Equivariant Polynomials for Graph Neural Networks

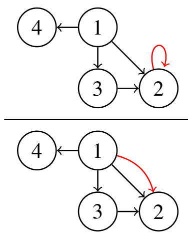
Omri Puny [*1] **Derek Lim** [*2] **Bobak T. Kiani** [*2] **Haggai Maron** [3] **Yaron Lipman** [1 4]

Abstract

Graph Neural Networks (GNN) are inherently limited in their expressive power. Recent seminal works (Xu et al., 2019; Morris et al., 2019b) introduced the Weisfeiler-Lehman (WL) hierarchy as a measure of expressive power. Although this hierarchy has propelled significant advances in GNN analysis and architecture developments, it suffers from several significant limitations. These include a complex definition that lacks direct guidance for model improvement and a WL hierarchy that is too coarse to study current GNNs. This paper introduces an alternative expressive power hierarchy based on the ability of GNNs to calculate equivariant polynomials of a certain degree. As a first step, we provide a full characterization of all equivariant graph polynomials by introducing a concrete basis, significantly generalizing previous results. Each basis element corresponds to a specific multi-graph, and its computation over some graph data input corresponds to a tensor contraction problem. Second, we propose algorithmic tools for evaluating the expressiveness of GNNs using tensor contraction sequences, and calculate the expressive power of popular GNNs. Finally, we enhance the expressivity of common GNN architectures by adding polynomial features or additional operations / aggregations inspired by our theory. These enhanced GNNs demonstrate state-of-the-art results in experiments across multiple graph learning benchmarks.

1. Introduction

In recent years, graph neural networks (GNNs) have become one of the most popular and extensively studied classes of



$P_H(\mathbf{X})_{j_2j_2} = \sum_{j_1,j_3,j_4=1}^{n} \mathbf{X}_{j_1j_4}\mathbf{X}_{j_1j_3}\mathbf{X}_{j_1j_2}\mathbf{X}_{j_3j_2} = \texttt{einsum}(12,13,14,32 \to 22, \mathbf{X},\mathbf{X},\mathbf{X},\mathbf{X})_{j_2j_2}$

$P_H(\mathbf{X})_{j_1j_2} = \sum_{j_3,j_4=1}^{n} \mathbf{X}_{j_1,j_2}\mathbf{X}_{j_1,j_3}\mathbf{X}_{j_1,j_4}\mathbf{X}_{j_3,j_2} = \texttt{einsum}(12,13,14,32 \to 12, \mathbf{X},\mathbf{X},\mathbf{X},\mathbf{X})_{j_1j_2}$

Figure 1. Example of two basis elements of equivariant graph polynomials: node-valued (top), and edge-valued (bottom). Basis elements P_H can be described by tensor contraction networks H (left), corresponding to a matching `einsum` expression.

machine learning models for processing graph-structured data. However, one of the most significant limitations of these architectures is their limited expressive power. In recent years, the Weisfeiler-Lehman (WL) hierarchy has been used to measure the expressive power of GNNs (Morris et al., 2019b; Xu et al., 2019; Morris et al., 2021). The introduction of the WL hierarchy marked an extremely significant step in the graph learning field, as researchers were able to evaluate and compare the expressive power of their architectures, and used higher-order WL tests to motivate the development of new, more powerful architectures.

The WL hierarchy, however, is not an optimal choice for either purpose. First, its definition is rather complex and not intuitive, particularly for $k \geq 3$. One implication is that it is often difficult to analyze WL expressiveness of a particular architecture class. As a result, many models lack a theoretical understanding of their expressive power. A second implication is that WL does not provide practical guidance in the search for more expressive architecture. Lastly, as was noted in recent works (Morris et al., 2022; 2019a), the WL test appears to be too coarse to be used to evaluate the expressive power of current graph models. As an example, many architectures (e.g., (Frasca et al., 2022)) are strictly more powerful than 2-WL and bounded by 3-WL, and there is no clear way to compare them.

The goal of this paper is to offer an alternative expressive power hierarchy, which we call *polynomial expressiveness* that mitigates the limitations of the WL hierarchy. Our proposed hierarchy relies on the concept of graph polynomials, which are, for graphs with n nodes, polynomial functions $P : \mathbb{R}^{n^2} \to \mathbb{R}^{n^2}$ that are also permutation equivariant —

[*]Equal contribution [1]Weizmann Institute of Science [2]MIT CSAIL [3]NVIDIA Research [4]Meta AI Research Centre for Artificial Intelligence. Correspondence to: Omri Puny <omri.puny@weizmann.ac.il>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

that is, well defined on graph data. The polynomial expressiveness hierarchy is based on a natural and simple idea — the ability of GNNs to compute or approximate equivariant graph polynomials up to a certain degree.

This paper provides a number of theoretical and algorithmic contributions aimed at defining the polynomial hierarchy, providing tools to analyze the polynomial expressive power of GNNs, and demonstrating how this analysis can suggest practical improvements in existing models that give state-of-the-art performance in GNN benchmarks.

First, while some polynomial functions were used in GNNs in the past (Maron et al., 2019; Chen et al., 2019b; Azizian & Lelarge, 2021), a complete characterization of the space of polynomials is lacking. In this paper, we provide the first characterization of graph polynomials with arbitrary degrees. In particular, we propose a basis for this vector space of polynomials, where each basis polynomial P_H of degree d corresponds to a specific multi-graph H with d edges. This characterization provides a significant generalization of known results, such as the basis of constant and linear equivariant functions on graphs (Maron et al., 2018). Furthermore, this graphical representation H can be viewed as a type of a tensor network, which provides a concrete way to compute those polynomials by performing a series of tensor (node) contractions. This is illustrated in Figure 1.

As a second contribution, we propose tools for measuring polynomial expressiveness of graph models and placing them in the hierarchy. This is accomplished by analyzing tensor networks using standard contraction operators, similar to those found in Einstein summation (`einsum`) algorithms. Using these, we analyze two popular graph models: Message Passing Neural Networks (MPNNs) and Provably Powerful Graph Networks (PPGNs). This is done by first studying the polynomial expressive power of prototypical versions of these algorithms, which we define.

Our third contribution demonstrates how to improve MPNN and PPGN by using the polynomial hierarchy. Specifically, we identify polynomial basis elements that are not computable by existing graph architectures and add those polynomial basis elements to the model as feature layers. Also, we add two simple operations to the PPGN architecture (matrix transpose and diagonal / off-diagonal MLPs) to achieve the power of a Prototypical edge-based graph model. After precomputing the polynomial features, we achieve strictly better than 3-WL expressive power while only requiring $O(n^2)$ memory — to the best of our knowledge this is the first equivariant model to achieve this. We demonstrate that these additions result in state-of-the-art performance across a wide variety of datasets.

2. Equivariant Graph Polynomials

We represent a graph with n nodes as a matrix $\mathbf{X} \in \mathbb{R}^{n^2}$, where edge values are stored at off-diagonal entries, $\mathbf{X}_{ij}$, $i \neq j, i, j \in [n] = \{1, 2, \ldots, n\}$, and node values are stored at diagonal entries $\mathbf{X}_{ii}$, $i \in [n]$.

An *equivariant graph polynomial* is a matrix polynomial map $P : \mathbb{R}^{n^2} \to \mathbb{R}^{n^2}$ that is also equivariant to node permutations. More precisely, P is a polynomial map if each of its entries, $P(\mathbf{X})_{ij}$, $i, j \in [n]$, is a polynomial in the inputs $\mathbf{X}_{rs}$, $r, s \in [n]$. P is equivariant if it satisfies

$$P(g \cdot \mathbf{X}) = g \cdot P(\mathbf{X}), \tag{1}$$

for all permutations $g \in S_n$, where S_n denotes the permutation group on $[n]$, and g acts on a matrix $\mathbf{Y}$ as usual by

$$(g \cdot \mathbf{Y})_{ij} = \mathbf{Y}_{g^{-1}(i), g^{-1}(j)}. \tag{2}$$

2.1. P_H: Basis for equivariant graph polynomials

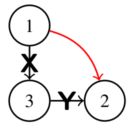
We next provide a full characterization of equivariant graph polynomials by enumerating a particular basis, denoted P_H. In later sections we use this basis to analyze expressive properties of graph models and improve expressiveness of existing GNNs.

The basis elements P_H of degree d equivariant graph polynomials are enumerated from non-isomorphic *directed multi-graphs*, $H = (V, E, (a, b))$, where $V = [m]$ is the node set; $E = \{(r_1, s_1), \ldots, (r_d, s_d)\}$, $r_i, s_i \in [m]$, the edge set, where parallel edges and self-loops are allowed; and $a, b \in V$ is a pair of not necessarily distinct nodes representing the output dimension. The pair (a, b) will be marked in our graphical notation as a red edge.

Defining the basis P_H will be facilitated by the use of Einstein summation operator defined next. Note that the multi-graph H can be represented as the following string that encodes both its list of edges and the single red edge: $H \cong {}'r_1 s_1, \ldots, r_d s_d \to ab'$. The `einsum` operator is:

$$\begin{aligned}
&\texttt{einsum}(H, \mathbf{X}^1, \ldots, \mathbf{X}^d)_{i_a, i_b} = \\
&\texttt{einsum}(r_1 s_1, \ldots, r_d s_d \to ab, \mathbf{X}^1, \ldots, \mathbf{X}^d)_{i_a, i_b} = \\
&\sum_{\substack{j_1, \ldots j_m \in [n] \\ j_a = i_a, j_b = i_b}} \mathbf{X}^1_{j_{r_1}, j_{s_1}} \cdots \mathbf{X}^d_{j_{r_d}, j_{s_d}}
\end{aligned}$$

Figure 2 shows how matrix multiplication can be defined using a corresponding multigraph H and Einstein summation. Such multigraphs span the basis of equivariant polynomials:



$\texttt{einsum}(H, \mathbf{X}, \mathbf{Y})_{j_1 j_2} =$

$\texttt{einsum}(13, 32 \rightarrow 12, \mathbf{X}, \mathbf{Y}) =$

$\sum_{j_3=1}^{n} \mathbf{X}_{j_1 j_3} \mathbf{Y}_{j_3 j_2} = (\mathbf{XY})_{j_1 j_2}$

Figure 2. Example of matrix multiplication, $\mathbf{XY}$. Computation defined by a multigraph H and Einstein summation.

Theorem 2.1 (Graph equivariant basis). *A basis for all equivariant graph polynomials $P : \mathbb{R}^{n^2} \rightarrow \mathbb{R}^{n^2}$ of degree $\leq d$ is enumerated by directed multigraphs $H = (V, E, (a, b))$, where $|V| \leq \min\{n, 2 + 2d\}$, $|E| \leq d$, and $V \setminus \{a, b\}$ does not contain isolated nodes. The polynomial basis elements corresponding to H are*

$$P_H(\mathbf{X}) = \texttt{einsum}(H, \overbrace{\mathbf{X}, \ldots, \mathbf{X}}^{d\ times}). \tag{3}$$

An explicit formula for P_H can be achieved by plugging in the definition of $\texttt{einsum}$ and equation 3:

$$P_H(\mathbf{X})_{i_a, i_b} = \sum_{\substack{j_1, \ldots j_m \in [n] \\ j_a = i_a, j_b = i_b}} \prod_{(r,s) \in E} \mathbf{X}_{j_r, j_s}. \tag{4}$$

Figure 1 depicts an example of graph equivariant basis elements P_H corresponding to two particular multigraphs H. Note that a repeated pair (a, a) in H leads to a *node-valued* equivariant polynomial, while a distinct pair (a, b), $a \neq b$ leads to an *edge-valued* equivariant polynomial. Furthermore, we make the convention that if E is empty then $\prod_{(r,s) \in E} \mathbf{X}_{i_r, i_s} = 1$. The number of such polynomials increases exponentially with the degree of the polynomial; the first few counts of degree d equivariant graph polynomials are 2 ($d = 0$), 15 ($d = 1$), 117 ($d = 2$), 877 ($d = 3$), 6719 ($d = 4$), ... Further details and proofs of these sequences are provided in Appendix I. The full proof of Theorem 2.1 is provided in Appendix B.

Proof idea for Theorem 2.1. Since the set of monomials form a basis of all (not necessarily invariant) polynomials $P : \mathbb{R}^{n^2} \rightarrow \mathbb{R}^{n^2}$, we can project them onto the space of equivariant polynomials via the symmetrization (Reynolds) operator to form a basis for the equivariant polynomials. This projection operation will group the monomials into orbits that form the equivariant basis.

To find these orbits, the basic idea is to consider the monomials in the input variables $\{\mathbf{X}_{i,j} : i, j \in [n]\}$ and an additional variable $\{\delta_{i,j} : i, j \in [n]\}$ to denote the possible output entries of the equivariant map. Any given monomial $M(\mathbf{X}, \delta^{a,b})$ takes the form

$$M(\mathbf{X}, \delta^{a,b})_{i,j} = \delta_{i,j}^{a,b} \prod_{r,s=1}^{n} \mathbf{X}_{r,s}^{\mathbf{A}_{r,s}}, \tag{5}$$

where $\mathbf{A} \in \mathbb{N}_0^{n^2}$, $\mathbb{N}_0 = \{0, 1, \ldots\}$ is the matrix of powers, and $\delta_{i,j}^{a,b} = 1$ if $a = i$, $b = j$, and $\delta_{i,j}^{a,b} = 0$ otherwise. A natural way to encode these monomials is with *labeled* multi-graphs $H = (V, E, (a, b))$, where $V = [n]$, E is defined by the adjacency matrix $\mathbf{A}$, and (a, b) is a special (red) edge. We therefore denote $M = M_H$.

These monomials can be projected onto equivariant polynomials via the Reynolds operator that takes the form,

$$\begin{aligned} Q_H(\mathbf{X}) &= \sum_{g \in S_n} g \cdot M_H(g^{-1} \cdot \mathbf{X}, \delta^{g(a), g(b)}) \\ &= \sum_{g \in S_n} M_{g \cdot H}(\mathbf{X}, \delta^{a,b}), \end{aligned} \tag{6}$$

where the action of $g \in S_n$ on the multi-graph H is defined (rather naturally) as node relabeling of H using the permutation g.

From the above, we note: (i) Q_H sums all monomials with multi-graphs in the orbit of H, namely $[H] = \{g \cdot H | g \in S_n\}$. This shows that, in contrast to M_H, Q_H is represented by an *unlabeled* multi-graph H and enumerated by *non-isomorphic* multi-graphs H. (ii) Since the symmetrization is a projection operator, any equivariant polynomial is spanned by Q_H. (iii) Since each Q_H is a sum of M_H belonging to a different orbit, and since orbits are disjoint, the set $\{Q_H\}$ for non-isomorphic H is linearly independent. These three points establish that $\{Q_H\}$ for non-isomorphic multi-graphs H is a basis of equivariant graph polynomials.

Noting that $Q_H(\mathbf{X})_{ij}$ includes only terms for which $\delta^{g(a), g(b)} = \delta^{i,j}$, the explicit form below can be derived:

$$Q_H(\mathbf{X})_{i_a, i_b} = \sum_{\substack{j_1 \neq \ldots \neq j_m \in [n] \\ j_a = i_a, j_b = i_b}} \prod_{(r,s) \in E} \mathbf{X}_{j_r, j_s}. \tag{7}$$

Q_H is similar to P_H in equation 4, except we only sum over non-repeated indices. The proof in Appendix B shows that P_H is also a basis for such equivariant polynomials.

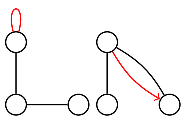


Figure 4. Simple H corresponding to simple graph data.

Simple graphs. It is often the case that the input data $\mathbf{X}$ is restricted to some subdomain of $\mathbb{R}^{n^2}$, e.g., symmetric $0/1$ matrices with diagonal entries set to zero correspond to simple graph data. In such cases, polynomials P_H that correspond to different multi-graphs H can coincide, resulting in a smaller basis. For simple graph data $\mathbf{X}$, existence of self loops in H would result in $P_H(\mathbf{X}) = 0$, parallel edges in H can be replaced with single edges without changing the value of $P_H(\mathbf{X})$, and since the direction of black edges in H do not change the value of $P_H(\mathbf{X})$ we can consider only *undirected* multi-graphs H. That is, for simple graph data it is enough to

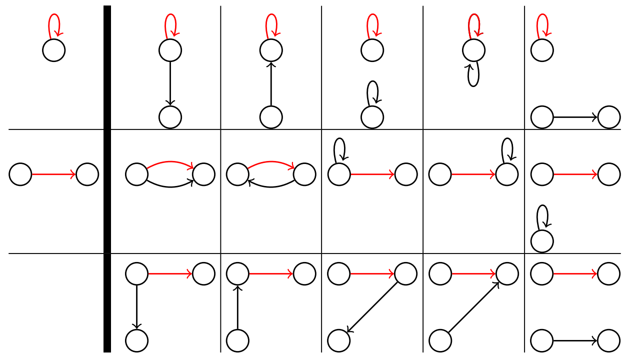


Figure 3. Basis of equivariant constant (left of bold line) and linear (right of bold line) graph polynomials.

consider simple graphs H (ignoring the red edge). Figure 4 shows two examples of H for simple graph data.

Example: linear basis. Employing Theorem 2.1 for the $d = 0, 1$ case reproduces the graph equivariant constant and linear functions from Maron et al. (2018). Figure 3 depicts the graphical enumeration of the 2 constant and 15 linear basis elements.

Computing $P_H(\mathbf{X})$ with tensor contractions. A useful observation for the graph model analysis performed later is that computing $P_H(\mathbf{X})$ is equivalent to a tensor contraction guided by H. Similarly to `einsum`, computing $P_H(\mathbf{X})$ can be done iteratively in multiple ways by finding a sequence of contraction paths for H where we start with each edge of H endowed with $\mathbf{X}$ and our end goal is to have a single black edge aligned with the red edge. Figure 5 provides an example of computing a 4th degree polynomial,

$$P_H(\mathbf{X})_{j_1,j_2} = \sum_{j_3,j_4=1}^{n} \mathbf{X}_{j_1 j_4}\mathbf{X}_{j_1 j_3}\mathbf{X}_{j_3 j_2}\mathbf{X}_{j_1 j_2}.$$

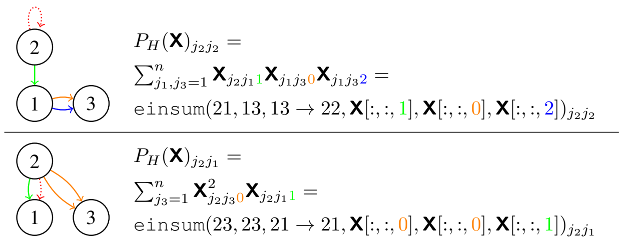
The computation of the polynomial is decomposed to a sequence of operations, portrayed in the figure. Each step is labeled by the tensor contraction operation and the corresponding explicit computation. Nodes colored in gray correspond to contracted nodes whose indices are summed in the `einsum`. The output of each contraction step is represented by a new black edge (labeled as $\mathbf{Y}$, $\mathbf{Z}$ and $\mathbf{R}$ in our example).

