \right\}\!\!\right\} \right) \Big) . \end{aligned} \qquad (6)$$

The final graph representation is:

$$\boldsymbol{z}_G = \Psi \left(\text{AGG} \left(\left\{\!\!\left\{ \boldsymbol{z}_C \mid C \in G \right\}\!\!\right\} \right) \right) .$$

If G is connected, $\boldsymbol{z}_G = \boldsymbol{z}_C$ for the single component.

4.5 Expressive Power Study

We prove that **Co-Sparsified GNNs are as powerful as the 2-FWL GNNs** in detecting substructures and distinguishing non-isomorphic graphs. Since graph isomorphism is a special case of substructure detection, this establishes full 2-FWL expressive power.

Theorem 4. *Let Q be a query subgraph. A Co-Sparsified 2-FWL GNN can detect Q if and only if a standard 2-FWL GNN can, under injective aggregation and consistent initialization. Thus, the two models are* equally expressive *in substructure detection.*

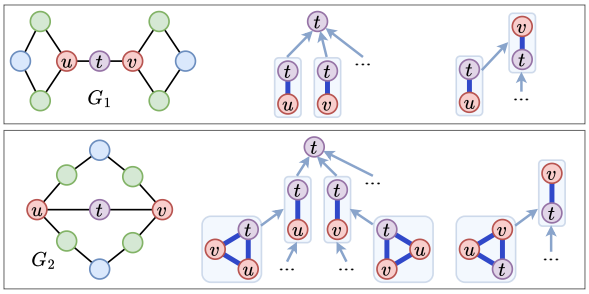

Figure 3: **Expressive power of Co-Sparsified GNNs.** It distinguishes graphs G_1 and G_2—indistinguishable by 1-WL but distinguishable by 2-FWL—by capturing structural differences in node and pair representations, enabling correct subgraph detection.

Proof. The expressive power of 2-FWL stems from its ability to model 2-node and 3-node interactions, enabling it to count paths, cycles, and other symmetric patterns. Co-Sparsify uses the same update rules but restricts 3-node interactions to biconnected components and 2-node interactions to connected components.

We show this sparsification preserves all essential structural information. When a subgraph Q—like a triangle or 4-cycle—lies within a biconnected component, all required 3-node interactions are retained, so Q is detected exactly as in 2-FWL. When Q spans multiple blocks (e.g., a path through a cut node), its structure decomposes into segments (u,t) and (t,v). Since 2-FWL already updates all such pairs, their representations $\boldsymbol{h}(u,t)$ and $\boldsymbol{h}(t,v)$ fully encode the path—aggregating over (u,t,v) adds no new information and can be safely omitted.

For disconnected queries, global topology (e.g., component count, size) is captured via unchanged readout operations. Crucially, the ability to distinguish complex patterns—such as two internally disjoint paths—depends on biconnected structure, which our method fully preserves.

By induction on message-passing layers, and under injective aggregation, Co-Sparsify computes pair representations that induce the same distinctions as 2-FWL. It detects the same subgraphs and separates the same graph classes—proving that only redundant computations are removed. □

As shown in Figure 3, Co-Sparsified 2-FWL GNNs achieve the same expressive power as standard 2-FWL GNNs on G_1 and G_2.

4.6 Computational Efficiency

Standard 2-FWL GNNs update $O(n^2)$ 2-tuples using $O(n^3)$ 3-tuple interactions. Our method reduces this to $O\left(\sum_i n_i^2\right)$ pairs (summing over connected components of size n_i) and $O\left(\sum_j n_{b_j}^3\right)$ triples (summing over biconnected components of size n_{b_j}). In graphs with sparse or fragmented higher-order connectivity (e.g., molecular), this results in significant reductions—from cubic to sub-quadratic or sub-cubic complexity—without expressivity loss.

Dataset	Counting	QM9	ZINC-12k	ZINC-250k	FRANK.	NCI1	NCI109	ENZYMES
#Graphs	5,000	130,831	12,000	249,456	4,337	4,110	4,127	600
Avg. #Nodes	18.8	18.0	23.2	23.1	16.9	29.9	29.7	32.6
Avg. #Edges	31.3	18.7	24.9	24.9	17.9	32.3	32.1	62.1

Table 1: Dataset statistics.

