Efficient Subgraph GNNs via Graph Products and Coarsening

Editors: List of editors' names

Abstract

Subgraph Graph Neural Networks (Subgraph GNNs) improve message-passing GNNs by representing graphs as a set of subgraphs, achieving strong performance, but their complexity limits applications to larger graphs. Previous methods use random or learnable sampling of subgraph subsets, but these lead to suboptimal subgraph selections or restricted subset sizes, causing performance drops. This paper presents a new framework to overcome these challenges. We use a graph coarsening function to cluster nodes into super-nodes with induced connectivity. The product of the coarsened and original graph reveals an implicit structure, where subgraphs are tied to specific node sets. By applying generalized message-passing to this graph product, we create an efficient and powerful Subgraph GNN. Unlike previous methods, our approach allows flexible subgraph selection and is compatible with standard training. Additionally, we uncover new permutation symmetries in the resulting node feature tensor, which we leverage by designing linear equivariant layers for our Subgraph GNN architecture. Extensive experiments on several datasets show our method is more flexible than previous approaches, effortlessly handling any number of subgraphs while consistently outperforming baselines.

Keywords: Subgraph GNNs, Equivariance, Symmetries

1. Introduction

Subgraph GNNs Bevilacqua et al. (2022); Frasca et al. (2022); Zhang and Li (2021); Cotta et al. (2021); Papp et al. (2021); Qian et al. (2022); Zhang et al. (2023); Bar-Shalom et al. (2024) enhance MPNN expressiveness by transforming graphs into bags of subgraphs, showing strong results on graph benchmarks. However, they face quadratic time complexity due to message-passing across all subgraphs. Approaches like random sampling Cotta et al. (2021); Bevilacqua et al. (2022); Bar-Shalom et al. (2024) or learning Bevilacqua et al. (2024); Kong et al. (2024); Qian et al. (2022) to select subgraphs and reduce the bag size aim to mitigate this but often lead to suboptimal performance or slow, complex training.

Our approach. We propose a Subgraph GNN that flexibly generates and processes variable-sized subgraph policies, achieving strong results without complex training. Building on Bar-Shalom et al. (2024), we replace the first element in the Cartesian product $G \square G$ with a coarsened

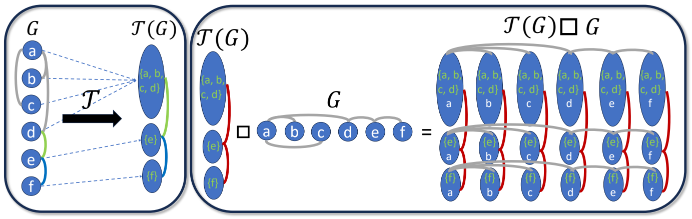


Figure 1: Product graph construction. **Left:** Transforming the graph into a coarse graph; **Right:** Cartesian product of the coarsened and original graph: the vertical axis represents the subgraph dimension (super-nodes), and the horizontal axis represents the node dimension (nodes).

graph $\mathcal{T}(G)$, leading to a smaller product graph $\mathcal{T}(G)\square G$, as illustrated in Figure 1(right). This allows flexible subgraph selection by adjusting the coarsening, shown in Figure 1(left).

Our method uses message passing on $\mathcal{T}(G)\square G$, leveraging unstudied symmetries in the sparse node feature tensor. We incorporate affine equivariant operations and a node-marking technique to improve performance. As shown in Figure 2, Section 4, experiments on Zinc-12k demonstrate our method outperforms baselines in the small bag setting and competes with state-of-the-art Subgraph GNNs in the full bag setting.

Contributions. This paper introduces a flexible Subgraph GNN framework for constructing and processing subgraphs of any size, a characterization of affine invariant/equivariant layers for this new node feature tensors, a theoretical analysis demonstrating the expressivity benefits of the approach, and a comprehensive experimental evaluation showing state-of-the-art results across both small and large bag sizes.

2. Preliminaries

Notation. Let $\mathcal{G}$ be a family of undirected graphs, with $G = (V, E)$ representing a graph in this family. The adjacency matrix $A \in \mathbb{R}^{n\times n}$ defines graph connectivity, and the feature matrix $X \in \mathbb{R}^{n\times d}$ represents node features. The set of nodes is V, edges are E, and $|V| = n$. We use $v_1 \sim_A v_2$ to indicate neighboring nodes in A. Let $[n] = \{1, 2, \dots, n\}$, and $\mathcal{P}([n])$ be its power set.

Subgraph GNNs as graph cartesian products. Bar-Shalom et al. (2024) showed that Subgraph GNNs and specifically the (node-based) maximally expressive variant Zhang et al. (2023), GNN-SSWL+, can be simulated using the Cartesian product of two graphs and applying message passing on the resulting product graph. The Cartesian product of G_1 and G_2, denoted $G_1\square G_2$, creates a graph with vertex set $V(G_1)\times V(G_2)$. The adjacency and node feature matrices of the product graph are $\mathcal{A}$ and $\mathcal{X}$, where $\mathcal{A}_{G_1\square G_2} = A_1 \otimes I + I \otimes A_2$. As we shall see, our framework utilizes a cartesian product of the original graph and a coarsened version of it, as illustrated in Figure 1 (right).

3. Coarsening-Based Subgraph GNN

Overview. This section introduces the *Coarsening-based Subgraph GNN* (CS-GNN) framework. The main idea is to select and process subgraphs in a principled and flexible manner through the following approach: (1) coarsen the original graph via a coarsening function, $\mathcal{T}$ – see Figure 1(left); (2) Obtain the product graph – Figure 1(right) defined by the combination of two adjacencies, $\mathcal{A}_{\mathcal{T}(G)}$ (red edges), $\mathcal{A}_G$ (grey edges), which arise from the graph Cartesian product operation (details follow); (3) leveraging the symmetry of this product graph to develop *symmetry-based* updates, described by $\mathcal{A}_{\text{Equiv}}$ (this part is not visualized in Figure 1). The general update of our suggested layer takes the following form [1],

$$\mathcal{X}^{t+1}(S,v) = f^t\Big(\mathcal{X}(S,v)^t, \tag{1}$$
$$\underbrace{\{\!\!\{\mathcal{X}(S',v')^t\}\!\!\}_{(S',v')\sim_{\mathcal{A}_G}(S,v)}}_{\text{Original connectivity (horizontal)}}, \underbrace{\{\!\!\{\mathcal{X}(S',v')^t\}\!\!\}_{(S',v')\sim_{\mathcal{A}_{\mathcal{T}(G)}}(S,v)}}_{\text{Induced connectivity (vertical)}}, \underbrace{\{\!\!\{\mathcal{X}(S',v')^t\}\!\!\}_{(S',v')\sim_{\mathcal{A}_{\text{Equiv}}}(S,v)}}_{\text{Symmetry-based updates}}\Big),$$

1. Omitting edge features

where the superscript ^{t} indicates the layer index. In what follows, we further elaborate on these three steps. We note that each connectivity in Equation (1) is processed using a different MPNN.

3.1. Product Graph Definition

A maximally expressive node-based Subgraph GNN can be achieved via the Cartesian product of a graph G with itself, $G \square G$ Bar-Shalom et al. (2024). We extend this by using a coarsened version of G, denoted $\mathcal{T}(G)$, as the left operand.

Graph coarsening. Given a graph $G = (V, E)$ with n nodes and adjacency matrix A, the coarsened graph $\mathcal{T}(G)$ has an adjacency matrix $A^{\mathcal{T}} \in \mathbb{R}^{2^n \times 2^n}$, where nodes represent super-nodes (subsets of $[n]$). The connectivity is sparse and induced by $A(v, u) = 1$ if any node in two subsets is connected in G.

In our implementation, we use spectral clustering to partition the graph into T clusters, creating a coarsened graph with fewer nodes and edges. We emphasize that the space complexity of $\mathcal{T}(G)$ is upper-bounded by that of the original graph.

Defining $\mathcal{T}(G)\square G$. The product graph $\mathcal{T}(G)\square G$ is defined by the tensors $\mathcal{A}_{\mathcal{T}(G)\square G} \in \mathbb{R}^{(2^n \times n)\times(2^n \times n)}$ and $\mathcal{X} \in \mathbb{R}^{2^n \times n \times d}$. The connectivity is defined by: $\mathcal{A}_{\mathcal{T}(G)\square G} = \overbrace{A^{\mathcal{T}} \otimes I}^{\triangleq \mathcal{A}_{\mathcal{T}(G)}} + \overbrace{I \otimes A}^{\triangleq \mathcal{A}_G}$. This product graph induces horizontal ($\mathcal{A}_G$) and vertical ($\mathcal{A}_{\mathcal{T}(G)}$) updates, visualized in Figure 1(right) via grey and red edges, respectively.

3.2. Symmetry Based Updates

Using a coarsening function and graph Cartesian product, we derived $\mathcal{A}_G$ and $\mathcal{A}_{\mathcal{T}(G)}$, enabling message-passing on the product graph (see Equation (1)). Building on recent Subgraph GNNs Frasca et al. (2022); Bar-Shalom et al. (2024); Zhang et al. (2023), in this section we construct the *Symmetry-based updates* ($\mathcal{A}_{\text{Equiv}}$) by studying the symmetry structure of the node feature tensor $\mathcal{X}(S, v)$. We present here the final results; detailed discussion and derivations are available in Appendix F. For clarity, we change the notation from nodes (v) to indices (i).

Symmetries of our product graph. Since the node order in G is arbitrary, our architecture must be equivariant to permutations in both G and $\mathcal{T}(G)$. The adjacency and feature matrices, $\mathcal{A} \in \mathbb{R}^{(2^n \times n)\times(2^n \times n)}$ and $\mathcal{X} \in \mathbb{R}^{2^n \times n \times d}$, respect the symmetries of the symmetric group S_n. For any $\sigma \in S_n$, the transformation rules are:

$$(\sigma \cdot \mathcal{A})(S_1, i_1, S_2, i_2) = \mathcal{A}(\sigma^{-1}(S_1), \sigma^{-1}(i_1), \sigma^{-1}(S_2), \sigma^{-1}(i_2)), \tag{2}$$

$$(\sigma \cdot \mathcal{X})(S, i) = \mathcal{X}(\sigma^{-1}(S), \sigma^{-1}(i)). \tag{3}$$

Equivariant Bias and Invariant Layers. The bias vectors are in $\mathbb{R}^{2^n \times n}$, defined by the indicators of the partition induced by the orbits under S_n. For pairs $(S, i) \in \mathcal{P}([n]) \times [n]$, γ^{k^+} includes pairs with $|S| = k$ and $i \notin S$, and γ^{k^-} with $i \in S$, resulting in the partition:

$$(\mathcal{P}([n]) \times [n])/_\sim \triangleq \{\gamma^{k^*} : k = 1, \ldots, n; * \in \{+, -\}\}. \tag{4}$$

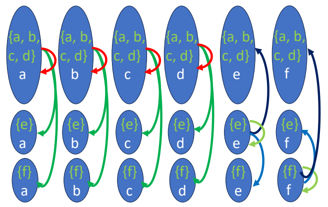
The bias tensor basis is defined as:

$$\mathbf{B}^{\gamma}_{S,i} = \begin{cases} 1, & \text{if } (S,i) \in \gamma; \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

Weight Matrices. Similarly, weight matrix elements (S_1, i_1, S_2, i_2) are partitioned by six conditions, including the sizes of S_1, S_2 and $S_1 \cap S_2$. The weight tensor basis is defined as:

$$\mathbf{B}^{\Gamma}_{S_1,i_1;S_2,i_2} = \begin{cases} 1, & \text{if } (S_1, i_1, S_2, i_2) \in \Gamma; \\ 0, & \text{otherwise,} \end{cases} \tag{6}$$

where Γ is an equivalence class, or orbit. These tensors form an orthogonal basis for invariant and equivariant layers. We revert to the original notation, using v for nodes instead of i.



Using the symmetry-based updates. Any linear equivariant layer can be realized through an MPNN applied to a fully connected graph with appropriate edge features, as stated in Lemma 24. The construction of A_{Equiv} uses these edge features derived from the parameter-sharing scheme. To maintain efficiency (and align with GNN-SSWL+ in our full-bag setting), we use a subset of basis vectors to construct A_{Equiv} (see inset – the parameter-sharing scheme is represented by edges with matching colors). Nodes (S, v) that satisfy $v \in S$ send messages to nodes (S', v') where $v = v'$.

Node Marking. Instead of using the bias term for node marking Papp and Wattenhofer (2022), we propose a more expressive variant: $\mathcal{X}_{S,v} \leftarrow \sum_{u \in S} z_{\text{SPD}(v,u)}$ where $\text{SPD}(v,u)$ is the shortest path distance in G. In Appendix D we justify this choice.

Pooling. After stacking layers, we apply a pooling layer to obtain the graph representation: $\rho(\mathcal{X}^{\mathtt{T}}) = \mathtt{MLP}^{\mathtt{T}}\left(\sum_S \left(\sum_{v=1}^{n} \mathcal{X}^{\mathtt{T}}(S,v)\right)\right)$ with $\mathtt{T}$ as the final layer.

We conclude this section by refering to Appendix A, where a detailed table of contents references various theoretical aspects of our model, thoroughly discussed in the appendix.

4. Experiments

We conduct an extensive set of experiments across six datasets, focusing here on evaluating CS-GNN using the ZINC12k Sterling and Irwin (2015) molecular benchmark under a 500k parameter budget. For full experimental details and results, see Appendix G. We compare CS-GNN against random, learned, and full-bag baselines, and as illustrated in Figure 2, CS-GNN outperforms all efficient baselines (sometimes significantly), achieving state-of-the-art results in the full-bag setting.

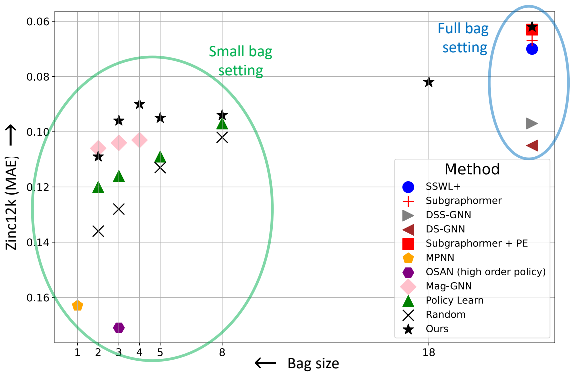


Figure 2: Performance of Subgraph GNNs with varying subgraph counts: Our method excels in smaller bag sizes and matches state-of-the-art performance in the full-bag setting.

References

Guy Bar-Shalom, Beatrice Bevilacqua, and Haggai Maron. Subgraphormer: Unifying subgraph GNNs and graph transformers via graph products. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=6djDWVTUEq.

Beatrice Bevilacqua, Fabrizio Frasca, Derek Lim, Balasubramaniam Srinivasan, Chen Cai, Gopinath Balamurugan, Michael M Bronstein, and Haggai Maron. Equivariant subgraph aggregation networks. *International Conference on Learning Representations*, 2022.

Beatrice Bevilacqua, Moshe Eliasof, Eli Meirom, Bruno Ribeiro, and Haggai Maron. Efficient subgraph gnns by learning effective selection policies. *International Conference on Learning Representations*, 2024.

Lukas Biewald. Experiment tracking with weights and biases, 2020. URL https://www.wandb.com/. Software available from wandb.com.

Leonardo Cotta, Christopher Morris, and Bruno Ribeiro. Reconstruction for powerful graph representations. In *Advances in Neural Information Processing Systems*, volume 34, 2021.

Vijay Prakash Dwivedi, Chaitanya K Joshi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking graph neural networks. *Journal of Machine Learning Research*, 24(43):1–48, 2023.

Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with pytorch geometric. *arXiv preprint arXiv:1903.02428*, 2019.

Fabrizio Frasca, Beatrice Bevilacqua, Michael Bronstein, and Haggai Maron. Understanding and extending subgraph gnns by rethinking their symmetries. *Advances in Neural Information Processing Systems*, 35:31376–31390, 2022.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. PMLR, 2017.

Rafael Gómez-Bombarelli, Jennifer N Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamín Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D Hirzel, Ryan P Adams, and Alán Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 4(2):268–276, 2018.

Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. Strategies for pre-training graph neural networks. *arXiv preprint arXiv:1905.12265*, 2019.

Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems*, 33:22118–22133, 2020.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *International Conference on Learning Representations*, 2016.

Lecheng Kong, Jiarui Feng, Hao Liu, Dacheng Tao, Yixin Chen, and Muhan Zhang. Maggnn: Reinforcement learning boosted graph neural network. *Advances in Neural Information Processing Systems*, 36, 2024.

Jungmin Kwon, Jeongseop Kim, Hyunseo Park, and In Kwon Choi. Asam: Adaptive sharpness-aware minimization for scale-invariant learning of deep neural networks. In *International Conference on Machine Learning*, 2021.

Haggai Maron, Heli Ben-Hamu, Nadav Shamir, and Yaron Lipman. Invariant and equivariant graph networks. *arXiv preprint arXiv:1812.09902*, 2018.

Pál András Papp and Roger Wattenhofer. A theoretical comparison of graph neural network extensions. In *International Conference on Machine Learning*, pages 17323–17345. PMLR, 2022.

Pál András Papp, Karolis Martinkus, Lukas Faber, and Roger Wattenhofer. Dropgnn: Random dropouts increase the expressiveness of graph neural networks. *Advances in Neural Information Processing Systems*, 34:21997–22009, 2021.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Chendi Qian, Gaurav Rattan, Floris Geerts, Mathias Niepert, and Christopher Morris. Ordered subgraph aggregation networks. *Advances in Neural Information Processing Systems*, 35:21030–21045, 2022.

Teague Sterling and John J Irwin. Zinc 15–ligand discovery for everyone. *Journal of chemical information and modeling*, 55(11):2324–2337, 2015.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *International Conference on Learning Representations*, 2018.

Chulhee Yun, Suvrit Sra, and Ali Jadbabaie. Small relu networks are powerful memorizers: a tight analysis of memorization capacity. *Advances in Neural Information Processing Systems*, 32, 2019.

Bohang Zhang, Guhao Feng, Yiheng Du, Di He, and Liwei Wang. A complete expressiveness hierarchy for subgraph gnns via subgraph weisfeiler-lehman tests. *International Conference on Machine Learning*, 2023.

Muhan Zhang and Pan Li. Nested graph neural networks. In *Advances in Neural Information Processing Systems*, volume 34, 2021.

Lingxiao Zhao, Wei Jin, Leman Akoglu, and Neil Shah. From stars to subgraphs: Uplifting any GNN with local structure awareness. In *International Conference on Learning Representations*, 2022.

Appendix A. Appendix: Table of Contents

The appendix is organized as follows:

- In Appendix B, we provide some basic definitions that will be used in later sections in the appendix.
- In Appendix C we discuss some theoretical aspects of our model implementation, and its relation to Equation (1).
- In Appendix D we define four natural general node marking policies and analyze their theoretical effects on our model, as well as their relation to other node-based node marking policies. Finally, we provide a principled derivation of one of these policies using the natural symmetry of our base object.
- In Appendix E.1 we compare our model to node-based subgraph GNNs, which are the most widely used variant of subgraph GNNs. Additionally, we demonstrate that different choices of coarsening functions can recover various existing subgraph GNN designs.
- In Appendix E.2 we demonstrate how our model can leverage the information provided by the coarsening function in an effective way, comparing its expressivity to a natural baseline which also leverages the coarsening function. We show that for all coarsening functions, we are at least as expressive as the baseline and that for some coarsening functions, our model is strictly more expressive.
- In Appendix F we delve deeper into the characterization of all linear maps $L : \mathbb{R}^{\mathcal{P}([n])\times[n]} \to \mathbb{R}^{\mathcal{P}([n])\times[n]}$ that are equivariant to the action of the symmetric group.
- In Appendix G, we provide additional results and detailed explanations of the experimental setup, including instructions on how to reproduce our findings.
- In Appendix H we provide detailed proofs to all propositions in this paper.

Appendix B. Basic Definitions

We devote this section to formally defining the key concepts of this paper, as well as introducing new useful notation. We start by defining the two principle components of our pipeline, the cartesian product graph and the coarsening function:

Definition 1 (Cartesian Product Graph) *Given two graphs G_1 and G_2, their Cartesian product $G_1 \square G_2$ is defined as:*

- *The vertex set $V(G_1 \square G_2) = V(G_1) \times V(G_2)$.*
- *Vertices (u_1, u_2) and (v_1, v_2) in $G_1 \square G_2$ are adjacent if:*
  - *$u_1 = v_1$ and u_2 is adjacent to v_2 in G_2, or*
  - *$u_2 = v_2$ and u_1 is adjacent to v_1 in G_1.*

Definition 2 (Coarsening Function) *A Coarsening function $\mathcal{T}(\cdot)$ is defined as a function that, given a graph $G = (V, E)$ with vertex set $V = [n]$ and adjacency matrix $A \in \mathbb{R}^{n\times n}$,* *takes A as input and returns a set of "super-nodes" $\mathcal{T}(A) \subseteq \mathcal{P}([n])$. The function $\mathcal{T}(\cdot)$ is considered equivariant if, for any permutation $\sigma \in S_n$, the following condition holds:*

$$\mathcal{T}(\sigma \cdot A) = \sigma \cdot \mathcal{T}(A). \tag{7}$$

Here, $\sigma \cdot A$, and $\sigma \cdot \mathcal{T}(A)$ represent the group action of the symmetric group S_n on $\mathbb{R}^{n \times n}$, and $\mathcal{P}([n])$ respectively.

