Breaking the Limits of Message Passing Graph Neural Networks

Muhammet Balcilar [1 2] **Pierre Héroux** [1] **Benoit Gaüzère** [3] **Pascal Vasseur** [1 4] **Sébastien Adam** [1] **Paul Honeine** [1]

Abstract

Since the Message Passing (Graph) Neural Networks (MPNNs) have a linear complexity with respect to the number of nodes when applied to sparse graphs, they have been widely implemented and still raise a lot of interest even though their theoretical expressive power is limited to the first order Weisfeiler-Lehman test (1-WL). In this paper, we show that if the graph convolution supports are designed in spectral-domain by a non-linear custom function of eigenvalues and masked with an arbitrary large receptive field, the MPNN is theoretically more powerful than the 1-WL test and experimentally as powerful as a 3-WL existing models, while remaining spatially localized. Moreover, by designing custom filter functions, outputs can have various frequency components that allow the convolution process to learn different relationships between a given input graph signal and its associated properties. So far, the best 3-WL equivalent graph neural networks have a computational complexity in $\mathcal{O}(n^3)$ with memory usage in $\mathcal{O}(n^2)$, consider non-local update mechanism and do not provide the spectral richness of output profile. The proposed method overcomes all these aforementioned problems and reaches state-of-the-art results in many downstream tasks.

1. Introduction

In the past few years, finding the best inductive bias for relational data represented as graphs has gained a lot of interest in the machine learning community. Node-based message passing mechanisms relying on the graph structure have given rise to the first generation of Graph Neural Networks (GNNs) called Message Passing Neural Networks (MPNNs) (Gilmer et al., 2017). These algorithms spread each node features to the neighborhood nodes using trainable weights. These weights can be shared with respect to the distance between nodes (Chebnet GNN) (Defferrard et al., 2016), to the connected nodes features (GAT for graph attention network) (Veličković et al., 2018) and/or to edge features (Bresson & Laurent, 2018). When considering sparse graphs, the memory and computational complexity of such approaches are linear with respect to the number of nodes. As a consequence, these algorithms are feasible for large sparse graphs and thus have been applied with success on many downstream tasks (Dwivedi et al., 2020).

Despite these successes and these interesting computational properties, it has been shown that MPNNs are not powerful enough (Xu et al., 2019). Considering two non-isomorphic graphs that are not distinguishable by the first order Weisfeiler-Lehman test (known as the 1-WL test), existing maximum powerful MPNNs embed them to the same point. Thus, from a theoretical expressive power point of view, these algorithms are not more powerful than the 1-WL test. Beyond the graph isomorphism issue, it has also been shown that many other combinatorial problems on graph cannot be solved by MPNNs (Sato et al., 2019).

In (Maron et al., 2019b; Keriven & Peyré, 2019), it has been proven that in order to reach universal approximation, higher order relations are required. In this context, some powerful models that are equivalent to the 3-WL test were proposed. For instance, (Maron et al., 2019a) proposed the model PPGN (Provably Powerful Graph Network) that mimics the second order Folklore WL test (2-FWL), which is equivalent to the 3-WL test. In (Morris et al., 2019), they proposed to use message passing between 1, 2 and 3 order node tuples hierarchically, thus reaching the 3-WL expressive power. However, using such relations makes both memory usage and computational complexities grown exponentially. Thus, it is not feasible to have universal approximation models in practice.

In order to increase the theoretical expressive power of MPNNs by keeping the linear complexity mentioned above, some researchers proposed to partly randomize node features (Abboud et al., 2020; Sato et al., 2020) or to add a unique label (Murphy et al., 2019) in order to have the ability to distinguish two non-isomorphic graphs that are not distinguished by the 1-WL test. These solutions need massively training samples and involve slow convergence.

[1]LITIS Lab, University of Rouen Normandy, France [2]InterDigital, France [3]LITIS Lab, INSA Rouen Normandy, France [4]MIS Lab, Université de Picardie Jules Verne, France. Correspondence to: Muhammet Balcilar <muhammetbalcilar@gmail.com>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

(Bouritsas et al., 2020; Dasoulas et al., 2020) proposed to use a preprocessing step to extract some features that cannot be extracted by MPNNs. Thus, the expressive power of their GNN is improved. However, these handcrafted features need domain expertise and a feature selection process among an infinite number of possibilities.

All these studies target more theoretically powerful models, closer to universal approximation. However, this does not always induce a better generalization ability. Since most of the realistic problems are given with many node/edge features (which can be either continuous or discrete), there is almost no pair of graphs that are not distinguishable by the 1-WL test in practice. In addition, theoretically more powerful methods use non-local updates, breaking one of the most important inductive bias in Euclidean learning named locality principle (Battaglia et al., 2018). These may explain why theoretical powerful methods cannot outperform MPNNs on many downstream tasks, as reported in (Dwivedi et al., 2020). On the other hand, it is obvious that 1-WL equivalent GNNs are not expressive enough since they are not able to count some simple structural features such as cycles or triangles (Arvind et al., 2020; Chen et al., 2020; Bouritsas et al., 2020; Vignac et al., 2020), which are informative for some social or chemical graphs. Finally, another important aspect mentioned by a recent paper (Balcilar et al., 2021) concerns the spectral ability of GNN models. It is shown that a vast majority of the MPNNs actually work as low-pass filters, thus reducing their expressive power.

In this paper, we propose to design graph convolution in the spectral domain with custom non-linear functions of eigenvalues and by masking the convolution support with desired length of receptive field. In this way, we have (i) a spatially local updates process, (ii) linear memory and computational complexities (except the eigendecomposition in preprocessing step), (iii) enough spectral ability and (iv) a model that is theoretically more powerful than the 1-WL test, and experimentally as powerful as PPGN. Experiments show that the proposed model can distinguish pairs of graphs that cannot be distinguished by 1-WL equivalent MPNNs. It is also able to count some substructures that 1-WL equivalent MPNNs cannot. Its spectral ability enables to produce various kind of spectral components in the output, while the vast majority of the GNNs including higher order WL equivalent models do not. Finally, thanks to the sparse matrix multiplication, it has linear time complexity except the eigendecomposition in preprocessing step.

