The Expressive Power of Graph Neural Networks

Xiaoning Guo
xg2084@nyu.edu

Dengtai Wang
dw2691@nyu.edu

Hanyu Zhou
hz1649@nyu.edu

Jing Tao
jt3149@nyu.edu

Miaozhi Yu
my1251@nyu.edu

Abstract

Recently, Graph Neural Networks (GNNs) have shown promising results in many graph-related machine learning problems. Despite the overwhelming success in GNNs applied to many domains, research shows that GNNs can sometimes fail to solve even the simplest graph tasks. In this survey paper, we explore the latest theoretical research on the expressive power of graph neural networks, their representation limits and how researchers attempt to overcome these limits.

1. Introduction

Graph Neural Networks (GNNs) are a powerful class of neural networks that have been applied to many areas of machine learning that require representing data as relationships defined by connections of nodes and edges. These applications include recommendation systems (Wang et al., 2021), protein interface prediction (Fout et al., 2017), community detection (Chen et al., 2017), physics simulations (Sanchez-Gonzalez et al., 2020) and more. While GNNs have shown promising empirical results, the theoretical understanding of their representation limits is still a working progress.

In this survey paper, we summarize how researchers examine the expressive power of GNNs using the Wiesfeiler-Lehman hierarchy. In fact, it was discovered that message passing neural networks (or traditional GNNs) are no more powerful than the 1-WL algorithm in distinguishing non-isomorphic graphs which means that GNNs cannot solve even the simplest graph tasks. We then discuss examples where researchers attempt to overcome these limits by introducing higher order GNNs. However, these approaches come at a cost of intractable computational complexity and non-locality. Finally, we describe the latest work in GNNs based on algebraic topology and compare them to previous GNN architectures.

2. WL Hierarchy

2.1 Graph Isomorphism Test

One way to examine the power of Graph Neural Networks is to see whether Graph Neural Networks can distinguish different types of graph structures. This is a classical question in graph theory known as the graph isomorphism problem.

In this section, we first briefly introduce graph isomorphism and the Weisfeiler-Lehman graph isomorphism test (Leman, 2018).

Definition 1. *Two finite graphs, G and G', are isomorphic if there exists a permutation of vertices of the first graph such that permuting both vertices of each pair in E is a bijection between two graphs.*

In other words, if two graphs are isomorphic then they only differ by the labels of the vertices and edges. There is a complete structural equivalence between two such graphs. 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 which can distinguish all pairs of non-isomorphic graphs is a universal approximator. Since it is not known if the graph isomorphism problem can be solved in polynomial time or not, this problem is neither NP-complete nor P, but NP-intermediate. The Weisfeiler-Lehman Test (WL-test) produces for each graph a canonical form. If the canonical forms of two graphs are not equivalent, then the graphs are definitively not isomorphic. However, it is possible for two non-isomorphic graphs to share a canonical form, so this test alone cannot provide conclusive evidence that two graphs are isomorphic. To illustrate this idea, we start with all nodes of identical color in Figure 1 [1]. At each step, the algorithm aggregates the colors of nodes and their neighbourhoods representing them as multisets, and hashes the aggregated color multisets into unique new colors. The algorithm stops upon reaching a stable coloring.