A coarsening function allows us to naturally define a graph structure on the "super-nodes" obtained from a given graph in the following way:

Definition 3 (Coarsened Graph) *Given a coarsening function $\mathcal{T}(\cdot)$ and a graph $G = (V, E)$ with vertex set $V = [n]$, adjacency matrix $A \in \mathbb{R}^{n \times n}$, we abuse notation and define the coarsened graph $\mathcal{T}(G) = (V^{\mathcal{T}}, E^{\mathcal{T}})$ as follows:*

- $V^{\mathcal{T}} = \mathcal{T}(A)$
- $E^{\mathcal{T}} = \{\{S, S'\} \mid S, S' \in \mathcal{T}(A),\ \exists i \in S, i' \in S' \text{ s.t. } A_{i,i'} = 1\}$.

The adjacency matrix of the coarsened graph can be expressed in two ways. The dense representation $A^{\mathcal{T}}_{dense} \in \mathbb{R}^{|V^{\mathcal{T}}| \times |V^{\mathcal{T}}|}$ is defined by:

$$A^{\mathcal{T}}_{dense}(S, S') = \begin{cases} 1 & \{S, S'\} \in E^{\mathcal{T}} \\ 0 & \textit{otherwise.} \end{cases} \tag{8}$$

The sparse representation $A^{\mathcal{T}}_{sparse} \in \mathbb{R}^{\mathcal{P}([n]) \times \mathcal{P}([n])}$ is defined by:

$$A^{\mathcal{T}}_{sparse}(S, S') = \begin{cases} 1 & S, S' \in V^{\mathcal{T}}, \{S, S'\} \in E^{\mathcal{T}} \\ 0 & \textit{otherwise.} \end{cases} \tag{9}$$

We note that if the coarsened graph $\mathcal{T}(G)$ has a corresponding node feature map $\mathcal{X} : V^{\mathcal{T}} \to \mathbb{R}^d$, it also has sparse and dense vector representations defined similarly. Though the dense representation seems more natural, the sparse representation is also useful, as the symmetric group S_n acts on it by:

$$\sigma \cdot A^{\mathcal{T}}_{\text{sparse}}(S, S') = A^{\mathcal{T}}_{\text{sparse}}(\sigma^{-1}(S), \sigma^{-1}(S')). \tag{10}$$

When the type of representation is clear from context, we abuse notation and write $A^{\mathcal{T}}$. Note also that in the above discussion, we have used the term "node feature map". Throughout this paper, in order to denote the node features of a graph $G = (V, E)$ with $|V| = n$, we use both the vector representation $X \in \mathbb{R}^{n \times d}$ and the map representation $\mathcal{X} : V \to \mathbb{R}^d$ interchangeably. Now, recalling that our pipeline is defined to create and update a node feature map $\mathcal{X}(S, v)$ supported on the nodes of the product graph $G \square \mathcal{T}(G)$, we define a general node marking policy, the following way:

Definition 4 (General Node Marking Policy) *A general node marking policy $\pi(\cdot, \cdot)$, is a function which takes as input a graph $G = (V, E)$, and a coarsening function $\mathcal{T}(\cdot)$, and returns a node feature map $\mathcal{X} : V^{\mathcal{T}} \times V \to \mathcal{R}^d$.*

In Appendix D We provide four different node marking policies, and analyze the effect on our pipeline. We now move on to define the general way in which we update a given node feature map on the product graph.

Definition 5 (General CS-GNN Layer Update) *Given a graph $G = (V, E)$ and a coarsening function $\mathcal{T}(\cdot)$, let $\mathcal{X}^t(S, v) : V \times V^{\mathcal{T}} \to \mathcal{R}^d$ denote the node feature map at layer t. The general CS-GNNlayer update is defined by:*

$$\begin{aligned}\mathcal{X}^{t+1}(S,v) = f^t\Bigg(&\mathcal{X}^t(S,v),\\ &agg_1^t\{\!\{(\mathcal{X}^t(S,v'), e_{v,v'}) \mid v' \sim_G v\}\!\},\\ &agg_2^t\{\!\{(\mathcal{X}^t(S',v), \tilde{e}_{S,S'}) \mid S' \sim_{G^{\mathcal{T}}} S\}\!\},\\ &agg_3^t\{\!\{(\mathcal{X}^t(S',v), z(S,v,S',v)) \mid S' \in V^{\mathcal{T}} s.t.\ v \in S'\}\!\},\\ &agg_4^t\{\!\{(\mathcal{X}^t(S,v'), z(S,v,S,v')) \mid v' \in V s.t.\ v' \in S\}\!\}\Bigg).\end{aligned} \tag{11}$$

Here, f^t is an arbitrary (parameterized) continuous function, agg_i^t, $i = 1, \ldots 4$ are learnable permutation invariant aggregation functions, $e_{v,v'}, \tilde{e}_{S,S'}$ are the (optional) edge features of G and $\mathcal{T}(G)$ respectively and the function $z : \mathcal{P}([n]) \times [n] \times \mathcal{P}([n]) \times [n] \to \mathbb{R}^d$ maps each tuple of indices $\mathbf{v} = (S, v, S', v')$ to a vector uniquely encoding the orbit of $\mathbf{v}$ under the action of S_n as described in 68.

We note that for brevity, the notation used in the main body of the paper omits the aggregation functions $\text{agg}_1^t, \ldots, \text{agg}_4^t$ and the edge features from the formulation of some of the layer updates. However, we explicitly state each component of the update, as we heavily utilize them in later proofs. We also note that this update is different than the general layer update presented in Equation (1), as it doesn't use all global updates characterized in 6. The reason for this is that some of the global updates have an asymptotic runtime of $\tilde{\mathcal{O}}(n^2)$where n is the number of nodes in the input graph. As our goal was to create models that improve on the scalability of standard subgraph GNNs which have an asymptotic runtime of $\tilde{\mathcal{O}}(n^2)$, We decided to discard some of the general global updates and keep only the ones that are induced by the last two entries in equation 11 which all have a linear runtime. After a stacking of the layers in Equation (11), we employ the following pooling procedure on the final node feature map $\mathcal{X}^T$:

$$\rho(\mathcal{X}^T) = \texttt{MLP}_2\left(\sum_{S\ \in V^{\mathcal{T}}}\left(\texttt{MLP}_1\big(\sum_{v \in V} \mathcal{X}^T(S,v)\big)\right)\right). \tag{12}$$

Finally, we define the set of all functions that can be expressed by our model:

Definition 6 (Expressivity of Family of Graph Functions) *Let $\mathcal{F}$ be a family of graph functions, we say that $\mathcal{F}$ can express a graph function $g(\cdot)$ if for every finite family of graphs $\mathcal{G}$ there exists a function $f \in \mathcal{F}$ such that:*

$$f(G) = g(G) \quad \forall G \in \mathcal{G}. \tag{13}$$

Here, $\mathcal{G}$ is a finite family of graphs if all possible values of node/edge features of the graphs in $\mathcal{G}$ form a finite set, and the maximal size of the graphs within $\mathcal{G}$ is bounded.

Definition 7 (Family of Functions Expressed By CS-GNN) *Let π be a general node marking policy and $\mathcal{T}$ be a coarsening function. Define $\mathcal{S}(\mathcal{T}, \pi)$ to be the family of graph functions, which when given input graph $G = (V, E)$, first compute $\mathcal{X}^0(S, v)$ using $\pi(G, \mathcal{T})$, then update this node feature map by stacking T layers of the form 11, and finally pooling $\mathcal{X}^0(S, v)$ using equation 12. We define CS-GNN$(\mathcal{T}, \pi)$ to be the set of all functions that can be expressed by $\mathcal{S}(\mathcal{T}, \pi)$.*

Appendix C. Theoretical Validation of Implementation Details

In this section, we provide implementation details of our model and prove that they enable us to recover the conceptual framework of the model discussed thus far. First, we note that in Section 3.2, we characterized all equivariant linear maps $L : \mathbb{R}^{\mathcal{P}([n])\times[n]} \to \mathbb{R}^{\mathcal{P}([n])\times[n]}$ in order to incorporate them into our layer update. Given the high dimensionality of the space of all such linear maps, and in order to save parameters, we demonstrate that it is possible to integrate these layers into our layer update by adding edge features to a standard MPNN model. This is formalized in the following proposition:

Lemma 8 (Parameter Sharing as MPNN) *Let $B_1, \dots B_k : \mathbb{R}^{n\times n}$ be orthogonal matrices with entries restricted to 0 or 1, and let $W_1, \dots W_k \in \mathbb{R}^{d\times d'}$ denote a sequence of weight matrices. Define $B_+ = \sum_{i=1}^{k} B_i$ and choose $z_1, \dots z_k \in \mathbb{R}^{d^*}$ to be a set of unique vectors representing an encoding of the index set. The function that represents an update via parameter sharing:*

$$f(X) = \sum_{i=1}^{k} B_i X W_i, \tag{14}$$

can be implemented on any finite family of graphs $\mathcal{G}$, by a stack of MPNN layers of the following form Gilmer et al. (2017),

$$m_v^l = \sum_{u\in N_{B_+}(v)} M^l(X_u^l, e_{u,v}), \tag{15}$$

$$X_v^{l+1} = U^l(X_v^l, m_v^l), \tag{16}$$

where U^l, M^l are multilayer perceptrons (MLPs). The inputs to this MPNN are the adjacency matrix B_+, node feature vector X, and edge features – the feature of edge (u, v) is given by:

$$e_{u,v} = \sum_{i=1}^{k} z_i \cdot B_i(u, v). \tag{17}$$

Here, $B_i(u, v)$ denotes the (u, v) entry to matrix B_i.

The proof is given in Appendix H. The analysis in Section 3.2 demonstrates that the basis of the space of all equivariant linear maps $L : \mathbb{R}^{\mathcal{P}([n])\times[n]} \to \mathbb{R}^{\mathcal{P}([n])\times[n]}$ satisfies the conditions of Theorem 24. Additionally, we notice that some of the equivariant linear functions have an asymptotic runtime of $\tilde{\mathcal{O}}(n^2)$ where n is the number of nodes in the input graph. As our main goal is to construct a more scalable alternative to node-based subgraph GNNs, which also have a runtime of $\tilde{\mathcal{O}}(n^2)$, we limit ourselves to a subset of the basis for which all maps run in linear time. This is implemented by adding edge features to the adjacency matrices A_{P_1} and A_{P_2}, defined later in this section.

We now move on to discussing our specific implementation of the general layer update from Theorem 5.

Given a graph $G = (V, E)$ and a coarsening function $\mathcal{T}$, we aim to implement this general layer update by combining several standard message passing updates on the product graph $G \square \mathcal{T}(G)$. In the next two definitions, we define the adjacency matrices supported on the node set $V \times V^{\mathcal{T}}$, which serve as the foundation for these message passing procedures, and formalize the procedures themselves.

Definition 9 (Adjacency Matrices on Product Graph) *Let $G = (V, E)$ be a graph with adjacency matrix A and node feature vector X, and let $\mathcal{T}(\cdot)$ be a coarsening function. We define the following four adjacency matrices on the vertex set $V^{\mathcal{T}} \times V$:*

$$A_G(S, v, S', v') = \begin{cases} 1 & v \sim_G v',\ S = S' \\ 0 & \textit{otherwise.} \end{cases} \tag{18}$$

$$A_{\mathcal{T}(G)}(S, v, S', v') = \begin{cases} 1 & S \sim_{\mathcal{T}(G)} S',\ v = v' \\ 0 & \textit{otherwise.} \end{cases} \tag{19}$$

$$A_{P_1}(S, v, S', v') = \begin{cases} 1 & v \in S',\ v = v' \\ 0 & \textit{otherwise.} \end{cases} \tag{20}$$

$$A_{P_2}(S, v, S', v') = \begin{cases} 1 & v' \in S,\ S' = S \\ 0 & \textit{otherwise.} \end{cases} \tag{21}$$

Given edge features $\{e_{v,v'} \mid v \sim_G v'\}$ and $\{\tilde{e}_{S,S'} \mid s \sim_{\mathcal{T}(G)} s'\}$ corresponding to the graphs G and $\mathcal{T}(G)$, respectively, we can trivially define the edge features corresponding to A_G and $A_{G^{\mathcal{T}}}$ as follows:

$$e_G(S, v, S', v') = e_{v,v'}, \tag{22}$$
$$e_{\mathcal{T}(G)}(S, v, S', v') = \tilde{e}_{S,S'}. \tag{23}$$

In addition, for $i = 1, 2$, we define the edge features corresponding to adjacency matrices A_{P_i} as follows:

$$e_{P_i}(S, v, S', v') = z(S, v, S', v'). \tag{24}$$

Here, the function $z : \mathcal{P}([n]) \times [n] \times \mathcal{P}([n]) \times [n] \to \mathbb{R}^d$ maps each tuple $\mathbf{v} = (S, v, S', v')$ to a vector uniquely encoding the orbit of $\mathbf{v}$ under the action of S_n as described in Equation 68.

Definition 10 (CS-GNN Update Implementation) *Given a graph $G = (V, E)$, and a coarsening function $\mathcal{T}(\cdot)$, let $A_1 \ldots A_4$ enumerate the set of adjacency matrices $\{A_G, A_{\mathcal{T}(G)}, A_{P_1}, A_{P_2}\}$. We define a CS-GNN layer update in the following way:*

$$\mathcal{X}_i^t(S, v) = U_i^t \left((1 + \epsilon_i^t) \cdot \mathcal{X}^t(S, v) + \sum_{(S',v') \sim_{A_i} (S,v)} M^t(\mathcal{X}^t(S', v') + e_i(S, v, S', v')) \right). \tag{25}$$

$$\mathcal{X}^{t+1}(S,v) = U^t_{fin}\left(\sum_{i=1}^{4} \mathcal{X}^t_i(S,v)\right). \tag{26}$$

Here $\mathcal{X}^t(S,v)$ and $\mathcal{X}^{t+1}(S,v)$ denote the node feature maps of the product graph at layers t and $t+1$, respectively. $e^1(S,v,S',v'),\dots,e^4(S,v,S',v')$ denote the edge features associated with adjacency matrices $A_1,\dots,A_4$. $\epsilon^t_1,\dots,\epsilon^t_4$ represent learnable parameters in $\mathbb{R}$, and $U^t_1,\dots,U^t_4$, U^t_{fin}, M^t all refer to multilayer perceptrons.

The next proposition states that using the layer update defined in equations 25 and 26 is enough to efficiently recover the general layer update defined in equation 11.

Proposition 11 (Equivalence of General Layer and Implemented Layer) *Let $\mathcal{T}(\cdot)$ be a coarsening function, π be a generalized node marking policy, and $\mathcal{G}$ be a finite family of graphs. Applying a stack of t general layer updates as defined in Equation 11 to the node feature map $\mathcal{X}(S,v)$ induced by $\pi(G,\mathcal{T})$, can be effectively implemented by applying a stack of t layer updates specified in Equations 25 and 26 to $\mathcal{X}(S,v)$. Additionally, the depths of all MLPs that appear in 25 and 26 can be bounded by 4.*

Appendix D. Node Marking Policies – Theoretical Analysis

In this section, we define and analyze various general node marking policies, starting with four natural choices.

Definition 12 (Four General Node Marking policies) *Let $G=(V,E)$ be a graph with adjacency matrix $A\in\mathbb{R}^{n\times n}$ and node feature vector $X\in\mathbb{R}^{n\times d}$, and let $\mathcal{T}(\cdot)$ be a coarsening function. All of the following node marking policies take the form:*

$$\pi(G,\mathcal{T}) = \mathcal{X}(S,v) = [X_u, b_\pi(S,v)], \tag{27}$$

where $[\cdot,\cdot]$ denotes the concatenation operator. We focus on four choices for $b_\pi(S,v)$:

1. ***Simple Node Marking:***

$$b_\pi(S,v) = \begin{cases} 1 & \text{if } v\in S, \\ 0 & \text{if } v\notin S. \end{cases} \tag{28}$$

We denote this node marking policy by π_S.

2. ***Node + Size Marking:***

$$b_\pi(S,v) = \begin{cases} (1,|S|) & \text{if } v\in S, \\ (0,|S|) & \text{if } v\notin S. \end{cases} \tag{29}$$

We denote this node marking policy by π_{SS}.

3. ***Minimum Distance:***

$$b_\pi(S,v) = \min_{v'\in S} d_G(v,v') \tag{30}$$

where $d_G(v,v')$ is the shortest path distance between nodes v and v' in the original graph. We denote this node marking policy by π_{MD}.

4. ***Learned Distance Function:***

$$b_\pi(S,v) = \phi(\{d_G(v,v') \mid v' \in S\}) \tag{31}$$

where $\phi(\cdot)$ is a learned permutation-invariant function. We denote this node marking policy by π_{LD}.

We note that when using the identity coarsening function $\mathcal{T}(G) = G$, our general node marking policies output node feature maps supported on the product $V \times V$. Thus, they can be compared to node marking policies used in node-based subgraph GNNs. In fact, in this case, both π_{S} and π_{SS} reduce to classical node-based node marking, while π_{MD} and π_{LD} reduce to distance encoding. The definitions of these can be found in Zhang et al. (2023). Interestingly, even though in the case of node-based subgraph GNNSs, both distance encoding and node marking were proven to be maximally expressive Zhang et al. (2023), in our case for some choices of $\mathcal{T}$, π_{LD} is strictly more expressive than the other three choices. The exact effect of each generalized node marking policy on the expressivity of our model is explored in the following two propositions.

Proposition 13 (Equal Expressivity of Node Marking Policies) *For any coarsening function $\mathcal{T}(\cdot)$ the following holds:*

$$\textit{CS-GNN}(\mathcal{T},\pi_S) = \textit{CS-GNN}(\mathcal{T},\pi_{SS}) = \textit{CS-GNN}(\mathcal{T},\pi_{MD}). \tag{32}$$

Proposition 14 (Expressivity of Learned Distance Policy) *For any coarsening function $\mathcal{T}(\cdot)$ the following holds:*

$$\textit{CS-GNN}(\mathcal{T},\pi_S) \subseteq \textit{CS-GNN}(\mathcal{T},\pi_{LD}). \tag{33}$$

In addition, for some choices of $\mathcal{T}(\cdot)$ the containment is strict.

The proofs of both propositions can be found in Appendix H. Finally, we provide a principled approach to deriving a generalized node marking policy based on symmetry invariance, and prove its equivalence to π_{SS}. Given a graph $G = (V,E)$ with $V = [n]$, adjacency matrix A, and node feature vector $X \in \mathbb{R}^{n\times d}$, along with a coarsening function $\mathcal{T}(\cdot)$. We define an action of the symmetric group S_n on the space $\mathbb{R}^{\mathcal{P}([n])\times[n]}$ as follows:

$$\sigma \cdot \mathcal{X}(S,v) = \mathcal{X}(\sigma^{-1}(S), \sigma^{-1}(v)) \quad \text{for } \sigma \in S_n, \mathcal{X} \in \mathbb{R}^{\mathcal{P}([n])\times[n]}. \tag{34}$$

Now, for each orbit $\gamma \in (\mathcal{P}([n]) \times [n])/S_n$, we define $\mathbf{1}_\gamma \in \mathbb{R}^{\mathcal{P}([n])\times[n]}$ as follows:

$$\mathbf{1}_\gamma(S,v) = \begin{cases} 1 & (S,v) \in \gamma, \\ 0 & \text{otherwise.} \end{cases} \tag{35}$$

Choosing some enumeration of the orbit set $(\mathcal{P}([n])\times[n])/S_n = \{\gamma_1,\ldots,\gamma_k\}$, We now define the invariant generalized node marking policy π_{inv} by first setting:

$$b^{\mathrm{sparse}}_{\pi_{\mathrm{inv}}}(S,v) : \mathcal{P}([n]) \times [n] \to \mathbb{R}^k$$

and

$$b_{\pi_{\mathrm{inv}}} : V^{\mathcal{T}} \times V \to \mathbb{R}^k$$

as follows:

$$b^{\text{sparse}}_{\pi_{\text{inv}}}(S, v) = [\mathbf{1}_{\gamma_1}(S, v), \dots, \mathbf{1}_{\gamma_k}(S, v)] \qquad S \in \mathcal{P}(V),\ v \in V, \tag{36}$$
$$b_{\pi_{\text{inv}}}(S, v) = b^{\text{sparse}}_{\pi_{\text{inv}}}(S, v) \qquad S \in V^{\mathcal{T}},\ v \in V. \tag{37}$$

Then, we define the node feature map induced by π_{inv} as:

$$\mathcal{X}^{\pi_{\text{inv}}}(S, v) = [X_v, b_{\pi_{\text{inv}}}(S, v)]. \tag{38}$$

Interestingly, π_{inv}, derived solely from the group action of S_n on $\mathcal{P}([n]) \times [n]$, is equivalent to the generalized node marking policy π_{SS}. This is stated more rigorously in the following proposition:

Proposition 15 (Node + Size Marking as Invariant Marking) *Given a graph $G = (V, E)$ with node feature vector $X \in \mathbb{R}^{n\times d}$, and a coarsening function $\mathcal{T}(\cdot)$, let $\mathcal{X}^{\pi_{SS}}, \mathcal{X}^{\pi_{inv}}$ be the node feature maps induced by π_{SS} and π_{inv} respectively. Recall that:*

$$\mathcal{X}^{\pi_{SS}}(S, v) = [X_v, b_{\pi_{SS}}(S, v)], \tag{39}$$
$$\mathcal{X}^{\pi_{inv}}(S, v) = [X_v, b_{\pi_{inv}}(S, v)]. \tag{40}$$

The following now holds:

$$b_{\pi_{inv}}(S, v) = OHE(b_{\pi_{SS}}(S, v)) \quad \forall S \in V^{\mathcal{T}},\ \forall v \in V. \tag{41}$$

Here, OHE denotes a one-hot encoder, independent of the choice of both G and $\mathcal{T}$.