The paper is structured as follows. In Section 2, we set the notations and the general framework used in the following. Section 3 is dedicated to the characterization of WL test, which is the backbone of our theoretical analysis. It is followed by our findings in Section 4 on analysing the expressive power of MPNNs and our solutions to improve expressive power of MPNNs in Section 5. The experimental results and conclusion are the last two section of this paper.

2. Generalization of Spectral and Spatial MPNN

Let G be a graph with n nodes and an arbitrary number of edges. Connectivity is given by the adjacency matrix $A \in \{0,1\}^{n\times n}$ and features are defined on nodes by $X \in \mathbb{R}^{n\times f_0}$, with f_0 the length of feature vectors. For any matrix X, we used X_i, $X_{:j}$ and $X_{i,j}$ to refer to its i-th column vector, j-th row vector and (i,j)-th entry, respectively. A graph Laplacian is given by $L = D - A$ (or $L = I - D^{-1/2}AD^{-1/2}$) where $D \in \mathbb{R}^{n\times n}$ is the diagonal degree matrix and I is the identity. Through an eigendecomposition, L can be written by $L = Udiag(\boldsymbol{\lambda})U^T$ where each column of $U \in \mathbb{R}^{n\times n}$ is an eigenvector of L, $\boldsymbol{\lambda} \in \mathbb{R}^n$ gathers the eigenvalues of L and $diag(\cdot)$ creates a diagonal matrix whose diagonal elements are from a given vector. We use superscripts to refer to vectors or matrices evolving through iterations or layers. For instance, $H^{(l)} \in \mathbb{R}^{n\times f_l}$ refers to the node representation on layer l whose feature dimension is f_l.

GNN models rely on a set of layers where each layer takes the node representation of the previous layer $H^{(l-1)}$ as input and produces a new representation $H^{(l)}$, with $H^{(0)} = X$. According to the domain which is considered to design the layer computations, GNNs are generally classified as either spectral or spatial (Wu et al., 2019; Chami et al., 2020). Spectral GNNs rely on the spectral graph theory (Chung, 1997). In this framework, signals on graphs are filtered using the eigendecomposition of the graph Laplacian (Shuman et al., 2013). By transposing the convolution theorem to graphs, the spectral filtering in the frequency domain can be defined by $x_{flt} = Udiag(\Omega(\boldsymbol{\lambda}))U^\top x$, where $\Omega(.)$ is the desired filter function which needs to be learnt by backpropagation. On the other hand, spatial GNNs, such as GCN (graph convolutional network) (Kipf & Welling, 2017) and GraphSage (Hamilton et al., 2017), consider two operators, one that aggregates the connected nodes messages and one that updates the concerned node representation.

In a recent paper (Balcilar et al., 2021), it was explicitly shown that both spatial and spectral GNNs are MPNN, taking the general form

$$H^{(l+1)} = \sigma\Big(\sum_s C^{(s)} H^{(l)} W^{(l,s)}\Big), \tag{1}$$

where $C^{(s)} \in \mathbb{R}^{n\times n}$ is the s-th convolution support that defines how the node features are propagated to the neighboring nodes and $W^{(l,s)} \in \mathbb{R}^{f_l \times f_{l+1}}$ is the trainable matrix for the l-th layer and s-th support. Within this generalization, GNNs differ from each other by the design of the convolution supports $C^{(s)}$. If the supports are designed in the

spectral domain by $\Phi_s(\boldsymbol{\lambda})$, the convolution support needs to be written as $C^{(s)} = U diag(\Phi_s(\boldsymbol{\lambda}))U^\top$.

One can see that as long as $C^{(s)}$ matrices are sparse (number of edges is defined by some constant multiplied by the number of nodes), MPNN in Eq.1 has linear memory and computational complexities with respect to the number of nodes. Because, the valid entries in $C^{(s)}$ that we need to keep is linear with respect to the number of nodes and thank to the sparse matrix multiplication $C^{(s)}H^{(l)}$ takes linear time with respect to the number of edges thus nodes as well.

3. Characterization of Weisfeiler-Lehman

The universality of a GNN is based on its ability to embed two non-isomorphic graphs to distinct points in the target feature space. A model that can distinguish all pairs of non-isomorphic graphs is a universal approximator. Since the graph isomorphism problem is NP-intermediate (Takapoui & Boyd, 2016), the Weisfeiler-Lehman Test (abbreviated WL-test), which gives sufficient but not enough evidence of graph isomorphism, is frequently used for characterizing GNN expressive power. The classical vertex coloring WL test can be extended by taking into account higher order of node tuple within the iterative process. These extensions are denoted as k-WL test, where k is equals to the order of the tuple. These tests are described in Appendix A.

It is shown in (Arvind et al., 2020) that for $k \geq 2$, $(k+1)$-WL $>$ (k)-WL, i.e., higher order of tuple leads to a better ability to distinguish two non-isomorphic graphs. For $k = 1$, this statement is not true, and 2-WL is not more powerful than 1-WL (Maron et al., 2019a). To clarify this point, the Folkore WL (FWL) test has been defined such that 1-WL=1-FWL, but for $k \geq 2$, we have $(k+1)$-WL $\approx$ (k)-FWL (Maron et al., 2019a).

In literature, some confusions occur among the two versions. Some papers use WL test order (Morris et al., 2019; Maron et al., 2019a), while others use FWL order under the name of WL such as in (Abboud et al., 2020; Arvind et al., 2020; Takapoui & Boyd, 2016). In this paper, we explicitly mention both WL and FWL equivalent.

In order to better understand the capability of WL tests, some papers attempt to characterize these tests using a first order logic (Immerman & Lander, 1990; Barceló et al., 2019). Consider two unlabeled and undirected graphs represented by their adjacency matrices A_G and A_H. These two graphs are said k-WL (or k-FWL) equivalent, and denoted $A_G \equiv_{k-WL} A_H$, if they are indistinguishable by a k-WL (or k-FWL) test.

Recently (Brijder et al., 2019; Geerts, 2020) proposed a new Matrix Language called MATLANG. This language includes different operations on matrices and makes some explicit connections between specific dictionaries of operations and the 1-WL and 3-WL tests. Expressive power varies with the operations included in each dictionnary.