2.2. Generalizations and discussion

Invariant graph polynomials. This approach also gives a basis for the invariant polynomials $P : \mathbb{R}^{n^2} \to \mathbb{R}$. In this case, we let H be a directed multigraph without a red edge, and define $P_H(\mathbf{X}) = \sum_{j_1,\dots,j_m\in[n]} \prod_{r,s\in E(H)} \mathbf{X}_{j_r,j_s}$. Computing P_H then corresponds to contracting H to the trivial graph (with no nodes or edges). Our equivariant basis is a generalization of previous work, which used invariant polynomials analogous to P_H or the alternative basis Q_H to study properties of graphs (Thiéry, 2000; Lovász, 2012; Komiske et al., 2018).

Subgraph counting. The previous work on invariant polynomials mentioned above as well as our proof of Theorem 2.1 suggest Q_H (see equation 7) as another basis of equivariant graph polynomials. In Appendix G, we show that when applied to binary input $\mathbf{X} \in \{0,1\}^{n\times n}$, Q_H performs subgraph counting; essentially, $Q_H(\mathbf{X})_{i_a,i_b}$ is proportional to the number of subgraphs of $\mathbf{X}$ isomorphic to H such that i_a is mapped to a and i_b is mapped to b. This Q_H basis is interpretable, but does not lend itself to efficient vectorized computation or the tensor contraction perspective that the P_H basis has.

Equivariant polynomials for attributed graphs. Our basis for equivariant graph polynomials can be extended to cover the more general case of *attributed graphs* (i.e., graphs with $\mathbb{R}^f$ features attached to nodes and/or edges), $P : \mathbb{R}^{n^2\times f} \to \mathbb{R}^{n^2}$. A similar basis to P_H can be used in this case, as described in Appendix F. Figure 6 visualizes this extension.



$P_H(\mathbf{X})_{j_2 j_2} =$
$\sum_{j_1,j_3=1}^{n} \mathbf{X}_{j_2 j_1 1}\mathbf{X}_{j_1 j_3 0}\mathbf{X}_{j_1 j_3 2} =$
$\texttt{einsum}(21, 13, 13 \to 22, \mathbf{X}[:,:,1], \mathbf{X}[:,:,0], \mathbf{X}[:,:,2])_{j_2 j_2}$

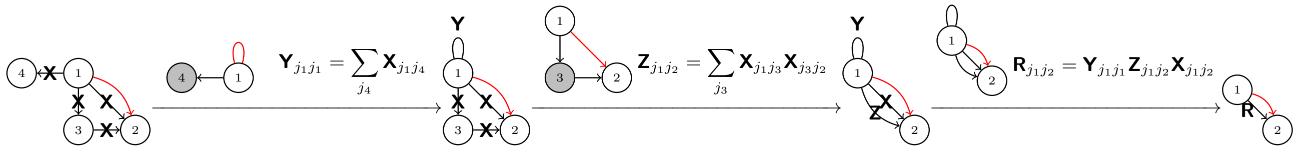
$P_H(\mathbf{X})_{j_2 j_1} =$
$\sum_{j_3=1}^{n} \mathbf{X}^2_{j_2 j_3 0}\mathbf{X}_{j_2 j_1 1} =$
$\texttt{einsum}(23, 23, 21 \to 21, \mathbf{X}[:,:,0], \mathbf{X}[:,:,0], \mathbf{X}[:,:,1])_{j_2 j_1}$

Figure 6. Basis elements of equivariant polynomials from $\mathbb{R}^{n^2\times 3} \to \mathbb{R}^{n^2}$. The output edge is indicated by a dotted red edge and the feature dimension is indexed by three colors for index zero (orange), index one (green), and index two (blue).

3. Expressive Power of Graph Models

In this section we evaluate the expressive power of equivariant graph models from the new, yet natural hierarchy arising from equivariant graph polynomials. By graph model, $\mathcal{F} = \{F\}$, we mean any collection of equivariant functions $F : \mathbb{R}^{n^2} \to \mathbb{R}^{n^k}$, where $k = 1$ corresponds to a family of node-valued functions, $F(\mathbf{X}) \in \mathbb{R}^n$, and $k = 2$ to node and edge-valued functions, $F(\mathbf{X}) \in \mathbb{R}^{n^2}$. For expositional simplicity we focus on graph data $\mathbf{X}$ representing simple graphs, but note that the general graph data case can be analysed using similar methods. We will use two notions of polynomial expressiveness: exact and approximate. The exact case is used for analyzing Prototypical graph models, whereas the approximate case is used for analyzing practical graph models.

Definition 3.1. A graph model $\mathcal{F}$ is d node/edge polynomial *exact* if it can compute all the degree d polynomial basis elements $P_H(\mathbf{X})$ for every simple graph $\mathbf{X}$.



$\mathbf{Y}_{j_1j_1} = \sum_{j_4} \mathbf{X}_{j_1j_4}$
$\mathbf{Z}_{j_1j_2} = \sum_{j_3} \mathbf{X}_{j_1j_3}\mathbf{X}_{j_3j_2}$
$\mathbf{R}_{j_1j_2} = \mathbf{Y}_{j_1j_1}\mathbf{Z}_{j_1j_2}\mathbf{X}_{j_1j_2}$

Figure 5. Computation of $P_H(\mathbf{X})$ with a sequence of tensor contractions: The polynomial $P_H(\mathbf{X})$ (left-most) is computed when a single black edge parallel to the red edge is left (right-most); above each arrow is the tensor contraction applied (contracted nodes are in gray).

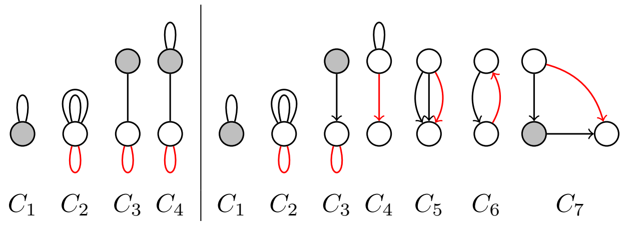


Figure 7. Prototypical node-based (left) and edge-based (right) graph models' contraction banks. Gray nodes indicate nodes that are contracted. Explicit formula of each element can be found in Appendix D.

Definition 3.2. A graph model $\mathcal{F}$ is d node/edge polynomial *expressive* if for arbitrary $\epsilon > 0$ and degree d polynomial basis element $P_H(\mathbf{X})$ there exists an $F \in \mathcal{F}$ such that $\max_{\mathbf{X} \text{ simple}} |F(\mathbf{X}) - P(\mathbf{X})| < \epsilon$.

As a primary application of the equivariant graph basis P_H, we develop tools here for analyzing the polynomial expressive power of graph models $\mathcal{F}$. We define *Prototypical* graph models which provide a structure to analyze or improve existing popular GNNs such as MPNN (Gilmer et al., 2017) and PPGN (Maron et al., 2019).

3.1. Prototypical graph models

We consider graph computation models, $\mathcal{F}_\mathcal{B}$, that are finite sequences of tensor contractions taken from a bank of primitive contractions $\mathcal{B}$.

$$\mathcal{F}_\mathcal{B} = \left\{ C_{i_1} C_{i_2} \cdots C_{i_\ell} \,|C_{i_j} \in \mathcal{B} \right\}, \tag{8}$$

where the bank $\mathcal{B} = \{C_1, \dots, C_k\}$ consists of multi-graphs $C_i = (V_i, E_i, (a_i, b_i))$, each representing a different primitive tensor contraction. A model $\mathcal{F}_\mathcal{B}$ can compute a polynomial $P_H(\mathbf{X}) = \mathtt{einsum}(H, \mathbf{X}, \dots, \mathbf{X})$ if it can contract H to the red edge by applying a finite sequence of contractions from its bank. If there exists such a sequence then P_H is deemed computable by $\mathcal{F}_\mathcal{B}$, otherwise it is not computable by $\mathcal{F}_\mathcal{B}$. For example, the model with the bank presented in Figure 8 can compute P_H in Figure 5; removing any element from this model, will make $P_H(\mathbf{X})$ non-computable. We recap:

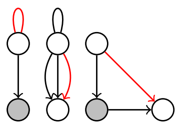


Figure 8. A bank $\mathcal{B}$ of a model $\mathcal{F}_\mathcal{B}$ that can compute the example in Figure 5.

Definition 3.3. The polynomial P_H is *computable by* $\mathcal{F}$ iff there exists a sequence of tensor contractions from $\mathcal{F}$ that computes $P_H(\mathbf{X}) = \mathtt{einsum}(H, \mathbf{X}, \dots, \mathbf{X})$.

We henceforth focus on two Prototypical models: the node-based model $\mathcal{F}_n$ and edge-based model $\mathcal{F}_e$. Their respective contraction banks are depicted in Figure 7, each motivated by the desire to achieve polynomial exactness (see Definition 3.1) and contract multi-graphs H where a member of the bank can always be used to contract nodes with up to N neighbors. Taking $N = 1$ results in the node-based bank in Figure 7 (left), and $N = 2$ in the edge-based bank in Figure 7 (right). These choices are not unique — other contraction banks can satisfy these requirements.

Lemma 3.4. *$\mathcal{F}_n$ (for simple graphs) and $\mathcal{F}_e$ (for general graphs) can always contract a node in H iff its number of neighbors is at-most* 1 *and* 2*, respectively.*

A few comments are in order. Node-based contractions can only add self-edges during the contraction process (i.e., new node-valued data) thus requiring only $O(n)$ additional memory to perform computation. Further note that since we assume simple graph data, H is also a simple graph, and no directed edges (i.e., non-symmetric intermediate tensors) are created during contraction. Contraction banks with undirected graphs suffice in this setting. We later show that the node-based model acts analogously to message-passing. The edge-based model targets exactness over both node and edge valued polynomial. It generates new edges that can be directed even if $\mathbf{X}$ is simple, and thus includes directed contractions in its bank. The edge-based model will later be connected to the graph models PPGN (Maron et al., 2019) and Ring-GNN (Chen et al., 2019b). We later show that the node-based model is 1-WL expressive and the edge-based model is 3-WL expressive.

Deciding computability of $P_H(\mathbf{X})$ with $\mathcal{F}_\mathcal{B}$. A key component in analyzing the expressive power of a Prototypical model is determining which polynomials $P_H(\mathbf{X})$ can be computed with $\mathcal{F}$, given H and $\mathbf{X}$ encoding simple graph data. A naive algorithm traversing all possible enumerations of nodes in H and their contractions would lead to a combinatorial explosion that is too costly — especially since this procedure needs to be repeated for a large number of polynomials. Here, we show that at least for contraction banks $\mathcal{F}_n$ and $\mathcal{F}_e$, Algorithm 1 is a linear time (in $|V|, |E|$),

Algorithm 1 Decide if P_H is computable by $\mathcal{F}_n$ or $\mathcal{F}_e$.

```
**Input:** contraction bank $\mathcal{F} \in \{\mathcal{F}_n, \mathcal{F}_e\}$, multi-graph $H$
set $d = 1$ for $\mathcal{F} = \mathcal{F}_n$, or $d = 2$ for $\mathcal{F} = \mathcal{F}_e$.
set $doneContracting = false$
while not $doneContracting$ do
    if exists a node in $V \setminus \{a, b\}$ with $\leq d$ neighbors then
        contract the node using $\mathcal{F}$
    else
        $doneContracting = true$
    end if
end while
if $V \setminus \{a, b\}$ is empty then
    return computable
else
    return non-computable
end if
```

greedy algorithm for deciding computability of a given polynomial. Algorithm 1 finds a sequence of contractions using the greedy step until no more nodes are left to contract. That is, it terminates when all nodes, aside from a, b, have more than 1 or 2 neighbors for $\mathcal{F}_n$ or $\mathcal{F}_e$, respectively. If it terminates with just $\{a, b\}$ as vertices it deems P_H computable and otherwise it deems P_H non-computable. To show correctness of this algorithm we prove:

Theorem 3.5. *Let H be some multi-graph and $\mathcal{F}_{\mathcal{B}} \in \{\mathcal{F}_n, \mathcal{F}_e\}$. Further, let H' be the multi-graph resulting after contracting a single node in H using one or more operations from $\mathcal{B}$ to H. Then, H is $\mathcal{F}_{\mathcal{B}}$-computable iff H' is $\mathcal{F}_{\mathcal{B}}$-computable.*

To verify the correctness of this procedure, note that the algorithm has to terminate after at most $|V| - |\{a, b\}|$ node contractions. Now consider two cases: if the algorithm terminates successfully, it must have found a sequence of tensor contractions to compute $P_H(\mathbf{X})$. If it terminates unsuccessfully, the theorem implies its last network H' is computable iff the input network H is computable. Now since there is *no* further node contraction possible to do in H' using operations from $\mathcal{B}$ it is not computable by definition, making H not computable.

Polynomial exactness. To compute the d polynomial exactness (see Definition 3.1) for the node-based and edge-based Prototypical graph models we enumerate all non-isomorphic simple graphs H with up to d edges and one red edge and run Algorithm 1 on each H. This reveals that the node-based model $\mathcal{F}_n$ is 2-node-polynomial-exact, while the edge-based model $\mathcal{F}_e$ is 5-node-polynomial-exact and 4-edge-polynomial-exact. See Figure 9 for the lowest degree polynomials, represented by H with the smallest number of edges, that are non-computable for $\mathcal{F}_n$ and $\mathcal{F}_e$.

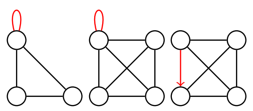


Figure 9. The smallest non-computable H for: $\mathcal{F}_n$ (left: node-valued), and $\mathcal{F}_e$ (middle: node-valued; right: edge-valued).

k-WL expressive power. For simple graphs, there is a natural connection between our Prototypical graph models and the k-WL graph isomorphism tests. This stems from a result of Dvořák (2010); Dell et al. (2018), which states that two graphs $\mathbf{X}^{(1)}$ and $\mathbf{X}^{(2)}$ are k-FWL equivalent if and only if $\hom(H, \mathbf{X}^{(1)}) = \hom(H, \mathbf{X}^{(2)})$ for all H of tree-width at most k. Recall that $\hom(H, \mathbf{X})$ is the number of homomorphisms from H to $\mathbf{X}$ (where H has no red edges), which we show is equivalent to the output of $P_H(\mathbf{X})$ for the invariant polynomial P_H in Appendix G. By showing that the Prototypical node-based graph model can contract any H of tree-width 1 (and no others), and that the Prototypical edge-based graph model can contract any H of tree-width at most 2 (and no others), we thus have the following result.

Proposition 3.6. *The Prototypical node-based model can distinguish a pair of simple graphs if and only if 1-WL can. The Prototypical edge-based model can distinguish a pair of simple graphs if and only if 3-WL / 2-FWL can.*

This proposition indicates that $\mathcal{F}_n$ and $\mathcal{F}_e$ can contract an invariant polynomial, represented by a graph H, if and only if the tree-width of H is 1 and 2, respectively. Therefore computability of *invariant* P_H can be decided by checking the tree width of H. We leave generalizing this approach to equivariant P_H to future work.

3.2. GNNs and their expressive power

In this section we turn our attention to commonly used graph neural networks (GNNs), and provide lower bounds on their polynomial expressive power as in Definition 3.2. The Message Passing Neural Network (MPNN) we consider consists of layers of the form

$$\mathbf{Y}^{(k+1)} = \mathtt{m}\left[\mathbf{X}\mathbf{Y}^{(k)}, \mathbf{1}\mathbf{1}^T\mathbf{Y}^{(k)}, \mathbf{Y}^{(k)}\right], \quad (9)$$

where the intermediate tensor variables are $\mathbf{Y} \in \mathbb{R}^{n \times d}$, $\mathbf{1} \in \mathbb{R}^n$ is the vector of all ones, $\mathbf{Y}^{(0)} = \mathbf{1}$, brackets indicate concatenation in the feature dimension, and $\mathtt{m}$ means a multilayer perceptron (MLP) applied to the feature dimension.

As an application of the Prototypical edge-based model, we propose and implement a new model architecture (PPGN++) that is at least as expressive as the full versions of PPGN/Ring-GNN (Maron et al., 2019; Chen et al., 2019b) (which incorporate all 15 linear basis elements), but is more efficient — PPGN++ uses a smaller number of "primitive" operations than the full PPGN/Ring-GNN, and does not need parameters for each linear basis element:

$$\mathbf{Z}^{(k+1)} = \bar{\mathtt{m}}_3\left[\bar{\mathtt{m}}_1\left[\mathbf{Z}^{(k)}, \mathbf{Z}^{(k)T}\right] \circledast \bar{\mathtt{m}}_2(\mathbf{Z}^{(k)}), \mathbf{Z}^{(k)}\right], \quad (10)$$

where $\mathbf{Z} \in \mathbb{R}^{n^2 \times d}$ are intermediate tensor variables, $\mathbf{Z}^{(0)} = \mathbf{X}$, $\circledast$ performs matrix multiplication of matching features, and $\bar{\mathrm{m}}_i$, for $i \in [3]$, is a *pair* of MLPs: one applied to all diagonal and off-diagonal features of $\mathbf{Z}$ separately.

We lower bound the polynomial expressiveness of MPNN and PPGN++ in the next theorem:

Theorem 3.7. *PPGN++ is at-least* 4 *edge polynomial expressive and* 5 *node polynomial expressive. MPNN is at-least* 2 *node polynomial expressive.*

Proof idea. We prove the theorem in two steps. First, showing that an MPNN or PPGN++ layer can approximate any primitive contraction $C \in \mathcal{B}$ from the bank of the Prototypical node based $\mathcal{F}_n$ or edge based $\mathcal{F}_e$ models, respectively. Second, we use a lemma from Lim et al. (2022) stating that layer-wise universality leads to overall universality. The complete proof is in Appendix D.

Comparison of PPGN++ and PPGN. Proposition 3.6 and the proof of Theorem 3.7 indicate that PPGN++ is 3-WL/2-FWL expressive for simple graphs, similarly to PPGN (Maron et al., 2019). However, the following proposition shows that there is a significant expressiveness gap between PPGN and PPGN++ in approximating equivariant polynomials.

Proposition 3.8. *PPGN is at most* 0 *edge polynomial expressive.*

Proof idea. We claim that PPGN is at most 0 edge polynomial expressive by proving that there exist a linear polynomial (the transpose operator) that cannot be approximated by PPGN. The proof shows that for an input tensor of the form

$$\mathbf{Z} = \begin{bmatrix} a & a \\ b & b \end{bmatrix}, \quad a, b \in \mathbb{R},$$

a PPGN model cannot approximate the transpose operator $P_H(\mathbf{Z}) = \mathbf{Z}^T$ since it preserves the row structure. The complete proof is in Appendix E.

3.3. Increasing the expressive power of GNNs

Theorem 3.7 proves a lower bound on the polynomial expressiveness of two popular GNN models — a natural question is how to increase the expressiveness beyond the lower bound. Polynomial expressiveness provides a simple path forward to add network operations or input features that complement these architectures with polynomials that are otherwise uncomputable. In our study, we add input features to enhance expressiveness.