Target	DTNN	MPNN	PPGN	NGNN	KP-GIN′	I^2-GNN	N^2-GNN	PPGN+RRWP	CoSp-PPGN+RRWP
μ	**0.244**	0.358	**0.231**	0.433	0.358	0.428	0.333	0.332	0.321
α	0.95	0.89	0.382	0.265	0.233	0.230	**0.193**	0.200	**0.183**
$\varepsilon_{\text{HOMO}}$	0.00388	0.00541	0.00276	0.00279	0.00240	0.00261	**0.00217**	0.00236	**0.00214**
$\varepsilon_{\text{LUMO}}$	0.00512	0.00623	0.00287	0.00276	0.00236	0.00267	**0.00210**	0.00231	**0.00210**
$\Delta\varepsilon$	0.0112	0.0066	0.00406	0.00390	0.00333	0.00380	**0.00304**	0.00327	**0.00299**
$\langle R^2 \rangle$	17.0	28.5	16.7	20.1	16.51	18.64	**14.47**	16.24	**14.25**
ZPVE	0.00172	0.00216	0.00064	0.00015	0.00017	0.00014	0.00013	**0.00012**	**0.00012**
U_0	2.43	2.05	0.234	0.205	0.0682	0.211	0.0247	**0.0111**	**0.0149**
U	2.43	2.00	0.234	0.200	0.0696	0.206	0.0315	**0.01175**	**0.01373**
H	2.43	2.02	0.229	0.249	0.0641	0.269	0.0182	**0.01266**	**0.01637**
G	2.43	2.02	0.238	0.253	0.0484	0.261	0.0178	**0.01278**	**0.01767**
C_v	0.27	0.42	0.184	0.0811	0.0869	**0.0730**	**0.0760**	0.08589	0.07619

Table 2: MAE results on QM9 (top 2 results are bold).

Target	NGNN	GIN-AK+	I^2-GNN	N^2-GNN	PPGN	**Ours**
Tailed Triangle	0.1044	0.0043	0.0011	0.0025	0.0026	0.0016
Chordal Cycle	0.0392	0.0112	0.0010	0.0019	0.0015	0.0014
4-Path	0.0244	0.0075	0.0041	0.0042	0.0041	0.0029
Tri.-Rec.	0.0729	0.1311	0.0013	0.0055	0.0144	0.0049
3-Cycles	0.0003	0.0004	0.0003	0.0002	0.0003	0.0004
4-Cycles	0.0013	0.0041	0.0016	0.0024	0.0009	0.0007
5-Cycles	0.0402	0.0133	0.0028	0.0039	0.0036	0.0032
6-Cycles	0.0439	0.0238	0.0082	0.0075	0.0071	0.0056

Table 3: Substructure counting results (normalized MAE; lower is better).

5 Experiments

We evaluate Co-Sparsified 2-FWL GNNs on both synthetic and real-world tasks. Our experiments focus on CoSp-PPGN, the sparsified variant of PPGN (Maron et al. 2019). Following the original PPGN formulation, CoSp-PPGN adopts element-wise multiplication for $\phi^{(l)}$ in Equation 5:

$$\begin{aligned} &\phi^{(l)}\left(\boldsymbol{h}_{\text{sp}}^{(l-1)}(u,t), \boldsymbol{h}_{\text{sp}}^{(l-1)}(t,v)\right) \\ &= \varphi^{(l)}\left(\boldsymbol{h}_{\text{sp}}^{(l-1)}(u,t) \otimes \boldsymbol{h}_{\text{sp}}^{(l-1)}(t,v)\right). \end{aligned} \tag{7}$$

5.1 Dataset statistics

We summarize the statistics of all datasets used in our experiments in Table 1.

5.2 Experimental Setup

All experiments are conducted on a platform with an NVIDIA GeForce RTX 4090 24GB GPU, Intel(R) Core(TM) i7-14700K CPU, and 64GB RAM. We implement local connectivity computation using the Block-Cut tree algorithm from SageMath [3]. GNN modeling, training, and evaluation are implemented using PyTorch [4] and PyTorch Geometric [5].