Definition 1. *$ML(\mathcal{L})$ is a matrix language with an allowed operation set $\mathcal{L} = \{op_1, \dots op_n\}$, where $op_i \in \{., +, ^\top, diag, tr, \mathbf{1}, \odot, \times, f\}$. The possible operations are matrices multiplication and addition, matrix transpose, vector diagonalization, matrix trace computation, column vector full of 1, element-wise matrix multiplication, matrix/scalar multiplication and element-wise custom function operating on scalars or vectors.*

Definition 2. *$e(X) \in \mathbb{R}$ is a sentence in $ML(\mathcal{L})$ if it consists of any possible consecutive operations in $\mathcal{L}$, operating on a given matrix X and resulting in a scalar value.*

As an example, $e(X) = \mathbf{1}^\top X^2 \mathbf{1}$ is a sentence of $ML(\mathcal{L})$ with $\mathcal{L} = \{.^\top, \mathbf{1}\}$, computing the sum of all elements of square matrix X. In the following, we are interested in languages $\mathcal{L}_1, \mathcal{L}_2$ and $\mathcal{L}_3$ that have been used for characterizing the WL-test in (Geerts, 2020). These results are given next.

Remark 1. *Two adjacency matrices are indistinguishable by the 1-WL test if and only if $e(A_G) = e(A_H)$ for all $e \in \mathcal{L}_1$ with $\mathcal{L}_1 = \{.^\top, \mathbf{1}, diag\}$. Hence, all possible sentences in $\mathcal{L}_1$ are the same for 1-WL equivalent adjacency matrices. Thus, $A_G \equiv_{1-WL} A_H \leftrightarrow A_G \equiv_{ML(\mathcal{L}_1)} A_H$. (see Theorem 7.1 in (Geerts, 2020))*

Remark 2. *$ML(\mathcal{L}_2)$ with $\mathcal{L}_2 = \{.^\top, \mathbf{1}, diag, tr\}$ is strictly more powerful than $\mathcal{L}_1$, i.e., than the 1-WL test, but less powerful than the 3-WL test. (see Theorem 7.2 and Example 7.3 in (Geerts, 2020))*

Remark 3. *Two adjacency matrices are indistinguishable by the 3-WL test if and only if they are indistinguishable by any sentence in $ML(\mathcal{L}_3)$ with $\mathcal{L}_3 = \{.^\top, \mathbf{1}, diag, tr, \odot\}$. Thus, $A_G \equiv_{3-WL} A_H \leftrightarrow A_G \equiv_{ML(\mathcal{L}_3)} A_H$. (see Theorem 9.2 in (Geerts, 2020))*

Remark 4. *Enriching the operation set to $\mathcal{L}^+ = \mathcal{L} \cup \{+, \times, f\}$ where $\mathcal{L} \in (\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_3)$ does not improve the expressive power of the language. Thus, $A_G \equiv_{ML(\mathcal{L})} A_H \leftrightarrow A_G \equiv_{ML(\mathcal{L}^+)} A_H$. (see Proposition 7.5 in (Geerts, 2020))*

4. How Powerful are MPNNs?

This section presents some results about the theoretical expressive power of state-of-the-art MPNNs. Those results are derived using the MATLANG language (Geerts, 2020) and more precisely the remarks of the preceding section. Proofs of the theorems are given in Appendix B.

Theorem 1. *MPNNs such as GCN, GAT, GraphSage, GIN (defined in Appendix H) cannot go further than operations in $\mathcal{L}_1^+$. Thus, they are not more powerful than the 1-WL test.*

This result has already been given in (Xu et al., 2019), which proposed GIN-ϵ (GIN for Graph Isomorphism Network) and

showed that it is the unique MPNN which is provably exact the same powerful with the 1-WL test, while the rest of MPNNs are known to be less powerful than 1-WL test.

Chebnet is also known to be not more powerful than the 1-WL test. However, the next theorem states that it is true if the maximum eigenvalues are the same for both graphs. For a pair of graphs whose maximum eigenvalues are not equal, Chebnet is strictly more powerful than the 1-WL test.

Theorem 2. *Chebnet is more powerful than the 1-WL test if the Laplacian maximum eigenvalues of the non-regular graphs to be compared are not the same. Otherwise Chebnet is not more powerful than 1-WL.*

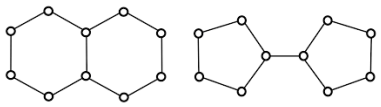

Figure 1. Decalin (G) and Bicyclopentyl (H) graphs are $\mathcal{L}_1$ and also 1-WL equivalent, but Chebnet can distinguish them.

Figure 1 shows two graphs that are 1-WL equivalent and are generally used to show how MPNNs fail. However, their normalized Laplacian's maximum eigenvalues are not the same. Thus, Chebnet can project these two graphs to different points in feature space. Details can be found in Appendix C.

As stated in the introduction, comparison with the WL-test is not the only way to characterize the expressive power of GNNs. Powerful GNNs are also expected to be able to count relevant substructures in a given graph for specific problems. The following theorems describe the matrix language required to be able to count the graphlets illustrated in Figure 2, which are called 3-star, triangle, tailed triangle and 4-cycle.

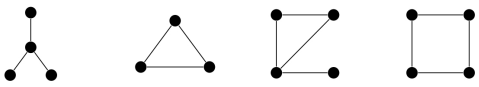

Figure 2. Sample of patterns: 3-star, triangle, tailed triangle and 4-cycle graphlets used in our analysis.

Theorem 3. *3-star graphlets can be counted by sentences in $\mathcal{L}_1^+$.*

Theorem 4. *Triangle and 4-cycle graphlets can be counted by sentences in $\mathcal{L}_2^+$.*

Theorem 5. *Tailed triangle graphlets can be counted by sentences in $\mathcal{L}_3^+$.*

These theorems show that 1-WL equivalent MPNNs can only count 3-star patterns, while 3-WL equivalent MPNNs can count all graphlets shown in Figure 2.

(Dehmamy et al., 2019) has shown that a MPNN is not able to learn node degrees if the MPNN has not an appropriate convolution support (e.g. A). Therefore, to achieve a fair comparison, we assume that node degrees are included as a node feature. Note however, that the number of 3-star graphlets centered on a node can be directly derived from its degrees (see Appendix B.3). Therefore, any graph agnostic MLP can count the number of 3-star graphlets given the node degree.