Suppose we have a d' polynomial expressive GNN model (with a corresponding d' exact Prototypical graph model $\mathcal{F}_{\mathcal{B}}$) that we want to extend it to be $d > d'$ polynomial expressive. For every $\ell \in \mathbb{N}$, $d' + 1 \leq \ell \leq d$, we can compute all $\mathcal{F}_{\mathcal{B}}$ *non-computable* ℓ-degree basis elements of P_H using Algorithm 1, considering all non-isomorphic, simple and connected H. Indeed any H with two disconnected components corresponds to a multiplication of two lower degree polynomials approximable by the GNN itself (or using lower degree polynomial features). Any non-computable polynomials discovered in this process are added as node/edge input features to the architecture, effectively increasing the polynomial expressiveness of the model to d.

Table 1. Numbers of non-computable polynomials (left) out of all relevant polynomials (right) for the Prototypical models.

	$d=3$	$d=4$	$d=5$	$d=6$	$d=7$
$\mathcal{F}_n$	2/8	6/18	23/49	85/144	308/446
$\mathcal{F}_e$	0/18	0/53	1/174	11/604	72/2193

In Table 1 we list, for each Prototypical model and degree d, the number of polynomials that are found non-computable by the Prototypical models (left), out of the total number of relevant polynomials P_H (right). For the node based model we count only node-valued polynomials, while for the edge based model we count both node and edge-valued polynomials. Note that the number of non-computable polynomials is substantially smaller than the total number, especially in $\mathcal{F}_e$. Since polynomials are calculated at the data preprocessing step, there is an upfront computational cost for this procedure that must be accounted for. Finding the optimal contraction path that minimizes runtime complexity for a general matrix polynomial is an NP-hard problem (Biamonte et al., 2015) with a naive upper bound in runtime complexity of $O(n^d)$. An empirical evaluation of the preprocessing time is in Appendix A; in our experiments, preprocessing time is small compared to training time.

4. Related Work

Relation to Homomorphisms and Subgraph Counts. Past work has studied invariant polynomials on graphs (Thiéry, 2000; Lovász, 2012; Komiske et al., 2018). Viewed as functions on binary inputs, the basis consists of functions that count homomorphisms or injective homomorphisms of H into an input graph $\mathbf{X}$. Homomorphisms are related to the P_H basis, and injective homomorphisms are related to Q_H (see Appendix G). Also, equivariant homomorphism counts that relate to our P_H or Q_H has been studied (Mančinska & Roberson, 2020; Grohe et al., 2021; Maehara & NT, 2019; Bouritsas et al., 2022; Barceló et al., 2021; Welke et al., 2022). However, these works do not exhibit a basis of equivariant polynomials. Also, our tensor contraction interpretation and analysis does not appear in past work.

Expressivity Measures for Graph Models. The k-WL hierarchy has been widely used for studying graph machine learning (Morris et al., 2021), starting with the works of Morris et al. (2019b) and Xu et al. (2019), which show an equivalence between message passing neural networks and 1-WL. Tensor methods resembling k-WL such as k-IGN (Maron et al., 2018) and PPGN-like methods (Maron et al., 2019; Azizian & Lelarge, 2021) achieve k-WL power (Azizian & Lelarge, 2021; Geerts & Reutter, 2022), but scale in memory as n^k or n^{k-1} for n-node graphs. Morris et al. (2019a; 2022) define new k-WL variants with locality and sparsity biases, which gives a finer hierarchy and offers a trade-off between efficiency and expressiveness.

Various works measure the expressivity of graph neural networks by the types of subgraphs that they can count (Chen et al., 2020; Tahmasebi et al., 2020; Arvind et al., 2020). On simple graphs, subgraph counting of H is equivalent to evaluating an invariant polynomial Q_H. Additional works have studied the ability of graph models to compute numerous other graph properties. For instance, graph machine learning models have been studied in the context of approximating combinatorial algorithms (Sato et al., 2019), solving biconnectivity problems (Zhang et al., 2023), computing spectral invariants (Lim et al., 2022), distinguishing rooted graphs at the node level (Chen et al., 2021), and computing various other graph properties (Garg et al., 2020). As opposed to our framework, these expressivity measures generally do not induce a hierarchy of increasing expressivity, and they often do not directly suggest improvements for graph models

A matrix query language (MATLANG) (Brijder et al., 2019; Geerts, 2021) and a more general tensor language (TL) (Geerts & Reutter, 2022) have been used to study expressive power of GNNs (Balcilar et al., 2021; Geerts & Reutter, 2022). These languages define operations and ways to compose them for processing graphs in a permutation equivariant or invariant way. Our edge-based Prototypical model result gives a new perspective on a result of Geerts (2021), which shows that MATLANG can distinguish any two graphs that 2-FWL / 3-WL can. Indeed, our edge-based graph model includes the five linear algebra operations that form the 3-WL expressive MATLANG. While the operations of MATLANG were included in a somewhat ad-hoc manner ("motivated by operations supported in linear algebra package" (Geerts, 2021)), our framework shows that these are the at-most quadratic equivariant polynomials that are required to contract all tree-width 2 graphs.

Other Expressive GNNs. Various approaches have been used to develop expressive graph neural networks. One approach adds node or edge features, oftentimes positional or structural encodings, to base graph models (Sato et al., 2021; Abboud et al., 2021; Bouritsas et al., 2022; Lim et al., 2022; Zhang et al., 2023; Li et al., 2020; Loukas, 2020). Subgraph GNNs treat an input graph as a collection of subgraphs (Bevilacqua et al., 2022; Frasca et al., 2022; Qian et al., 2022; Cotta et al., 2021; Zhao et al., 2021; You et al., 2021; Zhang & Li, 2021). Some models utilize modified message passing and higher-order convolutions (Bodnar et al., 2021a;b; Thiede et al., 2021; de Haan et al., 2020). One can also take a base model and perform group averaging or frame averaging to make it have the desired equivariances while preserving expressive power (Murphy et al., 2019; Puny et al., 2022).

5. Experiments

In this section we demonstrate the impact of increasing the polynomial expressive power of GNNs. We test two families of models. PPGN++ (d) uses the architecture in equation 10, derived using our edge based Prototypical model, and achieves d polynomial expressive power by pre-computing polynomial features found in Subsection 3.3; missing (d) notation means using just PPGN++ without pre-computed features. GatedGCN (d) uses the base MPNN architecture of (Bresson & Laurent, 2017) with the d-expressive polynomials pre-computed. We experiment with 4 datasets: a graph isomorphism dataset SR (Bodnar et al., 2021b), which measures the ability of GNNs to distinguish strongly regular graphs; and 3 real-world molecular property prediction datasets including ZINC, ZINC-full (Dwivedi et al., 2020) and Alchemy (Chen et al., 2019a).

5.1. Graph Isomorphism Expressiveness

Distinguishing non-isomorphic graphs from families of strongly regular graphs is a challenging task (Bodnar et al., 2021a;b). The SR dataset (Bouritsas et al., 2022) is composed of 9 strongly regular families. This dataset is challenging since any pair of graphs in the SR dataset cannot be distinguished by the 3-WL algorithm. This experiment is done without any training (same procedure as in (Bodnar et al., 2021b)) and the evaluation is done by randomly initialized models. For every family in the dataset, we iterate over all pairs of graphs and report the fraction that the model determines are isomorphic. Two graphs are considered isomorphic if the L_2 distance between their embeddings is smaller than a certain threshold ($\epsilon = 0.01$). This procedure was repeated for 5 different random seeds and the averaged fraction rate was reported in Figure 10. This figure portrays the expressiveness boost gained by using high degree polynomial features. While the base models, GatedGCN and PPGN (and PPGN++), cannot distinguish any pair of graphs (as theoretically expected), adding higher degree polynomial features significantly improves the ability of the model to distinguish non-isomorphic graphs in this dataset. Optimal results of 0% failure rate are obtained for PPGN++ (d) with $d \geq 6$ (i.e., adding at-least degree 6 polynomial features). For the GatedGCN model, although we do not reach the 0% failure rate, adding the right polynomial features makes GatedGCN outperform 3-WL based models in distinguishing non-isomorphic graphs in this dataset.

Table 2. Results on Alchemy (Chen et al., 2019a) and ZINC-full (Dwivedi et al., 2020) datasets. Lower is better, best models are marked in **bold**.

Model	ZINC-Full	Alchemy
	Test MAE	Test MAE
GIN (Xu et al., 2019)	.088 ± .002	.180 ± .006
δ-2-GNN (Morris et al., 2019a)	.042 ± .003	.118 ± .001
SpeqNet (Morris et al., 2022)	-	.115 ± .001
PF-GNN (Dupty et al., 2022)	-	.111 ± .010
HIMP (Fey et al., 2020)	.036 ± .002	-
SignNet (Lim et al., 2022)	.024 ± .003	.113 ± .002
CIN (Bodnar et al., 2021a)	.022 ± .002	-
PPGN (Maron et al., 2019)	.022 ± .003	.113 ± .001
PPGN++	.022 ± .001	.111 ± .002
PPGN++ (5)	**.020 ± .001**	.110 ± .001
PPGN++ (6)	**.020 ± .001**	**.109 ± .001**

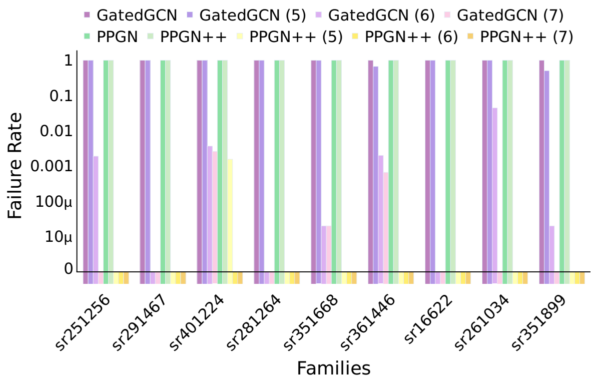


Figure 10. Failure rate (log scale) for distinguishing SR graphs, the lower the better.

5.2. Real-World Datasets

The efficacy of increasing polynomial expressive power on real-world data (molecular graphs datasets) was evaluated on 3 graph regression tasks: ZINC, ZINC-full and Alchemy.
Training. We followed the training protocol mentioned in (Dwivedi et al., 2020) for ZINC and ZINC-full and the protocol from (Lim et al., 2022) for Alchemy. All of our trained models obey a $500K$ parameter budget. Further details regarding the training procedure and model parameters can be found in Appendix A.
Baselines. The baseline results for the ZINC $12K$ experiment were obtained from (Zhao et al., 2022), except for PPGN and GatedGCN, which we calculated. For ZINC-full and Alchemy we used the results from (Lim et al., 2022).
Results. The mean absolute error (MAE) over the test set is reported in Table 3 for ZINC $12K$ and Table 2 for ZINC-full and alchemy. In both tables, PPGN++ (6) achieves SOTA results across all 3 datasets. In addition, for both test model families there is a clear correlation between higher d (polynomial expressiveness) and test error. Furthermore, PPGN++ (5) and PPGN++ (6), which produce the top results in all 3 experiments, are the only 2 models (including all baselines) which are provably strictly more powerful than 3-WL. Our results add evidence to the guiding hypothesis that increases in expressivity facilitate improved downstream performance.

Table 3. Results on ZINC $12K$ (Dwivedi et al., 2020) dataset. Lower is better, best model is marked in **bold**.

Model	Test MAE
GCN (Kipf & Welling, 2016)	.321 ± .009
GIN (Xu et al., 2019)	.163 ± .003
PNA (Corso et al., 2020)	.140 ± .006
GSN (Bouritsas et al., 2022)	.115 ± .012
PF-GNN (Dupty et al., 2022)	.122 ± .010
GIN-AK (Zhao et al., 2021)	.080 ± .001
CIN (Bodnar et al., 2021a)	.079 ± .006
SetGNN (Zhao et al., 2022)	.075 ± .003
GatedGCN (Bresson & Laurent, 2017)	.265 ± .015
GatedGCN (4)	.150 ± .005
GatedGCN (5)	.138 ± .003
GatedGCN (6)	.106 ± .003
PPGN (Maron et al., 2019)	.079 ± .005
PPGN++	.076 ± .003
PPGN++ (5)	.072 ± .005
PPGN++ (6)	**.071 ± .001**

6. Conclusions

We propose a novel framework for evaluating the expressive power of GNNs by evaluating their ability to approximate equivariant graph polynomials. Our first step was introducing a basis for those polynomials of any degree. We then utilized Prototypical graph models to determine the computability of these polynomials with practical GNNs. This led to a method for increasing the expressivity of GNNs through the use of precomputed polynomial features, resulting in a significant improvement in empirical performance.

Future research could focus on several promising directions. One direction can reduce the number of features passed into GNNs by working with a generating set of polynomials rather than the complete basis. Additionally, incorporating features on nodes and edges, as outlined in Section 2.2 and Appendix F, could further improve performance. Another exciting avenue for exploration can incorporate otherwise uncomputable polynomials as computational primitives into GNN layers, rather than as features, to increase expressiveness. Finally, it would be beneficial to study Prototypical graph models to identify families with optimal properties related to expressiveness, memory/computation complexity, and the size of the contraction bank.

7. Acknowledgments

OP is supported by a grant from Israel CHE Program for Data Science Research Centers. DL is supported by an NSF Graduate Fellowship. We thank Nicolas Usunier for his insightful remarks and the anonymous reviewers for their helpful comments.

References

Abboud, R., Ceylan, I. I., Grohe, M., and Lukasiewicz, T. The surprising power of graph neural networks with random node initialization. In *IJCAI*, 2021.

Antoneli, F., Dias, A. P. S., and Matthews, P. C. Invariants, equivariants and characters in symmetric bifurcation theory. *Proceedings of the Royal Society of Edinburgh Section A: Mathematics*, 138(3):477–512, 2008.

Arvind, V., Fuhlbrück, F., Köbler, J., and Verbitsky, O. On weisfeiler-leman invariance: Subgraph counts and related graph properties. *Journal of Computer and System Sciences*, 113:42–59, 2020.

Azizian, W. and Lelarge, M. Expressive power of invariant and equivariant graph neural networks. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=lxHgXYN4bwl`.

Balcilar, M., Héroux, P., Gauzere, B., Vasseur, P., Adam, S., and Honeine, P. Breaking the limits of message passing graph neural networks. In *International Conference on Machine Learning*, pp. 599–608. PMLR, 2021.

Barceló, P., Geerts, F., Reutter, J., and Ryschkov, M. Graph neural networks with local graph parameters, 2021. URL `https://arxiv.org/abs/2106.06707`.

Bedratyuk, L. A new formula for the generating function of the numbers of simple graphs. *arXiv preprint arXiv:1512.06355*, 2015.

Bevilacqua, B., Frasca, F., Lim, D., Srinivasan, B., Cai, C., Balamurugan, G., Bronstein, M. M., and Maron, H. Equivariant subgraph aggregation networks. In *International Conference on Learning Representations*, 2022.

Biamonte, J. D., Morton, J., and Turner, J. Tensor network contractions for #SAT. *Journal of Statistical Physics*, 160(5):1389–1404, jun 2015. doi: 10.1007/s10955-015-1276-z. URL `https://doi.org/10.1007%2Fs10955-015-1276-z`.

Bodlaender, H. L. A partial k-arboretum of graphs with bounded treewidth. *Theoretical computer science*, 209 (1-2):1–45, 1998.

Bodnar, C., Frasca, F., Otter, N., Wang, Y. G., Liò, P., Montúfar, G., and Bronstein, M. Weisfeiler and lehman go cellular: Cw networks, 2021a. URL `https://arxiv.org/abs/2106.12575`.

Bodnar, C., Frasca, F., Wang, Y. G., Otter, N., Montúfar, G., Liò, P., and Bronstein, M. Weisfeiler and lehman go topological: Message passing simplicial networks, 2021b. URL `https://arxiv.org/abs/2103.03212`.

Bouritsas, G., Frasca, F., Zafeiriou, S., and Bronstein, M. M. Improving graph neural network expressivity via subgraph isomorphism counting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(1):657–668, 2022.

Bresson, X. and Laurent, T. Residual gated graph convnets, 2017. URL `https://arxiv.org/abs/1711.07553`.

Brijder, R., Geerts, F., Bussche, J. V. D., and Weerwag, T. On the expressive power of query languages for matrices. *ACM Transactions on Database Systems (TODS)*, 44(4): 1–31, 2019.

Chen, G., Chen, P., Hsieh, C.-Y., Lee, C.-K., Liao, B., Liao, R., Liu, W., Qiu, J., Sun, Q., Tang, J., Zemel, R., and Zhang, S. Alchemy: A quantum chemistry dataset for benchmarking ai models, 2019a. URL `https://arxiv.org/abs/1906.09427`.

Chen, L., Chen, Z., and Bruna, J. On graph neural networks versus graph-augmented mlps. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=tiqI7w64JG2`.

Chen, Z., Villar, S., Chen, L., and Bruna, J. On the equivalence between graph isomorphism testing and function approximation with gnns. *Advances in neural information processing systems*, 32, 2019b.

Chen, Z., Chen, L., Villar, S., and Bruna, J. Can graph neural networks count substructures? *Advances in neural information processing systems*, 33:10383–10395, 2020.

Corso, G., Cavalleri, L., Beaini, D., Liò, P., and Veličković, P. Principal neighbourhood aggregation for graph nets, 2020. URL `https://arxiv.org/abs/2004.05718`.

Cotta, L., Morris, C., and Ribeiro, B. Reconstruction for powerful graph representations. *Advances in Neural Information Processing Systems*, 34:1713–1726, 2021.

de Haan, P., Cohen, T. S., and Welling, M. Natural graph networks. *Advances in Neural Information Processing Systems*, 33:3636–3646, 2020.

Dell, H., Grohe, M., and Rattan, G. Lov'asz meets weisfeiler and leman. In *45th International Colloquium on Automata, Languages, and Programming (ICALP 2018)*, volume 107, pp. 40. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018.

Derksen, H. and Kemper, G. *Computational invariant theory*. Springer, 2015.

Diestel, R. *Graph Theory (Graduate Texts in Mathematics)*. Springer, August 2005. ISBN 3540261826. URL `http://www.amazon.ca/exec/obidos/redirect?tag=citeulike04-20{&}path=ASIN/3540261826`.

Duffin, R. J. Topology of series-parallel networks. *Journal of Mathematical Analysis and Applications*, 10(2):303–318, 1965.

Dupty, M. H., Dong, Y., and Lee, W. S. PF-GNN: Differentiable particle filtering based approximation of universal graph representations. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=oh4TirnfSem`.

Dvořák, Z. On recognizing graphs by numbers of homomorphisms. *Journal of Graph Theory*, 64(4):330–342, 2010.

Dwivedi, V. P., Joshi, C. K., Luu, A. T., Laurent, T., Bengio, Y., and Bresson, X. Benchmarking graph neural networks. 2020. doi: 10.48550/ARXIV.2003.00982. URL `https://arxiv.org/abs/2003.00982`.

Fey, M. and Lenssen, J. E. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

Fey, M., Yuen, J.-G., and Weichert, F. Hierarchical inter-message passing for learning on molecular graphs, 2020. URL `https://arxiv.org/abs/2006.12179`.