The proof of proposition 15 can be found in Appendix H.

Appendix E. Expressive Power of CS-GNN

E.1. Recovering Subgraph GNNs

In this section, we demonstrate that by choosing suitable coarsening functions, our architecture can replicate various previous subgraph GNN designs. We begin by focusing on node-based models, which are the most widely used type. We define a variant of these models which was proven in Zhang et al. (2023) to be maximally expressive, and show that our approach can recover it.

Definition 16 (Maximally Expressive Subgraph GNN) *We define MSGNN(π_{NM}) as the set of all functions expressible by the following procedure:*

1. ***Node Marking:*** *The representation of tuple $(u, v) \in V \times V$ is initially given by:*

$$\mathcal{X}^0(u, v) = \begin{cases} 1 & \text{if } u = v, \\ 0 & \text{if } u \neq v. \end{cases} \tag{42}$$

2. ***Update:*** *The representation of tuple* (u, v) *is updated according to:*

$$\begin{aligned}\mathcal{X}^{t+1}(u,v) = f^t\Big(&\mathcal{X}^t(u,v), \mathcal{X}^t(u,u), \mathcal{X}^t(v,v), \\ &agg_1^t\{\!\{(\mathcal{X}^t(u,v'), e_{v,v'}) \mid v' \sim v\}\!\}, \\ &agg_2^t\{\!\{(\mathcal{X}^t(v,u'), e_{u,u'}) \mid u' \sim u\}\!\}\Big).\end{aligned} \tag{43}$$

3. ***Pooling:*** *The final node feature vector* $\mathcal{X}^T(u, v)$ *is pooled according to:*

$$MLP_2\left(\sum_{u\in V} MLP_1\left(\sum_{v\in V} \mathcal{X}^T(u,v)\right)\right). \tag{44}$$

Here, for any $t \in [T]$, f^t *is any continuous (parameterized) functions,* $agg_1^t, , agg_2^t$ *are any continuous (parameterized) permutation-invariant functions and* MLP_1, MLP_2 *are multi-layer preceptrons.*

Proposition 17 (CS-GNN Can Implement MSGNN) *Let* $\mathcal{T}(\cdot)$ *be the identity coarsening function defined by:*

$$\mathcal{T}(G) = \{\{v\} \mid v \in V\} \quad \forall G = (V, E). \tag{45}$$

The following holds:

$$CS\text{-}GNN(\mathcal{T}, \pi_S) = MSGNN(\pi_{NM}). \tag{46}$$

The proof of proposition 17 can be found in Appendix H. We observe that, similarly, by selecting the coarsening function:

$$\mathcal{T}(G) = E \quad \forall G = (V, E), \tag{47}$$

one can recover edge-based subgraph GNNs. An example of such a model is presented in Bevilacqua et al. (2022) (DS-GNN), where it was proven capable of distinguishing between two 3-WL indistinguishable graphs, despite having an asymptotic runtime of $\tilde{\mathcal{O}}(m^2)$, where m is the number of edges in the input graph. This demonstrates our model's ability to achieve expressivity improvements while maintaining a (relatively) low asymptotic runtime by exploiting the graph's sparsity through the coarsening function. Finally, we note that by selecting the coarsening function:

$$\mathcal{T}(G) = \{S \in \mathcal{P}(V) \mid |S| = k\} \quad G = (V, E), \tag{48}$$

We can recover an unordered variant of the k-OSAN model presented in Qian et al. (2022).

E.2. Comparison to Theoretical Baseline

In this section, we demonstrate how our model can leverage the information provided by the coarsening function $\mathcal{T}(\cdot)$ in an effective way. First, we define a baseline model that incorporates $\mathcal{T}$ in a straightforward manner. We then prove that, for any $\mathcal{T}(\cdot)$, our model is at least as expressive as this baseline. Additionally, we show that for certain choices of $\mathcal{T}(\cdot)$, our model exhibits strictly greater expressivity. To construct the baseline model, we first provide the following definition:

Definition 18 (Coarsened Sum Graph) *Given a graph $G = (V, E)$ and a coarsening function $\mathcal{T}(\cdot)$, we define the coarsened sum graph $G_+^{\mathcal{T}} = (V_+^{\mathcal{T}}, E_+^{\mathcal{T}})$ by:*

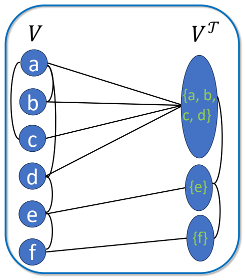
- $V_+^{\mathcal{T}} = V \cup V^{\mathcal{T}}$.
- $E_+^{\mathcal{T}} = E \cup E^{\mathcal{T}} \cup \{\{S, v\} \mid S \in V^{\mathcal{T}},\ v \in V\ v \in S\}$.

If graph G had a node feature vector $X \in \mathbb{R}^{n\times d}$, we define the node feature vector of $G_+^{\mathcal{T}}$ as:

$$X_v = \begin{cases} [X_v, 1] & v \in V \\ 0_{d+1} & v \in V^{\mathcal{T}} \end{cases}. \tag{49}$$

Here we concatenated a 1 to the end of node features of V to distinguish them from the nodes of $V^{\mathcal{T}}$.

The connectivity of the sum graph (for our running example Figure 1) is visualized inset.



We now define our baseline model:

Definition 19 (Coarse MPNN) *Let $\mathcal{T}(\cdot)$ be a coarsening function. Define MPNN$_+(\mathcal{T})$ as the set of all functions which can be expressed by the following procedure:*

1. ***Preprocessing:*** *We first construct the sum graph $G_+^{\mathcal{T}}$ of the input graph G, along with a node feature map $\mathcal{X}^0 : V_+^{\mathcal{T}} \to \mathbb{R}^d$ defined according to equation 49.*
2. ***Update:*** *The representation of node $v \in V_+^{\mathcal{T}}$ is updated according to:*

$$\begin{aligned} \text{For } v \in V: \quad & \mathcal{X}^{t+1}(v) = f_V^t\left(\mathcal{X}^t(v), agg_1^t\{\!\{(\mathcal{X}^t(u), e_{u,v}) \mid u \sim_G v\}\!\},\right. \\ & \left. agg_2^t\{\!\{\mathcal{X}^t(S) \mid S \in V^T, v \in S\}\!\}\right), \\ \text{For } S \in V^{\mathcal{T}}: \quad & \mathcal{X}^{t+1}(S) = f_{V^{\mathcal{T}}}^t\left(\mathcal{X}^t(S), agg_1^t\{\!\{(\mathcal{X}^t(S'), e_{S,S'}) \mid S' \sim_{\mathcal{T}(G)} S\}\!\},\right. \\ & agg_2^t\{\!\{\mathcal{X}^t(v) \mid v \in V, v \in S\}\}\!\}). \end{aligned} \tag{50}$$

3. ***Pooling:*** *The final node feature vector $\mathcal{X}^T(\cdot)$ is pooled according to:*

$$MLP\left(\sum_{v\in V_+^{\mathcal{T}}} \mathcal{X}^T(v)\right). \tag{51}$$

Here, for $t \in [T]$, f_V^t,, $f_{V^{\mathcal{T}}}^t$ are continuous (parameterized) functions and , $agg_1^t, agg_2^t T$ are continuous (parameterized) permutation invariant functions. Finally, we notice that for the trivial coarsening function defined by

$$\mathcal{T}_\emptyset(G) = \emptyset, \tag{52}$$

the update in Equation (50) devolves into a standard MPNN update, as defined in Gilmer et al. (2017) and so we define:

$$MPNN = MPNN_{+}(\mathcal{T}_{\emptyset}). \tag{53}$$

In essence, given an input graph $G = (V, E)$, the MPNN$_+(\mathcal{T})$ pipeline first constructs the coarsened graph $\mathcal{T}(G)$. It then adds edges between each super-node $S \in V^{\mathcal{T}}$ and the nodes it is comprised of (i.e., any $v \in S$). This is followed by a standard message passing procedure on the graph. The following two propositions suggest that this simple approach to incorporating $\mathcal{T}$ into a GNN pipeline is less powerful than our model.

Proposition 20 (CS-GNN Is at Least as Expressive as Coarse MPNN) *For any coarsening function $\mathcal{T}(\cdot)$ the following holds:*

$$MPNN \subseteq MPNN_{+}(\mathcal{T}) \subseteq CS\text{-}GNN(\mathcal{T}, \pi_S) \tag{54}$$

Proposition 21 (CS-GNN Can Be More Expressive Than MPNN+) *Let $\mathcal{T}(\cdot)$ be the identity coarsening function defined by:*

$$\mathcal{T}(G) = \{\{v\} \mid v \in V\} \quad G = (V, E). \tag{55}$$

The following holds:

$$MPNN = MPNN_{+}(\mathcal{T}). \tag{56}$$

Thus:

$$MPNN_{+}(\mathcal{T}) \subset CS\text{-}GNN(\mathcal{T}, \pi_S), \tag{57}$$

where this containment is strict.

The proofs to the last two propositions can be found in Appendix H. Theorem 21 demonstrates that CS-GNNis strictly more expressive than MPNN$_+$ when using the identity coarsening function. However, this result extends to more complex coarsening functions as well. We briefly discuss one such example. Let $\mathcal{T}(\cdot)$ be the coarsening function defined by:

$$\mathcal{T}_{\triangle}(G) = \{v_1, v_2, v_3 \mid G[v_1, v_2, v_3] \cong \triangle\}, \tag{58}$$

i.e. for an input graph G, the set of super-nodes is composed of all triplets of nodes whose induced subgraph is isomorphic to a triangle. To see that CS-GNN is strictly more expressive then MPNN$_+$ when using $\mathcal{T}_{\triangle}(\cdot)$, we look at the two graphs G and H depicted in Figure 3. In the figure, we see the two original graphs, G and H, their corresponding sum graphs $G_{+}^{\mathcal{T}_{\triangle}}$ and $H_{+}^{\mathcal{T}_{\triangle}}$, and a subgraph of their corresponging product graphs $G \square \mathcal{T}_{\triangle}(G)$ and $H \square \mathcal{T}_{\triangle}(H)$ induced by the sets $\{(S_0, v) \mid v \in V_G\}$ and $\{(S_0, v) \mid v \in V_H\}$ respectively (this can be thought of as looking at a single subgraph from the bag of subgraphs induced by CS-GNN). One can clearly see that both the original graphs and their respective sum graphs are 1-WL indistinguishable. On the other hand, the subgraphs induced by our method are 1-WL distinguishable. Since for both G and H the "bag of subgraphs" induced by CS-GNN is composed of 6 isomorphic copies of the same graph, this would imply that our method can distinguish between G and H, making it strictly mor expressive then MPNN$_+$.

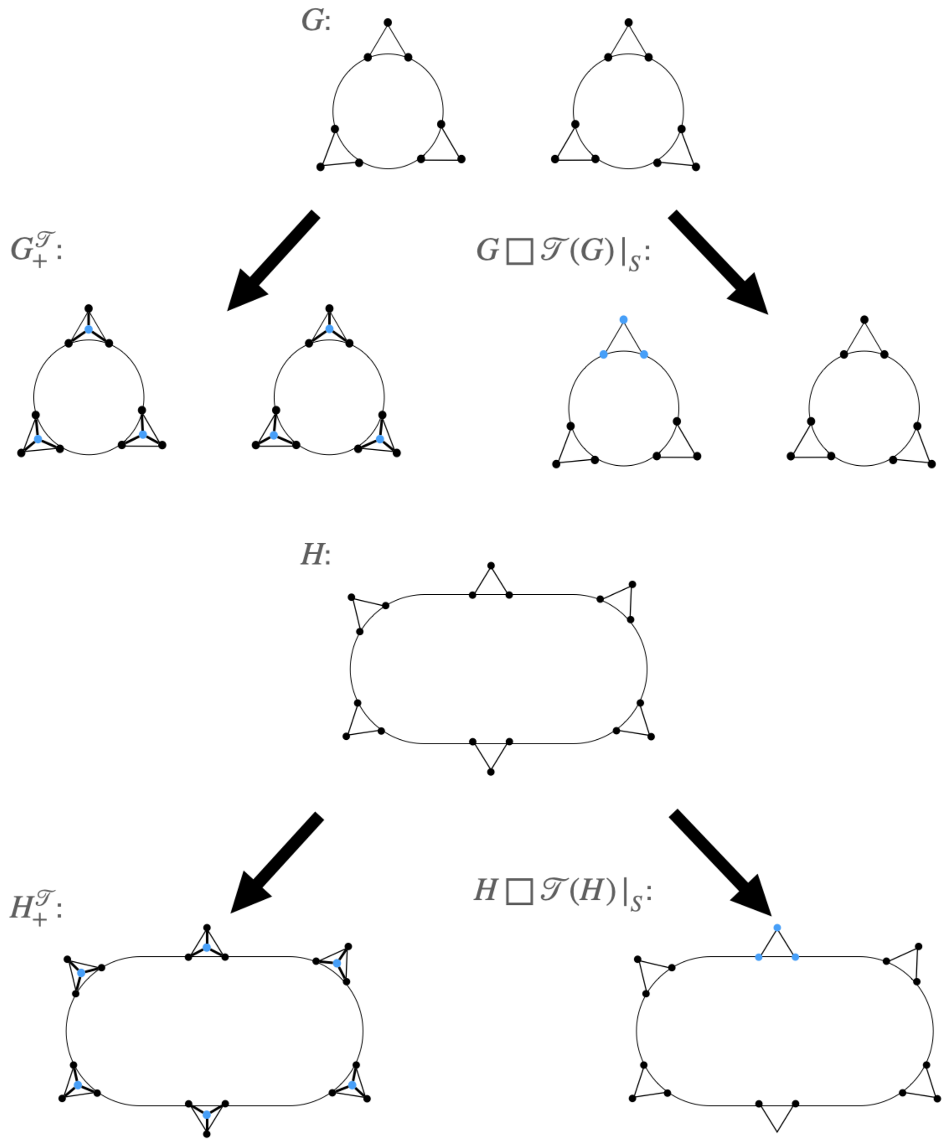

G:
$G_{+}^{\mathscr{T}}$:
$G \square \mathscr{T}(G)|_S$:
H:
$H_{+}^{\mathscr{T}}$:
$H \square \mathscr{T}(H)|_S$:

Figure 3: Rows 1 and 3 depict two 1-WL indistinguishable graphs¿ Rows 2 and 4 depict the sum graph of each of these graphs, as well as one subgraph of their product graphs induced by all node, super-node tuples whose super-node is fixed.

Appendix F. Linear Invariant (Equivariant) Layer – Extended Section

We introduce some key notation. In the matrix $\mathcal{X}$, the i-th row corresponds to the i-th subset S arranged in the lexicographic order of all subsets of $[n]$, namely, $[\{0\}, \{0,1\}, \{0,2\}, \ldots, \{0,1,2,\ldots,n\}]$. Each i-th position in this sequence aligns with the i-th row index in $\mathcal{X}$. It follows, that the standard basis for such matrices in $\mathbb{R}^{2^n \times n}$ is expressed as $\mathbf{e}^{(S)} \cdot \mathbf{e}^{(i)^T}$, where $\mathbf{e}^{(S)}$ is a 1-hot vector, with the value 1 positioned according to S in the lexicographic order. For a matrix $X \in \mathbb{R}^{a \times b}$, the operation of vectorization, denoted by $\text{vec}(X)$, transforms X into a single column vector in $\mathbb{R}^{ab\times 1}$ by sequentially stacking its columns; in the context of $\mathcal{X}$, the basis vectors of those vectors are $\mathbf{e}^{(i)} \otimes \mathbf{e}^{(S)}$. The inverse process, reshaping a vectorized matrix back to its original format, is denoted as $[\text{vec}(X)] = X$. We also denote an arbitrary permutation by $\sigma \in S_n$. The actions of permutations on vectors, whether indexed by sets or individual indices, are represented by $\mathbf{P}_{\mathcal{S}} \in \text{GL}(2^n)$ and $\mathbf{P}_{\mathcal{I}} \in \text{GL}(n)$, respectively. This framework acknowledges S_n as a subgroup of the larger permutation group S_{2^n}, which permutes all 2^n positions in a given vector $\mathbf{v}_S \in \mathbb{R}^{2^n}$.

Let $\mathbf{L} \in \mathbb{R}^{1\times 2^n\cdot n}$ be the matrix representation of a general linear operator $\mathcal{L} : \mathbb{R}^{2^n\times n} \to \mathbb{R}$ in the standard basis. The operator $\mathcal{L}$ is order-invariant iff

$$\mathbf{L}\,\text{vec}(\mathbf{P}_{\mathcal{S}}^T \mathcal{X} \mathbf{P}_{\mathcal{I}}) = \mathbf{L}\,\text{vec}(\mathcal{X}). \tag{59}$$

Similarly, let $\mathbf{L} \in \mathbb{R}^{2^n\cdot n\times 2^n\cdot n}$ denote the matrix for $\mathcal{L} : \mathbb{R}^{2^n\times n} \to \mathbb{R}^{2^n\times n}$. The operator $\mathcal{L}$ is order-equivariant if and only if

$$[\mathbf{L}\,\text{vec}(\mathbf{P}_{\mathcal{S}}^T \mathcal{X} \mathbf{P}_{\mathcal{I}})] = \mathbf{P}_{\mathcal{S}}^T [\mathbf{L}\,\text{vec}(\mathcal{X})] \mathbf{P}_{\mathcal{I}}. \tag{60}$$

Using properties of the Kronecker product (see Appendices F.1 and F.2 for details), we derive the following conditions for invariant and equivariant linear layers:

$$\text{Invariant } \mathbf{L}: \quad \mathbf{P}_{\mathcal{I}} \otimes \mathbf{P}_{\mathcal{S}}\,\text{vec}(\mathbf{L}) = \text{vec}(\mathbf{L}), \tag{61}$$

$$\text{Equivariant } \mathbf{L}: \quad \mathbf{P}_{\mathcal{I}} \otimes \mathbf{P}_{\mathcal{S}} \otimes \mathbf{P}_{\mathcal{I}} \otimes \mathbf{P}_{\mathcal{S}}\,\text{vec}(\mathbf{L}) = \text{vec}(\mathbf{L}). \tag{62}$$

Solving Equations (61) and (62). Let $\sigma \in S_n$ denote a permutation corresponding to the permutation matrix $\mathbf{P}$. Let $\mathbf{P} \star \mathbf{L}$ denote the tensor that results from expressing $\mathbf{L}$ after renumbering the nodes in $V^{\mathcal{T}}, V$ according to the permutation σ. Explicitly, for $\mathbf{L} \in \mathbb{R}^{2^n\times n}$, the $(\sigma(S), \sigma(i))$-entry of $\mathbf{P} \star \mathbf{L}$ equals to the (S, i)-entry of $\mathbf{L}$. The matrix that corresponds to the operator $\mathbf{P}\star$ in the standard basis, $\mathbf{e}^{(i)} \otimes \mathbf{e}^{(S)}$ is the kronecker product $\mathbf{P}_{\mathcal{I}} \otimes \mathbf{P}_{\mathcal{S}}$. Since $\text{vec}(\mathbf{L})$ is exactly the coordinate vector of the tensor $\mathbf{L}$ in the standard basis we have,

$$\text{vec}(\mathbf{P} \star \mathbf{L}) = \mathbf{P}_{\mathcal{I}} \otimes \mathbf{P}_{\mathcal{S}}\,\text{vec}(\mathbf{L}), \tag{63}$$

following the same logic, the following holds for the equivariant case, where $\mathbf{L} \in \mathbb{R}^{2^n\cdot n\times 2^n\cdot n}$,

$$\text{vec}(\mathbf{P} \star \mathbf{L}) = \mathbf{P}_{\mathcal{I}} \otimes \mathbf{P}_{\mathcal{S}} \otimes \mathbf{P}_{\mathcal{I}} \otimes \mathbf{P}_{\mathcal{S}}\,\text{vec}(\mathbf{L}). \tag{64}$$

Given Equations (61) and (63) and Equations (62) and (64), it holds that we should focus on solving,

$$\mathbf{P} \star \mathbf{L} = \mathbf{L}, \quad \forall \mathbf{P} \text{ permutation matrices}, \tag{65}$$

for both cases where $\mathbf{L} \in \mathbb{R}^{2^n\times n}$ and $\mathbf{L} \in \mathbb{R}^{2^n\times n\times 2^n\times n}$, corresponding to the bias term, and linear term.