5. MPNN Beyond 1-WL

In this section, we present two new MPNN models. The first one, called GNNML1 is shown to be as powerful as the 1-WL test. The second one, called GNNML3 exploits the theoretical results of (Geerts, 2020) to break the limits of 1-WL and reach 3-WL equivalence experimentally. GNNML1 relies on the node update schema given by :

$$H^{(l+1)}=\sigma\big(H^{(l)}W^{(l,1)}+AH^{(l)}W^{(l,2)}+H^{(l)}W^{(l,3)}\odot H^{(l)}W^{(l,4)}\big) \tag{2}$$

where $W^{(l,s)}$ are trainable parameters. Using this model, the new representation of a node consists of a sum of three terms : (i) a linear transformation of the previous layer representation of the node, (ii) a linear transformation of the sum of the previous layer representations of its connected nodes and (iii) the element-wise multiplication of two different linear transformations of the previous layer representation of the node.

The expressive power of GNNML1 is defined by the following theorem. Its proof is given in Appendix B:

Theorem 6. *GNNML1 can produce every possible sentences in $ML(\mathcal{L}_1)$ for undirected graph adjacency A with monochromatic edges and nodes. Thus, GNNML1 is exactly as powerful as the 1-WL test.*

Hence, this model has the same ability as the 1-WL test to distinguish two non-isomorphic graphs, i.e., the same as GIN. This is explained by the third term in the sum of Eq.(2) since it can produce feature-wise multiplication on each layer. Since node representation is richer, we also assume that it would be more powerful for counting substructures. This assumption is validated by experiments in Section 6.

To reach more powerful models than 1-WL, theoretical results (see Remarks 1, 2 and 3 in Section 3) show that a model that can produce different outputs than $\mathcal{L}_1^+$ language is needed. More precisely, according to Remarks 2 and 3, trace (tr) and element-wise multiplication ($\odot$) operations are required to go further than 1-WL.

In order to illustrate the impact of the trace operation, one can use 1-WL equivalent Decalin and Bicyclopentyl graphs in Figure 1. It is easy to show that $tr(A_G^5) = 0$ but $tr(A_H^5) = 20$, $tr(A^5)$ giving the number of 5-length closed

walks. Thus, if a model can apply a trace operator over some power of adjacency, it can easily distinguish these two graphs. Computational details concerning this example are given in Appendix C.

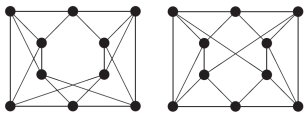

Figure 3. Cospectral and 4-regular graphs from (Van Dam & Haemers, 2003) are $\mathcal{L}_1$ and $\mathcal{L}_2$ equivalent.

Despite this interesting property of the trace operator, it is not sufficient to distinguish cospectral graphs, since cospectral graphs (see Figure 3) have the same number of closed walks of any length (see Proposition 5.1 in (Geerts, 2020)). In such cases, element-wise multiplication is useful. As an example, the sentence $e(A) = \mathbf{1}^\top f((A \odot A^2)^2\mathbf{1})$ where $f(x) = x \odot x$ for any vector x, gives $e(A_G) = 6032$ and $e(A_H) = 5872$ for the graphs of Figure 3. Thus, element-wise multiplication helps distinguishing these two graphs. The calculation details can be found in Appendix D.

As shown by these examples, a model enriched by element-wise multiplication and trace operator can go further than the 1-WL test. However, these operations need to keep the power of the adjacency matrix explicitly and to multiply these dense matrices to each other by matrix or element-wise multiplication. Such a strategy is actually used by higher order GNNs such as (Maron et al., 2019a; Morris et al., 2019), which are provably more powerful than existing MPNNs.

However, MPNNs cannot calculate the power of a given adjacency explicitly. Indeed, a MPNN layer multiplies the previous representation of the nodes by sparse adjacency matrix or more generally sparse convolution supports C in Eq.(1). More precisely, if the given node features are $H^{(0)} = \mathbf{1}$, a MPNN can calculate $C^3\mathbf{1}$ by 3 layered MPNN computing $C(C(C\mathbf{1}))$ but not by $(C^3)\mathbf{1}$. Since a MPNN does not keep C^3 explicitly, it cannot take its trace or multiply element-wise to another power of support. This is a major disadvantage of MPNNs, but it explains why MPNNs need just linear time and memory complexity, making them useful in practice.

A solution to the problem mentioned above is to design graph convolution supports by the element-wise multiplication of the s-power of the adjacency matrix and a given receptive field, i.e., by $C^{(s)} = M \odot A^s$ where M masks the components of the powered matrix and keeps the convolution support sparse. $M = A + I$ is an example of mask that gives a maximum 1-length receptive field. This model cannot calculate all possible element-wise multiplications between all possible matrices, but it can produce any sentence in a form of $(M \odot A^s)^l$ where $l \in [0, l_{max}]$ is the layer number and $s \in [0, s_{max}]$ is the pre-computed power of convolution supports. In this proposition, the receptive field mask and the number of power of adjacency should be computed in a pre-processing step. However, we cannot initially know which power of adjacency matrix is necessary for a given problem. One solution is to tune it as an hyper-parameter of the model. Another problem of this approach is that using powers of adjacency makes the convolution supports filled with high values that have to be normalized.

To overcome these problems, we propose through our GNNML3 model to design convolution supports in the spectral domain as functions of eigenvalues of the normalized Laplacian matrix or of the adjacency matrix. The following theorem, with proof given in Appendix B, shows that such supports can be written as power series of the graph Laplacian or the adjacency matrix.