Frasca, F., Bevilacqua, B., Bronstein, M. M., and Maron, H. Understanding and extending subgraph GNNs by rethinking their symmetries. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing Systems*, 2022.

Garg, V., Jegelka, S., and Jaakkola, T. Generalization and representational limits of graph neural networks. In *International Conference on Machine Learning*, pp. 3419–3430. PMLR, 2020.

Geerts, F. On the expressive power of linear algebra on graphs. *Theory of Computing Systems*, 65(1):179–239, 2021.

Geerts, F. and Reutter, J. L. Expressiveness and approximation properties of graph neural networks. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=wIzUeM3TAU`.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. In *International conference on machine learning*, pp. 1263–1272. PMLR, 2017.

Grohe, M., Rattan, G., and Seppelt, T. Homomorphism tensors and linear equations. *arXiv preprint arXiv:2111.11313*, 2021.

Harary, F. and Palmer, E. M. *Graphical enumeration*. Elsevier, 2014.

Hardy, G. H., Wright, E. M., et al. *An introduction to the theory of numbers*. Oxford university press, 1979.

Hornik, K. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251–257, 1991. ISSN 0893-6080. doi: https://doi.org/10.1016/0893-6080(91)90009-T. URL `https://www.sciencedirect.com/science/article/pii/089360809190009T`.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks, 2016. URL `https://arxiv.org/abs/1609.02907`.

Komiske, P. T., Metodiev, E. M., and Thaler, J. Energy flow polynomials: A complete linear basis for jet substructure. *Journal of High Energy Physics*, 2018(4):1–54, 2018.

Li, P., Wang, Y., Wang, H., and Leskovec, J. Distance encoding: Design provably more powerful neural networks for graph representation learning. *Advances in Neural Information Processing Systems*, 33:4465–4478, 2020.

Lim, D., Robinson, J., Zhao, L., Smidt, T., Sra, S., Maron, H., and Jegelka, S. Sign and basis invariant networks for spectral graph representation learning. *arXiv preprint arXiv:2202.13013*, 2022.

Loukas, A. What graph neural networks cannot learn: depth vs width. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=B1l2bp4YwS`.

Lovász, L. *Large networks and graph limits*, volume 60. American Mathematical Soc., 2012.

Maehara, T. and NT, H. A simple proof of the universality of invariant/equivariant graph neural networks. *arXiv preprint arXiv:1910.03802*, 2019.

Mančinska, L. and Roberson, D. E. Quantum isomorphism is equivalent to equality of homomorphism counts from planar graphs. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 661–672. IEEE, 2020.

Maron, H., Ben-Hamu, H., Shamir, N., and Lipman, Y. Invariant and equivariant graph networks. *arXiv preprint arXiv:1812.09902*, 2018.

Maron, H., Ben-Hamu, H., Serviansky, H., and Lipman, Y. Provably powerful graph networks. *Advances in neural information processing systems*, 32, 2019.

Molien, T. Uber die invarianten der linearen substitutionsgruppen. *Sitzungber. Konig. Preuss. Akad. Wiss. (J. Berl. Ber.)*, 52:1152–1156, 1897.

Morris, C., Rattan, G., and Mutzel, P. Weisfeiler and leman go sparse: Towards scalable higher-order graph embeddings, 2019a. URL `https://arxiv.org/abs/1904.01543`.

Morris, C., Ritzert, M., Fey, M., Hamilton, W. L., Lenssen, J. E., Rattan, G., and Grohe, M. Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pp. 4602–4609, 2019b.

Morris, C., Lipman, Y., Maron, H., Rieck, B., Kriege, N. M., Grohe, M., Fey, M., and Borgwardt, K. Weisfeiler and leman go machine learning: The story so far. *arXiv preprint arXiv:2112.09992*, 2021.

Morris, C., Rattan, G., Kiefer, S., and Ravanbakhsh, S. Speqnets: Sparsity-aware permutation-equivariant graph networks, 2022. URL `https://arxiv.org/abs/2203.13913`.

Murphy, R., Srinivasan, B., Rao, V., and Ribeiro, B. Relational pooling for graph representations. In *International Conference on Machine Learning*, pp. 4663–4673. PMLR, 2019.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: An imperative style, high-performance deep learning library, 2019. URL `https://arxiv.org/abs/1912.01703`.

Pólya, G. Kombinatorische anzahlbestimmungen für gruppen, graphen und chemische verbindungen. *Acta mathematica*, 68:145–254, 1937.

Puny, O., Atzmon, M., Smith, E. J., Misra, I., Grover, A., Ben-Hamu, H., and Lipman, Y. Frame averaging for invariant and equivariant network design. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=zIUyj55nXR`.

Qian, C., Rattan, G., Geerts, F., Niepert, M., and Morris, C. Ordered subgraph aggregation networks. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing Systems*, 2022.

Rampášek, L., Galkin, M., Dwivedi, V. P., Luu, A. T., Wolf, G., and Beaini, D. Recipe for a general, powerful, scalable graph transformer, 2023.

Sato, R., Yamada, M., and Kashima, H. Approximation ratios of graph neural networks for combinatorial problems. *Advances in Neural Information Processing Systems*, 32, 2019.

Sato, R., Yamada, M., and Kashima, H. Random features strengthen graph neural networks. In *Proceedings of the 2021 SIAM International Conference on Data Mining (SDM)*, pp. 333–341. SIAM, 2021.

Segol, N. and Lipman, Y. On universal equivariant set networks. *arXiv preprint arXiv:1910.02421*, 2019.

Tahmasebi, B., Lim, D., and Jegelka, S. Counting substructures with higher-order graph neural networks: Possibility and impossibility results. *arXiv preprint arXiv:2012.03174*, 2020.

Thiede, E., Zhou, W., and Kondor, R. Autobahn: Automorphism-based graph neural nets. *Advances in Neural Information Processing Systems*, 34:29922–29934, 2021.

Thiéry, N. M. Algebraic invariants of graphs; a study based on computer exploration. *ACM SIGSAM Bulletin*, 34(3): 9–20, 2000.

Tucker, A. *Applied combinatorics*. John Wiley & Sons, Inc., 1994.

Welke, P., Thiessen, M., and Gärtner, T. Expectation complete graph representations using graph homomorphisms. In *The First Learning on Graphs Conference*, 2022. URL `https://openreview.net/forum?id=8GJyW4i2oST`.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=ryGs6iA5Km`.

You, J., Gomes-Selman, J. M., Ying, R., and Leskovec, J. Identity-aware graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 10737–10745, 2021.

You, Y., Li, J., Reddi, S., Hseu, J., Kumar, S., Bhojanapalli, S., Song, X., Demmel, J., Keutzer, K., and Hsieh, C.-J. Large batch optimization for deep learning: Training bert in 76 minutes, 2019. URL `https://arxiv.org/abs/1904.00962`.

Zhang, B., Luo, S., Wang, L., and Di, H. Rethinking the expressive power of gnns via graph biconnectivity. *arXiv preprint arXiv:2301.09505*, 2023.

Zhang, M. and Li, P. Nested graph neural networks. *Advances in Neural Information Processing Systems*, 34: 15734–15747, 2021.

Zhao, L., Jin, W., Akoglu, L., and Shah, N. From stars to subgraphs: Uplifting any gnn with local structure awareness, 2021. URL `https://arxiv.org/abs/2110.03753`.

Zhao, L., Shah, N., and Akoglu, L. A practical, progressively-expressive GNN. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing Systems*, 2022. URL `https://openreview.net/forum?id=WBv9Z6qpA8x`.

A. Implementation Details

A.1. Datasets

SR. The SR dataset (Bouritsas et al., 2022) is composed of 9 families of Strongly Regular graphs. Each family has a 4 dimensional representation: n the number of nodes in the graph, d the degree of each node, λ the number of mutual neighbors of adjacent nodes and μ the number of mutual neighbors of non-adjacent nodes. Table 4 shows the size of each Strongly Regular family from the dataset.

Table 4. Sizes of Strongly Regular Families (Bouritsas et al., 2022)

Familty	(16,6,2,2)	(25,12,5,6)	(26,10,3,4)	(28,12,6,4)	(29,14,6,7)	(35,16,6,8)	(35,18,9,9)	(36,14,4,6)	(40,12,2,4)
Number of Graphs	2	15	10	4	41	3854	227	180	28

ZINC. The ZINC dataset is a molecular graph dataset composed of $\sim 250K$ molecules. The regression criterion is a molecular property known as the constrained solubility. Each molecule has both node features and edge features. Node features represent the type of heavy atoms (4 types) and edge features the type of bonds between them (28). The average number of nodes in a graph is 23.15 and the number of edges is 49.8. There are two versions of the dataset used for learning: ZINC $12K$ which has train/val/test split of $10000/1000/1000$ and ZINC-full with a $2200011/24445/5000$ split. Both data splits can be obtained from (Fey & Lenssen, 2019)

Alchemy. Alchemy is also a molecular graph dataset composoed of 12000 graphs ($10000/1000/1000$ split taken from (Lim et al., 2022)). The average number of nodes is 10.1 and the number of edges is 20.9. The Regression target in this dataset is a 12-dimensional vector composed of a collection molecular properties : dipole moment, polarizability, HOMO, LUMO, gap, R^2, zero point energy, internal energy, internal energy at $298.15K$, enthalpy at $298.15K$, free energy at $298.15K$ and heat capacity at $298.15K$. Each graph has node features (6-dimensional atom type indicator) and edge features (4-dimensional bond type indicator).

A.2. Training Protocol

ZINC. For the ZINC and ZINC-full experiments we followed the training protocol from (Dwivedi et al., 2020). The protocol includes parameter budget ($500K$), predefined 4 random seeds and a learning rate decay scheme that reduces the rate based on the validation error (factor 0.5 and patience factor of 10 epochs). Initial learning rate was set to 0.002 and training stopped when reached 10^{-5}. Batch size was set to 128. Test error at last epoch was reported. When using polynomial features, we removed the polynomials that had no response over the dataset. Namely, let $f : \mathbb{R}^{n^2} \to \mathbb{R}^{n^2}$ be an equivariant polynomial and $\mathcal{X} = \{\mathbf{X}\}$ be a graph dataset. f does not have a response over $\mathcal{X}$ if $\forall \mathbf{X} \in \mathcal{X}$, $f(\mathbf{X}) = 0$. Similarly to (Barceló et al., 2021) we normalized the additional features to have a unit norm. For PPGN++ we used $1/1$ of the edge based 5-degree polynomials and $8/11$ of the 6-degree polynomials. For GatedGCN we used $2/2$ of the node based 3-degree polynomials, $6/6$ of the 4-degree polynomials, $23/23$ of the 5-degree polynomials and $83/85$ of the 6-degree polynomials. models were trained using the LAMB optimizer (You et al., 2019) on a single Nvidia V-100 GPU. The models were trained using the PyTorch framework (Paszke et al., 2019).

Alchemy. We followed the training protocol from (Lim et al., 2022). The protocol includes averaging results on 5 random seeds and learning rate decay scheme that reduces the rate based on the validation error (factor 0.5 and patience factor of 20 epochs). Initial learning rate was set to 10^{-3} and training stopped when reached 10^{-5}. Batch size was set to 128. Test error at last epoch was reported. When using polynomial features, we removed the polynomials that had no response over the dataset and normalized them in the same way as in the ZINC experiment. For PPGN++ we used $1/1$ of the edge based 5-degree polynomials and $8/11$ of the 6-degree polynomials. models were trained using the LAMB optimizer (You et al., 2019) on a single Nvidia V-100 GPU. The models were trained using the PyTorch framework (Paszke et al., 2019).

A.3. Architectures

GatedGCN. We used the model as it defined in (Bresson & Laurent, 2017) and implemented in (Lim et al., 2022). For the ZINC $12K$ experiment we to used a 16-layer model (same baseline as used in (Lim et al., 2022)) with feature dimension of size 77 for the baseline model and 75 for the models with polynomial features. For the SR dataset we used a 4-layer network with hidden dimension size of 150. The polynomial features were added to the initial input node features via concatenation.

PPGN++. The PPGN++ architecture is based on the PPGN architecture (Maron et al., 2019). The original PPGN layer is defined by the following equation:

$$\mathbf{Z}^{(k+1)} = \mathtt{m}_3 \left[\mathtt{m}_1(\mathbf{Z}^{(k)}) \circledast \mathtt{m}_2(\mathbf{Z}^{(k)}), \mathbf{Z}^{(k)} \right]$$

For $\mathbf{Z} \in \mathbb{R}^{n^2 \times d}$. While this layer definition cannot approximate all $C \in \mathcal{B}$ from $\mathcal{F}_e$, it is possible to naively incorporate all the linear and constant basis (Maron et al., 2018) to obtain full approximation power. As mentioned in Section 3.2 we suggest to add this expressiveness to the layer in a more compact manner:

$$\mathbf{Z}^{(k+1)} = \bar{\mathtt{m}}_3 \left[\bar{\mathtt{m}}_1 \left[\mathbf{Z}^{(k)}, \mathbf{Z}^{(k)T} \right] \circledast \bar{\mathtt{m}}_2(\mathbf{Z}^{(k)}), \mathbf{Z}^{(k)} \right]$$

where

$$\bar{\mathtt{m}}_i = \left(\bar{\mathtt{m}}_i^{\text{diag}}, \bar{\mathtt{m}}_i^{\text{off-diag}} \right),$$

defines a separate MLP for diagonal elements and off-diagonal elements. In practice, we implement this separation by adding an identity matrix as additional feature before applying an MLP on the tensor's features.

For the ZINC $12K$ experiment we used a 8-layer network with hidden dimension size of 95. For ZINC-full and Alchemy we used a 6-layer network with hidden dimension size of 110. We ran parameter search over the number of layers $L \in \{4, 6, 8\}$ and hidden dimension size $h \in \{95, 110, 130\}$ while maintaining the $500K$ parameter budget. For the SR experiment we used a 4-layer network with hidden dimension of size 75. The polynomial features were added to the initial input features via concatenation.

A.4. Timing

Table 5 shows a runtime comparison between the preprocessing require to compute polynomial features and training a PPGN++ (6) model on the ZINC $12K$ dataset. The time it takes to compute polynomials of degree 7 is non-negligible and most likely that for higher degrees (or in cases of larger graphs) the runtime will be longer and intractable from some degree. However, for SOTA results which we report in Section 5 we only use up to 6 degree polynomial features and the added time used for computing those features is equivalent to only 3 training epochs. Moreover, comparing the running time of other methods puts in perspective the computational time required for computing polynomial features. SetGNN (Zhao et al., 2022) reports that the epoch running of their best ZINC model (0.075 compared to 0.071 of PPGN++ (6)) is around 25 seconds. In addition GraphGPS (Rampášek et al., 2023), a state of the art Graph Transformer (test error of 0.07 on the ZINC dataset), takes $\sim$ 11.7 hours to train.

Table 5. Runtime comparison on ZINC $12K$: preprocessing vs. training.

	Time (Seconds)
finding all $\mathcal{F}_e$ non-computable polynomials up to degree 7.	5
compute all 5 degree polynomial features for the entire ZINC $12K$ dataset.	10
compute all 6 degree polynomial features for the entire ZINC $12K$ dataset.	23
compute all 7 degree polynomial features for the entire ZINC $12K$ dataset.	310
Average runtime of training PPGN++ (6) on ZINC $12K$	4110 (15.5 per epoch)

B. Proof of Theorem 2.1.

General definitions and setup. We denote an input graph data points represented by $\mathbf{X} \in \mathbb{R}^{n^2}$. We denote by the vector space of all polynomials $P : \mathbb{R}^{n^2} \to \mathbb{R}^{n^2}$ by $\mathfrak{P} = \mathbb{R}^{n^2} \otimes \mathbb{R}[\mathbf{X}]$, where $\otimes$ is the tensor product and $\mathbb{R}[\mathbf{X}]$ denotes the module of polynomials with indeterminate $\mathbf{X}_{11}, \ldots, \mathbf{X}_{nn}$. The space of polynomials $\mathfrak{P}$ is spanned by the monomial basis

$$M(\mathbf{X}) = \delta^{a,b} \otimes \prod_{r,s=1}^{n} \mathbf{X}_{r,s}^{\mathbf{A}_{r,s}} \tag{11}$$

where $\mathbf{A} \in \mathbb{N}_0^{n^2}$, $\mathbb{N}_0 = \{0, 1, \ldots\}$, and $\delta^{a,b} \in \mathbb{R}^{n^2}$ is a matrix satisfying

$$\delta_{i,j}^{a,b} = \begin{cases} 1 & \text{if } a = i\text{,b=j} \\ 0 & \text{o/w} \end{cases}$$

That is $\delta^{a,b}$, $a, b \in [n]$ is a basis for $\mathbb{R}^{n^2}$.

The degree of a polynomial is the maximal degree of its monomials defined by

$$\deg M(\mathbf{X}) = \sum_{r,s=1}^{n} \mathbf{A}_{r,s} \tag{12}$$

We denote by $\mathfrak{P}_d$ the space of all polynomials of degree at most d.

Enumerating monomials with multi-graphs H. Next, we define $H = (V, E, (a, b))$ to be a multi-graph with node set $V = [n]$, and edge multiset defined by the matrix $\mathbf{A}$, that is (r, s) appears $k \in \mathbb{N}_0$ times in E iff $\mathbf{A}_{r,s} = k$. Lastly (a, b) is the red edge. We can therefore identify monomials with multi-graphs H, i.e.,

$$M_H = M, \tag{13}$$

where M is defined in equation 11.