Model	# Param	ZINC-Subset	ZINC-Full
CIN (2021)	∼100k	0.079 ± 0.006	0.022 ± 0.002
GPS (2022)	424k	0.070 ± 0.004	-
Specformer (2023)	∼500k	0.066 ± 0.003	-
ESAN (2022)	446k	0.097 ± 0.006	0.025 ± 0.003
SUN (2022)	526k	0.083 ± 0.003	0.024 ± 0.003
KP-GIN′ (2022)	489k	0.093 ± 0.007	-
I^2-GNN (2023)	-	0.083 ± 0.001	0.023 ± 0.001
SSWL+ (2023a)	387k	0.070 ± 0.005	0.022 ± 0.002
GRIT (2023b)	∼500k	0.059 ± 0.002	0.023 ± 0.001
N^2-GNN (2023)	316k/414k	0.059 ± 0.002	0.022 ± 0.002
Local 2-GNN (2024)	-	0.069 ± 0.001	0.024 ± 0.002
PPGN+RRWP	478K	$\mathbf{0.055 \pm 0.002}$	$\mathbf{0.020 \pm 0.002}$
CoSp-PPGN+RRWP	478K	$\mathbf{0.050 \pm 0.001}$	$\mathbf{0.018 \pm 0.002}$

Table 4: MAE results on ZINC (top 2 results are bold).

Dataset	FRANK.	NCI1	NCI109	ENZYMES
3WL-GNN	58.68 ± 1.93	78.39 ± 1.54	77.97 ± 2.22	54.17 ± 6.25
UnionGNNs	68.02 ± 1.47	82.24 ± 1.24	82.34 ± 1.93	68.17 ± 5.70
CoSp-PPGN	$\mathbf{77.65 \pm 1.35}$	$\mathbf{82.87 \pm 1.87}$	$\mathbf{82.91 \pm 1.22}$	$\mathbf{74.50 \pm 6.45}$

Table 5: Classification accuracy (%) on TUD^2 benchmarks. Results are mean ± standard deviation over 10 runs. Best results are in bold.

[3] https://www.sagemath.org
[4] https://pytorch.org
[5] https://pyg.org

5.3 Substructure Counting

Task and Dataset. We evaluate CoSp-PPGN's substructure counting capability on the synthetic dataset from (Zhao et al. 2022; Huang et al. 2023), consisting of 5,000 randomly generated graphs. Following standard protocols, we use a 0.3/0.2/0.5 train/validation/test split. The task involves node-level regression to count various substructures: tailed triangles, chordal cycles, 4-paths, triangle-rectangles, and cycles of lengths 3–6.

Baselines. We compare against Identity-aware GNN (ID-GNN) (You et al. 2021), Nested GNN (NGNN) (Zhang and Li 2021), GNN-AK+ (Zhao et al. 2022), I^2-GNN (Huang et al. 2023), N^2-GNN (Feng et al. 2023), and PPGN (Maron et al. 2019). Baseline results are taken from (Chen et al. 2025; Feng et al. 2023).

Results. Table 3 presents the normalized test MAE averaged over five runs with different random seeds. CoSp-PPGN achieves state-of-the-art performance, matching or surpassing PPGN across all substructure counting tasks.

5.4 Real-World Benchmarks

Datasets. We evaluate CoSp-PPGN on molecular graph datasets QM9 (Wu et al. 2018; Ramakrishnan et al. 2014) and ZINC (Dwivedi et al. 2023). QM9 contains 130K+ molecules with 12 molecular properties for regression (0.8/0.1/0.1 train/val/test split). ZINC has two variants—ZINC-subset (12K graphs) and ZINC-full (250K graphs)—for graph-level regression. We additionally evaluate on TUD[6] graph classification benchmarks.

Baselines. For QM9, we compare against NGNN and KP-GIN′ (Feng et al. 2022), DTNN and MPNN (Wu et al. 2018), I^2-GNN (Huang et al. 2023), N^2-GNN (Feng et al. 2023), and PPGN+RRWP (Chen et al. 2025). For ZINC, baselines include CIN (Bodnar et al. 2021), GPS (Rampásek et al. 2022), Specformer (Bo et al. 2023), ESAN (Bevilacqua et al. 2022), SUN (Frasca et al. 2022), KP-GIN′ (Feng et al. 2022), I^2-GNN (Huang et al. 2023), SSWL+ (Zhang et al. 2023a), GRIT (Zhang et al. 2023b), N^2-GNN (Feng et al. 2023), and Local 2-GNN (Zhang et al. 2024). TUD baselines are from UnionGNNs (Xu et al. 2024).