Bias. To this end, let us define an equivalence relation in the index space of a tensor in $\mathbb{R}^{2^n\times n}$. Given a pair $(S, i) \in \mathcal{P}([n]) \times [n]$, we define γ^{k^+} to correspond to all pairs (S, i) such that $|S| = k$ and $i \notin S$. Similarly, γ^{k^-} corresponds to all pairs (S, i) such that $|S| = k$ and $i \in S$. We denote this equivalence relation as follows:

$$(\mathcal{P}([n]) \times [n])/_\sim \triangleq \{\gamma^{k^*} : k = 1, \ldots, n; * \in \{+, -\}\}. \tag{66}$$

For each set-equivalence class $\gamma \in (\mathcal{P}([n]) \times [n])_\sim$, we define a basis tensor, $\mathbf{B}^\gamma \in \mathbb{R}^{2^n \times n}$ by setting:

$$\mathbf{B}^\gamma_{S,i} = \begin{cases} 1, & \text{if } (S, i) \in \gamma; \\ 0, & \text{otherwise.} \end{cases} \tag{67}$$

Following similar reasoning, consider elements $(S_1, i_1, S_2, i_2) \in (\mathcal{P}([n]) \times [n] \times \mathcal{P}([n]) \times [n])$. We define a partition according to six conditions: the relationship between i_1 and i_2, denoted as $i_1 \leftrightarrow i_2$, which is determines by the condition: $i_1 = i_2$ or $i_1 \neq i_2$; the cardinalities of S_1 and S_2, denoted as k_1 and k_2, respectively; the size of the intersection $S_1 \cap S_2$, denoted as $k^\cap$; the membership of i_l in S_l for $l \in \{1, 2\}$, denoted as $\delta_{\text{same}} \in \{1, 2, 3, 4\}$; and the membership of i_{l_1} in S_{l_2} for distinct $l_1, l_2 \in \{1, 2\}$, denoted as $\delta_{\text{diff}} \in \{1, 2, 3, 4\}$. The equivalence relation thus defined can be represented as:

$$(\mathcal{P}([n]) \times [n] \times \mathcal{P}([n]) \times [n])/_\sim \triangleq \{\Gamma^{\leftrightarrow; k_1; k_2; k^\cap; \delta_{\text{same}}; \delta_{\text{diff}}}\}. \tag{68}$$

For each set-equivalence class $\Gamma \in (\mathcal{P}([n]) \times [n] \times \mathcal{P}([n]) \times [n])/_\sim$, we define a basis tensor, $\mathbf{B}^\Gamma \in \mathbb{R}^{2^n \times n \times 2^n \times n}$ by setting:

$$\mathbf{B}^\Gamma_{S_1, i_1; S_2, i_2} = \begin{cases} 1, & \text{if } (S_1, i_1, S_2, i_2) \in \Gamma; \\ 0, & \text{otherwise.} \end{cases} \tag{69}$$

The following two proposition summarizes the results in this section,

Lemma 22 (γ (Γ) are orbits) *The sets $\{\gamma^{k^*} : k = 1, \ldots, n; * \in \{+, -\}\}$ and $\{\Gamma^{\leftrightarrow; k_1; k_2; k^\cap; \delta_{same}; \delta_{diff}}\}$ are the orbits of S_n on the index space $(\mathcal{P}([n]) \times [n])$ and $(\mathcal{P}([n]) \times [n] \times (\mathcal{P}([n]) \times [n])$, respectively.*

Proposition 23 (Basis of Invariant (Equivariant) Layer) *The tensors $\mathbf{B}^\gamma$ ($\mathbf{B}^\Gamma$) in Equation (67) (Equation (69)) form an orthogonal basis (in the standard inner product) to the solution of Equation (61) (Equation (62)).*

The proofs are given in Appendix H.

F.1. Full Derivation of Equation (61).

Our goal is to transition from the equation,

$$\mathbf{L}\,\mathrm{vec}(\mathbf{P}_\mathcal{S}^T \mathcal{X} \mathbf{P}_\mathcal{I}) = \mathbf{L}\,\mathrm{vec}(\mathcal{X}) \tag{59}$$

to the form,

$$\mathbf{P}_\mathcal{I} \otimes \mathbf{P}_\mathcal{S}\,\mathrm{vec}(\mathbf{L}) = \mathrm{vec}(\mathbf{L}) \tag{61}$$

We introduce the following property of the Kronecker product,

$$\mathrm{vec}(\mathbf{ABC}) = (\mathbf{C}^T \otimes \mathbf{A})\mathrm{vec}(\mathbf{B}). \tag{70}$$

Using Equation (70) on the left side of Equation (59), we obtain

$$\mathbf{L}\mathbf{P}_{\mathcal{I}}^T \otimes \mathbf{P}_{\mathcal{S}}^T \,\mathrm{vec}(\mathcal{X}) = \mathbf{L}\,\mathrm{vec}(\mathcal{X}), \tag{71}$$

since this should be true for any $\mathcal{X} \in \mathbb{R}^{2^n \times n}$, we derive

$$\mathbf{L}\mathbf{P}_{\mathcal{I}}^T \otimes \mathbf{P}_{\mathcal{S}}^T = \mathbf{L}. \tag{72}$$

Applying the transpose operation on both sides, and noting that $(\mathbf{P}_{\mathcal{I}}^T \otimes \mathbf{P}_{\mathcal{S}}^T)^T = \mathbf{P}_{\mathcal{I}} \otimes \mathbf{P}_{\mathcal{S}}$, we obtain

$$\mathbf{P}_{\mathcal{I}} \otimes \mathbf{P}_{\mathcal{S}}\mathbf{L}^T = \mathbf{L}^T. \tag{73}$$

Recalling that $\mathbf{L} \in \mathbb{R}^{1\times 2^n \cdot n}$, and thus $\mathbf{L}^T \in \mathbb{R}^{2^n \cdot n \times 1}$, we find that $\mathbf{L}^T = \mathrm{vec}(\mathbf{L})$. Substituting this back into the previous equation we achieve Equation (61).

F.2. Full Derivation of Equation (62).

Our goal is to transition from the equation,

$$[\mathbf{L}\,\mathrm{vec}(\mathbf{P}_{\mathcal{S}}^T \mathcal{X} \mathbf{P}_{\mathcal{I}})] = \mathbf{P}_{\mathcal{S}}^T [\mathbf{L}\,\mathrm{vec}(\mathcal{X})]\mathbf{P}_{\mathcal{I}} \tag{60}$$

to the form,

$$\mathbf{P}_{\mathcal{I}} \otimes \mathbf{P}_{\mathcal{S}} \otimes \mathbf{P}_{\mathcal{I}} \otimes \mathbf{P}_{\mathcal{S}}\,\mathrm{vec}(\mathbf{L}) = \mathrm{vec}(\mathbf{L}). \tag{62}$$

Applying the property in Equation (70), after the reverse operation of the vectorization, namely,

$$[\mathrm{vec}(\mathbf{ABC})] = [(\mathbf{C}^T \otimes \mathbf{A})\mathrm{vec}(\mathbf{B})] \tag{74}$$

on the right hand side of Equation (60), for

$$\mathbf{A} \triangleq \mathbf{P}_{\mathcal{S}}^T; \tag{75}$$
$$\mathbf{B} \triangleq [\mathbf{L}\,\mathrm{vec}(\mathcal{X})]; \tag{76}$$
$$\mathbf{C} \triangleq \mathbf{P}_{\mathcal{I}}, \tag{77}$$

we obtain,

$$[\mathbf{L}\,\mathrm{vec}(\mathbf{P}_{\mathcal{S}}^T \mathcal{X} \mathbf{P}_{\mathcal{I}})] = [\mathbf{P}_{\mathcal{I}}^T \otimes \mathbf{P}_{\mathcal{S}}^T \mathbf{L}\,\mathrm{vec}(\mathcal{X})]. \tag{78}$$

Thus, by omitting the revere-vectorization operation,

$$\mathbf{L}\,\mathrm{vec}(\mathbf{P}_{\mathcal{S}}^T \mathcal{X} \mathbf{P}_{\mathcal{I}}) = \mathbf{P}_{\mathcal{I}}^T \otimes \mathbf{P}_{\mathcal{S}}^T \mathbf{L}\,\mathrm{vec}(\mathcal{X}). \tag{79}$$

Noting that $(\mathbf{P}_{\mathcal{I}}^T \otimes \mathbf{P}_{\mathcal{S}}^T)^{-1} = \mathbf{P}_{\mathcal{I}} \otimes \mathbf{P}_{\mathcal{S}}$, and multiplying by this inverse both sides (from the left), we obtain,

$$\mathbf{P}_{\mathcal{I}} \otimes \mathbf{P}_{\mathcal{S}}\mathbf{L}\,\mathrm{vec}(\mathbf{P}_{\mathcal{S}}^T \mathcal{X} \mathbf{P}_{\mathcal{I}}) = \mathbf{L}\,\mathrm{vec}(\mathcal{X}). \tag{80}$$

Applying, again, the property in Equation (70), we obtain,

$$\mathbf{P}_{\mathcal{I}} \otimes \mathbf{P}_{\mathcal{S}}\mathbf{L}\mathbf{P}_{\mathcal{I}}^T \otimes \mathbf{P}_{\mathcal{S}}^T \,\mathrm{vec}(\mathcal{X}) = \mathbf{L}\,\mathrm{vec}(\mathcal{X}). \tag{81}$$

Since this should be true for any $\mathcal{X} \in \mathbb{R}^{2^n \times n}$, we derive,

$$\mathbf{P}_{\mathcal{I}} \otimes \mathbf{P}_{\mathcal{S}} \mathbf{L} \mathbf{P}_{\mathcal{I}}^T \otimes \mathbf{P}_{\mathcal{S}}^T = \mathrm{vec}(\mathbf{L}). \tag{82}$$

Again, applying Equation (70) on the left side, where,

$$\mathbf{A} \triangleq \mathbf{P}_{\mathcal{I}} \otimes \mathbf{P}_{\mathcal{S}}; \tag{83}$$

$$\mathbf{B} \triangleq \mathbf{L}; \tag{84}$$

$$\mathbf{C} \triangleq \mathbf{P}_{\mathcal{I}}^T \otimes \mathbf{P}_{\mathcal{S}}^T, \tag{85}$$

we get the following equality,

$$\mathbf{P}_{\mathcal{I}} \otimes \mathbf{P}_{\mathcal{S}} \mathbf{L} \mathbf{P}_{\mathcal{I}}^T \otimes \mathbf{P}_{\mathcal{S}}^T = \mathbf{P}_{\mathcal{I}} \otimes \mathbf{P}_{\mathcal{S}} \otimes \mathbf{P}_{\mathcal{I}} \otimes \mathbf{P}_{\mathcal{S}}\, \mathrm{vec}(\mathbf{L}). \tag{86}$$

By substituting this to the left side of Equation (82) we obtain Equation (62).

F.3. Comparative Parameter Reduction in Linear Equivariant Layers

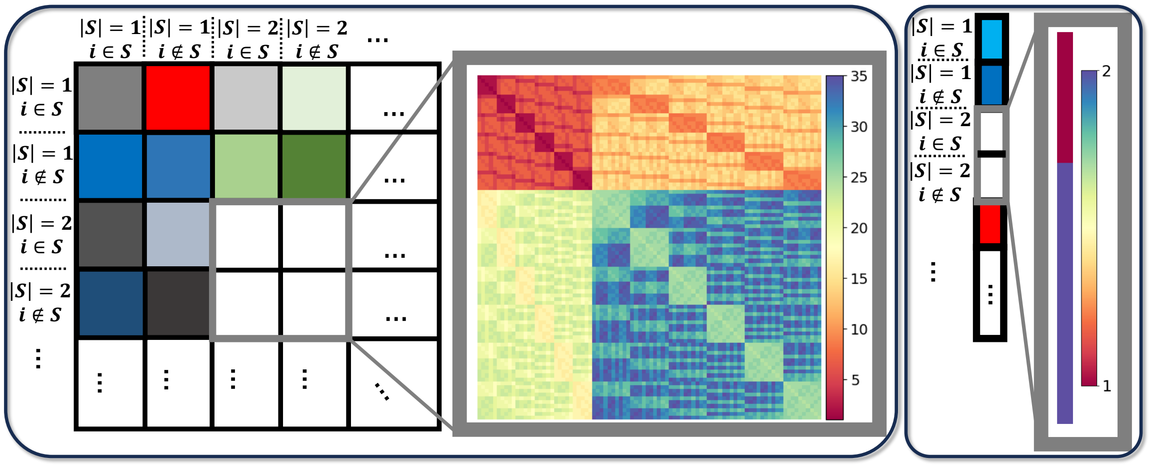


Figure 4: Visualization via heatmaps (different colors correspond to different parameters) of the parameter-sharing scheme determined by symmetries for a graph with $n = 6$ nodes, zooming-in on the block which corresponds to sets of size two. **Left:** Visualization of the weight matrix for the equivariant basis $\mathbf{B}^{\Gamma}_{S_1,i_1;S_2,i_2}$ (a total of 35 parameters in the block). **Right:** Visualization of the bias vector for the invariant basis $\mathbf{B}^{\gamma}_{S,i}$ (a total of 2 parameters in the block). Symmetry-based updates reduce parameters more effectively than previously proposed linear equivariant layers by treating indices as unordered tuples (see Appendix F.3 for a discussion).

To demonstrate the effectiveness of our parameter-sharing scheme, which results from considering unordered tuples rather than ordered tuples, we present the following comparison. 3-IGNs Maron et al. (2018) are structurally similar to our approach, with the main difference being that they consider indices as ordered tuples, while we consider them as sets. Both approaches use a total of six indices, as shown in the visualized block in Figure 4, making 3-IGNs a natural comparator. By leveraging our scheme, we reduce the number of parameters from 203 (the number of parameters in 3-IGNs) to just 35!

Table 1: Overview of the graph learning datasets.

Dataset	# Graphs	Avg. # nodes	Avg. # edges	Directed	Prediction task	Metric
Zinc-12k Sterling and Irwin (2015)	12,000	23.2	24.9	No	Regression	Mean Abs. Error
Zinc-Full Sterling and Irwin (2015)	249,456	23.2	49.8	No	Regression	Mean Abs. Error
ogbg-molhiv Hu et al. (2020)	41,127	25.5	27.5	No	Binary Classification	AUROC
ogbg-molbace Hu et al. (2020)	1513	34.1	36.9	No	Binary Classification	AUROC
ogbg-molesol Hu et al. (2020)	1,128	13.3	13.7	No	Regression	Root Mean Squ. Error

Appendix G. Experimental Details and Further Results

G.1. Dataset Description

In this section we overview the five different datasets considered; this is summarized in Table 1.

Zinc-12k and Zinc-Full Datasets Sterling and Irwin (2015); Gómez-Bombarelli et al. (2018); Dwivedi et al. (2023). The Zinc-12k dataset includes 12,000 molecular graphs sourced from the ZINC database, a compilation of commercially available chemical compounds. These molecular graphs vary in size, ranging from 9 to 37 nodes, where each node represents a heavy atom, covering 28 different atom types. Edges represent chemical bonds and there are three types of bonds. The main goal when using this dataset is to perform regression analysis on the constrained solubility (logP) of the molecules. The dataset is divided into training, validation, and test sets with 10,000, 1,000, and 1,000 molecular graphs respectively. The full version, Zinc-Full, comprises approximately 250,000 molecular graphs, ranging from 9 to 37 nodes and 16 to 84 edges per graph. These graphs also represent heavy atoms, with 28 distinct atom types, and the edges indicate bonds between these atoms, with four types of bonds present.

ogbg-molhiv, ogbg-molbace, ogbg-molesol Datasets Hu et al. (2020) These datasets are used for molecular property prediction and have been adopted by the Open Graph Benchmark (OGB, MIT License) from MoleculeNet. They use a standardized featurization for nodes (atoms) and edges (bonds), capturing various chemophysical properties.

We note that for all datasets, we used the random splits provided by the public benchmarks.

G.2. Experimental Details

Implementation Details. Our implementation of Equation (1) is given by:

$$\mathcal{X}^{(l+1)} = \text{MLP}\left(\sum_{i=1}^{3} \texttt{MPNN}^{(l+1,i)}\left(\mathcal{X}, \mathcal{A}_i\right)\right), \tag{87}$$

where $\mathcal{A}_1 = \mathcal{A}_G$, $\mathcal{A}_2 = \mathcal{A}_{\mathcal{T}(G)}$, and $\mathcal{A}_3 = \mathcal{A}_{\text{Equiv}}$.

We use a GINE Hu et al. (2019) base encoder. Given an adjacency matrix $\mathcal{A}$, and defining $e_{(S',v'),(S,v)}$ to denote the edge features from node (S',v') to node (S,v), it takes the following form:

$$\mathcal{X}(S,v) = \texttt{MLP}\Big((1+\epsilon)\cdot\mathcal{X}(S,v) + \sum_{(S',v')\sim_{\mathcal{A}}(S,v)} \texttt{ReLU}\big(\mathcal{X}(S',v') + e_{(S',v'),(S,v)}\big)\Big). \tag{88}$$

We note that for the symmetry-based updates, we switch the `ReLU` to an `MLP` to align with the theoretical analyses[2] (Appendix C), stating that we can implement the equivariant update developed in Section 3.2. A more thorough discussion regarding the implementation of the symmetry-based updates is given in Appendix G.4.

Our experiments were conducted using the PyTorch Paszke et al. (2019) and PyTorch Geometric Fey and Lenssen (2019) frameworks (resp. BSD and MIT Licenses), using a single NVIDIA L40 GPU, and for every considered experiment, we show the mean ± std. of 3 runs with different random seeds. Hyperparameter tuning was performed utilizing the Weight and Biases framework Biewald (2020) – see Appendix G.3. All our `MLP`s feature a single hidden layer equipped with a ReLU non-linearity function. For the encoding of atom numbers and bonds, we utilized learnable embeddings indexed by their respective numbers.

In the case of the OGBG-MOLHIV, OGBG-MOLESOL, OGBG-MOLBACE datasets, we follow Frasca et al. (2022), therefore adding a residual connection between different layers. Additionally, for those datasets (except OGBG-MOLHIV), we used linear layers instead of `MLP`s inside the GIN layers. Moreover, for these datasets, the following pooling mechanism was employed

$$\rho(\mathcal{X}) = \texttt{MLP}\left(\sum_{S}\left(\frac{1}{n}\sum_{v=1}^{n}\mathcal{X}(s,v)\right)\right). \tag{89}$$

G.3. HyperParameters

In this section, we detail the hyperparameter search conducted for our experiments. Besides standard hyperparameters such as learning rate and dropout, our specific hyperparameters are:

1. **Laplacian Dimension:** This refers to the number of columns used in the matrix U, where $L = U^T \lambda U$, for the spectral clustering in the coarsening function.
2. **SPD Dimension:** This represents the number of indices used in the node marking equation. To clarify, since $|S|$ might be large, we opt for using the first k indices that satisfy $i \in S$, sorted according to the SPD distance.

SPD Dimension. For the Laplacian dimension, we chose a fixed value of 10 for all bag sizes for both ZINC-12K and ZINC-FULL datasets. For OGBG-MOLHIV, we used a fixed value of 1, since the value 10 did not perform well. For the OGBG-MOLESOL and OGBG-MOLBACE datasets, we searched over the two values $\{1, 2\}$.

Laplacian Dimension. For the Laplacian dimension, we searched over the values $\{1, 2\}$ for all datasets.

Standard Hyperparameters. For ZINC-12K, we used a weight decay of 0.0003 for all bag sizes, except for the full bag size, for which we used 0.0001.

All of the hyperparameter search configurations are presented in Table 2, and the selected hyperparameters are presented in Table 3.

2. The theoretical analysis assumes the usage of an `MLP` for all three considered updates.

Table 2: Hyperparameters search for CS-GNN.