Theorem 7. *A convolution support given by*

$$C'^{(s)} = U diag(\Phi_s(\lambda))U^\top, \tag{3}$$

where $\Phi_s(\lambda) = exp(-b(\lambda - f_s)^2)$, $f_s \in [\lambda_{min}, \lambda_{max}]$ *is a scalar design parameter of each convolution support and* $b > 0$ *is a general scalar design parameter, can be expressed as a linear combination of all powers of graph Laplacian (or adjacency) as follows, with* $\alpha_{s,i} = \frac{\Phi_s^{(i)}(0)}{i!}$*:*

$$C'^{(s)} = \alpha_{s,0}L^0 + \alpha_{s,1}L^1 + \alpha_{s,2}L^2 + \dots. \tag{4}$$

Since design parameters f_s of each matrix are different, each $C'^{(s)}$ in Eq.(4) consists of different linear combinations of power series of the graph Laplacian (or adjacency). Thus, necessary powers of the graph Laplacian (or adjacency) and its diagonal part (for trace operation) can be learned and their element-wise multiplication can be produced by:

$$C = M \odot mlp_4\left(mlp_1(C')|mlp_2(C') \odot mlp_3(C')\right), \tag{5}$$

where $C' = [C'^{(1)}|\dots|C'^{(s)}] \in \mathbb{R}^{n\times n\times S}$ stacks initial convolution supports on the third dimension, $mlp_k(.)$ is a trainable model performed on third dimension of a given tensor, $C = [C^{(1)}|\dots|C^{(S)}] \in \mathbb{R}^{n\times n\times S}$ sparsify convolution support by defined receptive field mask M. The forward calculation of one layer MPNN becomes:

$$H^{(l+1)} = \sigma\Big(\sum_s (C^{(s)}H^{(l)}W^{(l,s)})|mlp_5(H^{(l)}) \odot mlp_6(H^{(l)})\Big) \tag{6}$$

where we concatenate MPNN representation under learned convolution with element-wise product of node representations as in GNNML1.

There is an infinite number of selections of $\Phi_s(\lambda)$ that make the convolution support written by power series of graph Laplacian (or adjacency). However, we can design each convolution support to be sensitive on each band of spectrum

Algorithm 1 GNNML3 Preprocessing Step

Input: adjacency $A \in \mathbb{R}^{n\times n}$, receptive field mask $M \in \{0,1\}^{n\times n}$, number of supports $S \in \mathbb{N}$, frequency responses function of each support $\Phi_1(\lambda)\dots\Phi_S(\lambda)$
Output: extracted edge features $C' \in \mathbb{R}^{m\times S}$
Set basis matrix: $B = I - D^{-1/2}AD^{-1/2}$ or $B = A$.
Eigendecomposition: $Udiag(\lambda)U^\top = B$
for $s = 1$ **to** S **do**
  $C'_{:s} = sparse2vec\left(M \odot (Udiag(\Phi_s(\lambda))U^\top)\right)$
end for

Algorithm 2 GNNML3 Forward calculation

Input: extracted edge features $C' \in \mathbb{R}^{m\times S}$, initial node feature $H^{(0)} \in \mathbb{R}^{n\times f_0}$, receptive field mask $M \in \{0,1\}^{n\times n}$, number of layers L, number of supports S
Output: new node representation $H^{(L)}$
for $l = 0$ **to** $L-1$ **do**
  $\tilde{C}=mlp_{l,4}(mlp_{l,1}(C')|mlp_{l,2}(C')\odot mlp_{l,3}(C'))$
  for $s = 1$ **to** S **do**
    $C^{(s)}=vec2sparse(\tilde{C}_{:s},M)$
  end for
  $H^{(l+1)}=\sigma\left(\sum_s (C^{(s)}H^{(l)}W^{(l,s)})|mlp_{l,5}(H^{(l)})\odot mlp_{l,6}(H^{(l)})\right)$
end for

(f_s) by given bandwidth (b). Therefore, our model will be able to learn properties depending on the spectrum of graph signal.

Algorithm 1 calculates the initial convolution supports (C'). Since the supports are computed for valid indices in the receptive field mask (where $M_{i,j} = 1$), one can see C' as extracted edge features where the edge indices are defined by M. In application, a function $sparse2vec : \mathbb{R}^{n\times n} \to \mathbb{R}^m$ converts the sparse matrix to a vector by just keeping the components on valid indices of the mask. Algorithm 2 shows the forward calculation of the model for just one graph. To make the representation as simple as possible, we prefer to use tensor representation in Eq.(5). However, implementation of Algorithm 2 just apply $mlp_k(.)$ to the valid indices defined by receptive field mask M. Thus, C, C' have the dimension of $\mathbb{R}^{m\times S}$, where m shows number of valid indices in M and $mlp_k(.)$ applies on columns of C'. Beside, we use a function $vec2sparse : \mathbb{R}^m \to \mathbb{R}^{n\times n}$ that converts the vector to the sparse convolution support according to a given mask M.

The limit of the proposed method is similar to the limit of 3-WL (or 2-FWL) test. For instance, it fails to distinguish strongly regular graphs, that can be defined by 3 parameters: the degree of the nodes, the number of common neighbours of adjacent node pairs, and the number of common neighbours of non-adjacent node pairs. Such graphs are provably known to be 3-WL equivalent (Arvind et al., 2020). In Appendix E, a strongly regular graphs pair and the result of a sample sentence in $\mathcal{L}_3$ are presented.

6. Experimental Results

This section presents the experimental results obtained by the proposed models GNNML1 and GNNML3. All codes and datasets are available online [1]. We use GCN, GAT, GIN and Chebnet as 1-WL MPNN baselines and PPGN as 3-WL baseline (see Appendix H). Experiments aim to answer four questions:

Q1: How many pairs of non-isomorphic simple graphs that are either 1-WL or 3-WL equivalent are not distinguished by the models?
Q2: Can the models generalize the counting of some substructures in a given graph?
Q3: Can the models learn low-pass, high-pass and band-pass filtering effects and generalize the classification problem according to the frequency of the signal?
Q4: Can the models generalize downstream graph classification and regression tasks?

In order to perform experimental expressive power tests, we use graph8c and sr25 datasets[2]. Graph8c is composed of all the 11 117 possible connected non-isomorphic simple graphs with 8 nodes. We compare all possible pairs of graphs of this dataset, leading to more than 61M comparisons. According to our test, we found that 312 pairs out of 61M are 1-WL equivalent and none of the pairs are 3-WL equivalent. The sr25 dataset contains strongly regular graphs where each graph has 25 nodes, each node's degree is 12, connected nodes share 5 common neighbours and non-connected nodes share 6 common neighbors. Sr25 consists of 15 graphs, leading to 105 different pairs for comparison.