Results. Table 2 shows QM9 test MAE results. CoSp-PPGN+RRWP achieves top-two performance on 10 out of 12 targets, empirically validating that our sparsification preserves the full 2-FWL expressivity of PPGN. Table 4 reports average test MAE and standard deviation over 5 runs on ZINC ("-" indicates unreported values). CoSp-PPGN+RRWP achieves state-of-the-art results, slightly outperforming PPGN+RRWP. Table 5 demonstrates state-of-the-art performance on TUD classification benchmarks.

Efficiency Analysis. CoSp-PPGN delivers substantial computational improvements over PPGN. Runtime is reduced by *13–60%*: from 9.3s to 7.9s per epoch on ZINC-subset, 456.9s to 403.6s on ZINC-Full, and 97.1s to 60.7s on QM9. Memory consumption decreases by *12–52%*: from 3.7GB to 3.0GB on ZINC-subset, 17.4GB to 15.6GB on ZINC-Full, and 6.4GB to 4.2GB on QM9. Block-cut tree preprocessing incurs negligible overhead ($\sim$1ms per graph on average), confirming our theoretical $O(n+m)$ complexity.

[6]https://chrsmrrs.github.io/datasets/

6 Limitations, Future Work, and Conclusions

Limitations: While CoSp-PPGN can be applied to benchmarks commonly used for 1-WL-equivalent GNNs and Graph Transformers, we focus on datasets where expressive power is the primary bottleneck—a common practice for HOGNNs. The lack of results on tasks like Peptides-struct (Dwivedi et al. 2022) reflects a broader challenge: *HOGNNs are more prone to over-squashing (Topping et al. 2022; Chen et al. 2022) due to combinatorial message aggregation*, wherein long-range dependencies are distorted during message passing in deep architectures. On Peptides-struct, standard CoSp-PPGN underperforms despite high expressivity. In contrast, a localized variant—restricting updates and aggregation to pairs within shortest-path distance ≤ 4—achieves a state-of-the-art average MAE of 0.245, outperforming Graph Transformers (Rampásek et al. 2022). Peak performance occurs with only *one message-passing layer*, and degrades with depth—indicating that the bottleneck is *generalization*, not expressivity. This limitation is shared across HOGNNs, not unique to our method.

Future Work: The limitations highlights a key open problem: *balancing expressivity and generalization*. To address this, we plan to explore adaptive receptive fields. Moreover, since HOGNNs beyond 2-FWL face prohibitive computational complexity, we aim to extend our sparsification framework to make them more scalable.

Conclusions: We propose *Co-Sparsify*, a connectivity-aware sparsification for 2-FWL GNNs that removes only expressivity-redundant computations. By restricting 3-node interactions to biconnected components and 2-node modeling to connected components, it preserves full 2-FWL expressivity while greatly improving efficiency. We prove its equivalence to 2-FWL and show that CoSp-PPGN achieves state-of-the-art accuracy on key tasks. This study demonstrates that *efficiency in expressive GNNs can arise from structural insight, not approximation*—enabling scalable, theoretically grounded models that preserve expressivity by design.

Acknowledgments

This work was supported by the Science and Technology Development Fund Macau SAR (0003/2023/RIC, 0052/2023/RIA1, 0031/2022/A, 001/2024/SKL for SKL-IOTSC) and Shenzhen-Hong Kong-Macau Science and Technology Program Category C (SGDX20230821095159012). This work was performed in part at SICC which is supported by SKL-IOTSC, University of Macau. Additionally, this work was funded in part by the National Natural Science Foundation of China (U2241210, T2541009) and the Shenzhen Science and Technology Innovation Commission Project under Grant JCYJ20241202124903006.

References

Bevilacqua, B.; Frasca, F.; Lim, D.; Srinivasan, B.; Cai, C.; Balamurugan, G.; Bronstein, M. M.; and Maron, H. 2022. Equivariant Subgraph Aggregation Networks. In *International Conference on Learning Representations (ICLR)*.

Bo, D.; Shi, C.; Wang, L.; and Liao, R. 2023. Specformer: Spectral Graph Neural Networks Meet Transformers. In *International Conference on Learning Representations (ICLR)*.

Bodnar, C.; Frasca, F.; Wang, Y.; Otter, N.; Montúfar, G. F.; Lió, P.; and Bronstein, M. M. 2021. Weisfeiler and Lehman Go Topological: Message Passing Simplicial Networks. In *International Conference on Machine Learning (ICML)*, 1026–1037.