Dataset	Bag size	Num. layers	Learning rate	Embedding size	Epochs	Batch size	Dropout	Laplacian dimension	SPD dimension
Zinc-12k	$T = 2$	6	0.0005	96	400	128	0	$\{1, 2\}$	10
Zinc-12k	$T \in \{3, 4, 5, 8, 18\}$	6	0.0007	96	400	128	0	$\{1, 2\}$	10
Zinc-12k	$T =$ "full"	6	0.0007	96	500	128	0	$\{1, 2\}$	10
Zinc-FULL	$T = 4$	6	0.0007	96	400	128	0	$\{1, 2\}$	10
Zinc-FULL	$T =$ "full"	6	$\{0.001, 0.0005\}$	96	500	128	0	$\{1, 2\}$	10
ogbg-molhiv	$T \in \{2, 5,$ "full"$\}$	2	0.01	60	100	32	0.5	$\{1, 2\}$	1
ogbg-molesol	$T \in \{2, 5,$ "full"$\}$	3	0.001	60	100	32	0.3	$\{1, 2\}$	$\{ 1, 2 \}$
ogbg-molbace	$T \in \{2, 5,$ "full"$\}$	$\{2, 3\}$	0.01	60	100	32	0.3	$\{1, 2\}$	$\{ 1, 2 \}$

Table 3: Chosen Hyperparameters for CS-GNN.

Dataset	Bag size	Num. layers	Learning rate	Embedding size	Epochs	Batch size	Dropout	Laplacian dimension	SPD dimension
Zinc-12k	$T = 2$	6	0.0005	96	400	128	0	1	10
Zinc-12k	$T = 3$	6	0.0007	96	400	128	0	2	10
Zinc-12k	$T = 4$	6	0.0007	96	400	128	0	1	10
Zinc-12k	$T = 5$	6	0.0007	96	400	128	0	1	10
Zinc-12k	$T = 8$	6	0.0007	96	400	128	0	1	10
Zinc-12k	$T = 18$	6	0.0007	96	400	128	0	1	10
Zinc-12k	$T =$ "full"	6	0.0007	96	500	128	0	N/A	10
Zinc-FULL	$T = 4$	6	0.0007	96	400	128	0	1	10
Zinc-FULL	$T =$ "full"	6	0.0005	96	500	128	0	N/A	N/A
ogbg-molhiv	$T = 2\}$	2	0.01	60	100	32	0.5	1	1
ogbg-molhiv	$T = 5$	2	0.01	60	100	32	0.5	1	1
ogbg-molhiv	$T =$ "full"	2	0.01	60	100	32	0.5	N/A	N/A
ogbg-molesol	$T = 2$	3	0.001	60	100	32	0.3	1	2
ogbg-molesol	$T = 5$	3	0.001	60	100	32	0.3	1	2
ogbg-molesol	$T =$ "full"	3	0.001	60	100	32	0.3	N/A	N/A
ogbg-molbace	$T = 2$	3	0.01	60	100	32	0.3	1	1
ogbg-molbace	$T = 5$	3	0.01	60	100	32	0.3	1	2
ogbg-molbace	$T =$ "full"	3	0.01	60	100	32	0.3	N/A	N/A

Optimizers and Schedulers. For the ZINC-12k and Zinc-FULL datasets, we employ the Adam optimizer paired with a ReduceLROnPlateau scheduler,factor set to 0.5, patience at 40[3], and a minimum learning rate of 0. For the ogbg-molhiv dataset, we utilized the ASAM optimizer (Kwon et al., 2021) without a scheduler. For both ogbg-molesol and ogbg-molbace, we employed a constant learning rate without any scheduler.

G.4. Implementation of Linear Equivariant and Invariant layers – Extended Section

In this section, in a more formal discussion, we specify how to integrate those invariant and equivariant layers to our proposed architecture. We start by drawing an analogy between parameter sharing in linear layers and the operation of an MPNN on a fully connected graph with edge features in the following lemma,

Lemma 24 (Parameter Sharing as MPNN) *Let $B_1, \dots B_k : \mathbb{R}^{n \times n}$ be orthogonal matrices with entries restricted to 0 or 1, and let $W_1, \dots W_k \in \mathbb{R}^{d \times d'}$ denote a sequence of weight matrices. Define $B_+ = \sum_{i=1}^{k} B_i$ and choose $z_1, \dots z_k \in \mathbb{R}^{d^*}$ to be a set of unique vectors representing an encoding of the index set. The function, which represents an update via*

3. For Zinc-12k, $T \in \{2,$ "full"$\}$, we used a patience of 50.

parameter sharing:

$$f(X) = \sum_{i=1}^{k} B_i X W_i, \tag{90}$$

can be implemented by a stack of MPNN layers of the following form Gilmer et al. (2017),

$$m_u^l = \sum_{v \in N_{B_+}(u)} M^l(X_v^l, e_{u,v}),, \tag{91}$$

$$X_u^{l+1} = U^l(X_v^l, m_v^l), \tag{92}$$

where U^l, M^l are multilayer preceptrons (MLPs). The inputs to this MPNN are the adjacency matrix B_+, node feature vector X, and edge features – the feature of edge (u, v) is given by:

$$e_{u,v} = \sum_{i=1}^{k} z_i \cdot B_i(u, v). \tag{93}$$

Here, $B_i(u, v)$ denotes the (u, v) entry to matrix B_i.

The proof is given in Appendix H.

Thus, our implementation for the global update is as follows,

$$\mathcal{X}(S, i) = \texttt{MLP}\left((1+\epsilon)\cdot\mathcal{X}(S, i) + \sum_{(S', i') \sim \mathcal{A}_{\text{Equiv}}(S,i)} \texttt{MLP}\Big(\mathcal{X}(S', i') + e_{(S',i'),(S,i)}\Big)\right), \tag{94}$$

where $e_{(S',i'),(S,i)} = \sum_\Gamma z_\Gamma \cdot \mathbf{B}^\Gamma_{S,i;S',i'}$ and z_Γ are orthogonal 1-hot vectors for different Γ's. The connectivity $\mathcal{A}_{Equiv}$ is such that $\mathcal{A}_{Equiv}(S, v, S', v')$ contains the value one iff $v \in S, v = v'$. This corresponds to choosing only several Γ's in the partition, and since each Γ is invariant to the permutation, this choice still maintains equivariance.

G.5. Additional Results

We experimented extensively over five different datasets to answer the following questions: *(Q1) Can `CS-GNN` outperform efficient Subgraph GNNs operating on small bags? (Q2) Does the additional symmetry-based updates boost performance? (Q3) Does `CS-GNN` in the full-bag setting validate its theory and match state-of-the-art full-bag Subgraph GNNs?*

Baselines. For each task, we include several baselines. The Random baseline corresponds to random subgraph selection. We report the best performing random baseline from all prior work Bevilacqua et al. (2024); Kong et al. (2024); Qian et al. (2022); Bar-Shalom et al. (2024). The other two (non-random) baselines are: (1) Learned Bevilacqua et al. (2024); Kong et al. (2024); Qian et al. (2022), which represents methods that learn the specific subgraphs to be used; and (2) Full Zhang et al. (2023); Bar-Shalom et al. (2024), which corresponds to full-bag Subgraph GNNs.

ZINC and Zinc-FULL We experimented with both the ZINC-12k and Zinc-FULL datasets (Sterling and Irwin, 2015; Gómez-Bombarelli et al., 2018; Dwivedi et al., 2023),

Table 4: Test results on the Zinc-12k molecular dataset under $500k$ parameter budget. The top two results are reported as **First** and **Second**.

Method	Bag size	ZINC (MAE ↓)
GCN Kipf and Welling (2016)	$T = 1$	0.321 ± 0.009
GIN Xu et al. (2018)	$T = 1$	0.163 ± 0.004
OSAN Qian et al. (2022)	$T = 2$	0.177 ± 0.016
Random Kong et al. (2024)	$T = 2$	0.131 ± 0.005
PL Bevilacqua et al. (2024)	$T = 2$	0.120 ± 0.003
Mag-GNN Kong et al. (2024)	$T = 2$	**0.106** ± 0.014
Ours	$T = 2$	0.109 ± 0.005
Random Kong et al. (2024)	$T = 3$	$0.124 \pm$ N/A
Mag-GNN Kong et al. (2024)	$T = 3$	0.104 $\pm$ N/A
Ours	$T = 3$	**0.096** ± 0.005
Random Kong et al. (2024)	$T = 4$	$0.125 \pm$ N/A
Mag-GNN Kong et al. (2024)	$T = 4$	**0.101** $\pm$ N/A
Ours	$T = 4$	**0.090** ± 0.003
Random Bevilacqua et al. (2024)	$T = 5$	0.113 ± 0.006
PL Bevilacqua et al. (2024)	$T = 5$	0.109 ± 0.005
Ours	$T = 5$	**0.095** ± 0.003
Random Bevilacqua et al. (2024)	$T = 8$	0.102 ± 0.003
PL Bevilacqua et al. (2024)	$T = 8$	**0.097** ± 0.005
Ours	$T = 8$	**0.094** ± 0.006
Ours	$T = 18$	**0.082** ± 0.003
NGNN Zhang and Li (2021)	Full	$0.111_{\pm 0.003}$
DS-GNN Bevilacqua et al. (2022)	Full	$0.116_{\pm 0.009}$
DSS-GNN Bevilacqua et al. (2022)	Full	$0.102_{\pm 0.003}$
GNN-AK Zhao et al. (2022)	Full	$0.105_{\pm 0.010}$
GNN-AK+ Zhao et al. (2022)	Full	$0.091_{\pm 0.002}$
SUN Frasca et al. (2022)	Full	$0.083_{\pm 0.003}$
OSAN Qian et al. (2022)	Full	$0.154_{\pm 0.008}$
GNN-SSWL+ Zhang et al. (2023)	Full	0.070 ± 0.005
Subgraphormer Bar-Shalom et al. (2024)	Full	0.067 ± 0.007
Subgraphormer+PE Bar-Shalom et al. (2024)	Full	**0.063** ± 0.001
Ours	Full	**0.062** ± 0.0007

adhering to a $500k$ parameter budget as prescribed. As shown in Table 4 (which corresponds to Figure 2 in tabular form), CS-GNN outperforms all efficient baselines by a significant margin, with at least a +0.008 MAE improvement for bag sizes $T \in \{3, 4, 5\}$. Additionally, in the full-bag setting, our method recovers state-of-the-art results.

For the Zinc-FULL dataset, The results are summarized in Table 5.

Table 5: Comparison over the ZINC-FULL molecular dataset under 500k parameter budget. The best performing method is highlighted in **blue**, while the second best is highlighted in **red**.

Model ↓ / Dataset →	ZINC-FULL (MAE ↓)
MAG-GNN Kong et al. (2024) ($T = 4$)	**0.030**$\pm$0.002
Ours ($T = 4$)	**0.027**$\pm$0.002
GNN-SSWL Zhang et al. (2023) ($T =$ "full")	0.026$\pm$0.001
GNN-SSWL+ Zhang et al. (2023) ($T =$ "full")	0.022$\pm$0.001
Subgraphormer Bar-Shalom et al. (2024)($T =$ "full")	**0.020**$\pm$0.002
Subgraphormer + PE Bar-Shalom et al. (2024) ($T =$ "full")	0.023$\pm$0.001
Ours ($T =$ "full")	**0.021**$\pm$0.001

OGB. We tested our framework on several datasets from the OGB benchmark collection Hu et al. (2020). Table 7 shows the performance of our method compared to both efficient and full-bag Subgraph GNNs. Our CS-GNN outperforms all baselines across all datasets for bag sizes $T \in \{2, 5\}$, except for the MOLHIV dataset with $T = 2$, where PL achieves the best results and our method ranks second. In the full-bag setting, CS-GNN is slightly outperformed by the top-performing Subgraph GNNs but still offers comparable results.

Table 6: Ablation study.

Bag size	w/	w/o
T=2	**0.109**$\pm$0.005	0.143$\pm$0.003
T=3	**0.096**$\pm$0.005	0.101$\pm$0.006
T=4	**0.090**$\pm$0.003	0.106$\pm$0.001
T=5	**0.095**$\pm$0.003	0.104$\pm$0.005

Ablation study – symmetry-based updates. We assessed the impact of the symmetry-based update on the performance of CS-GNN. Specifically, we asked, *do the symmetry-based updates significantly contribute to the performance of CS-GNN?* To evaluate this, we conducted several experiments using the ZINC-12K dataset across various bag sizes, $T \in \{2, 3, 4, 5\}$, comparing CS-GNN with and without the symmetry-based update. The results are summarized in Table 6. It is clear that the symmetry-based updates play a key role in the performance of CS-GNN. For a bag size of $T = 2$, the inclusion of the symmetry-based update improves the MAE by a significant 0.034. For other bag sizes, the improvements range from 0.005 to 0.016, clearly demonstrating the benefits of including the symmetry-based updates.

Discussion. In what follows, we address research questions **Q1** to **Q3**.

1. Tables 4 and 7 clearly demonstrate that we outperform efficient Subgraph GNNs (which operate on a small bag) in the vast majority of dataset and bag size combinations.

2. Our ablation study on the ZINC-12K dataset, as shown in Table 6, clearly demonstrates the benefits of the symmetry-based updates across all the considered bag sizes.

Table 7: Results on OGB datasets. The top two results are reported as **First** and **Second**.

Model ↓ / Dataset →	Bag size	MOLHIV (ROC-AUC ↑)	MOLBACE (ROC-AUC ↑)	MOLESOL (RMSE ↓)
GIN Xu et al. (2018)	$T = 1$	75.58±1.40	72.97±4.00	1.173±0.057
Random Bevilacqua et al. (2024)	$T = 2$	77.55±1.24	75.36±4.28	0.951±0.039
PL Bevilacqua et al. (2024)	$T = 2$	**79.13**±0.60	**78.40**±2.85	**0.877**±0.029
Mag-GNN Kong et al. (2024)	$T = 2$	77.12±1.13	-	-
Ours	$T = 2$	**77.72**±0.76	**80.58**±1.04	**0.850**±0.024
OSAN Qian et al. (2022)	$T = 3$	-	-	**0.959**±0.184
OSAN Qian et al. (2022)	$T = 5$	-	76.30±3.00	-
PL Bevilacqua et al. (2024)	$T = 5$	**78.49**±1.01	**78.39**±2.28	**0.883**±0.032
Random Bevilacqua et al. (2024)	$T = 5$	77.30±2.56	78.14±2.36	0.900±0.032
Ours	$T = 5$	**79.09**±0.90	**79.64**±1.43	**0.863**±0.029
GNN-SSWL+ Zhang et al. (2023)	Full	**79.58**±0.35	**82.70**±1.80	0.837±0.019
Subgraphormer	Full	**80.38**±1.92	81.62±3.55	0.832±0.043
Subgraphormer + PE	Full	79.48±1.28	**84.35**±0.65	**0.826**±0.010
Ours	Full	79.44±0.87	80.71±1.76	**0.814**±0.021

Table 8: Run time comparison over the Zinc-12k dataset. Time taken at train for one epoch and at inference on the test set. All values are in milliseconds.

Method	Train time (for a single epoch; ms)	Test time (ms)	MAE ↓
GIN Xu et al. (2018)	1370.10 ± 10.97	84.81 ± 0.26	0.163 ± 0.004
OSAN Qian et al. (2022) ($T = 2$)	2964.46 ± 30.36	227.93 ± 0.21	0.177 ± 0.016
PL Bevilacqua et al. (2024) ($T = 2$)	2489.25 ± 9.42	150.38 ± 0.33	0.120 ± 0.003
Ours ($T = 2$)	2764.60 ± 234	383.14 ± 15.74	0.109 ± 0.005

3. On the Zinc-12k dataset (see Table 4), CS-GNN achieves state-of-the-art results compared to Subgraph GNNs. On the OGB datasets (see Table 7), our performance is comparable to these top-performing Subgraph GNNs.

Runtime comparison. We compare the training time and prediction performance on the Zinc-12k dataset. For all methods, we report the training and inference times on the entire training and test sets, respectively, using a batch size of 128. Our experiments were conducted using an NVIDIA L40 GPU, while for the baselines, we used the timing reported in Bevilacqua et al. (2024), which utilized an RTX A6000 GPU. The runtime comparison is presented in Table 8.

G.6. Zinc12k Product Graph Visualization

In this subsection, we visualize the product graph derived from the first graph in the Zinc12k dataset. Specifically, we present the right part of Figure 1, for the case of the real-world graphs in the Zinc12k dataset. We perform this visualization for different cluster sizes, $T \in \{2, 3, 4, 5, 8, 12\}$, which also define the bag size, hence the notation T. The nodes in the product graph, $\mathcal{T}(G)\square G$, are (S, v), where S is the coarsened graph node (again a tuple), and v is the node index (of a node from the original graph). For better clarity, we color the nodes (S, v) with $v \in S$ using different colors, while reserving the gray color exclusively for nodes (S, v) where $v \notin S$. The product graphs are visualized in Figures 5 to 10 below.

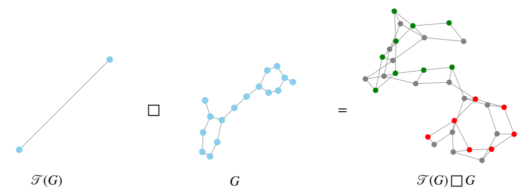


Figure 5: $T = 2$.

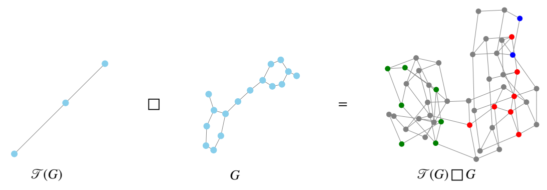


Figure 6: $T = 3$.

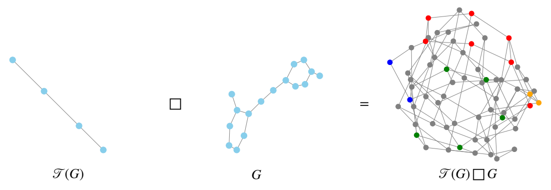


Figure 7: $T = 4$.

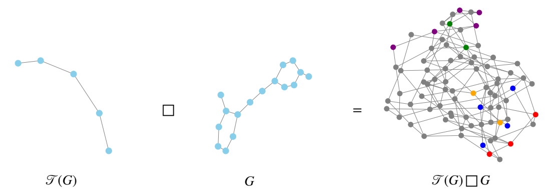


Figure 8: $T = 5$.

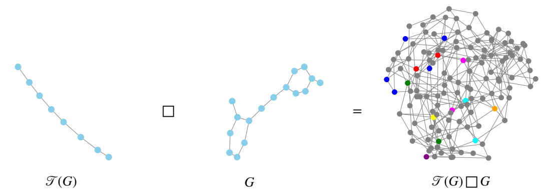


Figure 9: $T = 8$.

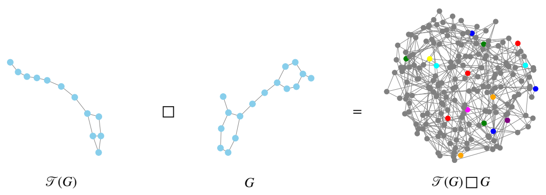


Figure 10: $T = 12$.

Appendix H. Proofs

H.1. Proofs of Appendix C

We first state the memorization theorem, proven in Yun et al. (2019) , which will be heavily used in a lot of the proofs in this section.

Theorem 25 (Memorization Theorem) *Consider a dataset* $\{x_j, y_j\}_{j=1}^N \in \mathbb{R}^d \times \mathbb{R}^{d_y}$ *, with each* x_j *being distinct and every* $y_j \in \{0,1\}^{d_y}$*. There exists a 4-layer fully connected ReLU neural network* $f_\theta : \mathbb{R}^d \to \mathbb{R}^{d_y}$ *that perfectly maps each* x_j *to its corresponding* y_j*, i.e.,* $f_\theta(x_j) = y_j$ *for all* j*.*

We now restate and prove the propositions and lemmas of Appendix C.