Moreover, we use the EXP dataset (Abboud et al., 2020), having 600 pairs of 1-WL equivalent graphs. This dataset also includes a binary classification task. Depending on graph features, each graph of a pair of 1-WL equivalent graphs is assigned to two different classes. We split the dataset into 400, 100, and 100 pairs for train, validation and test sets respectively. The test set is used to measure the generalization ability: a model that fails to distinguish 1-WL equivalent graphs inevitably fails to learn this task.

We use 3-layer graph convolution followed by sum readout layer, and then a linear layer to convert the readout layer representation into a 10-length feature vector. We keep the parameter budget around 30K for all methods. For graph8c, sr25 and EXP tasks, there is no learning. Model weights are randomly initialized and 10-length graph representations are compared by the Manhattan distance. If the distance is less than 10^{-3} in all 100 independent runs, we assume the pairs are similar. For EXP-classification task, we train the model and pick the best one according to validation set performance and report its performance on test set.

[1] https://github.com/balcilar/gnn-matlang
[2] http://users.cecs.anu.edu.au/∼bdm/data/graphs.html

Table 1. Number of undistinguished pairs of graphs in graph8c, sr25 and EXP datasets and binary classification accuracy on EXP dataset. An ideal method does not find any pair similar and classifies graphs with 100% accuracy. The number of pairs is 61M for graph8c, 105 pairs for sr25 and 600 for EXP.

MODEL	GRAPH8C	SR25	EXP	EXP-CLASSIFY
MLP	293K	105	600	50%
GCN	4775	105	600	50%
GAT	1828	105	600	50%
GIN	386	105	600	50%
CHEBNET	44	105	71	82%
PPGN	**0**	105	**0**	**100%**
GNNML1	333	105	600	50%
GNNML3	**0**	105	**0**	**100%**

Table 1 presents the obtained results. One can see that 99.5% of the graphs in graph8c dataset can be distinguished even by graph agnostic method MLP (293K out of 61M is not separable by MLP). This can be explained by the fact that the node degrees has been added as node features. Hence, all methods initially know the result of first iteration of 1-WL test. Thus, MLP (and also first iteration of 1-WL test) can distinguish pairs of graphs when multiset of node degrees are not same. GNNML1 and GIN's result is very closed to the theoretical limit of 1-WL test which is 312 pairs for graph8c dataset. The difference can be explained by threshold value to make decision if the two representations are equal and/or the number of layers in the model. It is possible that 1-WL test may need more than 3 iteration to distinguish some pairs. Due to having less expressive power of GCN and GAT compare to the 1-WL test, their performances are worse than 1-WL test. Since graph8c dataset has 1-WL equivalent non-regular graph pairs that have different maximum eigenvalue, Chebnet could detect these pairs and reaches better performance than theoretical limit of 1-WL test as stated by Theorem 2.

On EXP dataset, composed of 1-WL equivalent graph pairs, MPNNs cannot distinguish any pair of graphs, except Chebnet which is able to distinguish all the pairs with different maximum eigenvalues. In EXP there is no regular graphs and only 71 graph pairs have similar maximum eigenvalues. Chebnet fails on these pairs but distinguishes the others, as stated by Theorem 2. One can note that using a fixed value for maximum eigenvalue (e.g. $\lambda_{max} = 2$ as it is usually done in practice) reduces Chebnet performance to those of MPNNs.

Similarly to results on EXP, 1-WL equivalent MPNNs except Chebnet fail to predict of EXP classification task and do not perform better than random prediction. On the contrary, PPGN and GNNML3 have perfect results on graph8c, EXP and EXP-classify tasks thanks to their 3-WL equivalence. However, since strongly regular graphs are 3-WL equivalent, no model less or as powerful as 3-WL test can distinguish the pairs in sr25 dataset. To obtain a better result on this dataset, we need to go further than 3-WL (see Appendix E). These experiments reply to **Q1**.

Table 2. Median of test set MSE error for graphlet counting problem on random graph dataset over 10 random runs.

MODEL	3-STARS	CUSTOM	TRIANGLE	TAILED-TRI	4-CYCLES
MLP	**1.0E-4**	4.58E-1	3.13E-1	2.22E-1	1.73E-1
GCN	**1.0E-4**	3.22E-3	2.43E-1	1.42E-1	1.14E-1
GAT	**1.0E-4**	4.57E-3	2.47E-1	1.44E-1	1.12E-1
GIN	**1.0E-4**	1.47E-3	2.06E-1	1.18E-1	1.21E-1
CHEBNET	**1.0E-4**	**7.68E-4**	2.01E-1	1.15E-1	9.60E-2
PPGN	**1.0E-4**	**9.19E-4**	**1.00E-4**	**2.61E-4**	**3.30E-4**
GNNML1	**1.0E-4**	**2.75E-4**	2.45E-1	1.32E-1	1.14E-1
GNNML3	**1.0E-4**	**7.24E-4**	**4.44E-4**	**3.18E-4**	**6.62E-4**

To bring an answer to **Q2**, we propose to count 3-star, triangle, tailed-triangle and 4-cycle substructures (Fig. 2). In addition to these 4 graphlets, we also create another task (noted as CUSTOM in Table 2) that aims to approximate a custom sentence $e_c \in \mathcal{L}_1^+$, $e_c(A) = \mathbf{1}^\top A\, diag(exp(-A^2\mathbf{1}))A\mathbf{1}$ with A the graph adjacency matrix. Since $e_c \in ML(\mathcal{L}_1^+)$, it may be learnable by 1-WL equivalent MPNNs. We used the RandomGraph dataset (Chen et al., 2020) with same partitioning: 1500, 1000 and 2500 graphs for train, validation and test respectively. To create the ground truth of number of graphlets, we count them according to theorem proofs in Appendix B.3, B.4, B.5 and normalized the number to a unitary standard deviations, to keep the errors in the same scale as in Table 2. We use 4 convolution layers, a graph readout layer computing a sum and followed by 2 fully connected layers. All methods parameter budget is around 30K. We keep the maximum number of iterations to 200 and we stop the algorithm if the error goes below 10^{-4}.