Action of permutations S_n on polynomials. We consider the group of permutations S_n that consists of bijections $g : [n] \to [n]$. The action of S_n on a matrix $\mathbf{X}$ is defined in the standard way in equation 2, i.e.,

$$(g \cdot \mathbf{X})_{i,j} = \mathbf{X}_{g^{-1}(i),g^{-1}(j)} \tag{14}$$

where the inverse is used to make this a left action. We define $\mathfrak{P}^{S_n}$ to be the space of permutation equivariant polynomials, namely $P \in \mathfrak{P}$ that satisfy

$$g \cdot P(\mathbf{X}) = P(g \cdot \mathbf{X})$$

for all $g \in S_n$ and $\mathbf{X} \in \mathbb{R}^{n^2}$. A standard method of projecting a polynomial in $\mathfrak{P}$ onto the equivariant polynomials $\mathfrak{P}^{S_n}$ is via the symmetrization (Reynolds) operators:

$$\bar{P}(\mathbf{X}) = \sum_{g \in S_n} g \cdot P(g^{-1} \cdot \mathbf{X}) \tag{15}$$

Let us verify that indeed $\bar{P} \in \mathfrak{P}^{S_n}$:

$$\begin{aligned} \bar{P}(h \cdot \mathbf{X}) &= \sum_{g \in S_n} g \cdot P(g^{-1} \cdot (h \cdot \mathbf{X})) \\ &= \sum_{g \in S_n} g \cdot P((h^{-1}g)^{-1} \cdot \mathbf{X}) \\ &= \sum_{g \in S_n} hg \cdot P(g^{-1} \cdot \mathbf{X}) \\ &= h \cdot \bar{P}(\mathbf{X}) \end{aligned}$$

Symmetrization of monomials. The key part of the proof is computing the symmetrization of the monomial basis M_H via the symmetrization operator:

$$\begin{aligned}
Q_H(\mathbf{X})_{i,j} &= \sum_{g\in S_n} \left[g \cdot M_H(g^{-1}\cdot \mathbf{X})\right]_{i,j} \\
&= \sum_{g\in S_n} \left[M_H(g^{-1}\cdot \mathbf{X})\right]_{g^{-1}(i),g^{-1}(j)} \\
&= \sum_{g\in S_n} \left[\delta^{a,b} \otimes \prod_{r,s=1}^{n} (g^{-1}\cdot \mathbf{X})_{r,s}^{\mathbf{A}_{r,s}}\right]_{g^{-1}(i),g^{-1}(j)} \\
&= \sum_{g\in S_n} \left[\delta^{a,b} \otimes \prod_{r,s=1}^{n} \mathbf{X}_{g(r),g(s)}^{\mathbf{A}_{r,s}}\right]_{g^{-1}(i),g^{-1}(j)} \\
&= \sum_{g\in S_n} \left[\delta^{a,b} \otimes \prod_{r,s=1}^{n} \mathbf{X}_{r,s}^{\mathbf{A}_{g^{-1}(r),g^{-1}(s)}}\right]_{g^{-1}(i),g^{-1}(j)} \\
&= \sum_{g\in S_n} \delta^{a,b}_{g^{-1}(i),g^{-1}(j)} \prod_{r,s=1}^{n} \mathbf{X}_{r,s}^{\mathbf{A}_{g^{-1}(r),g^{-1}(s)}} \\
&= \sum_{g\in S_n} \delta^{g(a),g(b)}_{i,j} \prod_{r,s=1}^{n} \mathbf{X}_{r,s}^{\mathbf{A}_{g^{-1}(r),g^{-1}(s)}}
\end{aligned}$$

where in the second and fourth equality we used the action definition (equation 23), in the fifth equality we re-enumerated $(r, s) \in [n] \times [n]$ with $(r', s') = (g(r), g(s))$, and the last equality uses the fact that $a = g^{-1}(i)$ and $b = g^{-1}(j)$ iff $g(a) = i$ and $g(b) = j$.

Now let us define the action of S_n on the multi-graph H, also in a natural manner: $g \cdot H$ is the multi-graph that results from relabeling each node $i \in [n]$ in H as $g(i) \in [n]$. The multi-graph $g \cdot H$ is isomorphic to H and $(r, s) \in E$ with multiplicity ℓ iff $(g(r), g(s)) \in g \cdot E$ with multiplicity ℓ. If we let $\mathbf{A}$ be the adjacency matrix of $\mathbf{H}$ then $g \cdot \mathbf{A}$ (defined via equation 23) is the adjacency of $g \cdot H$, i.e., $(g \cdot \mathbf{A})_{i,j} = \mathbf{A}_{g^{-1}(i),g^{-1}(j)}$. Furthermore, the red edge in $g \cdot H$ is $(g(a), g(b))$. With these definitions, the above equation takes the form

$$Q_H(\mathbf{X}) = \sum_{g\in S_n} M_{g\cdot H}(\mathbf{X}) \tag{16}$$

Equation 16 is the key to the proof. It shows that Q_H is a sum over all monomials corresponding to the orbit of H under node relabeling g, therefore, any two isomorphic multi-graphs $H \cong H'$ would correspond to the same equivariant polynomials $Q_H = Q_{H'}$. Differently put, in contrast to M_H that are enumerated by *labeled* multi-graphs H, Q_H are enumerated by *non-isomorphic* multi-graphs H. Note that if H has isolated nodes (i.e., not touching any edge), these can be discarded without changing Q_H, so for degree d polynomials we really just need to consider graphs with d edges and a single red edge with no isolated nodes, so the maximal number of nodes is at most $\min\{2d + 2, n\}$.

We next show that $\{Q_H\}$, corresponding to all non-isomorphic H with up to d edges, is a basis for $\mathfrak{P}_d$. First, we claim it spans $\mathfrak{P}_d$. Indeed, since every polynomial $P \in \mathfrak{P}_d$ can be written as a linear combination of monomials M_H, $P = \sum_k c_k M_{H_k}(\mathbf{X})$. Now,

$$P(\mathbf{X}) = \bar{P}(\mathbf{X}) = \sum_k c_k \bar{M}_{H_k}(\mathbf{X}) = \sum_k c_k Q_{H_k}(\mathbf{X})$$

where in the first equality we used the fact that the symmetrization operator fixes P, i.e., $\bar{P} = P$, and in the second equality the fact that the symmetrization operator is linear. Next, we claim that $\{Q_H\}$ for non-isomorphic H is an independent set. This is true since each Q_H is a sum over the orbit of H, $\{g \cdot H | g \in S_n\}$, and the orbits are disjoint sets. Therefore, since the set of all monomials, M_H, is independent, also $\{Q_H\}$ is independent.

Formula for Q_H. We found that Q_H is a basis for the equivariant graph polynomials $\mathfrak{P}_d$. Let us write down an explicit formula for it next. The (i_a, i_b) entry of $Q_H(\mathbf{X})$ takes the form

$$\begin{aligned} Q_H(\mathbf{X})_{i_a,i_b} &= \sum_{g\in S_n} \delta^{g(a),g(b)}_{i_a,i_b} \prod_{r,s=1}^{n} \mathbf{X}^{\mathbf{A}_{r,s}}_{g(r)g(s)} \\ &= \sum_{g\in S_n} \delta^{g(a),g(b)}_{i_a,i_b} \prod_{(r,s)\in E} \mathbf{X}_{g(r)g(s)} \\ &= \sum_{\substack{j_1\neq\cdots\neq j_m\in[n] \\ j_a=i_a, j_b=i_b}} \prod_{(r,s)\in E} \mathbf{X}_{j_r,j_s} \end{aligned} \tag{17}$$

where in the third equality we denote $j_1 = g(1), j_2 = g(2), \ldots, j_m = g(m)$, and $j_1 \neq \cdots \neq j_m \in [n]$ stands for all assignments of different indices $j_1, \ldots, j_m \in [n]$.

Note that Q_H is proved a basis but is still different from P_H in equation 4 in that it does not sum over repeated indices. The fact that allowing repeated indices is still a basis is proved next. This seemingly small change of basis is crucial for our tensor network connection and analysis in the paper.

P_H is a basis. We now prove that P_H defined in equation 4 is a basis. For convenience we repeat it below:

$$P_H(\mathbf{X})_{i_a,i_b} = \sum_{\substack{j_1,\ldots,j_m\in[n] \\ j_a=i_a, j_b=i_a}} \prod_{(r,s)\in E} \mathbf{X}_{j_r,j_s} \tag{18}$$

Denote by $\mathcal{H}_m$ the set of all multigraphs $H = (V, E, (a, b))$ with $|V| \leq m$. Since the cardinality of $\{P_H\}_{H\in\mathcal{H}_m}$ is at most that of $\{Q_H\}_{H\in\mathcal{H}_m}$ it is enough to show that $\{P_H\}_{H\in\mathcal{H}_m}$ spans the same space as $\{Q_H\}_{H\in\mathcal{H}_m}$.

The proof follows an induction on m. For the base $m = 1$ consider all multigraphs $H = (V, E, (a, a))$, where $V = \{a\}$. In this case both equation 17 and equation 18 have vacant sums and

$$P_H(\mathbf{X})_{i_a,i_a} = \prod_{(r,s)\in E} \mathbf{X}_{i_r,i_s} = Q_H(\mathbf{X})_{i_a,i_a},$$

where for all $(r, s) \in E$ we have $r, s \in \{a\}$.

Now, for $m \geq 2$, assume

$$\text{span}\,\{P_H\}_{H\in\mathcal{H}_{m-1}} = \text{span}\,\{Q_H\}_{H\in\mathcal{H}_{m-1}}$$

and consider an arbitrary $H \in \mathcal{H}_m \setminus \mathcal{H}_{m-1}$.

If $m = 2$, and $a \neq b$, then $H = (V, E, (a, b))$, and $V = \{a, b\}$. In this case again both equation 17 and equation 18 have vacant sums and

$$P_H(\mathbf{X})_{i_a,i_b} = \prod_{(r,s)\in E} \mathbf{X}_{i_r,i_s} = Q_H(\mathbf{X})_{i_a,i_b},$$

where for all $(r, s) \in E$ we have $r, s \in \{a, b\}$.

In all other cases, consider the space of tuples $[n]^m = \{(j_1, \ldots, j_m) | j_i \in [n], i \in [m]\}$, and the action of S_n on this collection via $g \cdot (j_1, \ldots, j_m) = (g(j_1), \ldots, g(j_m))$. The orbits, denoted $o_1, \ldots, o_B$ correspond to equality patterns of indices, and $B = \text{Bell}(m)$, the Bell number of m. By convention we define o_1 to be the orbit

$$o_1 = [(1, 2, \ldots, m)]$$

where we use the orbit notation $[(j_1, \ldots, j_m)] = \{(g(j_1), \ldots, g(j_m)) | g \in S_n\}$. Now, we decompose the index set $\{(j_1, \ldots, j_m) \in [n]^m | j_a = i_a, j_b = i_b\}$ to disjoint index sets by intersecting it with o_ℓ, $\ell \in [B]$. Note that some of these index sets may be empty; we let $c_\ell = 1$ in case this index set is not empty, and $c_\ell = 0$ otherwise.

$$\begin{aligned}
P_H(\mathbf{X})_{i_a,i_b} &= \sum_{\ell=1}^{B} \sum_{\substack{(j_1,\dots,j_m)\in[n]^m\cap o_\ell \\ j_a=i_a, j_b=i_b}} \prod_{(r,s)\in E} \mathbf{X}_{j_r,j_s} \\
&= \sum_{\substack{j_1\neq\dots\neq j_m \\ j_a=i_a, j_b=i_b}} \prod_{(r,s)\in E} \mathbf{X}_{j_r,j_s} + \sum_{\ell=2}^{B} \sum_{\substack{(j_1,\dots,j_m)\in[n]^m\cap o_\ell \\ j_a=i_a, j_b=i_b}} \prod_{(r,s)\in E} \mathbf{X}_{j_r,j_s} \\
&= Q_H(\mathbf{X})_{i_a,i_b} + \sum_{\ell=2}^{B} \sum_{\substack{(j_1,\dots,j_m)\in[n]^m\cap o_\ell \\ j_a=i_a, j_b=i_b}} \prod_{(r,s)\in E} \mathbf{X}_{j_r,j_s}
\end{aligned}$$

For $\ell \geq 2$ consider the polynomial

$$\sum_{\substack{(j_1,\dots,j_m)\in[n]^m\cap o_\ell \\ j_a=i_a, j_b=i_b}} \prod_{(r,s)\in E} \mathbf{X}_{j_r,j_s}$$

In case $c_\ell = 1$, this polynomial corresponds to $Q_{H_\ell}(\mathbf{X})_{i_a,i_b}$, where we denote by $H_\ell = (V_\ell, E_\ell, (a,b))$ the multigraph that results from unifying nodes in H that correspond to equal indices in o_ℓ. We therefore have

$$P_H(\mathbf{X})_{i_a,i_b} = Q_H(\mathbf{X})_{i_a,i_b} + \sum_{\ell=2}^{B} c_\ell Q_{H_\ell}(\mathbf{X})_{i_a,i_b}$$

Since for all $\ell \geq 2$ there is at-least one pair of equal indices in o_k, $|V_\ell| \leq m-1$. We can therefore use the induction assumption and express these polynomials using polynomials in $\{P_H\}_{H\in\mathcal{H}_{m-1}}$. This shows that Q_H can be spanned by $\{P_H\}_{H\in\mathcal{H}_m}$. Since $H \in \mathcal{H}_m \setminus \mathcal{H}_{m-1}$ was arbitrary this shows that all $Q_H \in \mathcal{H}_m \setminus \mathcal{H}_{m-1}$ can be spanned by elements in $\{P_H\}_{H\in\mathcal{H}_m}$. Now using the induction assumption again and the fact that $\mathcal{H}_{m-1} \subset \mathcal{H}_m$ we get that

$$\operatorname{span}\{Q_h\}_{H\in\mathcal{H}_m} \subset \operatorname{span}\{P_h\}_{H\in\mathcal{H}_m}$$

as required.

C. Proof of Theorem 3.5

***Theorem** 3.5.* Let H be some multi-graph and $\mathcal{F}_\mathcal{B} \in \{\mathcal{F}_n, \mathcal{F}_e\}$. Further, let H' be the multi-graph resulting after contracting a single node in H using one or more operations from $\mathcal{B}$ to H. Then, H is $\mathcal{F}$-computable iff H' is $\mathcal{F}$-computable.

Proof. We will use Lemma 3.4 and two auxiliary lemmas:

Lemma C.1. *All the tensor contractions used in $\mathcal{F}_n$ and $\mathcal{F}_e$ only affect the 1-ring neighborhood of the contracted node.*

Lemma C.2. *Assumption (I) for H_k and H'_k imply that the k-th node can be contracted from H'_k.*

For conciseness we will use $\mathcal{F} = \mathcal{F}_\mathcal{B}$ to denote a graph model. If H' is $\mathcal{F}$ computable then there exists a sequence of contractions $C_{i_1}, C_{i_2}, \dots, C_{i_k} \in \mathcal{F}$ that contracts H' to the red edge. Then $C, C_{i_1}, C_{i_2}, \dots, C_{i_k}$ is a sequence contracting H to the red edge. Therefore H is computable with $\mathcal{F}$.

The other direction is more challenging. We assume H is computable with $\mathcal{F}$ and need to prove H' is computable with $\mathcal{F}$. $H = (V = [m], E, (a,b))$ has some sequence of tensor contraction contracting all vertices in H until only a and b are left (a and b could be the same node). Without losing generality $a \leq b$, and we assume the order of the node contraction from H is $1, 2, \dots, a-1$. We will say that nodes $i, j \in [m]$ are neighbors (in H) if they share an edge.

A key property we use is proved in Lemma 3.4 that shows that using contractions from $\mathcal{B}$, we can always contract a node if it has at-most 1 and 2 neighbors for $\mathcal{F}_n$ and $\mathcal{F}_e$, respectively.

We have that $H' = (V', E', (a,b))$ resulted from H by contracting a single node using contractions in $\mathcal{F}$. Therefore $|V'| = m-1$, and we let c be the contracted node. Since c is contracted then necessarily $c \neq a, b$. Therefore c belongs to the H contraction series, and in our notation that means $c < a$. Now we will use the series $1, 2, \ldots, \bar{c}, \ldots, a-1$ (a bar indicates a missing index) as a node contraction series for H'. What we need to prove is that this is indeed a series of node contractions that can be implemented with tensor contractions from $\mathcal{B}$.

To show that we will prove by induction the following claim. We will compare the two node contraction sequences:

$$\begin{aligned} H: &\quad 1, 2, \ldots, c-1, c, c+1, \ldots, a-1 \\ H': &\quad 1, 2, \ldots, c-1, \emptyset, c+1, \ldots, a-1 \end{aligned}$$

where $\emptyset$ means no node contraction done. We enumerate these steps using $k = 1, \ldots, a$. We denote by H_k and H'_k the corresponding graphs *before* performing the k-th contraction. So $H_1 = H$, and $H'_1 = H'$. We claim the following holds at the k-th step:

(I) Any pair of nodes i, j where at-least one node is not c or a neighbor of c satisfy: i, j are neighbors in H_K iff they are neighbors in H'_k

Where we define the 1-ring of a node c to be the set of nodes that includes: c and all the nodes that share an edge with c. Before proving this by induction we note that if the induction hypothesis holds at the k-th step then the k-th node can be contracted from H' using operations from $\mathcal{B}$, see Lemma C.2.

Base case, $k = 1$: For $k = 1$ we compare the original H and H'. Consider two nodes i, j not in the 1-ring of c in H. Lemma C.1 asserts that contraction of a node only affect its immediate neighbors. Therefore any edge/no edge between i and j will be identical in H and H'.

Induction step: We assume by the induction assumption that the H_{k-1}, H'_{k-1} satisfy (I) and prove it for H_k, H'_k.

Consider the node $k-1$ that was contracted at the $k-1$ stage. Let $\{d, e\}$ be its neighbor set in H_{k-1} (could be empty, with a single node, or at-most two nodes). There are three cases: (i) $k-1 = c$, (ii) $k-1$ is a neighbor of c in H_{k-1} (i.e., $k-1 \in \{d, e\}$), and (iii) $k-1$ is not in the 1-ring of c in H_{k-1} (i.e., $k-1 \notin \{c, e, d\}$).

In case (i), its contraction will only affect its 1-ring in H_{k-1} (see Lemma C.1), and in H'_{k-1} no contraction will happen. Therefore assumption (I) can be carried to H_k and H'_k.

In case (ii), $k-1$ is a neighbor of c in H_{k-1}. Since the contraction of $k-1$ only affects its 1-ring in H_{k-1} (according to Lemma C.1) and the 1-ring of $k-1$ in H_{k-1} is included, aside of $k-1$, in the 1-ring of c at H_k. Therefore the neighboring relations in H_k and H'_k outside the 1-ring of c in H_k do not change. The induction assumption (I) on H_{k-1} and H'_{k-1} now implies the assumption holds for H_k and H'_k .

In case (iii), $k-1$ is not in the 1-ring of c in H_{k-1}. Then Lemma C.1 implies that the 1-ring of c in H_k will not change and the neighborhood changes in the 1-ring of $k-1$ will be identical to H_k and H'_k due to induction assumption (I). $\square$

***Lemma** 3.4.* $\mathcal{F}_n$ (for simple graphs) and $\mathcal{F}_e$ (for general graphs) can always contract a node in H iff its number of neighbors is at-most 1 and 2, respectively.

Proof. We start with $\mathcal{F}_n$: We will use contraction notations from the contraction banks presented in Figure 7, left. For simple graphs H is simple (see Section 2.1), and therefore does not have parallel edges. Applications of contractions from the bank of $\mathcal{F}_n$ cannot introduce parallel edges and therefore any two neighbors in the graph will share a single edge. Furthermore, using C_2 we can always reduce the number of self-loops generated during the tensor computation path to 1. Lastly, any node, with or without a single self-loop, and with at-most 1 neighbor is connected to it with at-most a single edge, and therefore C_3 or C_4 will be able to contract it.

For $\mathcal{F}_e$: We will use contraction notations from the contraction bank in Figure 7, right. First note that any number of self-loops and parallel edges can be reduced to 1 using C_2 and C_6, respectively. Now if a node with a single self-loop has no neighbors in H then is can be contracted with C_1. Now in the case a node i has 1 or 2 neighbors in H we can cancel its self-loop (if it has one) as follows. Let j denote one of its neighbors. Then we first apply C_5 between i (top) and j (bottom), then we apply C_6 if necessary to make the existing edge between i, j directing towards j, and lastly apply C_6 to have a single edge between i and j.

Lets recap: we have now a node i, without self-loops, with a single edge going to its 1 or 2 neighbors. We can change the direction of these edges by applying C_6, if required. Now, we can use C_3 or C_7 to contract node i.