Lemma 26 (Parameter Sharing as MPNN) *Let* $B_1, \dots B_k : \mathbb{R}^{n\times n}$ *be orthogonal matrices with entries restricted to 0 or 1, and let* $W_1, \dots W_k \in \mathbb{R}^{d\times d'}$ *denote a sequence of weight matrices. Define* $B_+ = \sum_{i=1}^k B_i$ *and choose* $z_1, \dots z_k \in \mathbb{R}^{d^*}$ *to be a set of unique vectors representing an encoding of the index set. The function that represents an update via parameter sharing:*

$$f(X) = \sum_{i=1}^{k} B_i X W_i, \tag{14}$$

can be implemented on any finite family of graphs $\mathcal{G}$*, by a stack of MPNN layers of the following form Gilmer et al. (2017),*

$$m_v^l = \sum_{u\in N_{B_+}(v)} M^l(X_u^l, e_{u,v}), \tag{15}$$

$$X_v^{l+1} = U^l(X_v^l, m_v^l), \tag{16}$$

where U^l, M^l *are multilayer perceptrons (MLPs). The inputs to this MPNN are the adjacency matrix* B_+*, node feature vector* X*, and edge features – the feature of edge* (u,v) *is given by:*

$$e_{u,v} = \sum_{i=1}^{k} z_i \cdot B_i(u,v). \tag{17}$$

Here, $B_i(u,v)$ *denotes the* (u,v) *entry to matrix* B_i*.*

Proof Since we are concerned only with input graphs G from a finite family of graphs (where "finite" means that the maximal graph size is bounded and all possible node and edge feature values come from a finite set), we assume that for any $v \in [n]$, $i \in [k]$, both the input feature vectors $X_v \in \mathbb{R}^d$ and the encoding vectors $z_i \in \mathbb{R}^{d^*}$ are one-hot encoded. We aim to show that under these assumptions, any function $f(\cdot)$ of the form 90 can be realized through a single-layer update detailed in Equations 92 , 91, where M is a 4 layer MLP , and U is a single linear layer. The proof involves the following steps:

1. Compute $[B_1X, \dots, B_kX]$ using the message function M.

2. Compute $f(X)$ using the update function U.

Step 1: We notice that for every $i \in [k]$, $v \in [n]$ we have:

$$(B_i X)_v = \sum_{B_i(v,u)=1} X_u = \sum_{u \in N_{B_+}(v)} X_u \cdot \mathbf{1}_{z_i}(e_{u,v}). \tag{95}$$

Here $\mathbf{1}_{z_i}$ is the indicator function of the set $\{z_i\}$. We notice that since X_u and z_i are one-hot encoded, there is a finite set of possible values for the pair $(X_u, e_{u,v})$. In addition, the function:

$$enc(X_u, e_{u,v}) = [X_u \cdot \mathbf{1}_{z_1}(e_{u,v}), \ldots, X_u \cdot \mathbf{1}_{z_k}(e_{u,v})] \tag{96}$$

outputs vectors in the set $\{0,1\}^{d \times k}$. Thus, employing the memorization theorem 25, we define a dataset $\{x_j, y_j\}_{j=1}^N$ by taking the x_js to be all possible (distinct) values of $(X_u, e_{u,v})$ with each corresponding y_i being the output enc(x_i). We note that there are finitely many such values as both X_u and $e_{u,v}$ are one-hot encoded. The theorem now tells us that there exists a a 4-layer fully connected ReLU neural network M such that:

$$M(X_u, e_{u,v}) = enc(X_u, e_{u,v}). \tag{97}$$

and so, equation 95 implies:

$$m_v = \sum_{u \in N_{B_+}(v)} M(X_u, e_{u,v}) = [(B_1 X)_v, \ldots, (B_k X)_v]. \tag{98}$$

Step 2: Define $P_i : \mathbb{R}^{k \times d} \to \mathbb{R}^d$ as the projection operator, extracting coordinates $d \cdot i + 1$ through $d \cdot (i+1)$ from its input vector:

$$P_i(V) = V|_{d \cdot i+1 : d \cdot (i+1)}. \tag{99}$$

We define the update function to be the following linear map:

$$U(X_v, m_v) = \sum_{i=1}^{k} P_i(m_v) W_i. \tag{100}$$

Combining equations 98 and 100 we get:

$$\tilde{X}_v = U(X_v, m_v) = \sum_{i=1}^{k} (B_i X)_v \cdot W_i = f(X)_v. \tag{101}$$

■

Proposition 27 (Equivalence of General Layer and Implemented Layer) *Let $\mathcal{T}(\cdot)$ be a coarsening function, π be a generalized node marking policy, and $\mathcal{G}$ be a finite family of graphs. Applying a stack of t general layer updates as defined in Equation 11 to the node feature map $\mathcal{X}(S, v)$ induced by $\pi(G, \mathcal{T})$, can be effectively implemented by applying a stack of t layer updates specified in Equations 25 and 26 to $\mathcal{X}(S, v)$. Additionally, the depths of all MLPs that appear in 25 and 26 can be bounded by 4.*

Proof For convenience, let us first restate the general layer update:

$$\begin{aligned}\mathcal{X}^{t+1}(S,v) = f^t\Big(&\mathcal{X}^t(S,v),\\ &\text{agg}_1^t\{\!\{(\mathcal{X}^t(S,v'), e_{v,v'}) \mid v' \sim_G v\}\!\},\\ &\text{agg}_2^t\{\!\{(\mathcal{X}^t(S',v), \tilde{e}_{S,S'}) \mid S' \sim_{G^{\mathcal{T}}} S\}\!\},\\ &\text{agg}_3^t\{\!\{(\mathcal{X}^t(S',v), z(S,v,S',v)) \mid S' \in V^{\mathcal{T}} \text{s.t. } v \in S'\}\!\},\\ &\text{agg}_4^t\{\!\{(\mathcal{X}^t(S,v'), z(S,v,S,v')) \mid v' \in V \text{s.t. } v' \in S\}\!\}\Big),\end{aligned} \tag{11}$$

as well as the two step implemented layer update:

$$\mathcal{X}_i^t(S,v) = U_i^t\left((1+\epsilon_i^t)\cdot\mathcal{X}^t(S,v) + \sum_{(S',v')\sim_{A_i}(S,v)} M^t(\mathcal{X}^t(S',v') + e_i(S,v,S',v'))\right). \tag{25}$$

$$\mathcal{X}^{t+1}(S,v) = \text{U}_{\text{fin}}^t\left(\sum_{i=1}^{4}\mathcal{X}_i^t(S,v)\right). \tag{26}$$

We aim to demonstrate that any general layer, which updates the node feature map $\mathcal{X}^t(S,v)$ at layer t to node feature map $\mathcal{X}^{t+1}(S,v)$ at layer $t+1$ as described in equation 11, can be effectively implemented using the layer update processes outlined in equations 25 and 26.

As we are concerned only with input graphs belonging to the finite graph family $\mathcal{G}$ (where "finite" indicates that the maximal graph size is bounded and all node and edge features have a finite set of possible values), we assume that the values of the node feature map $\mathcal{X}^t(S,v)$ and the edge feature vectors $e_i(S,v,S',v')$ are represented as one-hot vectors in $\mathbb{R}^k$. We also assume that the parameterized functions f^t and $\text{agg}_1^t, \dots \text{agg}_4^t$, which are applied in Equation 11 outputs one-hot vectors. Finally, we assume that there exists integers d, d^*, such that the node feature map values are supported on coordinates $1, \dots d$, the edge feature vectors are supported on coordinates $d+1, \dots d+d^*$, and coordinates $d+d^*+1, \dots k$ are used as extra memory space, with:

$$k > d \times d^* + d + d^*. \tag{102}$$

We note that the last assumption can be easily achieved using padding. The proof involves the following steps:

1. For $i = 1, \dots, 4$, Use the term:

$$m_i^t = \sum_{(S',v')\sim_{A_i}(S,v)} M^t(\mathcal{X}^t(S',v') + e_i(S,v,S',v')) \tag{103}$$

to uniquely encode:

$$\{\!\{(\mathcal{X}^t(S',v'), e_i(S,v,S',v')) \mid (S',v') \sim_{A_i} (S,v)\}\!\}. \tag{104}$$

2. Use the term:
$$\mathcal{X}^t_* = \sum_{i=1}^{4} \mathcal{X}^t_i(S, v) \tag{105}$$
to uniquely encode the input of f^t as a whole.

3. Implement the parameterized function f^t.

Step 1: Since we assume that node feature map values and edge feature vectors are supported on orthogonal sub-spaces of $\mathbb{R}^k$, the term:
$$\mathcal{X}^t(S, v) + e_i(S, v, S', v') \tag{106}$$
uniquely encodes the value of the tuple:
$$(\mathcal{X}^t(S, v), e_i(S, v, S', v')). \tag{107}$$

Since $\mathcal{X}^t(S, v)$ is a one-hot encoded vector with d possible values, while $e_i(S, v, S', v')$ is a one-hot encoded vector with d^* possible values, their sum has $d \cdot d^*$ possible values. Thus there exists a function:
$$\text{enc} : \mathbb{R}^k \to \mathbb{R}^k$$
which encodes each such possible value as a one-hot vector in $\mathbb{R}^k$ supported on the last $k - d - d^*$ coordinates (this is possible because of equation 102). Now, employing theorem 25, we define the x_js as all possible (distinct) values of 106, with each corresponding y_j being the output $\text{enc}(x_j)$. The theorem now tells us that there exists a 4-layer fully connected ReLU neural network capable of implementing the function $\text{enc}(\cdot)$. We choose M^t to be this network. Now since m^t_i, defined in equation 103 is a sum of one-hot encoded vectors, it effectively counts the number of each possible value in the set 104. This proves step 1.

Step 2: First, we note that:
$$\begin{aligned} &\{\!\{(\mathcal{X}^t(S, v'), e_{v,v'}) \mid v' \sim_G v\}\!\} \\ &= \{\!\{(\mathcal{X}^t(S', v'), e_G(S, v, S', v')) \mid (S, v) \sim_{A_G} (S', v')\}\!\} \end{aligned} \tag{108}$$
$$\begin{aligned} &\{\!\{(\mathcal{X}^t(S', v), e_{s',s}) \mid S' \sim_{\mathcal{T}(G)} S\}\!\} \\ &= \{\!\{(\mathcal{X}^t(S', v'), e_{\mathcal{T}(G)}(S, v, S', v')) \mid (S, v) \sim_{A_{\mathcal{T}(G)}} (S', v')\}\!\} \end{aligned} \tag{109}$$
$$\begin{aligned} &\{\!\{(\mathcal{X}^t(S', v), z(S, v, S', v')) \mid v \in S'\}\!\} \\ &= \{\!\{(\mathcal{X}^t(S', v'), e_{P_1}(S, v, S', v')) \mid (S, v) \sim_{A_{P_1}} (S', v')\}\!\} \end{aligned} \tag{110}$$
$$\begin{aligned} &\{\!\{(\mathcal{X}^t(S, v'), z(S, v, S', v')) \mid v' \in S\}\!\} \\ &= \{\!\{(\mathcal{X}^t(S', v'), e_{P_2}(S, v, S', v')) \mid (S, v) \sim_{A_{P_2}} (S', v')\}\!\} \end{aligned} \tag{111}$$

Now, since m^t_i and $\mathcal{X}^t(S, v)$ are supported on orthogonal sub-spaces of $\mathbb{R}^k$, the sum $\mathcal{X}^t(S, v) + m^t_i$ uniquely encodes the value of:
$$\left(\mathcal{X}^t(S, v), \{\!\{(\mathcal{X}^t(s, v), e_i(S, v, S', v')) \mid (S, v) \sim_{A_i} (S,' v')\}\!\}\right). \tag{112}$$

Thus, we choose $\epsilon_1^t, \ldots, epsilon_4^t$ to be all zeroes. To compute the aggregation functions $\mathrm{agg}_1^t, \ldots, \mathrm{agg}_4^t$ using these unique encodings, and to avoid repetition of the value $\mathcal{X}^t(S,v)$, we define auxiliary functions $\tilde{\mathrm{agg}}_i^t : \mathbb{R}^k \to \mathbb{R}^{k_i}$ for $i = 1, \ldots, 4$ as follows:

$$\tilde{\mathrm{agg}}_1^t(\mathcal{X}^t(S,v) + m_1^t) = \left(\mathcal{X}^t(S,v), \mathrm{agg}_1^t \{\!\{ (\mathcal{X}^t(S,v'), e_1(S,v,S',v')) \mid (S,v) \sim_{A_1} (S,' v') \}\!\}\right) \tag{113}$$

and for $i > 1$:

$$\tilde{\mathrm{agg}}_i^t(\mathcal{X}^t(S,v) + m_i^t) = \mathrm{agg}_i^l \{\!\{ (\mathcal{X}^t(s,v), e_i(S,v,S',v')) \mid (S,v) \sim_{A_i} (S,' v') \}\!\}. \tag{114}$$

Here, since we avoided repeating the value of $\mathcal{X}^t(S,v)$ by only adding it to the output of $\tilde{\mathrm{agg}}_1^t(\cdot)$, the expression:

$$\left(\tilde{\mathrm{agg}}_1^t(\mathcal{X}^t(S,v) + m_1^t), \ldots, \tilde{\mathrm{agg}}_4^t(\mathcal{X}^t(S,v) + m_4^t)\right) \tag{115}$$

is exactly equal to the input of f^t. In addition, since the function agg_i^t outputs one-hot encoded vectors, and the vector $\mathcal{X}^t(S,v)$ is one-hot encoded, the output of $\tilde{\mathrm{agg}}_i^t$ is always within the set $\{0,1\}^{k_i}$. Now for any input vector $X \in \mathbb{R}^k$ define:

$$V_1^t(X) = (\tilde{\mathrm{agg}}_1^t(X),\ 0_{k_2},\ 0_{k_3},\ 0_{k_4}). \tag{116}$$

$$V_2^t(X) = (0_{k_1},\ \tilde{\mathrm{agg}}_2^t(X),\ 0_{k_3},\ 0_{k_4}). \tag{117}$$

$$V_3^t(X) = (0_{k_1},\ 0_{k_2},\ \tilde{\mathrm{agg}}_3^t(X),\ 0_{k_4}). \tag{118}$$

$$V_4^t(X) = (0_{k_1},\ 0_{k_2},\ \ 0_{k_3}, \tilde{\mathrm{agg}}_4^t(X)). \tag{119}$$

We note that since the output of agg_i^t is always within the set $\{0,1\}^{k_i}$, the outputs of V_i^t is always within $\{0,1\}^{k_1+\cdots+k_4}$. Now for $i = 1, \ldots 4$, employing theorem 25 we define a dataset $\{x_j, y_j\}_{j=1}^N$ by taking the x_js as all possible (distinct) values of $\mathcal{X}^t(S,v) + m_i^t$, with each corresponding y_j being the output $V_i^t(x_j)$. We note that there are finitely many such values as both $\mathcal{X}^t(S,v)$ and m_i^t are one-hot encoded vectors. The theorem now tells us that there exists a a 4-layer fully connected ReLU neural network capable of implementing the function $V_i^t(\cdot)$. We choose U_i^t to be this network. Equations 116 - 119 now give us:

$$\sum_{i=1}^{4} \mathcal{X}_i^t(S,v) = \left(\tilde{\mathrm{agg}}_1^t(\mathcal{X}^t(S,v) + m_1^t), \ldots, \tilde{\mathrm{agg}}_4^t(\mathcal{X}^t(S,v) + m_4^t)\right). \tag{120}$$

which as stated before, is exactly the input to f^t. This proves step 2.

Step 3: We employ theorem 25 for one final time, defining a dataset $\{x_j, y_j\}_{j=1}^N$ by taking the x_js as all possible(distinct) values of:

$$\sum_{i=1}^{4} \mathcal{X}_i^t(S,v)$$

(which we showed is a unique encoding to the input of $f^t(\cdot)$), with each corresponding y_j being the output $f^t(x_j)$. We note that Given the finite nature of our graph set, there are finitely many such values. Recalling that $f^t(\cdot)$ outputs one-hot encoded vectors, The theorem now tells us that there exists a a 4-layer fully connected ReLU neural network capable of implementing the function $f^t(\cdot)$. We choose U^t_{fin} to be this network. This completes the proof. ■

H.2. Proofs of Appendix D

Proposition 28 (Equal Expressivity of Node Marking Policies) *For any coarsening function $\mathcal{T}(\cdot)$ the following holds:*

$$\textit{CS-GNN}(\mathcal{T}, \pi_S) = \textit{CS-GNN}(\mathcal{T}, \pi_{SS}) = \textit{CS-GNN}(\mathcal{T}, \pi_{MD}). \tag{32}$$

Proof Let $\Pi = \{\pi_{\text{S}}, \pi_{\text{SS}}, \pi_{\text{MD}}\}$ be the set of all relevant node initialization policies, and assume for simplicity that our input graphs have no node features (the proof can be easily adjusted to account for the general case). For each $\pi \in \Pi$, let $\mathcal{X}^\pi(S, v)$ denote the node feature map induced by general node marking policy π, as per Theorem 12. We notice it is enough to prove for each $\pi_1, \pi_2 \in \Pi$ that $\mathcal{X}^{\pi_1}(S, v)$ can be implemented by updating $\mathcal{X}^{\pi_2}(S, v)$ using a stack of T layers of type 50. Thus, we prove the following four cases:

- Node + Size Marking ⇒ Simple Node Marking.
- Minimum Distance ⇒ Simple Node Marking.
- Simple Node Marking ⇒ Node + Size Marking.
- Simple Node Marking ⇒ Minimum Distance.

Node + Size Marking ⇒ Simple Node Marking:

In this case, we aim to update the node feature map:

$$\mathcal{X}^0(S, v) = \mathcal{X}^{\pi_{SS}}(S, v) = \begin{cases} (1, |S|) & v \in S \\ (0, |S|) & v \notin S. \end{cases} \tag{121}$$

We notice that:

$$\mathcal{X}^0(S, v) = \langle (1, 0),\ \mathcal{X}^{\pi_{\text{S}}}(S, v) \rangle, \tag{122}$$

where $\langle \cdot, \cdot \rangle$ denotes the standard inner product in $\mathbb{R}^2$. Using a CS-GNN update as per equation 11, with the update function:

$$f^1(\mathcal{X}^0(S, v), \cdot, \cdot, \cdot, \cdot) = \langle (1, 0), \mathcal{X}^0(S, v) \rangle, \tag{123}$$

where $f(a, \cdot, \cdot, \cdot, \cdot)$ indicates that the function f depends solely on the parameter a, we obtain:

$$\mathcal{X}^1(S, v) = f^1(\mathcal{X}^0(S, v), \cdot, \cdot, \cdot, \cdot) = \mathcal{X}^{\pi_S}(S, v). \tag{124}$$

This implies that for any coarsening function $\mathcal{T}(\cdot)$, the following holds:

$$\text{CS-GNN}(\mathcal{T}, \pi_{\text{S}}) \subseteq \text{CS-GNN}(\mathcal{T}, \pi_{\text{SS}}). \tag{125}$$

Minimum Distance ⇒ Simple Node Marking:

In this case, we aim to update the node feature map:

$$\mathcal{X}^0(S,v) = \mathcal{X}^{\pi_{\text{MD}}}(S,v) = \min_{v \in s} d_G(u,v) \tag{126}$$

We notice that:

$$\mathcal{X}^{\text{S}}(S,v) = g(\mathcal{X}^0(S,v)) \tag{127}$$

where $g : \mathbb{R} \to \mathbb{R}$ is any continuous function such that:

1. $g(x) = 1 \; \forall x > \frac{1}{2}$,
2. $g(x) = 0 \; \forall x < \frac{1}{4}$.

Using a CS-GNN update as per equation 11, with the update function:

$$f^1(\mathcal{X}^0(S,v), \cdot, \cdot, \cdot, \cdot) = g(\mathcal{X}^0(S,v)), \tag{128}$$

we obtain:

$$\mathcal{X}^1(S,v) = f^1(\mathcal{X}^0(S,v), \cdot, \cdot, \cdot, \cdot) = \mathcal{X}^{\pi_S}(S,v). \tag{129}$$

This implies that for any coarsening function $\mathcal{T}(\cdot)$ the following holds:

$$\text{CS-GNN}(\mathcal{T}, \pi_{\text{S}}) \subseteq \text{CS-GNN}(\mathcal{T}, \pi_{\text{MD}}). \tag{130}$$

Simple Node Marking ⇒ Node + Size Marking: In this case, we aim to update the node feature map:

$$\mathcal{X}^0(S,v) = \mathcal{X}^{\pi_S}(S,v) = \begin{cases} 1 & v \in S \\ 0 & v \notin S. \end{cases} \tag{131}$$

We notice that:

$$\sum_{v' \in S} \mathcal{X}^0(S,v') = |S|. \tag{132}$$

Using a CS-GNN update as per Equation (11), with aggregation function:

$$\text{agg}_4^l \{\!\{ (\mathcal{X}^0(S,v'), z(S,v,S,v')) \mid v' \in S \}\!\} = \sum_{v' \in S} \mathcal{X}^0(S,v'), \tag{133}$$

and update function:

$$f^1\left(\mathcal{X}^0(S,v), \cdot, \cdot, \cdot, \sum_{v' \in S} \mathcal{X}^0(S,v')\right) = \left(\mathcal{X}^0(S,v), \sum_{v' \in S} \mathcal{X}^0(S,v')\right), \tag{134}$$

we obtain:

$$\mathcal{X}^1(S,v) = f^1\left(\mathcal{X}^0(S,v), \cdot, \cdot, \cdot, \sum_{v' \in S} \mathcal{X}^0(S,v')\right) = \mathcal{X}^{\pi_{SS}}(S,v). \tag{135}$$

This implies that for any coarsening function $\mathcal{T}(\cdot)$ the following holds:

$$\text{CS-GNN}(\mathcal{T}, \pi_{\text{SS}}) \subseteq \text{CS-GNN}(\mathcal{T}, \pi_{\text{S}}). \tag{136}$$

Simple Node Marking $\Rightarrow$ Minimum Distance:

In this case, we aim to update the node feature map:

$$\mathcal{X}^0(S, v) = \mathcal{X}^{\pi_S}(S, v) = \begin{cases} 1 & v \in S \\ 0 & v \notin S. \end{cases} \tag{137}$$

We shall prove that $\mathcal{X}^{\pi_{\text{MD}}}$ can be expressed by updating $\mathcal{X}^0(S, v)$ with a stack of CS-GNN layers. We do this by inductively showing that this procedure can express the following auxiliary node feature maps:

$$\mathcal{X}^t_*(S, v) = \begin{cases} \min_{v' \in S} d_G(v, v') + 1 & \min_{v' \in S} d_G(v, v') \leq t \\ 0 & \text{otherwise.} \end{cases} \tag{138}$$

We notice first that:

$$\mathcal{X}^0(S, v) = \mathcal{X}^0_*(S, v). \tag{139}$$

Now for the induction step, assume that there exists a stack of t CS-GNN layers such that:

$$\mathcal{X}^t(S, v) = \mathcal{X}^t_*(S, v). \tag{140}$$

We observe that equation:

$$\min_{v' \in S} d_G(v, v') = t + 1 \tag{141}$$

holds if and only if the following two conditions are met:

$$\min_{v' \in S} d_G(v, v') > t \tag{142}$$

$$\exists u \in N_G(v) \text{ s.t. } \min_{u' \in S} d_G(u, u') = t. \tag{143}$$

Equation 138 implies:

$$\min_{v' \in S} d_G(v, v') > t \Leftrightarrow \mathcal{X}^t(S, v) = 0. \tag{144}$$

In addition, since the node feature map $\mathcal{X}^t = \mathcal{X}^t_*$ is bounded by $t + 1$, Equation (138) implies:

$$\exists u \in N_G(v) \text{ s.t. } \min_{u' \in S} d_G(u, u') = t \Leftrightarrow \max\{\mathcal{X}^t(s, u) \mid v \sim_G u\} = t + 1. \tag{145}$$

Now, let $g_t : \mathbb{R}^2 \to \mathbb{R}$ be any continuous function such that for every pair of natural numbers $a, b \in \mathbb{N}$:

1. $g_t(a, b) = t + 2 \quad \text{if } a = 0, b = t + 1,$

2. $g_t(a, b) = a \quad$ otherwise.