The results in Table 2 are consistent with Theorems 3, 4, 5. 3-WL models are able to count graphlets and approximate our custom function (result $< 10^{-3}$), while 1-WL equivalent models can only count the 3-stars graphlet, as stated in Theorem 3. Custom function approximation results also show that GNNML1 and Chebnet provide better approximation of the target other MPNNs, which is again consistent with our analysis.

Question **Q3** concerns the spectral expressive power of models. Such an analysis is important when input-output relations depend on the spectral properties of the graph signal such as in image/signal processing applications. As shown in (Balcilar et al., 2021), the vast majority of existing MPNNs operate as low-pass filters which limits their capacity. To lead this analysis, we use the datasets presented in (Balcilar et al., 2021). First, we evaluate if the models can learn low-pass, high-pass and band-pass filtering effects, through a node regression problem. Model performances are thus reported R^2 using mean square error (MSE) loss. The original data consists in a 2-d grid graph of size 100x100.

Table 3. Spectral expressive analysis results. R^2 for LowPass, HighPass and BandPass node regression tasks, accuracy on graph classification task. Results are median of 10 different runs.

MODEL	LOW-PASS	HIGH-PASS	BAND-PASS	CLASSIFY
MLP	0.9749	0.0167	0.0027	50.0%
GCN	0.9858	0.0863	0.0051	77.9%
GAT	0.9811	0.0879	0.0044	85.3%
GIN	0.9824	0.2934	0.0629	87.6%
CHEBNET	**0.9995**	**0.9901**	**0.8217**	**98.2%**
PPGN	**0.9991**	**0.9925**	0.1041	91.2%
GNNML1	**0.9994**	**0.9833**	0.3802	92.8%
GNNML3	**0.9995**	**0.9909**	**0.8189**	**97.8%**

Table 4. Results on Zinc12K and MNIST-75 datasets. The values are the MSE for Zinc12K and the accuracy for MNIST-75. Edge features are not used even if they are available in the datasets. For Zinc12K, all models use node labels. For MNIST-75, the model uses superpixel intensive values and node degree as node features.

MODEL	ZINC12K	MNIST-75
MLP	0.5869 ± 0.025	$25.10\% \pm 0.12$
GCN	0.3322 ± 0.010	$52.80\% \pm 0.31$
GAT	0.3977 ± 0.007	$82.73\% \pm 0.21$
GIN	0.3044 ± 0.010	$75.23\% \pm 0.41$
CHEBNET	0.3569 ± 0.012	$\mathbf{92.08\% \pm 0.22}$
PPGN	$\mathbf{0.1589 \pm 0.007}$	$90.04\% \pm 0.54$
GNNML1	0.3140 ± 0.015	$84.21\% \pm 1.75$
GNNML3	$\mathbf{0.1612 \pm 0.006}$	$\mathbf{91.98\% \pm 0.18}$

Since the PPGN's memory and computational complexity is prohibitive with a reasonable computer, we select 3 different 30x30 regions of the original 2-d grid graph as training, validation and test sets. A second dataset consists of 5K planar graphs, split into 3K, 1K and 1K sets for train, validation and test. They are used to evaluate if the models can classify graphs into binary classes where the ground truth labels were determined according to the frequency of the signal on the graph. Since the problem is binary graph classification we use binary cross entropy loss.

The results of spectral expressive power analysis are presented in Table 3. Node regression results show that 1-WL equivalent existing MPNNs can mostly learn low-pass effects. By applying different weights to self node and neighbourhood, GNNML1 can learn high pass effect relatively well. PPGN also learns high-pass effect better than 1-WL equivalent methods. Band-pass can be generalized by Chebnet and GNNML3 thanks to the convolutions designed in spectral domain. The reason why the band-pass regression results are worse than the low and high-pass results is that the ground truth band-pass effect is created by very stiff frequency function and Chebnet also GNNML3 need more convolution supports to learn it. Because of non-local process in PPGN, it cannot learn the band-pass effect and provide no better result than 1-WL MPNNs in graph classification problem. Thus, Chebnet and GNNML3 give the best results on all spectral ability test, thanks to their spectral convolutions process.

For answering the last question **Q4**, we apply the different models on some common benchmark tasks and datasets. Table 4 and Table 5 present the performance of both baseline models and the proposed ones on these benchmark datasets. The results on Zinc12K and MNIST-75 datasets are very interesting because of the nature of these two problems. The solution of the Zinc12K dataset mostly depends on structural features of the graph. For instance, a recent study reaches 0.14 MAE by using handcrafted features, which cannot be extracted by a 3-WL equivalent model (Bouritsas et al., 2020). Obtained results confirm that models that are able to count substructures, such as PPGN and GNNML3, perform better than others with a large margin. On the other hand, since MNIST-75 dataset is based on image analysis, it needs a model with a higher spectral ability. Therefore, Chebnet and GNNML3 perform significantly better than other models on this task. Our proposal GNNML3 gives comparable results on other TU datasets in (Morris et al., 2020) such as MUTAG, ENZYMES, PROTEINS and PTC presented in Appendix F.

7. Conclusion

Despite a computational and memory efficiency, MPNN is known to have an expressive power limited to 1-WL test. MPNN is then unable to distinguish 1-WL equivalent graphs and cannot count some substructures of the graph. In this paper, we have presented new models, by translating the insights of MATLANG to the GNN world. This solution gives access to a new MPNN that is theoretically more powerful than the 1-WL test, and experimentally as powerful as 3-WL existing models for distinguishing non-isomorphic graphs and for counting substructures without feature engineering nor node permutations in the training phase. The proposed MPNN is also powerful in terms of spectral expressive ability, going beyond low-pass filtering, which is another expressive perspective of GNNs. Experimental results confirm the theorems stated in the paper. The proposed method has a big advantage over all studied MPNN on graph isomorphism and substructure counting tasks. With respect to the 3-WL equivalent baseline PPGN, the biggest advantage of our proposal is its complexity. Proposed GNNML3 needs linear memory and time complexity with respect to the number of nodes, while PPGN needs quadratic memory and cubic time complexity, making the model infeasible for large graphs. The second advantage over PPGN is that since it is created in the spectral domain, its convolution process takes care of signal frequencies, making it more efficient in terms of output signal frequency profile.