Bouritsas, G.; Frasca, F.; Zafeiriou, S.; and Bronstein, M. M. 2023. Improving Graph Neural Network Expressivity via Subgraph Isomorphism Counting. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 45(1): 657–668.

Chen, R.; Li, Y.; Wu, D.; Zhang, S.; Ip, P. L.; Iam, H. C.; and U, L. H. 2025. Enhanced Subgraph Learning in 2-FWL GNNs via Local Connectivity, Spectral, and Distance Encodings. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*. ACM.

Chen, R.; Zhang, S.; Li, Y.; et al. 2022. Redundancy-Free Message Massing for Graph Neural Networks. *Advances in Neural Information Processing Systems*, 35: 4316–4327.

Dwivedi, V. P.; Joshi, C. K.; Luu, A. T.; Laurent, T.; Bengio, Y.; and Bresson, X. 2023. Benchmarking Graph Neural Networks. *Journal of Machine Learning Research (JMLR)*, 24: 43:1–43:48.

Dwivedi, V. P.; Rampásek, L.; Galkin, M.; Parviz, A.; Wolf, G.; Luu, A. T.; and Beaini, D. 2022. Long Range Graph Benchmark. In *Conference on Neural Information Processing Systems (NeurIPS)*.

Feng, J.; Chen, Y.; Li, F.; Sarkar, A.; and Zhang, M. 2022. How Powerful are K-hop Message Passing Graph Neural Networks. In *Conference on Neural Information Processing Systems (NeurIPS)*.

Feng, J.; Kong, L.; Liu, H.; Tao, D.; Li, F.; Zhang, M.; and Chen, Y. 2023. Extending the Design Space of Graph Neural Networks by Rethinking Folklore Weisfeiler-Lehman. In *Conference on Neural Information Processing Systems (NeurIPS)*.

Frasca, F.; Bevilacqua, B.; Bronstein, M. M.; and Maron, H. 2022. Understanding and Extending Subgraph GNNs by Rethinking Their Symmetries. In *Conference on Neural Information Processing Systems (NeurIPS)*.

Göring, F. 2000. Short proof of Menger's theorem. *Discrete Mathematics*, 219(1-3): 295–296.

Hopcroft, J.; and Tarjan, R. 1973. Algorithm 447: efficient algorithms for graph manipulation. *Communications of the ACM*, 16(6): 372–378.

Huang, Y.; Peng, X.; Ma, J.; and Zhang, M. 2023. Boosting the Cycle Counting Power of Graph Neural Networks with IMATHENVDOLLARi-GNNs. In *International Conference on Learning Representations (ICLR)*.

Hussain, M. S.; Zaki, M. J.; and Subramanian, D. 2024. Triplet Interaction Improves Graph Transformers: Accurate Molecular Graph Learning with Triplet Graph Transformers. In *International Conference on Machine Learning (ICML)*, volume abs/2402.04538.

Kipf, T. N.; and Welling, M. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations (ICLR)*.

Li, J.; Zhang, T.; Tian, H.; Jin, S.; Fardad, M.; and Zafarani, R. 2022. Graph sparsification with graph convolutional networks. *International journal of data science and analytics*, 13(1): 33–46.

Li, Y.; Tarlow, D.; Brockschmidt, M.; and Zemel, R. S. 2016. Gated Graph Sequence Neural Networks. In *International Conference on Learning Representations (ICLR)*.

Maron, H.; Ben-Hamu, H.; Serviansky, H.; and Lipman, Y. 2019. Provably Powerful Graph Networks. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2153–2164.

Morris, C.; Rattan, G.; and Mutzel, P. 2020. Weisfeiler and Leman go sparse: Towards scalable higher-order graph embeddings. In *Conference on Neural Information Processing Systems (NeurIPS)*.

Morris, C.; Ritzert, M.; Fey, M.; Hamilton, W. L.; Lenssen, J. E.; Rattan, G.; and Grohe, M. 2019. Weisfeiler and Leman Go Neural: Higher-Order Graph Neural Networks. In *AAAI Conference on Artificial Intelligence (AAAI)*, 4602–4609.

Peng, H.; Gurevin, D.; Huang, S.; Geng, T.; Jiang, W.; Khan, O.; and Ding, C. 2022. Towards sparsification of graph neural networks. In *2022 IEEE 40th International Conference on Computer Design (ICCD)*, 272–279. IEEE.