Equations 141 - 145 imply:

$$\mathcal{X}_*^{t+1}(S, v) = g_t(\mathcal{X}^t(S, v), \max\{\mathcal{X}^t(s, u) \mid v \sim_G u\}). \tag{146}$$

Using a CS-GNN update as per Equation (11), with aggregation function:

$$\text{agg}_1^t\{\!\{(\mathcal{X}^t(S, v'), e_{v,v'}) \mid v' \sim_G v\}\!\} = \max_{v' \sim_G v} \mathcal{X}^t(S, v'). \tag{147}$$

and update function:

$$f^t(\mathcal{X}^t(S, v), \max_{v' \sim_G v} \mathcal{X}^t(S, v'), \cdot, \cdot, \cdot) = g_t(\mathcal{X}^t(S, v), \max_{v' \sim_G v} \mathcal{X}^t(S, v')) \tag{148}$$

we obtain:

$$\mathcal{X}^{t+1}(S, v) = f^t(\mathcal{X}^t(S, v), \max_{v' \sim_G v} \mathcal{X}^t(S, v'), \cdot, \cdot, \cdot) = \mathcal{X}_*^{t+1}(S, v). \tag{149}$$

This completes the induction step. Now, let $\mathcal{G}$ be a finite family of graphs, whose maximal vertex size is n. We notice that:

$$\mathcal{X}^{\pi_{\text{MD}}}(S, v) = \mathcal{X}_*^n(S, v) - 1, \tag{150}$$

Which implies that there exists a stack of n CS-GNN layers such that:

$$\mathcal{X}^0(S, v) = \mathcal{X}^{\pi_{\text{S}}}(S, v) \quad \text{and} \quad \mathcal{X}^n(S, v) = \mathcal{X}^{\pi_{\text{MD}}}(S, v). \tag{151}$$

This implies:

$$\text{CS-GNN}(\mathcal{T}, \pi_{\text{MD}}) \subseteq \text{CS-GNN}(\mathcal{T}, \pi_{\text{S}}). \tag{152}$$

This concludes the proof. ■

Proposition 29 (Expressivity of Learned Distance Policy) *For any coarsening function $\mathcal{T}(\cdot)$ the following holds:*

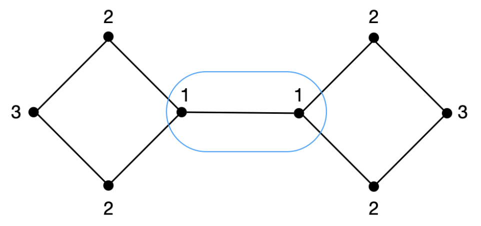
$$\textit{CS-GNN}(\mathcal{T}, \pi_S) \subseteq \textit{CS-GNN}(\mathcal{T}, \pi_{LD}). \tag{33}$$

In addition, for some choices of $\mathcal{T}(\cdot)$ the containment is strict.

Proof First, since we are concerned with input graphs belonging to a finite graph family $\mathcal{G}$, the learned function $\phi(\cdot)$ implemented by an MLP can express any continuous function on $\mathcal{G}$. This follows from Theorem 25 (see the proof of Theorem 11 for details). By choosing $\phi = \min(\cdot)$ in equation 31, it is clear that for any coarsening function $\mathcal{T}(\cdot)$ we have:

$$\text{CS-GNN}(\mathcal{T}, \pi_{\text{S}}) = \text{CS-GNN}(\mathcal{T}, \pi_{\text{MD}}) \subseteq \text{CS-GNN}(\mathcal{T}, \pi_{\text{LD}}). \tag{153}$$

We now construct a coarsening function $\mathcal{T}(\cdot)$ along with two graphs, G and H, and demonstrate that there exists a function in CS-GNN$(\mathcal{T}, \pi_{\text{LD}})$ that can separate G and H. However, every function in CS-GNN$(\mathcal{T}, \pi_{\text{S}})$ cannot separate the two.



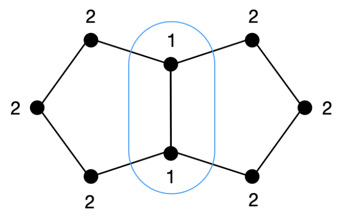


Figure 11: Graphs G and H defined in the proof of Theorem 14. In each graph, the circle marks the single super-node induced by $\mathcal{T}$, while the number next to each node u is the maximal SPD between u and the nodes that compose the super-node.

For an input graph $G = (V, E)$ define:

$$\mathcal{T}(G) = \{\{u \in V \mid \deg_G(u) = 3\}\}. \tag{154}$$

i.e., $\mathcal{T}(\cdot)$ returns a single super-node composed of all nodes with degree 3. Now, define $G = (V_G, E_G)$ as the graph obtained by connecting two cycles of size four by adding an edge between a single node from each cycle. Additionally, define $H = (V_H, E_H)$ as the graph formed by joining two cycles of size five along one of their edges. See Figure 11 for an illustration of the two graphs.

By choosing $\phi = \max(\cdot)$ in equation 31 a quick calculation shows that:

$$\sum_{S \in V_{\mathcal{T}(G)}} \sum_{v \in V_G} \mathcal{X}^{\pi_{\text{LD}}}(S, v) = 16, \tag{155}$$

while:

$$\sum_{S \in V_{\mathcal{T}(H)}} \sum_{v \in V_H} \mathcal{X}^{\pi_{\text{LD}}}(S, v) = 14. \tag{156}$$

Refer to Figure 11 for more details. Observe that:

$$f(G) = \sum_{s \in V_{\mathcal{T}(H)}} \sum_{u \in V_H} \mathcal{X}^{\pi_{\text{LD}}}(S, v) \in \text{CS-GNN}(\mathcal{T}, \pi_{\text{LD}}) \tag{157}$$

Thus it is enough to show that:

$$f(G) = f(H), \quad \forall f \in \text{CS-GNN}(\mathcal{T}, \pi_{\text{S}}). \tag{158}$$

To achieve this, we use the layer update as per Theorem 10, which was demonstrated in Theorem 11 to be equivalent to the general equivariant message passing update in Theorem 5. First, we observe that the graphs G and H are WL-indistinguishable. We then observe that since $|V^{\mathcal{T}}| = 1$, the graphs induced by the adjacency matrices A_G and A_H in Theorem 9 are isomorphic to the original graphs G and H, respectively, and therefore they are also WL-indistinguishable. Additionally, we notice that the graphs induced by the adjacency matrices $A_{\mathcal{T}(G)}$ and $A_{\mathcal{T}(H)}$ in Theorem 9 are both isomorphic to the fully disconnected graph with 8 nodes, making them WL-indistinguishable as well. WWe also observe that there exists a bijection $\sigma : V_G \to V_H$ that maps all nodes of degree 3 in G to all nodes of degree 3 in H. The definition of $\mathcal{T}(\cdot)$ implies that σ is an isomorphism between the adjacency matrices A_{P_i} corresponding to G and H, where $i = 1, 2$. Finally, we notice that for both G, and H, the node feature map induced by π_{S} satisfies:

$$\mathcal{X}^{\pi_{\text{S}}}(S, v) = \deg(v) - 2. \tag{159}$$

This node feature map can be easily implemented by the layer update in definition 10 and so it can be ignored. Since all four graphs corresponding to G that are induced by the adjacency matrices in Theorem 9, are WL-indistinguishable from their counterpart corresponding to H, and equation 25 in definition 10 is an MPNN update, which is incapable of distinguishing graphs that are WL-indistinguishable, we see that equation 158 holds, concluding the proof. ■

Proposition 30 (Node + Size Marking as Invariant Marking) *Given a graph $G = (V, E)$ with node feature vector $X \in \mathbb{R}^{n \times d}$, and a coarsening function $\mathcal{T}(\cdot)$, let $\mathcal{X}^{\pi_{SS}}, \mathcal{X}^{\pi_{inv}}$ be the node feature maps induced by π_{SS} and π_{inv} respectively. Recall that:*

$$\mathcal{X}^{\pi_{SS}}(S, v) = [X_v, b_{\pi_{SS}}(S, v)], \tag{39}$$
$$\mathcal{X}^{\pi_{inv}}(S, v) = [X_v, b_{\pi_{inv}}(S, v)]. \tag{40}$$

The following now holds:

$$b_{\pi_{inv}}(S, v) = OHE(b_{\pi_{SS}}(S, v)) \quad \forall S \in V^{\mathcal{T}},\ \forall v \in V. \tag{41}$$

Here, OHE denotes a one-hot encoder, independent of the choice of both G and $\mathcal{T}$.

Proof Let $G = (V, E)$ be a graph with $V = [n]$, and let $\mathcal{T}(\cdot)$ be a coarsening function. Recall that the maps $b_{\pi_{\text{SS}}}(\cdot, \cdot)$ and $b_{\pi_{\text{inv}}}(\cdot, \cdot)$ are both independent of the connectivity of G and are defined as follows:

$$b_{\pi_{\text{SS}}}(S, v) = \begin{cases} (1, |S|) & v \in S, \\ (0, |S|) & v \notin S. \end{cases} \tag{160}$$

$$b_{\pi_{\text{inv}}}(S, v) = [\mathbf{1}_{\gamma_1}(S, v), \ldots, \mathbf{1}_{\gamma_k}(S, v)]. \tag{161}$$

Here, $v \in [n]$, $S \in \mathcal{T}([n]) \subseteq \mathcal{P}([n])$, $\gamma_1, \ldots, \gamma_k$ is any enumeration of the set of all orbits $(\mathcal{P}([n]) \times [n])/S_n$, and $\mathbf{1}_{\gamma_i}$ denotes the indicator function of orbit γ_i. Since any tuple $(S, v) \in \mathcal{P}([n]) \times [n]$ belongs to exactly one orbit γ_i, we note that the right hand side of Equation (161) is a one-hot encoded vector. Thus, it suffices to show that for every $v, v' \in [n]$ and $S, S' \in \mathcal{P}([n])$, we have:

$$b_{\pi_{\text{SS}}}(S, v) = b_{\pi_{\text{SS}}}(S,' v') \Leftrightarrow b_{\pi_{\text{inv}}}(S, v) = b_{\pi_{\text{inv}}}(S,' v'). \tag{162}$$

This is equivalent to:

$$(\mathcal{P}([n]) \times [n])/S_n = \{\{(S,v) \mid |S| = i, \mathbf{1}_S(v) = j\} \mid i \in [n], j \in \{0,1\}\}\,. \tag{163}$$

Essentially, this means that each orbit corresponds to a choice of the size of s and whether $v \in S$ or not. To conclude the proof, it remains to show that for any two pairs $(S,v), (S,' v') \in \mathcal{P}([n]) \times [n]$, there exists a permutation $\sigma \in S_n$ such that:

$$\sigma \cdot (S,v) = (S,' v') \tag{164}$$

if and only if

$$|S| = |S'| \text{ and } \mathbf{1}_S(v) = \mathbf{1}_{S'}(v'). \tag{165}$$

Assume first that $\sigma \cdot (S,v) = (S',v')$, then $\sigma^{-1}(S) = S'$ and since σ is a bijection, $|S| = |S'|$. In addition $\sigma^{-1}(v) = v'$ thus:

$$v \in S \Leftrightarrow v' = \sigma^{-1}(v) \in \sigma^{-1}(S) = S'. \tag{166}$$

Assume now that:

$$|S| = |S'| \tag{167}$$
$$\mathbf{1}_S(v) = \mathbf{1}_{S'}(v') \tag{168}$$

It follows that for some $r, m \in [n]$:

$$|S \setminus \{v\}| = |S' \setminus \{v'\}| = r \quad \text{and} \quad |[n] \setminus (S \cup \{v\})| = |[n] \setminus (S' \cup \{v'\})| = m \tag{169}$$

Write:

$$\begin{aligned} &S \setminus \{v\} = \{i_1, \ldots, i_r\}, \quad S' \setminus \{v'\} = \{i'_1, \ldots, i'_r\}, \\ &[n] \setminus (S \cup \{v\}) = \{j_1, \ldots j_m\}, \quad [n] \setminus (S' \cup \{v'\}) = \{j'_1, \ldots j'_m\} \end{aligned}$$

and define:

$$\sigma(x) = \begin{cases} v' & x = v \\ i'_l & x = i_l, l \in [r] \\ j'_l & x = j_l, l \in [m] \end{cases} \tag{170}$$

We now have:

$$\sigma \cdot (S,v) = (S',v'). \tag{171}$$

This concludes the proof. ■

H.3. Proofs of Appendix E.1

Proposition 31 (CS-GNN Can Implement MSGNN) *Let $\mathcal{T}(\cdot)$ be the identity coarsening function defined by:*

$$\mathcal{T}(G) = \{\{v\} \mid v \in V\} \quad \forall G = (V, E). \tag{45}$$

The following holds:

$$\text{CS-GNN}(\mathcal{T}, \pi_S) = \text{MSGNN}(\pi_{NM}). \tag{46}$$

Proof Abusing notation, for a given graph $G = (V, E)$ we write $\mathcal{T}(G) = G$, $V^{\mathcal{T}} = V$. First, we observe that:

$$v \in \{u\} \Leftrightarrow u = v, \tag{172}$$

This implies that the initial node feature map $\mathcal{X}^0(u, v)$ induced by π_{S} is equivalent to the standard node marking described in equation 42. Additionally, we note that the pooling procedures for both models, as described in equations 12 and 51, are identical. Therefore, it is sufficient to show that the CS-GNN and MSGNN layer updates described in equations 11 and 43 respectively are also identical. For this purpose, let $\mathcal{X}^t(v, u)$ be a node feature map supported on the set $V \times V$. The inputs to the MSGNN layer are the following:

1. $\mathcal{X}^t(u, v)$.
2. $\mathcal{X}^t(u, u)$.
3. $\mathcal{X}^t(v, v)$.
4. $\text{agg}_1^t\{\{(\mathcal{X}^t(u, v'), e_{v,v'}) \mid v' \sim v\}\}$.
5. $\text{agg}_2^t\{\{(\mathcal{X}^t(u', v), e_{u,u'}) \mid u' \sim u\}\}$.

The inputs to the CS-GNN layer are the following:

1. $\mathcal{X}^t(S, v) \Rightarrow \mathcal{X}^t(u, v)$.
2. $\text{agg}_1^t\{\{(\mathcal{X}^t(S, v'), e_{v,v'}) \mid v' \sim_G v\}\} \Rightarrow \text{agg}_1^t\{\{(\mathcal{X}^t(u, v'), e_{v,v'}) \mid v' \sim v\}\}$.
3. $\text{agg}_2^t\{\{(\mathcal{X}^t(S', v), \tilde{e}_{S,S'}) \mid S' \sim_{G^{\mathcal{T}}} S\}\} \Rightarrow \text{agg}_2^t\{\{(\mathcal{X}^t(u, u'), e_{u,v'}) \mid v' \sim v\}\}$.
4. $\text{agg}_3^t\{\{(\mathcal{X}^t(S', v), z(S, v, S', v)) \mid \forall s' \in V^{\mathcal{T}} \text{s.t. } v \in S'\}\} \Rightarrow \{\{(X^t(v, v), z(u, v, v, v))\}\}$.
5. $\text{agg}_4^t\{\{(\mathcal{X}^t(S, v'), z(S, v, S, v')) \mid \forall u' \in V \text{s.t. } v' \in S\}\} \Rightarrow \{\{(X^t(u, u), z(u, v, u, u))\}\}$.

The terms $z(u, v, v, v)$ and $z(u, v, u, u)$ appearing in the last two input terms of the CS-GNN layer uniquely encode the orbit tuples (u, v, v, v) and (u, v, u, u) belong to respectively. Since these orbits depend solely on whether $u = v$, these values are equivalent to the node marking feature map $\mathcal{X}^0(u, v)$. Therefore, these terms can be ignored. Observing the two lists above, we see that the inputs to both update layers are identical (ignoring the $z(\cdot)$ terms), Thus, as both updates act on these inputs in the same way, the updates themselves are identical. and so

$$\text{MSGNN}(\pi_{\text{NM}}) = \text{CS-GNN}(\mathcal{T}, \pi_{\text{S}}). \tag{173}$$

■

H.4. Proofs of Appendix E.2

Proposition 32 (CS-GNN Is at Least as Expressive as Coarse MPNN) *For any coarsening function $\mathcal{T}(\cdot)$ the following holds:*

$$\textit{MPNN} \subseteq \textit{MPNN}_{+}(\mathcal{T}) \subseteq \textit{CS-GNN}(\mathcal{T}, \pi_S) \tag{54}$$

Proof For convenience, let us first restate the CS-GNN layer update:

$$\begin{aligned}\mathcal{X}^{t+1}(S,v) = f^t\Big(&\mathcal{X}^t(S,v),\\ &\text{agg}_1^t\{\!\{(\mathcal{X}^t(S,v'), e_{v,v'}) \mid v' \sim_G v\}\!\},\\ &\text{agg}_2^t\{\!\{(\mathcal{X}^t(S',v), \tilde{e}_{S,S'}) \mid S' \sim_{G^{\mathcal{T}}} S\}\!\},\\ &\text{agg}_3^t\{\!\{(\mathcal{X}^t(S',v), z(S,v,S',v)) \mid s' \in V^{\mathcal{T}}\text{s.t. } v \in S'\}\!\},\\ &\text{agg}_4^t\{\!\{(\mathcal{X}^t(S,v'), z(S,v,S,v')) \mid u' \in V\text{s.t. } v' \in S\}\!\}\Big),\end{aligned} \tag{11}$$

as well as the MPNN$_+$ layer update:

$$\begin{aligned}\text{For } v \in V: \quad & \mathcal{X}^{t+1}(v) = f_V^t\left(\mathcal{X}^t(v), \text{agg}_1^t\{\!\{(\mathcal{X}^t(v'), e_{v,v'}) \mid v \sim_G v'\}\!\},\right.\\ & \left.\text{agg}_2^t\{\!\{\mathcal{X}^t(S) \mid S \in V^T, v \in S\}\!\}\right),\\ \text{For } S \in V^{\mathcal{T}}: \quad & \mathcal{X}^{t+1}(S) = f_{V^{\mathcal{T}}}^t\left(\mathcal{X}^t(S), \text{agg}_1^t\{\!\{(\mathcal{X}^t(S'), e_{S,S'}) \mid S \sim_{G^T} S'\}\!\},\right.\\ & \left.\text{agg}_2^t\{\!\{\mathcal{X}^t(v) \mid v \in V, v \in S\}\}\!\}\right).\end{aligned} \tag{50}$$

We note that by setting $f_{V^{\mathcal{T}}}^t$ to be a constant zero and choosing f_V^t to be any continuous function that depends only on its first two arguments, the update in equation 50 becomes a standard MPNN layer. This proves:

$$\text{MPNN} \subseteq \text{MPNN}_{+}(T). \tag{174}$$

Next, we prove the following 2 Lemmas:

Lemma 33 *Given a graph $G = (V, E)$ such that $V = [n]$ with node feature vector $X \in \mathbb{R}^{n \times d}$, and a coarsening function $\mathcal{T}(\cdot)$, there exists a CS-GNN$(\mathcal{T}, \pi_S)$ layer such that:*

$$\mathcal{X}^1(S,v) = [0_{d+1}, X_v, 1] = [\tilde{\mathcal{X}}^0(S), \tilde{\mathcal{X}}^0(v)]. \tag{175}$$

Here $[\cdot,\cdot]$ denotes concatenation and $\tilde{\mathcal{X}}^0(\cdot)$ denotes the initial node feature map of the coarsened sum graph $G_{+}^{\mathcal{T}}$.