In the other direction if the number of neighbors is greater than 1 for $\mathcal{F}_n$ and 2 for $\mathcal{F}_e$ then inspection of the respective contraction banks shows that edges cannot be completely removed between nodes without contraction and no contraction operators for nodes with valence 2 and 3 exists for $\mathcal{F}_n$ and $\mathcal{F}_e$, respectively. □

***Lemma** C.1.* All the tensor contractions used in $\mathcal{F}_n$ and $\mathcal{F}_e$ only affect the 1-ring neighborhood of the contracted node.

Proof. Inspection of the tensor contraction banks of $\mathcal{F}_n$ and $\mathcal{F}_e$ (see Figure 7, contracted nodes are in gray) shows that any contraction of a node can introduce new edges in its 1-ring but does not affect neighboring relation outside the 1-ring. □

***Lemma** C.2.* Assumption (I) for H_k and H'_k imply that the k-th node can be contracted from H'_k.

Proof. Indeed, there are 3 options for the k-th node: (i) $k = c$, (ii) k is a neighbor of c in H_k, and (iii) k is not in the 1-ring of c in H_k.

Since k can be contracted from H_k by definition, Lemma 3.4 imply that we only need to show that k has at-most the same number of neighbors in H'_k in order to prove the lemma. We show that next.

In case (i): since $k = c$ no contraction is to take place in H'_k. In case (ii): hypothesis (I) imply that the number of neighbors of k in H'_k is at most that in H_k. In case (iii): Hypothesis (I) implies that k has the same neighbors in H_k and H'_k. □

D. Proof of Theorem 3.7

To prove Theorem 3.7 it is enough to show that MPNN and PPGN++ can approximate any polynomial computable by the matching Prototypical models, namely node based $\mathcal{F}_n$ and edge based $\mathcal{F}_e$. We show that in the following theorem:

Theorem D.1. *For any compact input domain, PPGN++ and MPNN can arbitrarily approximate any polynomial computable by the Prototypical graph models $\mathcal{F}_e$ and $\mathcal{F}_n$, respectively.*

Proof. The proof of this theorem has two parts. First, we will show that for $\epsilon > 0$, a single layer of PPGN++ and MPNN can approximate any contraction operation $C \in \mathcal{B}$ of the corresponding graph model. The second part will show that a composition of those layers can approximate any finite sequence $f \in \mathcal{F}_\mathcal{B}$, i.e any polynomial computable by $\mathcal{F}_\mathcal{B}$.

Part I. Let $\Omega \subset \mathbb{R}^{n\times d}$ be an arbitrary compact set and $\mathbf{Y}_i \in \mathbb{R}^n$ be the ith column of $\mathbf{Y} \in \Omega$. Consider $C \in \mathcal{B}_n$, the tensor contraction bank of $\mathcal{F}_n$. We will show that a single layer of MPNN (eq 9) can approximate C. To do so we will write the operations explicitly and verify that a MPNN layer can approximate them:

- $C_1 \to \mathbf{Y}_j^{(k+1)} = \mathbf{1}\mathbf{1}^T\mathbf{Y}_i^{(k)}$.
- $C_2 \to \mathbf{Y}_l^{(k+1)} = \mathbf{Y}_i^{(k)} \odot \mathbf{Y}_j^{(k)}$ where $\odot$ is element-wise product.
- $C_3 \to \mathbf{Y}_j^{(k+1)} = \mathbf{X}\mathbf{1}$.
- $C_4 \to \mathbf{Y}_j^{(k+1)} = \mathbf{X}\mathbf{Y}_i^{(k)}$.

In order for MPNN approximate the mentioned functions we need to argue that m can approximate several functions. We assume that simple functions, such as constant functions and feature retrieving, can be computed exactly for m. To justify approximation of element-wise product we use the universal approximation theorem (Hornik, 1991):

Theorem D.2. *The set of one hidden layer MLPs with a continuous σ, i.e, $M(\sigma) = span\left\{\sigma(w^T x + b) | w \in \mathbb{R}^n, b \in \mathbb{R}\right\}$ is dense in $C(\mathbb{R}^n)$ in the topology of uniform convergence over compact sets if and only if σ is not a polynomial.*

In the next case we set $\Omega \subset \mathbb{R}^{n^2\times d}$ be an arbitrary compact set ($\mathbf{Z}_i \in \mathbb{R}^{n^2}$ for $\mathbf{Z} \in \Omega$). We repeat the same process for PPGN++ (eq 10) and $C \in \mathcal{B}$ of $\mathcal{F}_e$:

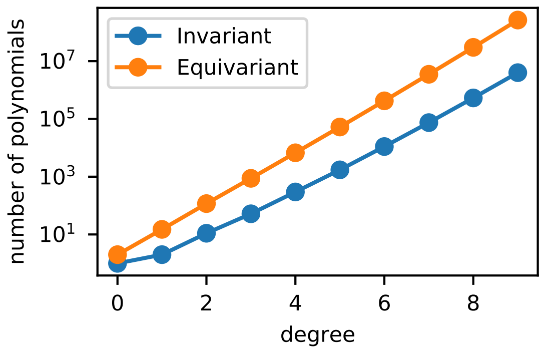


Figure 11. The number of graph invariant and equivariant polynomials scales exponentially with the degree of the polynomial. Here, we count the number of polynomials in the asymptotic limit of graphs of $n \to \infty$ nodes. Graphs with a fixed number of nodes will have fewer invariant polynomials. See Appendix I for further details on the generating functions and explicit counts.

- $C_1 \to \mathbf{Z}_j^{(k+1)} = \mathbf{1}\mathbf{1}^T(\text{diag}(\mathbf{Z}_i^{(k)})\mathbf{1}\mathbf{1}^T)$.
- $C_2 \to \mathbf{Z}_l^{(k+1)} = \text{diag}(\mathbf{Z}_i^{(k)} \cdot \mathbf{Z}_j^{(k)})$.
- $C_3 \to \mathbf{Z}_j^{(k+1)} = \text{diag}(\mathbf{1}\mathbf{1}^T\mathbf{Z}_i^{(k)})$.
- $C_4 \to \mathbf{Z}_j^{(k+1)} = \text{diag}(\mathbf{Z}_i^{(k)})\mathbf{1}\mathbf{1}^T$.
- $C_5 \to \mathbf{Z}_l^{(k+1)} = \mathbf{Z}_i^{(k)} \cdot \mathbf{Z}_j^{(k)}$.
- $C_6 \to \mathbf{Z}_j^{(k+1)} = \mathbf{Z}_i^{(k)T}$.
- $C_7 \to \mathbf{Z}_l^{(k+1)} = \mathbf{Z}_i^{(k)}\mathbf{Z}_j^{(k)}$.

Here the only addition is the diag function, which given a matrix returns a diagonal matrix with the matrix diagonal. This could computed using $\bar{\mathfrak{m}}_i$ since it computes different functions for the diagonal and off-diagonal elements.

Part II. This part of the proof will show that any sequence $f \in \mathcal{F}_e$ (or $\in \mathcal{F}_n$), namely a polynomial computable by this graph model, can be approximated by a composition of PPGN++ (or MPNN) layers. To prove that we can use Lemma 6 from (Lim et al., 2022) that states the following:

Lemma D.3 (Layer-wise universality implies universality)**.** *Let $\mathcal{Z} \subseteq \mathbb{R}^{d_0}$ be a compact domain, let $\mathcal{F}_1, \cdots, \mathcal{F}_L$ be a families of continuous functinos where $\mathcal{F}_i$ consists of functions from $\mathbb{R}^{d_{i-1}} \to \mathbb{R}^{d_i}$ for some $d_1, \cdots, d_L$. Let $\mathcal{F}$ be a family of functions $\{f_L \circ \cdots \circ f_1 : \mathcal{Z} \to \mathbb{R}^{d_L}, f_i \in \mathcal{F}_i\}$ that are compositions of functions $f_i \in \mathcal{F}_i$.*

For each i, let Φ_i, be a family of continuous functions that universally approximates $\mathcal{F}_i$. Then the family of compositions $\{\phi_L \circ \cdots \circ \phi_1 : \phi_i \in \Phi_i\}$ universally approximates $\mathcal{F}$.

Based on the first part proof, using this lemma while pluging in $\mathcal{B}$ as $\mathcal{F}_i$ for every i and the corresponding GNN layer as Φ_i (also for every i) shows that PPGN++ and MPNN are universal approximators for $\mathcal{F}_e$ and $\mathcal{F}_n$, respectively.

□

E. Proof of Proposition 3.8

Let $f : \mathbb{R}^{n^2 \times d} \to \mathbb{R}^{n^2 \times d'}$ be a PPGN block, defined by the following equation (as portrayed in (Maron et al., 2019)):

$$\mathbf{Z}^{(k+1)} = \bar{\mathfrak{m}}_3 \left[\bar{\mathfrak{m}}_1(\mathbf{Z}^{(k)}) \circledast \bar{\mathfrak{m}}_2(\mathbf{Z}^{(k)}), \mathbf{Z}^{(k)} \right]. \tag{19}$$

Also, let

$$\mathbf{Z} = \begin{bmatrix} a & a \\ b & b \end{bmatrix} \; a, b \in \mathbb{R}$$

to be an input tensor. A PPGN network (composition of PPGN blocks) cannot approximate the transpose operator $P_H(\mathbf{Z}) = \mathbf{Z}^T$ due to the fact that the PPGN block maintains the row structure of $\mathbf{Z}$, i.e

$$f(\mathbf{Z})_i = \begin{bmatrix} x & x \\ y & y \end{bmatrix} \; x, y \in \mathbb{R},$$

when $f(\mathbf{Z})_i \in \mathbb{R}^{n^2}$ (for $i \in [d']$) denotes the ith slice of $f(\mathbf{Z})$ along the last dimension. This holds since each PPGN block is composed of Siamese element-wise operations and matrix multiplications which preserve this structure. a simple induction can generalize this claim for a composition of blocks while the extension for larger size graphs is also trivial.

F. Equivariant Polynomials of Attributed Graphs

Repeating the derivations in Appendix B for the case of equivariant polynomials of attributed graphs, $P : \mathbb{R}^{n^2 \times f} \to \mathbb{R}^{n^2}$, suggests that we should enumerate each Q_H and P_H with a multi-graph $H = (V, E, (a, b))$, where there are also f types of edges *types* (e.g., colors); we denote by $(r, s; k) \in E$ an edge (r, s) with type $k \in [f]$. This gives the formulas

$$Q_H(\mathbf{X})_{i_a,i_b} = \sum_{\substack{j_1 \neq \ldots \neq j_m \in [n] \\ j_a = i_a, j_b = i_b}} \prod_{(r,s;k) \in E} \mathbf{X}_{j_r,j_s,k} \tag{20}$$

$$P_H(\mathbf{X})_{i_a,i_b} = \sum_{\substack{j_1, \ldots j_m \in [n] \\ j_a = i_a, j_b = i_b}} \prod_{(r,s;k) \in E} \mathbf{X}_{j_r,j_s,k} \tag{21}$$

where the degree of P_H, Q_H is the total number of edges of all types, counting multiplicities. The basis $\{Q_H\}$ (and consequently $\{P_H\}$) for equivariant polynomials in this case is achieved by considering all non-isomorphic H (comparing both edge types and multiplicity) with total number of edges up to f.

Equivariant maps $\mathbb{R}^{n^2 \times f} \to \mathbb{R}^{n^2}$ are isomorphically equivalent to the module $(\mathbb{R}[\mathbb{R}^{n^2 \times f}] \times \mathbb{R}^{n^2})^{S_n}$, where $\mathbb{R}[\mathbb{R}^{n^2 \times f}]$ is the polynomial ring on the vector space $\mathbb{R}^{n^2 \times f}$. Similarly to the proof in Appendix B, via the Reynolds operator applied to monomials in variables $\mathbf{X}_{ijk}$, we obtain orbits which as before, correspond to unique subgraphs. For a given monomial, if the variable $\mathbf{X}_{ijk}$ is contained in that monomial, then we add an edge between i to j and color that edge according to the index k. The k-index remains invariant under the group operation and is not permuted by the group action.

To continue the graphical notation, we label this basis by labeling its orbits according to the monomials that appear. We pick a given monomial in the orbit and then for each variable in that monomial, we add an edge with the appropriate color. As before, the equivariant output dimension is colored red, but we make such an edge dotted to more clearly differentiate it with other edges.

In this expanded graphical notation, we provide two examples below:

The proof that the above forms a basis follows directly from the proof in Appendix B. We follow the basic steps below.

We denote by the vector space of all polynomials $P : \mathbb{R}^{n^2 \times f} \to \mathbb{R}^{n^2}$ by $\mathfrak{P} = \mathbb{R}^{n^2 \times f} \otimes \mathbb{R}[\mathbf{X}]$, where $\otimes$ is the tensor product and $\mathbb{R}[\mathbf{X}]$ denotes the module of polynomials with indeterminate $\mathbf{X}_{11}, \ldots, \mathbf{X}_{nn}$. The space of polynomials $\mathfrak{P}$ is now spanned by the expanded monomial basis

$$M(\mathbf{X}) = \delta^{a,b} \otimes \prod_{r,s=1}^{n} \prod_{k=1}^{f} \mathbf{X}_{r,s,k}^{\mathbf{A}_{r,s,k}} \tag{22}$$

where $\mathbf{A} \in \mathbb{N}_0^{n^2 \times f}$, $\mathbb{N}_0 = \{0, 1, \ldots\}$, and $\delta^{a,b} \in \mathbb{R}^{n^2}$ is a matrix satisfying

$$\delta_{i,j}^{a,b} = \begin{cases} 1 & \text{if } a = i, b = j \\ 0 & \text{o/w} \end{cases}$$

Permuations now only act on the first two indices, i.e.,

$$(g \cdot \mathbf{X})_{i,j,k} = \mathbf{X}_{g^{-1}(i),g^{-1}(j),k}. \tag{23}$$

Given the last index is invariant to permutations, symmetrization of monomials continues as before where we add the feature dimension:

$$\begin{aligned} Q_H(\mathbf{X})_{i,j} &= \sum_{g \in S_n} \left[g \cdot M_H(g^{-1} \cdot \mathbf{X}) \right]_{i,j} \\ &= \sum_{g \in S_n} \delta_{i,j}^{g(a),g(b)} \prod_{r,s=1}^{n} \prod_{k=1}^{f} \mathbf{X}_{r,s,k}^{\mathbf{A}_{g^{-1}(r),g^{-1}(s),k}}. \end{aligned}$$

Q_H is a sum over all monomials corresponding to the orbit of H under node relabeling g. This forms an equivalence class over orbits of H (i.e., two graphs H and H' are in the same class if they can be obtained from one another via permutations). Since every polynomial is a sum over monomials, symmetrization over these monomials implies each symmetrization falls into one of these orbits. Therefore, similar to the proof as before, the multigraphs H compose the set of equivariant polynomials.

F.1. Equivariant set polynomials

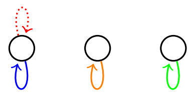
As a note, we show here via an example how to form set polynomials from the structure described before. In correspondence with the equivariant polynomials on sets ($\mathbb{R}^{n\times d} \to \mathbb{R}^n$), Segol & Lipman (2019) proved any polynomial in this setting takes the following form:

Theorem F.1 (Theorem 2 of (Segol & Lipman, 2019), paraphrased)**.** *Any equivariant map on sets $\mathbb{R}^{n\times d} \to \mathbb{R}^n$ can be generated by polynomials of the form*

$$P(\mathbf{X}) = \sum_{|\boldsymbol{\alpha}| \le n} \boldsymbol{b}_{\boldsymbol{\alpha}} q_{\boldsymbol{\alpha},j}(s_1, \dots, s_t), \tag{24}$$

where $\boldsymbol{b}_{\boldsymbol{\alpha}} = [\boldsymbol{x}_1^{\boldsymbol{\alpha}}, \dots, \boldsymbol{x}_n^{\boldsymbol{\alpha}}]^\top$, $s_j(\mathbf{X}) = \sum_{i=1}^{n} \boldsymbol{x}_i^{\boldsymbol{\alpha}_j}$ are the power sum symmetric polynomials indexed by $j \in \left[\binom{n+d}{d}\right]$ possible such polynomials up to degree d, and $q_{\boldsymbol{\alpha},j}(s_1, \dots, s_t)$ is a polynomial in its power sum polynomial inputs.

In our graphical language, set polynomials correspond to graphs with only multi-edges that are self loops. To recover the above theorem in our graphical notation, we consider each element in the sum above. We identify a given $\boldsymbol{b}_{\boldsymbol{\alpha}}$ with the self loops on the equivariant edge with the dotted line. The polynomial $q_{\boldsymbol{\alpha},j}$ is identified with the polynomial on the rest of the nodes, e.g. let us consider the below graph.

 (25)

As before, for colors indexed by index zero (orange), index one (green), and index two (blue), the above corresponds to the polynomial

$$P(\mathbf{X}) = \begin{bmatrix} \mathbf{X}_{1,2} \\ \mathbf{X}_{2,2} \\ \vdots \\ \mathbf{X}_{n,2} \end{bmatrix} \cdot \left(\sum_{i,j=1}^{n} \mathbf{X}_{i,0} \mathbf{X}_{j,1} \right). \tag{26}$$

By inspection, one can see that the above is of the form as stated in (Segol & Lipman, 2019), i.e. choose $q_{\boldsymbol{\alpha},j} = (\sum_{i=1}^{n} \boldsymbol{x}_i^{[1,0,0]})(\sum_{i=1}^{n} \boldsymbol{x}_i^{[0,1,0]})$ only for $\boldsymbol{\alpha} = [0,0,1]$ and $q_{\boldsymbol{\alpha},j} = 0$ otherwise.

G. Relationship between Equivariant Polynomials, Homomorphisms, and Subgraph Counting

There is a close correspondence between homomorphism counts, subgraph counts, and the evaluation of our polynomials on binary graphs $\mathbf{X}$, meaning directed graphs with self-loops but no multiedges, i.e. those graphs with adjacency in

$\{0,1\}^{n\times n}$. For binary graphs H and $\mathbf{X}$, a homomorphism is a function $\varphi : V(H) \to V(\mathbf{X})$ such that if $(r,s) \in E(H)$, then $(\varphi(r), \varphi(s)) \in E(\mathbf{X})$. An isomorphism is a bijective homomorphism whose inverse is also a homomorphism. We let $\mathrm{hom}(H, \mathbf{X})$ denote the number of homomorphisms from H to $\mathbf{X}$. We let $\mathrm{inj}(H, \mathbf{X})$ denote the number of injective homomorphisms from H to $\mathbf{X}$. The injective homomorphism number is closely related to subgraph counts. If we let $\mathrm{count}(H, \mathbf{X})$ denote the number of subgraphs isomorphic to H, and let $\mathrm{Aut}(H)$ denote the automorphism group of H (i.e. the set of isomorphisms from H to H), then $\mathrm{inj}(H, \mathbf{X}) = |\mathrm{Aut}(H)| \cdot \mathrm{count}(H, \mathbf{X})$. The $|\mathrm{Aut}(H)|$ term is due to overcounting when H has symmetries.