Puny, O.; Ben-Hamu, H.; and Lipman, Y. 2020. From graph low-rank global attention to 2-fwl approximation. *arXiv preprint arXiv:2006.07846*.

Ramakrishnan, R.; Dral, P. O.; Rupp, M.; and Von Lilienfeld, O. A. 2014. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific data*, 1(1): 1–7.

Rampásek, L.; Galkin, M.; Dwivedi, V. P.; Luu, A. T.; Wolf, G.; and Beaini, D. 2022. Recipe for a General, Powerful, Scalable Graph Transformer. In *Conference on Neural Information Processing Systems (NeurIPS)*.

Rampášek, L.; and Wolf, G. 2021. Hierarchical graph neural nets can capture long-range interactions. In *2021 IEEE 31st International Workshop on Machine Learning for Signal Processing (MLSP)*, 1–6. IEEE.

Rathee, M.; Zhang, Z.; Funke, T.; Khosla, M.; and Anand, A. 2021. Learnt sparsification for interpretable graph neural networks. *arXiv preprint arXiv:2106.12920*.

Topping, J.; Giovanni, F. D.; Chamberlain, B. P.; Dong, X.; and Bronstein, M. M. 2022. Understanding over-squashing and bottlenecks on graphs via curvature. In *International Conference on Learning Representations (ICLR)*.

Velickovic, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; and Bengio, Y. 2018. Graph Attention Networks. In *International Conference on Learning Representations (ICLR)*.

Wu, F.; Jr., A. H. S.; Zhang, T.; Fifty, C.; Yu, T.; and Weinberger, K. Q. 2019. Simplifying Graph Convolutional Networks. In *International Conference on Machine Learning (ICML)*, 6861–6871.

Wu, Z.; Ramsundar, B.; Feinberg, E. N.; Gomes, J.; Geniesse, C.; Pappu, A. S.; Leswing, K.; and Pande, V. 2018. MoleculeNet: a benchmark for molecular machine learning. *Chemical Science*, 9(2): 513–530.

Xu, J.; Zhang, A.; Bian, Q.; Dwivedi, V. P.; and Ke, Y. 2024. Union subgraph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38-14, 16173–16183.

Xu, K.; Hu, W.; Leskovec, J.; and Jegelka, S. 2019. How Powerful are Graph Neural Networks? In *International Conference on Learning Representations (ICLR)*.

Ying, Z.; You, J.; Morris, C.; Ren, X.; Hamilton, W.; and Leskovec, J. 2018. Hierarchical graph representation learning with differentiable pooling. *Advances in neural information processing systems*, 31.

You, J.; Selman, J. M. G.; Ying, R.; and Leskovec, J. 2021. Identity-aware Graph Neural Networks. In *AAAI Conference on Artificial Intelligence (AAAI)*, 10737–10745.

Zhang, B.; Feng, G.; Du, Y.; He, D.; and Wang, L. 2023a. A Complete Expressiveness Hierarchy for Subgraph GNNs via Subgraph Weisfeiler-Lehman Tests. In *International Conference on Machine Learning (ICML)*, 41019–41077.

Zhang, B.; Gai, J.; Du, Y.; Ye, Q.; He, D.; and Wang, L. 2024. Beyond Weisfeiler-Lehman: A Quantitative Framework for GNN Expressiveness. In *International Conference on Learning Representations (ICLR)*, volume abs/2401.08514.

Zhang, B.; Luo, S.; Wang, L.; and He, D. 2023b. Rethinking the Expressive Power of GNNs via Graph Biconnectivity. In *International Conference on Learning Representations (ICLR)*.

Zhang, M.; and Li, P. 2021. Nested Graph Neural Networks. In *Conference on Neural Information Processing Systems (NeurIPS)*, 15734–15747.

Zhao, L.; Jin, W.; Akoglu, L.; and Shah, N. 2022. From Stars to Subgraphs: Uplifting Any GNN with Local Structure Awareness. In *International Conference on Learning Representations (ICLR)*.

Zhao, L.; Shah, N.; and Akoglu, L. 2022. A Practical, Progressively-Expressive GNN. In *Conference on Neural Information Processing Systems (NeurIPS)*.

Zheng, C.; Zong, B.; Cheng, W.; Song, D.; Ni, J.; Yu, W.; Chen, H.; and Wang, W. 2020. Robust graph representation learning via neural sparsification. In *International conference on machine learning*, 11458–11468. PMLR.