Acknowledgments

This work was partially supported by the Normandy Region (grant RiderNet), the French Agence National de Recherche (grant APi, ANR-18-CE23-0014) and the PAUSE Program of Collège de France.

References

Abboud, R., Ceylan, İ. İ., Grohe, M., and Lukasiewicz, T. The surprising power of graph neural networks with random node initialization. *arXiv preprint arXiv:2010.01179*, 2020.

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*, 2020.

Balcilar, M., Renton, G., Héroux, P., Gaüzère, B., Adam, S., and Honeine, P. Analyzing the expressive power of graph neural networks in a spectral perspective. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=-qh0M9XWxnv.

Barceló, P., Kostylev, E. V., Monet, M., Pérez, J., Reutter, J., and Silva, J. P. The logical expressiveness of graph neural networks. In *International Conference on Learning Representations*, 2019.

Battaglia, P. W., Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

Bouritsas, G., Frasca, F., Zafeiriou, S., and Bronstein, M. M. Improving graph neural network expressivity via subgraph isomorphism counting. *arXiv preprint arXiv:2006.09252*, 2020.

Bresson, X. and Laurent, T. Residual gated graph convnets, 2018. URL https://openreview.net/forum?id=HyXBcYg0b.

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.

Chami, I., Abu-El-Haija, S., Perozzi, B., Ré, C., and Murphy, K. Machine learning on graphs: A model and comprehensive taxonomy. *arXiv preprint arXiv:2005.03675*, 2020.

Chen, Z., Chen, L., Villar, S., and Bruna, J. Can graph neural networks count substructures? *arXiv preprint arXiv:2002.04025*, 2020.

Chung, F. *Spectral graph theory*. American Mathematical Society, 1997.

Dasoulas, G., Dos Santos, L., Scaman, K., and Virmaux, A. Coloring graph neural networks for node disambiguation. In Bessiere, C. (ed.), *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pp. 2126–2132, 7 2020. doi: 10.24963/ijcai.2020/294. URL https://doi.org/10.24963/ijcai.2020/294.

Defferrard, M., Bresson, X., and Vandergheynst, P. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems*, pp. 3844–3852, 2016.

Dehmamy, N., Barabási, A.-L., and Yu, R. Understanding the representation power of graph neural networks in learning graph topology. In *NeurIPS*, pp. 15387–15397, 2019.

Dwivedi, V. P., Joshi, C. K., Laurent, T., Bengio, Y., and Bresson, X. Benchmarking graph neural networks. *arXiv preprint arXiv:2003.00982*, 2020.

Fey, M. and Lenssen, J. E. Fast graph representation learning with pytorch geometric. *arXiv preprint arXiv:1903.02428*, 2019.

Geerts, F. On the expressive power of linear algebra on graphs. *Theory of Computing Systems*, Oct 2020. ISSN 1433-0490. doi: 10.1007/s00224-020-09990-9. URL https://doi.org/10.1007/s00224-020-09990-9.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyal, O., and Dahl, G. E. Neural message passing from quantum chemistry. In *Proceedings of the International Conference on Machine Learning*, 2017.

Hamilton, W., Ying, Z., and Leskovec, J. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pp. 1024–1034, 2017.

Hammond, D. K., Vandergheynst, P., and Gribonval, R. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis*, 30(2):129–150, 2011.

Harary, F. and Manvel, B. On the number of cycles in a graph. *Matematický časopis*, 21(1):55–63, 1971.

Hornik, K., Stinchcombe, M., and White, H. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

Immerman, N. and Lander, E. Describing graphs: A first-order approach to graph canonization. In *Complexity theory retrospective*, pp. 59–81. Springer, 1990.

Keriven, N. and Peyré, G. Universal invariant and equivariant graph neural networks. In *Advances in Neural Information Processing Systems 32*, pp. 7092–7101. 2019.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.

Maron, H., Ben-Hamu, H., Serviansky, H., and Lipman, Y. Provably powerful graph networks. In *Advances in Neural Information Processing Systems*, pp. 2156–2167, 2019a.

Maron, H., Fetaya, E., Segol, N., and Lipman, Y. On the universality of invariant networks. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 4363–4371, Long Beach, California, USA, 09–15 Jun 2019b. PMLR.

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. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01): 4602–4609, Jul. 2019.

Morris, C., Kriege, N. M., Bause, F., Kersting, K., Mutzel, P., and Neumann, M. Tudataset: A collection of benchmark datasets for learning with graphs. *arXiv preprint arXiv:2007.08663*, 2020.

Murphy, R., Srinivasan, B., Rao, V., and Ribeiro, B. Relational pooling for graph representations. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 4663–4673, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

Pinar, A., Seshadhri, C., and Vishal, V. Escape: Efficiently counting all 5-vertex subgraphs. pp. 1431–1440, 2017.

Sato, R., Yamada, M., and Kashima, H. Approximation ratios of graph neural networks for combinatorial problems. In *Advances in Neural Information Processing Systems*, volume 32, pp. 4081–4090. Curran Associates, Inc., 2019.

Sato, R., Yamada, M., and Kashima, H. Random features strengthen graph neural networks. *arXiv preprint arXiv:2002.03155*, 2020.

Shuman, D. I., Narang, S. K., Frossard, P., Ortega, A., and Vandergheynst, P. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE signal processing magazine*, 30(3):83–98, 2013.

Takapoui, R. and Boyd, S. Linear programming heuristics for the graph isomorphism problem. *arXiv preprint arXiv:1611.00711*, 2016.

Van Dam, E. R. and Haemers, W. H. Which graphs are determined by their spectrum? *Linear Algebra and its applications*, 373:241–272, 2003.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., and Bengio, Y. Graph attention networks. In *International Conference on Learning Representations (ICLR)*, 2018.

Vignac, C., Loukas, A., and Frossard, P. Building powerful and equivariant graph neural networks with structural message-passing. In *NeurIPS*, 2020.

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., and Yu, P. S. A comprehensive survey on graph neural networks. arXiv preprint arXiv:1901.00596, 2019.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019.