G.1. Invariant Polynomials and Standard Homomorphisms

P_H and standard homomorphisms. Let H and $\mathbf{X}$ be binary graphs. Then it can be seen that the homomorphism count $\mathrm{hom}(H, \mathbf{X})$ can be written as (Lovász, 2012):

$$\mathrm{hom}(H, \mathbf{X}) = \sum_{\varphi : V(H) \to V(\mathbf{X})} \prod_{(r,s) \in E(H)} \mathbf{X}_{\varphi(r), \varphi(s)}, \tag{27}$$

where the sum ranges over all functions from $V(H)$ to $V(\mathbf{X})$. Choose an ordering $1, \ldots, m$ of the nodes of $V(H)$ and $1, \ldots, n$ of the nodes of $\mathbf{X}$; writing $\varphi(l) = j_l$, we see that $\mathrm{hom}(H, \mathbf{X})$ is exactly equivalent to our (invariant) polynomial basis element P_H:

$$P_H(\mathbf{X}) = \mathrm{hom}(H, \mathbf{X}) = \sum_{j_1, \ldots, j_m \in [n]} \prod_{(r,s) \in E(H)} \mathbf{X}_{j_r, j_s}. \tag{28}$$

If H has multiple edges, then let $\tilde{H}$ be the graph H with any multiple edges reduced to a single edge. If $\mathbf{X}$ is still a binary graph, it is easy to see that $P_H(\mathbf{X}) = P_{\tilde{H}}(\mathbf{X})$, so in this case $P_H(\mathbf{X}) = \mathrm{hom}(\tilde{H}, \mathbf{X})$.

Q_H and injective homomorphisms / subgraph counts. Once again, let H and $\mathbf{X}$ be binary graphs. The injective homomorphism number can be written in a similar form to the homomorphism count (Lovász, 2012):

$$\mathrm{inj}(H, \mathbf{X}) = \sum_{\substack{\varphi : V(H) \to V(\mathbf{X}) \\ \varphi \text{ injective}}} \prod_{(r,s) \in E(H)} \mathbf{X}_{\varphi(r), \varphi(s)}. \tag{29}$$

In this case, the sum ranges over all injective functions $\varphi : V(H) \to V(\mathbf{X})$. As in the non-injective case, we write $j_l = \varphi(l)$ for each $l = 1, \ldots, m$. By injectivity $j_1 \neq \ldots \neq j_m$. Thus, we have a corrspondence between $\mathrm{inj}(H, \mathbf{X})$ and our invariant basis polynomial Q_H:

$$Q_H(\mathbf{X}) = \mathrm{inj}(H, \mathbf{X}) = \sum_{j_1 \neq \ldots \neq j_m \in [n]} \prod_{(r,s) \in E(H)} \mathbf{X}_{\varphi(r), \varphi(s)} = |\mathrm{Aut}(H)| \cdot \mathrm{count}(H, \mathbf{X}). \tag{30}$$

G.2. Equivariant Polynomials and Homomorphism Tensors

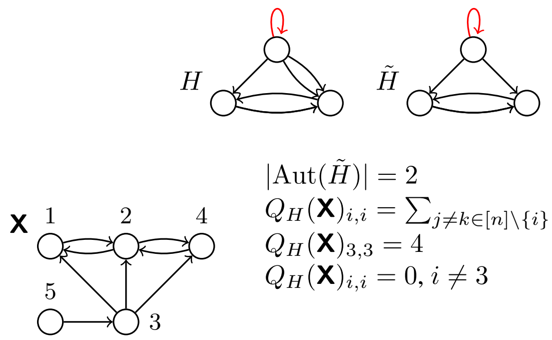
Here, we consider our equivariant polynomials basis elements $P_H, Q_H : \mathbb{R}^{n^2} \to \mathbb{R}^{n^2}$. On binary graphs, this will also correspond to counts of homomorphisms, except now we have to restrict the homomorphisms to preserve the red edge in an equivariant way. Let (a, b) be the red edge of H, and consider any two nodes (possibly the same) i_a, i_b of $\mathbf{X}$. Then we may define a tensor of homomorphism counts $\mathrm{Hom}(H, \mathbf{X}) \in \mathbb{R}^{n^2}$, where $\mathrm{Hom}(H, \mathbf{X})_{i_a, i_b}$ is the number of homomorphisms $\varphi : V(H) \to V(\mathbf{X})$ such that $\varphi(a) = i_a$ and $\varphi(b) = i_b$. We define a tensor $\mathrm{Inj}(H, \mathbf{X})$ of injective homomorphism counts similarly.

Following the arguments for the invariant case, it is easy to see that

$$P_H(\mathbf{X})_{i_a, i_b} = \mathrm{Hom}(H, \mathbf{X})_{i_a, i_b} \tag{31}$$

$$Q_H(\mathbf{X})_{i_a, i_b} = \mathrm{Inj}(H, \mathbf{X})_{i_a, i_b}. \tag{32}$$

Thus, this gives an interpretation of $P_H(\mathbf{X})_{i_a, i_b}$ and $Q_H(\mathbf{X})_{i_a, i_b}$, when H and $\mathbf{X}$ are binary graphs. We expand on this interpretation for Q_H in the next section.



$|\mathrm{Aut}(\tilde{H})| = 2$
$Q_H(\mathbf{X})_{i,i} = \sum_{j \neq k \in [n] \setminus \{i\}} \mathbf{X}_{i,j} \mathbf{X}_{i,k}^2 \mathbf{X}_{j,k}^2$
$Q_H(\mathbf{X})_{3,3} = 4$
$Q_H(\mathbf{X})_{i,i} = 0,\, i \neq 3$

Figure 12. Relationship between Q_H and subgraph counts. The directed multigraph H (top left) has its multiedges turned to single edges to form $\tilde{H}$ (top right). The size of the automorphism group $\mathrm{Aut}(\tilde{H})$ is 2, as we may swap the bottom left and bottom right node while preserving the graph structure. $Q_H(\mathbf{X})_{3,3} = 4$ because node 3 participates in 2 subgraphs isomorphic to $\tilde{H}$, and this subgraph count is scaled by $|\mathrm{Aut}(\tilde{H})| = 2$ to get $Q_H(\mathbf{X})_{3,3}$.

G.3. Q_H as Subgraph Counts

The basis Q_H can be interpreted as subgraph counts when the input is a simple binary graph $\mathbf{X} \in \{0,1\}^{n^2}$. For any directed multigraph H with red edge (a,b), let $\tilde{H}$ denote the same graph H but where any multiple black edges are collapsed to just one black edge; the difference between H and $\tilde{H}$ is that if H has more than one black edge from node i to j, then $\tilde{H}$ only has one edge. Then we have that

$$Q_H(\mathbf{X})_{i_a,i_b} = |\mathrm{Aut}(\tilde{H})| \cdot \mathrm{count}(\tilde{H}, \mathbf{X}, (i_a, i_b)), \tag{33}$$

where $\mathrm{count}(\tilde{H}, \mathbf{X}, (i_a, i_b))$ is the number of subgraphs of $\mathbf{X}$ that are isomorphic to $\tilde{H}$, after adding a red edge (i_a, i_b) to $\mathbf{X}$ and labelling edge (a,b) in $\tilde{H}$ as red. $|\mathrm{Aut}(\tilde{H})|$ is the size of the automorphism group of $\tilde{H}$, which contains the automorphisms $\varphi : V(H) \to V(H)$ that have $\varphi(a) = a$ and $\varphi(b) = b$.

Now, suppose H has a red self loop (a,a), and let i be any node in $\mathbf{X}$. We have that $Q_H(\mathbf{X})_{i_a,i_a}$ is equal to $|\mathrm{Aut}(\tilde{H})|$ multiplied by the number of subgraphs of $\mathbf{X}$ that are isomorphic to $\tilde{H}$, where the isomorphism maps i_a in $\mathbf{X}$ to a (the node with the self loop in H). Intuitively, this is proportional to the number of subgraphs isomorphic to $\tilde{H}$ that i_a participates in as the designated red-self-loop node. See Figure 12 for an illustration.

Derivation. Here, we derive the relationship between $Q_H(\mathbf{X})$ and subgraph counts. First, we write out some definitions more precisely. Let $\mathrm{Aut}(\tilde{H})$ denote the automorphism group of $\tilde{H}$. This is the set of permutations $\sigma : V(\tilde{H}) \to V(\tilde{H})$ such that $\sigma(a) = a$, $\sigma(b) = b$, and $(r,s) \in E(\tilde{H})$ if and only if $(\sigma(r), \sigma(s)) \in E(\tilde{H})$. Further, let $\mathrm{count}(\tilde{H}, \mathbf{X}, (i_a, i_b))$ denote the number of subgraphs of $\mathbf{X}$ isomorphic to $\tilde{H}$, where the isomorphism maps a to i_a and b to i_b. In other words, it is the number of choices (V', E') such that $V' \subseteq V(\mathbf{X})$ and $E' \subseteq E(\mathbf{X}) \cap (V' \times V')$ where there exists a bijection $\varphi : V(\tilde{H}) \to V'$ such that $\varphi(a) = i_a$, $\varphi(b) = i_b$, and $(r,s) \in E(\tilde{H})$ if and only if $(\varphi(r), \varphi(s)) \in E'$. We call any such map φ a subgraph isomorphism, and we may also view it as an injective function $V(\tilde{H}) \to V(\mathbf{X})$.

We will use the fact that for any subgraph isomorphism φ and any automorphism $\sigma \in \mathrm{Aut}(\tilde{H})$, the map $\varphi \circ \sigma : V(\tilde{H}) \to V(\mathbf{X})$ is also a subgraph isomorphism. Let (V', E') be the subgraph that φ maps to. To see that φ is a subgraph isomorphism, first note that $\varphi \circ \sigma$ is injective, $\varphi \circ \sigma(a) = \varphi(a) = i_a$, and $\varphi \circ \sigma(b) = \varphi(b) = i_b$. To see the edge preserving property, note that $(r,s) \in E(\tilde{H})$ if and only if $(\sigma(r), \sigma(s)) \in E(\tilde{H})$ since $\sigma \in \mathrm{Aut}(\tilde{H})$. Moreover, $(\sigma(r), \sigma(s)) \in E(\tilde{H})$ if and only if $(\varphi(\sigma(r)), \varphi(\sigma(s))) \in E'$ since φ is a subgraph isomorphism. Thus, $\varphi \circ \sigma$ is a subgraph isomorphism.

We now derive the formula for $Q_H(\mathbf{X})$ in terms of subgraph counts:

Proposition G.1. *If* $\mathbf{X} \in \{0,1\}^{n^2}$, *then*

$$Q_H(\mathbf{X})_{i_a,i_b} = |\mathrm{Aut}(\tilde{H})| \cdot \mathrm{count}(\tilde{H}, \mathbf{X}, (i_a, i_b)) \tag{34}$$

Proof. Note that we can write

$$Q_H(\mathbf{X})_{i_a,i_b} = \sum_{\substack{j_1 \neq \cdots \neq j_m \\ j_a = i_a, j_b = i_b}} \prod_{(r,s) \in E(\tilde{H})} \mathbf{X}_{j_r, j_s}, \tag{35}$$

where we replace the product over $E(H)$ with the product over $E(\tilde{H})$, due to the assumption that $\mathbf{X}$ is binary. Further, note that the summand $\prod_{(r,s)\in E} \mathbf{X}_{j_r,j_s}$ is either 0 or 1 for each setting of $(j_1, \ldots, j_m)$. We show that the number of nonzero terms of the sum in $Q_H(\mathbf{X})_{i_a,i_b}$ is equal to $|\mathrm{Aut}(\tilde{H})| \cdot \mathrm{count}(\tilde{H}, \mathbf{X}, (i_a, i_b))$.

Let $(j_1, \ldots, j_m)$ correspond to a nonzero term in the sum. Define $\varphi : V(H) \to V(\mathbf{X})$ by $\varphi(i) = j_i$. Note that φ is injective since $j_1 \neq \ldots \neq j_m$. Moreover, $\varphi(a) = j_a = i_a$ and $\varphi(b) = j_b = i_b$. Now, define

$$V_\varphi = \{\varphi(i) : i \in V(H)\}, \quad E_\varphi = \{(\varphi(r), \varphi(s)) : (r,s) \in E(\tilde{H})\}. \tag{36}$$

This is the vertex set and edge set of a subgraph in $\mathbf{X}$, because $\prod_{(r,s)\in E(\tilde{H})} \mathbf{X}_{j_r,j_s} = 1$ implies that $(\varphi(r), \varphi(s)) \in E(\mathbf{X})$ for all $(r,s) \in E(\tilde{H})$. Thus, φ corresponds to an isomorphism between $\tilde{H}$ and a subgraph of $\mathbf{X}$.

Suppose $(j_1, \ldots, j_m) \neq (\tilde{j}_1, \ldots, \tilde{j}_m)$ are both indices corresponding to nonzero summands, with corresponding subgraph isomorphisms φ and $\tilde{\varphi}$. Note that $\varphi \neq \tilde{\varphi}$, so each nonzero summand corresponds to a unique subgraph isomorphism φ. Hence, it suffices to show that the number of subgraph isomorphisms is $|\mathrm{Aut}(\tilde{H})| \cdot \mathrm{count}(\tilde{H}, \mathbf{X}, (i_a, i_b))$.

For each $l \in \{1, \ldots, \mathrm{count}(\tilde{H}, \mathbf{X}, (i_a, i_b))\}$, let $G_l = (V_l, E_l)$ be a subgraph of $\mathbf{X}$ that is isomorphic to $\tilde{H}$. Then choose a subgraph isomorphism $\varphi_l : V(H) \to V(\mathbf{X})$ associated to G_l. For this φ_l, as in our argument above we know that $\varphi_l \circ \sigma : V(H) \to V(\mathbf{X})$ is a subgraph isomorphism for every $\sigma \in \mathrm{Aut}(\tilde{H})$. Thus, there are at least $|\mathrm{Aut}(\tilde{H})| \cdot \mathrm{count}(\tilde{H}, \mathbf{X}, (i_a, i_b))$ subgraph isomorphisms $\varphi_l \circ \sigma$.

To show that there are at most $|\mathrm{Aut}(\tilde{H})| \cdot \mathrm{count}(\tilde{H}, \mathbf{X}, (i_a, i_b))$ subgraph isomorphisms, assume for sake of contradiction that $\psi : V(H) \to V(\mathbf{X})$ is a subgraph isomorphism that is not of the form $\varphi_l \circ \sigma$ above. Denote the vertex set and edge set of the associated subgraph as V_ψ and E_ψ, respectively. If $(V_\psi, E_\psi) \neq (V_l, E_l)$ for each l, then we have the existence of $\mathrm{count}(\tilde{H}, \mathbf{X}, (i_a, i_b)) + 1$ subgraphs in $\mathbf{X}$ isomorphic to $\tilde{H}$, which contradicts the definition of count. Thus, $(V_\psi, E_\psi) = (V_l, E_l)$ for some l. As φ_l and ψ are both bijective from $V(\tilde{H}) \to V_\psi$, there is a unique bijection $\sigma : V(\tilde{H}) \to V(\tilde{H})$ such that $\varphi_l = \psi \circ \sigma$.

We will show that $\sigma \in \mathrm{Aut}(\tilde{H})$, which contradicts our definition of ψ. Note that $\sigma(a) = a$ and $\sigma(b) = b$, because $\varphi_l(a) = \psi(a) = i_a$ and $\varphi_l(b) = \psi(b) = i_b$. Now, suppose $(r,s) \in E(\tilde{H})$. Since $(\psi \circ \sigma(r), \psi \circ \sigma(s)) = (\varphi(r), \varphi(s)) \in E_{\varphi_l} = E_\psi$, we know that $(\sigma(r), \sigma(s)) \in E(\tilde{H})$ as ψ is a subgraph isomorphism. On the other hand, if $(\sigma(r), \sigma(s)) \in E(\tilde{H})$, then $(\varphi_l(r), \varphi_l(s)) = (\psi \circ \sigma(r), \psi \circ \sigma(s)) \in E_\psi$ since ψ is a subgraph isomorphism, so that $(r,s) \in E(\tilde{H})$ because φ_l is a subgraph isomorphism. Thus, $\sigma \in \mathrm{Aut}(\tilde{H})$, and we are done.

□

H. k-WL Equivalence

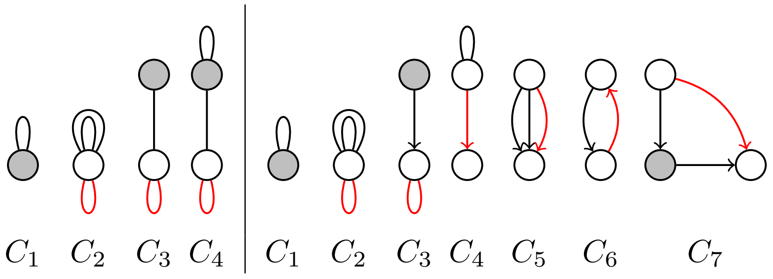


Figure 13. For convenience, we redraw our Prototypical graph models here. We show these the node-based model (left) is equivalent to 1-WL on simple graphs, and the edge-based model (right) is equivalent to 2-FWL / 3-WL on simple graphs.

In this section, we demonstrate that our studied Prototypical graph models achieve 1-WL and 3-WL/2-FWL expressive power, thus showing that our framework can be used to design and analyze k-WL style models. The key connection comes from a result of Dvořák (2010); Dell et al. (2018), which states that k-FWL indistinguishability is equivalent to $\mathrm{hom}(H, \mathbf{X})$ indistinguishability for all H of tree-width at most k.

Lemma H.1 (Dvořák (2010); Dell et al. (2018)). *Two simple graphs $\mathbf{X}^{(1)}$ and $\mathbf{X}^{(2)}$ are k-FWL distinguishable if and only if there is a graph H of tree-width at most k such that* $\mathrm{hom}(H, \mathbf{X}^{(1)}) \neq \mathrm{hom}(H, \mathbf{X}^{(2)})$

Recall from Appendix G that $\mathrm{hom}(H, \mathbf{X})$ is equal to the evaluation of the invariant polynomial $P_H(\mathbf{X})$ when $\mathbf{X} \in \{0,1\}^{n \times n}$. Thus, an Prototypical graph model can compute $\mathrm{hom}(H, \mathbf{X})$ if it can contract H into the trivial graph (that has zero nodes and zero edges) using contractions from its bank.

Proposition H.2. *The Prototypical node-based graph model can distinguish any two simple graphs if and only if 1-WL can.*

Proof. ($\Longleftarrow$) Suppose 1-WL can distinguish two simple graphs $\mathbf{X}^{(1)}$ and $\mathbf{X}^{(2)}$. Then there is a graph H of tree-width 1 such that $\mathrm{hom}(H, \mathbf{X}^{(1)}) \neq \mathrm{hom}(H, \mathbf{X}^{(2)})$. We will show that the node-based Prototypical model can contract H to the trivial graph. As H has tree-width 1, it is a forest or a tree. We can assume it is a tree as we can contract each tree connected component one by one if it is a forest. We will show that the node-based model can contract any tree T with or without self-loops.