Lemma 34 *Let $\tilde{\mathcal{X}}^t(\cdot)$ denote the node feature maps of G_{+}^{T} at layers t of a stack of MPNN$_+(\mathcal{T})$ layers. There exists a stack of $t+1$ CS-GNN$(\mathcal{T}, \pi_S)$ layers such that:*

$$\mathcal{X}^{t+1}(S,v) = [\tilde{\mathcal{X}}^t(S), \tilde{\mathcal{X}}^t(v)]. \tag{176}$$

Proof [proof of Lemma 33] Recall that the initial node feature map of CS-GNN($\mathcal{T}, \pi_{\mathrm{S}}$) is given by:

$$\mathcal{X}^0(S, v) = \begin{cases} [X_v, 1] & v \in S \\ [X_v, 0] & v \notin S. \end{cases} \tag{177}$$

In addition, the initial node feature map of MPNN$_+(\mathcal{T})$ is given by:

$$\tilde{X}^0(v) = \begin{cases} [X_v, 1] & v \in V \\ 0_{d+1} & v \in V^{\mathcal{T}}. \end{cases} \tag{178}$$

Thus, we choose a layer update as described in equation 11 with:

$$\mathcal{X}^1(S, v) = f^0(\mathcal{X}^0(S, v), \cdot, \cdot, \cdot, \cdot) = [0_{d+1}, \mathcal{X}^0(S, v)_{1:d}, 1] \tag{179}$$

Here, $f(a, \cdot, \cdot, \cdot)$ denotes that the function depends only on the parameter a, and $X_{a:b}$ indicates that only the coordinates a through b of the vector X are taken. This gives us:

$$\mathcal{X}^1(S, v) = [\tilde{\mathcal{X}}^0(S), \tilde{\mathcal{X}}^0(v)]. \tag{180}$$

■

Proof [proof of Lemma 34] We prove this Lemma by induction on t. We note that Lemma 33 provides the base case $t = 0$. Assume now that for a given stack of $t + 1$ MPNN$_+(\mathcal{T})$ layer updates, with corresponding node feature maps:

$$\tilde{\mathcal{X}}^i : V_+^{\mathcal{T}} \to \mathbb{R}^{d_i} \quad i = 1\ldots, t+1, \tag{181}$$

there exists a stack of $t + 1$ CS-GNN($\mathcal{T}, \pi_{\mathrm{S}}$) layers with node feature maps:

$$\mathcal{X}^i : V^{\mathcal{T}} \times V \to \mathbb{R}^{2d_i} \quad i = 1, \ldots, t+1, \tag{182}$$

such that:

$$\mathcal{X}^{t+1}(S, v) = [\tilde{\mathcal{X}}^t(S), \tilde{\mathcal{X}}^t(v)]. \tag{183}$$

We shall show that there exists a single additional CS-GNN($\mathcal{T}, \pi_{\mathrm{S}}$) layer update such that:

$$\mathcal{X}^{t+2}(S, v) = [\tilde{\mathcal{X}}^{t+1}(S), \tilde{X}^{t+1}(v)]. \tag{184}$$

For that purpose we define the following CS-GNN($\mathcal{T}, \pi_{\mathrm{S}}$) update (abusing notation, the left hand side refers to components of the CS-GNN($\mathcal{T}, \pi_{\mathrm{S}}$) update at layer $t+1$, while the right hand side refers to components of the MPNN$_+(\mathcal{T})$ update at layer t):

$$\begin{aligned} \mathrm{agg}_1^{t+1} &= \mathrm{agg}_1^t|_{1:d_t}, \\ \mathrm{agg}_2^{t+1} &= \mathrm{agg}_1^t|_{d_t+1:2d_t}, \\ \mathrm{agg}_3^{t+1} &= \mathrm{agg}_2^t|_{1:d_t}, \\ \mathrm{agg}_4^{t+1} &= \mathrm{agg}_2^t|_{d_t+1:2d_t}, \end{aligned} \tag{185}$$

$$f^{t+1}(a,b,c,d,e) = [f^t_V(a_{1:d_t}, b, d), f^t_{V^\mathcal{T}}(a_{d_t+1:2d_t}, c, e)]. \tag{186}$$

Here the operation $\text{agg}|_{a:b}$ initially projects all vectors in the input multi-set onto coordinates a through b, and subsequently passes them to the function agg. equations 185 , 186 guarantee that:

$$\begin{aligned}
\mathcal{X}^{t+2}(S,v)_{1:d_{t+1}} &= f^t_V\big(\mathcal{X}^t(S,v)_{1:d_t},\\
&\quad \text{agg}^t_1\{\{(S,v')_{1:d_t} \mid v \sim_G v'\}\},\\
&\quad \text{agg}^t_2\{\{(S',v)_{1:d_t} \mid v \in S'\}\}\big)\\
&= \tilde{X}^{t+1}(v),\\
\mathcal{X}^{t+2}(S,v)_{d_{t+1}+1:2d_{t+1}} &= f^t_{V^\mathcal{T}}\big(\mathcal{X}^t(S,v)_{d_t+1:2d_t},\\
&\quad \text{agg}^t_1\{\{(S',v)_{d_t+1:2d_t} \mid S' \sim_{\mathcal{T}(G)} S\}\},\\
&\quad \text{agg}^t_2\{\{(S,v')_{d_t+1:2d_t} \mid v' \in S\}\}\big)\\
&= \tilde{X}^{t+1}(S).
\end{aligned} \tag{187}$$

This proves the Lemma. ■

Now, for a given finite family of graphs $\mathcal{G}$ and a function $f \in \text{MPNN}_+(\mathcal{T})$, there exists a stack of T $\text{MPNN}_+(\mathcal{T})$ layers such that:

$$f(G) = U\left(\sum_{v \in V_+^\mathcal{T}} \tilde{\mathcal{X}}^T(v)\right) \quad \forall G \in \mathcal{G}. \tag{188}$$

Here, $\tilde{\mathcal{X}}^T : V_+^\mathcal{T} \to \mathbb{R}^{d_T}$ denotes the final node feature map, and U is an MLP. Lemma 34 now tells us that there exists a stack of $T+1$ $\text{CS-GNN}(\mathcal{T}, \pi_\text{S})$ layers such that:

$$\mathcal{X}^{T+1}(S,v) = [\tilde{\mathcal{X}}^T(S), \tilde{\mathcal{X}}^T(v)]. \tag{189}$$

Similarly to Lemma 33, we use one additional layer to pad $\mathcal{X}^{T+1}(S,v)$ as follows:

$$\mathcal{X}^{T+2}(S,v) = [\tilde{\mathcal{X}}^T(S), \tilde{\mathcal{X}}^T(v), 1]. \tag{190}$$

We notice that:

$$\begin{aligned}
\sum_{s \in V^\mathcal{T}} \mathcal{X}^{T+2}(S,v) &= \left[\sum_{S \in V^\mathcal{T}} \tilde{X}^T(S),\ \sum_{S \in V^\mathcal{T}} \tilde{X}^T(v),\ \sum_{S \in V^\mathcal{T}} 1\right]\\
&= \left[\sum_{S \in V^\mathcal{T}} \tilde{X}^T(S),\ |V^\mathcal{T}| \cdot \tilde{X}^T(v),\ |V^\mathcal{T}|\right].
\end{aligned} \tag{191}$$

Thus, in order to get rid of the $|V^{\mathcal{T}}|$ term, We define:

$$\text{MLP}_1(a,b,c) = [a, \frac{1}{c} \cdot b, 1], \quad a,b \in \mathbb{R}^{d_L}, c > 0. \tag{192}$$

We note that since we are restricted to a finite family of input graphs, the use of an MLP in equation 195 can be justified using Theorem 25 (see the proof of Theorem 11 for a detailed explanation).

Equations 191 and 195 imply:

$$\text{MLP}_1\left(\sum_{s \in V^{\mathcal{T}}} \mathcal{X}^{T+2}(S,v)\right) = \left[\sum_{S \in V^{\mathcal{T}}} \tilde{X}^T(S),\ \tilde{X}^T(v),\ 1\right] \tag{193}$$

Thus, similarly to equation 191:

$$\sum_{v \in V} \text{MLP}_1\left(\sum_{S \in V^{\mathcal{T}}} \mathcal{X}^{T+2}(S,v)\right) = \left[|V| \cdot \sum_{S \in V^{\mathcal{T}}} \tilde{X}^T(S),\ \sum_{v \in V} \tilde{X}^T(v),\ |V|\right] \tag{194}$$

And so, in order to get rid of the $|V|$ term, We define:

$$\text{MLP}_2(a,b,c) = U(a \cdot \frac{1}{c} + b, 1), \quad a,b \in \mathbb{R}^{d_T}, c > 0. \tag{195}$$

Thus for all $G \in \mathcal{G}$:

$$\begin{aligned}
&\text{MLP}_2\left(\sum_{v \in V} \text{MLP}_1\left(\sum_{S \in V^{\mathcal{T}}} \mathcal{X}^{T+2}(S,v)\right)\right) \\
&= \text{MLP}_2\left(\left[|V| \cdot \sum_{S \in V^{\mathcal{T}}} \tilde{X}^T(S),\ \sum_{v \in V} \tilde{X}^T(v),\ |V|\right]\right) \\
&= U\left(\sum_{v \in V_+^{\mathcal{T}}} \tilde{X}^T(v)\right) \\
&= f(G).
\end{aligned} \tag{196}$$

and so $f \in \text{CS-GNN}(\mathcal{T}, \pi_{\text{S}})$. This proves:

$$\text{MPNN}_+(T) \subseteq \text{CS-GNN}(\mathcal{T}, \pi_{\text{S}}). \tag{197}$$

■

Proposition 35 (CS-GNN Can Be More Expressive Than MPNN+) *Let $\mathcal{T}(\cdot)$ be the identity coarsening function defined by:*

$$\mathcal{T}(G) = \{\{v\} \mid v \in V\} \quad G = (V, E). \tag{55}$$

The following holds:

$$MPNN = MPNN_{+}(\mathcal{T}). \tag{56}$$

Thus:

$$MPNN_{+}(\mathcal{T}) \subset CS\text{-}GNN(\mathcal{T}, \pi_S), \tag{57}$$

where this containment is strict.

Proof First, using the notation $\tilde{v}$ to mark the single element set $\{v\} \in V^{\mathcal{T}}$, We notice that the MPNN$_{+}(\mathcal{T})$ layer update described in equation 50, becomes:

$$\begin{aligned} \text{For } v \in V: \quad & \mathcal{X}^{t+1}(v) = f_V^t\Big(\mathcal{X}^t(v), \mathcal{X}^t(\tilde{v}), \text{agg}^t\{\!\{(\mathcal{X}^t(v'), e_{v,v'}) \mid v' \sim_G v\}\!\},\Big), \\ \text{For } \tilde{v} \in V^{\mathcal{T}}: \quad & \mathcal{X}^{t+1}(\tilde{v}) = f_{V^{\mathcal{T}}}^t\Big(\mathcal{X}^t(\tilde{v}), \mathcal{X}^t(v), \text{agg}^t\{\!\{(\mathcal{X}^t(\tilde{v'}), e_{\tilde{v},\tilde{v'}}) \mid v \sim_G v'\}\!\}\Big). \end{aligned} \tag{198}$$

Now, for a given finite family of graphs $\mathcal{G}$ and a function $f \in$ MPNN$_{+}(\mathcal{T})$, there exists a stack of T MPNN$_{+}(\mathcal{T})$ layers such that:

$$f(G) = U\left(\sum_{v \in V_{+}^{\mathcal{T}}} \mathcal{X}^T(v)\right) \quad \forall G \in \mathcal{G}. \tag{199}$$

Here, $\mathcal{X}^T : V_{+}^{\mathcal{T}} \to \mathbb{R}^d$ denotes the final node feature map, and U is an MPL. We now prove by induction on t that there exists a stack of t standard MPNN layers, with corresponding node feature map $X^t : V \to \mathbb{R}^{2d_t}$ such that :

$$X^t(v) = [\mathcal{X}^t(v), \mathcal{X}^t(\tilde{v})]. \tag{200}$$

Here, $[\cdot,\cdot]$ stands for concatenation. We assume for simplicity that the input graph G does not have node features, though the proof can be easily adapted for the more general case. We notice that for the base case $t = 0$, equation 49 in definition 18 implies:

$$\mathcal{X}^0(v) = \begin{cases} 1 & v \in V, \\ 0 & v \in V^{\mathcal{T}}. \end{cases} \tag{201}$$

Thus, we define:

$$X^0(v) = (1, 0). \tag{202}$$

This satisfies Equation (200), establishing the base case of the induction. Assume now that Equation (200) holds for some $t \in [T]$. Let agg$^t, f_V^t, f_{V^{\mathcal{T}}}^t$ be the components of layer t, as in equation 198. We define:

$$\tilde{\text{agg}}^t = [\text{agg}^t|_{1:d_t}, \text{agg}^t|_{d_t+1:2d_t}]. \tag{203}$$

Here the operation $\text{agg}|_{a:b}$ initially projects all vectors in the input multi-set onto coordinates a through b, and subsequently passes them to the function agg.

Additionally, let d^* denote the dimension of the output of the function agg^t. We define:

$$\tilde{f}^t(a,b) = \left[f_V^t\left(a|_{1:d_t}, a|_{d_t+1:2d_t}, b|_{1:d^*}\right), f_V^t\left(a|_{d_t+1:2d_t}, a|_{1:d_t}, b|_{d^*+1:2d^*}\right)\right]. \tag{204}$$

Finally, we update our node feature map X^t using a standard MPNN update according to:

$$X^{t+1}(v) = \tilde{f}^l\left(X^t(v), \{\!\{(X^t(v'), e_{v,v'}) \mid v' \sim_G v\}\!\}\right). \tag{205}$$

equations 198, 200 and 205 now guarantee that:

$$X^{t+1}(v) = [\mathcal{X}^t(v), \mathcal{X}^{t+1}(\tilde{v})]. \tag{206}$$

This concludes the inductive proof. We now define:

$$\text{MLP}(x) = U(x|_{1:d_T}) + U(x|_{d_T+1:2d_T}). \tag{207}$$

This gives us:

$$U\Bigg(\sum_{v \in V_+^{\mathcal{T}}} \mathcal{X}^T(v)\Bigg) = \text{MLP}\Bigg(\sum_{v \in V} X^T(v)\Bigg) = f(G). \tag{208}$$

We have thus proven that $f \in \text{MPNN}$ and so:

$$\text{MPNN}_+(\mathcal{N}) \subseteq \text{MPNN}. \tag{209}$$

Combining this result with Proposition 20, we obtain:

$$\text{MPNN} = \text{MPNN}_+(\mathcal{T}). \tag{210}$$

Finally, since Proposition 17 tells us that $\text{CS-GNN}(\mathcal{T}, \pi_{\text{S}})$ has the same implementation power as the maximally expressive node policy subgraph architecture MSGNN, which is proven to be strictly more expressive than the standard MPNN, we have:

$$\text{MPNN}_+(\mathcal{T}) \subset \text{CS-GNN}(\mathcal{T}, \pi_{\text{S}}). \tag{211}$$

■

H.5. Proofs of Appendix F

Lemma 36 (γ (Γ) are orbits) *The sets $\{\gamma^{k^*} : k = 1, \dots, n; * \in \{+, -\}\}$ and $\{\Gamma^{\leftrightarrow;k_1;k_2;k^{\cap};\delta_{same};\delta_{diff}}\}$ are the orbits of S_n on the index space $(\mathcal{P}([n]) \times [n])$ and $(\mathcal{P}([n]) \times [n] \times (\mathcal{P}([n]) \times [n])$, respectively.*

Proof We will prove this lemma for γ. The proof for Γ follows similar reasoning; we also refer the reader to Maron et al. (2018) for a general proof.

We will prove this lemma through the following three steps.

(1). Given indices $(S, i) \in \mathcal{P}([n]) \times [n]$, there exists $\gamma \in (\mathcal{P}([n]) \times [n])_{\sim}$ such that $(S, i) \in \gamma$.

(2). Given indices $(S, i) \in \gamma$, for any $\sigma \in S_n$, it holds that $(\sigma^{-1}(S), \sigma^{-1}(i)) \in \gamma$.

(3). Given $(S, i) \in \gamma$ and $(S', i') \in \gamma$ (the same γ), it holds that there exists a $\sigma \in S_n$ such that $\sigma \cdot (S, i) = (S', i')$.

We prove in what follows.

(1). Given indices $(S, i) \in \mathcal{P}([n]) \times [n]$, w.l.o.g. we assume that $|S| = k$, thus if $i \in S$ ($i \notin S$) it holds that $(S, i) \in \gamma^{k^-}$ $\big((S, i) \in \gamma^{k^+}\big)$, recall Equation (66).

(2). Given indices $(S, i) \in \gamma$, note that any permutation $\sigma \in S_n$ does not change the cardinality of S nor the inclusion (or exclusion) of i in S. Recalling Equation (66), we complete this step.

(3). Given that $(S, i) \in \gamma$ and $(S', i') \in \gamma$, and recalling Equation (66), we note that $|S| = |S'|$ and that either both $i \in S$ and $i' \in S'$, or both $i \notin S$ and $i' \notin S'$.

(3.1). In **(3.1)** we focus on the case where $i \notin S$ and $i' \notin S'$. Let $S = \{i_1, \dots, i_k\}$ and $S' = \{i'_1, \dots, i'_k\}$. Then, we have $(\{i_1, \dots, i_k\}, j)$ and $(\{i'_1, \dots, i'_k\}, j')$. Define $\sigma \in S_n$ such that $\sigma(i_l) = i'_l$ for $l \in [k]$, and $\sigma(j) = j'$. Since $(\{i_1, \dots, i_k\}, j)$ consists of $k + 1$ distinct indices and $(\{i'_1, \dots, i'_k\}, j')$ also consists of $k + 1$ distinct indices, this is a valid $\sigma \in S_n$.

(3.2). Here, we focus on the case where $i \in S$ and $i' \in S'$. This proof is similar to **(3.1)**, but without considering the indices j and j', as they are included in S and S', respectively.

■

Proposition 37 (Basis of Invariant (Equivariant) Layer) *The tensors $\mathbf{B}^{\gamma}$ ($\mathbf{B}^{\Gamma}$) in Equation (67) (Equation (69)) form an orthogonal basis (in the standard inner product) to the solution of Equation (61) (Equation (62)).*

Proof

We prove this proposition for the invariant case. The equivariant case is proved similarly – we also refer the reader for Maron et al. (2018) for a general proof. We will prove this in three steps,

(1). For any $\gamma \in (\mathcal{P}([n]) \times [n])_{\sim}$ it holds that $\mathbf{B}^{\gamma}_{S,i}$ solves Equation (61).

(2). Given a solution $\mathbf{L}$ to Equation (61), it is a linear combination of the basis elements.

(3). We show that the basis vectors are orthogonal and thus linearly independent.

We prove in what follows.

(1). Given $\gamma \in (\mathcal{P}([n]) \times [n])_{\sim}$, we need to show that $\mathbf{B}^{\gamma}_{S,i} = \mathbf{B}^{\gamma}_{\sigma^{-1}(S),\sigma^{-1}(i)}$. Since any $\gamma \in (\mathcal{P}([n]) \times [n])_{\sim}$ is an orbit in the index space (recall Theorem 22), and $\mathbf{B}^{\gamma}_{S,i}$ are indicator vectors of the orbits this always holds.

(2). Given a solution $\mathbf{L}$ to Equation (61), it must hold that $\mathbf{L}_{S,i} = \mathbf{L}_{\sigma^{-1}(S),\sigma^{-1}(i)}$. Since the set $\{\gamma^{k^*} : k = 1, \dots, n; * \in \{+,-\}\}$ corresponds to the orbits in the index space with respect to S_n, $\mathbf{L}$ should have the same values over the index space of these orbits. Let's define these values as α^{γ} for each $\gamma \in \{\gamma^{k^*} : k = 1, \dots, n; * \in \{+,-\}\}$. Thus, we obtain that $\mathbf{L}' = \sum_{\gamma \in (\mathcal{P}([n]) \times [n])_{\sim}} \alpha^{\gamma} \cdot \mathbf{B}^{\gamma}$, since $\mathbf{B}^{\gamma}$ are simply indicator vectors of the orbits. This completes this step.

(3). Once again, since the basis elements are indicator vectors of disjoint orbits we obtain their orthogonality, and thus linearly independent. ■