Suppose T consists of one node. If T has no self-loops, it is the constant polynomial, which is a trivial case. Otherwise, the node has at least one self-loop, and C_1 and C_2 can of course contract it the trivial graph.

Now, suppose T consists of $m \geq 2$ nodes. Then there is a leaf (a node that has degree 1 when we ignore self-loops). If this leaf does not have a self-loop, we can contract it to remove this node using C_3. Otherwise, we can use C_2 to remove any multiple self-loops (if needed), and then use C_4 to remove the node once we are left with one self-loop. This gives a tree T' with at least one self-loop of $m-1$ nodes, so T' can be contracted by induction.

($\Longrightarrow$) Suppose 1-WL cannot distinguish the two simple graphs $\mathbf{X}^{(1)}$ and $\mathbf{X}^{(2)}$, so $\mathrm{hom}(H, \mathbf{X}^{(1)}) = \mathrm{hom}(H, \mathbf{X}^{(2)})$ for all H of tree-width 1. We show that the node-based contraction bank cannot contract any $\mathrm{hom}(H, \mathbf{X})$ for H of tree-width greater than 1.

Suppose H has tree-width greater than 1, so it cannot be a forest or a tree. Hence, H must have a cycle. However, the node-based model cannot contract any graph that has a cycle; this is because if it could, then the first node of the cycle that is contracted would have had at least two different neighbors, but such a node cannot be contracted by C_1, C_2, C_3 or C_4. Hence, the node-based graph model cannot distinguish $\mathbf{X}^{(1)}$ and $\mathbf{X}^{(2)}$. $\square$

Proposition H.3. *The Prototypical edge-based graph model can distinguish any two simple graphs if and only if 2-FWL / 3-WL can.*

Proof. ($\Longleftarrow$) Suppose 2-FWL can distinguish the two simple graphs $\mathbf{X}^{(1)}$ and $\mathbf{X}^{(2)}$, so there is a graph H of tree-width 2 such that $\mathrm{hom}(H, \mathbf{X}^{(1)}) \neq \mathrm{hom}(H, \mathbf{X}^{(2)})$. We will show that the edge-based model can contract H. We can assume H is connected because otherwise we can separately contract each connected component. Moreover, we can assume that H has no self-loops or multiple edges. This is because if a node has a self-loop and it has no neighbors, then C_2 and C_1 can remove the node, and if it has neighbors then C_4 can contract the self-loop and add an edge to a neighbor, thus forming a multiple edge. For multiple edges, C_6 can align the direction of the edges between if necessary, and then C_5 can remove the multiple edges.

Since H has tree-width 2, we know it is a partial 2-tree. Thus, there is an ordering of the m vertices of H, say $v_1, \ldots, v_m$, such that when deleting each vertex and all incident edges in turn, we only ever delete vertices of degree at most 2. We show that our edge-based Prototypical model can contract edges in this order by induction. For $i \in [m]$, suppose we have deleted nodes $v_1, \ldots, v_{i-1}$ (in the base case $i = 1$ we have not deleted any nodes), we are deleting v_i, the current graph has no multiple edges, and the any self-loops in the current graph belong to nodes with no neighbors

- If node v_i has degree zero, then it can be contracted to a trivial graph by C_1 and C_2.
- If node v_i has degree one, then we use C_3 to contract it, thus adding a self-loop to its neighbor. If its neighbor then has degree 0, then we do not need to remove self-loops for our induction. Otherwise, if its neighbor has nonzero degree, then we use C_4, C_6 (if necessary), and C_5 to remove the self-loop of the neighbor, and remove any multiple edges.
- If node v_i has degree two, then we use C_6 if necessary, then use C_7 to contract v_i, and use C_6 and C_5 to remove any multiple edges formed.

After any of these operations, we have deleted the node v_i, and maintained the assumptions of our induction. In particular, note that the degree of a node (ignoring self-loops) never increases, so we indeed only ever contract nodes of degree at most 2. Thus, the Prototypical edge model is capable of contracting H to a trivial graph.

($\Longrightarrow$) Consider a pair of graphs that are not distinguishable by 3-WL. Let $k > 3$ be the smallest integer such that k-WL distinguishes this pair. This pair of graphs differs in the number of homomorphisms for a subgraph that has at least tree-width $k - 1 \geq 3$ (Grohe et al., 2021) (e.g. see figure 2 of (Bouritsas et al., 2022) for explicit example). Series of eliminations of nodes of a subgraph form a tree decomposition of the subgraph via its chordal completion as described below. Subgraphs of tree-width ≥ 3 have at least one bag of 4 or more nodes in their tree decomposition. Since the Prototypical edge-based graph model can only contract with bags of size 3 or fewer, such a homomorphism count cannot be performed using the contractions.

To show that any set of contractions forms a valid tree decomposition of the graph, consider the chordal completion of the graph formed by the eliminations. This chordal completion consists of the original graph with all edges added between nodes that were part of a contraction which eliminated any other node. E.g., if nodes in $\{a, b, c\}$ were part of a contraction eliminating node a, then all edges between the nodes in $\{a, b, c\}$ are added to its chordal completion. Note that this chordal completion follows naturally in any contraction as once a node is eliminated in a contraction, an edge must be made between its remaining neighbors to store the output of the contraction.

Any chordal completion of a subgraph has the property that via the same elimination order of its construction, no more edges are added. I.e., the subgraph constructed by each eliminated node and its neighbors forms a clique. This elimination ordering forms a tree decomposition with bags consisting of the cliques in the elimination ordering. Thus, the tree-width of a graph is upper bounded by the size of the maximal clique in its chordal completion minus one (Diestel, 2005). Since the Prototypical edge based model can only contract up to 3 nodes at once, cliques of size at most 3 can be constructed in the chordal completion and the tree-width of such a tree decomposition is at most 2. $\square$

Another way to approach this result is through the relationship between graphs of tree-width 2 and series-parallel graphs. Any biconnected graph H of tree-width 2 is a series-parallel graph (more generally, a graph has tree-width 2 if and only if all biconnected components are series-parallel) (Bodlaender, 1998). Suppose H is series-parallel. Then it is known that it can be contracted to a single edge by two operations (Duffin, 1965):

(op1) Delete a node of exactly degree 2, and connecting its two neighbors.

(op2) If there are two edges between the same two nodes, delete one of the edges.

The edge-based Prototypical model can implement these two operations, so it can contract H into a single edge. The first operation (op1) is C_7 (matrix multiplication), possibly with a C_6 (matrix transpose) beforehand to align the directions of the edges. The second operation (op2) is C_5 (replace two parallel edges with one edge), again possibly with a C_6 (matrix transpose) beforehand to align the directions of the edges. After contraction to a single edge, the remaining operations C_1, C_2, C_3 can be used to contract H to the trivial graph.

If H is not biconnected, then we can use the block-cut tree of H to get the biconnected components. Then we can contract each biconnected component that is a leaf of the block-cut-tree (as it is a series parallel graph) in a way such that we add a self-loop to the cut vertex it is connected to. After pruning cut vertices appropriately, we can continue this process until reaching the trivial graph.

I. Counting of Invariant Polynomials of Symmetric Group

Various methods exist to count the number of invariant polynomials of the Symmetric group (in our case, also isomorphic to multigraphs) of a given form (Harary & Palmer, 2014; Pólya, 1937; Molien, 1897; Thiéry, 2000; Bedratyuk, 2015). Here, we follow a standard strategy to count the number of invariant polynomials by summing over partitions corresponding to cycle indices of the permutation group. Given S_n as the symmetric group on n elements, let $S_n^{(k)}$ be the symmetric group acting on the representation $X \in (\mathbb{R}^n)^{\otimes k}$. Let P_n denote the set of integer partitions of n where each partition $\boldsymbol{m} \in P_n$ is a length n vector whose j-th element is the number of elements of size j in the given partition. For example, for the partition of 4 into $(1, 1, 2)$, the corresponding value of $\boldsymbol{m} = [2, 1, 0, 0]$. Let $\boldsymbol{s}_i$ for $i \in \mathbb{N}$ be arbitrary variables for now (their meaning will become clear later). We define the cycle index $Z(S_n^{(k)})[\boldsymbol{s}_i]$ of $S_n^{(k)}$ as a power series in variables $\boldsymbol{s}_i$ for $i \in \mathbb{N}$ as

$$Z(S_n^{(k)})[\boldsymbol{s}_i] = \sum_{\boldsymbol{m} \in P_n} \frac{1}{\prod_{t=1}^n t^{\boldsymbol{m}_t} \boldsymbol{m}_t!} \prod_{i=1}^{k} \prod_{j_i=1}^{n} \boldsymbol{s}_{\mathrm{lcm}(j_1,\ldots,j_k)}^{\prod_{t=1}^k \boldsymbol{m}_{j_t} j_t / \mathrm{lcm}(j_1,\ldots,j_k)}, \tag{37}$$

where $\operatorname{lcm}(\cdot)$ is the least common multiple of the arguments.

As an example, we have

$$Z(S_3^{(2)})[\boldsymbol{s}_i] = \frac{1}{6}\boldsymbol{s}_1^9 + \frac{1}{2}\boldsymbol{s}_1\boldsymbol{s}_2^4 + \frac{1}{3}\boldsymbol{s}_3^3. \tag{38}$$

For the equivariant case, we need a weighted cycle index which we denote as $Z_W(S_n^{(k)} \times S_n^{(d)})[\boldsymbol{s}_i]$ which can be calculated as

$$Z_W(S_n^{(k)} \times S_n^{(d)})[\boldsymbol{s}_i] = \sum_{\boldsymbol{m}\in P_n} \frac{\boldsymbol{m}_1^d}{\prod_{t=1}^n t^{\boldsymbol{m}_t}\boldsymbol{m}_t!} \prod_{i=1}^k \prod_{j_i=1}^n \boldsymbol{s}_{\operatorname{lcm}(j_1,\ldots,j_k)}^{\prod_{t=1}^k \boldsymbol{m}_{j_t} j_t / \operatorname{lcm}(j_1,\ldots,j_k)}, \tag{39}$$

From here, we can generate the Molien series which counts the number of invariant/equivariant polynomials on $S_n^{(k)}$.

Theorem I.1. *The Molien series $M_{S_n^{(k)}}(x)$ for invariant polynomials on $S_n^{(k)}$ is generated by*

$$\textit{invariant: } M_{S_n^{(k)}}(x) = Z(S_n^{(k)})[\boldsymbol{s}_i = 1 + x^i + x^{2i} + \cdots], \tag{40}$$

and more generally, the Molien series for the polynomials which are equivariant to $S_n^{(d)}$ outputs and $S_n^{(k)}$ inputs is generated by:

$$\textit{equivariant: } M_{S_n^{(k)}\times S_n^{(d)}}(x) = Z_W(S_n^{(k)} \times S_n^{(d)})[\boldsymbol{s}_i = 1 + x^i + x^{2i} + \cdots]. \tag{41}$$

Proof. We enumerate the Molien series of order k invariant polynomials R_k^G in the invariant ring of polynomials R^G using Molien's formula (Derksen & Kemper, 2015; Molien, 1897). A similar proof can be obtained via the Pólya enumeration theorem (Tucker, 1994; Pólya, 1937).

Given a representation $\rho : G \to \mathrm{GL}\,(V)$, Molien's formula states that

$$M_G(x) = \sum_k \dim(R_k^G)x^k = |G|^{-1} \sum_{g\in G} \frac{1}{\det\left(\boldsymbol{I} - x\rho(g)\right)}. \tag{42}$$

First, let us consider the invariant setting for $S_n^{(k)}$ – the symmetric group with representation acting on the vector space $X \in (\mathbb{R}^n)^{\otimes k}$. Eigenvalues of a permutation matrix depend only on the cycle index of the permutation. Therefore, we decompose the sum of group elements in the symmetric group by their cycle indices

$$M_{S_n^{(k)}}(x) = \sum_{\boldsymbol{m}\in P_n} \frac{1}{\prod_{t=1}^n t^{\boldsymbol{m}_t}\boldsymbol{m}_t!} \frac{1}{\det\left(\boldsymbol{I} - x\rho(g_{\boldsymbol{m}}^{(k)})\right)}, \tag{43}$$

where $\rho(g_{\boldsymbol{m}^{(k)}})$ is any permutation with cycle index $\boldsymbol{m}_t$ for the representation on $S_n^{(k)}$. For $S_n^{(1)}$, the representation of the permutation group is the standard representation corresponding to permutations of indices of the vector space. For cycle index $\boldsymbol{m}$, the representation $\rho(g_{\boldsymbol{m}^{(1)}})$ has $\boldsymbol{m}_j$ eigenvalues equal to the j different powers of the j-th root of unity. Denoting $\boldsymbol{s}_i = 1 + x^i + x^{2i} + \cdots$, this then results in the following:

$$\frac{1}{\det\left(\boldsymbol{I} - x\rho(g_{\boldsymbol{m}}^{(1)})\right)} = \prod_{j_i=1}^n \boldsymbol{s}_{j_i}^{\boldsymbol{m}_{j_i}}. \tag{44}$$

$\rho(g_{\boldsymbol{m}^{(k)}})$ acts as a k-fold tensor product of the representation $\rho(g_{\boldsymbol{m}^{(1)}})$, i.e. $\rho(g_{\boldsymbol{m}^{(k)}}) = [\rho(g_{\boldsymbol{m}^{(1)}})]^{(\otimes k)}$. Therefore, to generalize the above formula to higher order k, we enumerate the possible eigenvalues of the tensor product of $\rho(g_{\boldsymbol{m}^{(1)}}$. Since the eigenvalues of a tensor product of operators are simply the product of the eigenvalues of the elements of the tensor product, we can perform this enumeration over products of the operators.

As we noted before, for cycle index $\boldsymbol{m}$, the representation $\rho(g_{\boldsymbol{m}^{(1)}})$ has $\boldsymbol{m}_j$ eigenvalues equal to the j different powers of the j-th root of unity. Given the product of two elements of this cycle index, we now consider the product of eigenvalues equal to the different powers of the j-th and k-th roots of unity. This results in all the eigenvalues which are products of the $\operatorname{lcm}(j,k)$-roots of unity where $\operatorname{lcm}(\cdot)$ denotes the least common multiple. Given $\boldsymbol{m}_j$ and $\boldsymbol{m}_k$ cycles of j-th and k-th order respectively, powers of the $\operatorname{lcm}(j,k)$-roots of unity will appear a total of $\boldsymbol{m}_j \boldsymbol{m}_k / \operatorname{lcm}(j,k)$ number of times. Generalizing this to products of more than two eigenvalues, we arrive at

$$\frac{1}{\det\left(\boldsymbol{I} - x\rho(g_{\boldsymbol{m}}^{(k)})\right)} = \prod_{i=1}^{k} \prod_{j_i=1}^{n} \boldsymbol{s}_{\operatorname{lcm}(j_1,\ldots,j_k)}^{\prod_{t=1}^{k} \boldsymbol{m}_{j_t} j_t / \operatorname{lcm}(j_1,\ldots,j_k)}. \tag{45}$$

Plugging the above into Equation (43), we obtain the desired result.

For the equivariant setting, we use the equivariant form of Molien's formula. For maps from a vector space V to another vector space W, we consider a representation $\rho : G \to \mathrm{GL}\,(V)$ acting on the input space and a representation $\sigma : G \to \mathrm{GL}\,(W)$ acting on the output space. Here, Molien's formula takes the form (Derksen & Kemper, 2015; Antoneli et al., 2008)

$$M_G(x) = \sum_k \dim(R_k^G) x^k = |G|^{-1} \sum_{g \in G} \frac{\operatorname{Tr}(\sigma(g)^{-1})}{\det\left(\boldsymbol{I} - x\rho(g)\right)}. \tag{46}$$

The above can be shown by noting that the module of equivariant polynomials corresponds to $(R[V] \otimes W)^G$ where $R[V]$ is the ring of polynomials on the vector space V. The representation of $S_n^{(d)}$ corresponds to the d-fold tensor product of the standard representation of the symmetric group. For a cycle index $\boldsymbol{m}$ this representation has one nonzero entries on the diagonal for each cycle of size 1. Therefore, there are $\boldsymbol{m}_1^d$ total nonzero entries each equal to one. Plugging this in, we arrive at the final solution. □

As an example, returning to Equation (38), we have for invariant polynomials on $S_3^{(2)}$:

$$\begin{aligned} M_{S_3^{(2)}}(x) &= \frac{1}{6}\big[(1 + x + x^2 + \cdots)^9 + 3(1 + x + x^2 + \cdots)(1 + x^2 + x^4 + \cdots)^4 \\ &\quad + 2(1 + x^3 + x^6 + \cdots)^3\big] \\ &= 1 + 2x + 10x^2 + \cdots \end{aligned} \tag{47}$$

Theorem I.1 quantifies the Molien series for polynomials on n nodes. To obtain the asymptotic limit $M_{S_\infty^{(2)}}(x)$ for invariant polynomials on graphs of arbitrary size, we note that $M_{S_n^{(2)}}(x)$ and $M_{S_\infty^{(2)}}(x)$ agree in powers x^c up to $c = \lfloor n/2 \rfloor$. Therefore, to generate the asymptotic series up to power n, it suffices to calculate the corresponding Molien series for $M_{S_{2n}^{(2)}}(x)$. A similar logic can be applied for equivariant polynomials which also have a "red" edge as well as described in the main text.

Corollary I.2. *The number of invariant polynomials quantified in the Molien series $M_{S_\infty^{(2)}}(x)$ for the asymptotic limit of $n \to \infty$ nodes agrees with $M_{S_n^{(2)}}(x)$ for n up to the first $\lfloor n/2 \rfloor$ degrees. Similarly, the number of equivariant polynomials quantified in the Molien series $M_{S_\infty^{(2)}}(x)$ for the asymptotic limit of $n \to \infty$ nodes agrees with $M_{S_n^{(2)}}(x)$ for n up to the first $\lfloor n/2 \rfloor - 1$ degrees.*

Counts of invariant polynomials. The Molien series of the number of invariant polynomials on graphs, i.e., $\mathbb{R}^{n^2} \to \mathbb{R}$ for sufficiently large n, begins with

$$\begin{aligned} &1, 2, 11, 52, 296, 1724, 11060, 74527, 533046, 3999187, \\ &31412182, 257150093, 2188063401, 19299062896, \\ &176059781439, 1657961491087, \ldots \end{aligned} \tag{48}$$

Counts of equivariant polynomials. The Molien series of the number of equivariant polynomials on graphs, i.e., $\mathbb{R}^{n^2} \to \mathbb{R}^{n^2}$ for sufficiently large n, begins with

$$\begin{aligned} &2, 15, 117, 877, 6719, 52505, 422824, 3508753, 30036833, \\ &265100322, 2410638644, 22563597944, 217175819474, \\ &2147355853088, 21790101729085, 226707665717377, \ldots \end{aligned} \tag{49}$$

For the setting of the standard representation of the symmetric group on nodes ($S_n^{(1)}$ in our notation), the above recovers the generating series for partitions for which there exist efficiently calculable recurrences via the pentagonal number theorem (Hardy et al., 1979). We do not know of a similarly more direct way to compute the Molien series above in the general case.