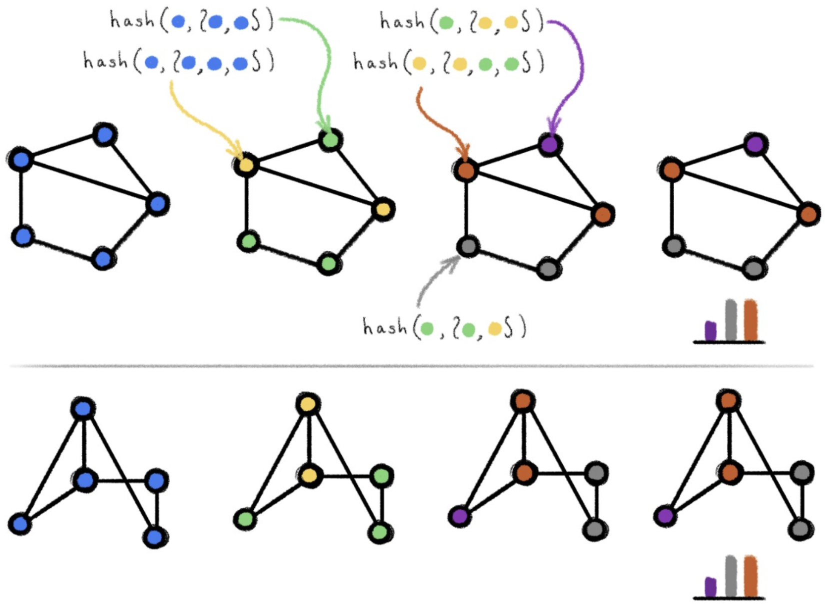


Figure 1: Example of running the Weisfeiler-Lehman test on two isomorphic graphs. The algorithm stops when a stable coloring is reached. If the colorings of two graphs are the same, they are possibly isomorphic. Image from Michael Bronstein.

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 equal to the order of the tuple. It is important to mention that an higher order of tuple leads to a better

1. https://towardsdatascience.com/expressive-power-of-graph-neural-networks-and-the-weisefeiler-lehman-test-b883db3c7c49

ability to distinguish two non-isomorphic graphs (with the exception for $k = 2$). 1-WL and 2-WL tests are equivalent, $(k + 1)$-WL is strictly stronger than k-WL for $k \geq 2$.

The WL-test does an iterative graph recoloring. At each step, the algorithm aggregates the color of nodes and their neighborhoods and when the coloring of two graphs diverges, they are deemed to be non-isomorphic. This neighborhood aggregation practice is very similar to the Message Passing Neural Network(MPNN) architecture, which in each of its layers, the features of each node is updated through aggregating itself and its neighborhood. It is proved that for a MPNN, if we choose the aggregation and update function to be multiset injective function, then the model is equivalent to a WL test(Xu et al., 2019).

The equivalence is established between traditional MPNN and the WL test. To attain more expressive power, it is possible to make a GNN to replicate higher order WL tests and to be strictly more powerful than a traditional MPNN. Since k-WL operates on k-tuples of nodes, the higher order GNN also operates on k-tuples of nodes and thus have memory complexity $O(n^k)$.

2.2 Universal Approximation Theorem

Another way to examine expressive power is through the lens of function approximation. In fact, Chen et al. (2019) proved that the study of expressive power through graph isomorphism testing and function approximation are equivalent. The universal approximation theorem is well known for traditional neural networks which says that neural networks with sufficient width or depth can approximate a large variety of functions provided that the weights are chosen appropriately (and non-liner activation functions). That is, for any arbitrary function $f(x)$, there exists a neural network $\hat{f}(x)$ such that $\sup_{x \in X} ||\hat{f}(x) - f(x)|| < \epsilon$.

Maron et al. (2019) derived a universal approximation theorem for permutation-invariant functions while Keriven and Peyré (2019) extended the theorem to equivariant functions. These two properties are especially important for graph neural networks where either the permutation of nodes should not change the outputs or permutation of nodes should also permute the outputs. However, these theoretical guarantees require higher order tensorization which may not be tractable in practice.

3. Message Passing Neural Networks

3.1 Graph Neural Networks Types

The vast majority of the GNN architectures fall under three different categories: graph convolution networks (Kipf and Welling, 2016), graph attention networks (Veličković et al., 2017), Message passing networks (Gilmer et al., 2017). In all three categories, permutation invariance is ensured by aggregating features $\mathbf{X}_{N(u)}$ of some potential transformation ψ with some permutation invariant function $\bigoplus$, then updating the feature of node u, followed by some transformation ϕ. Typically, ϕ and ψ are learnable.

In the GCN, the features of the neighbourhood nodes are directly aggregated with fixed weights,

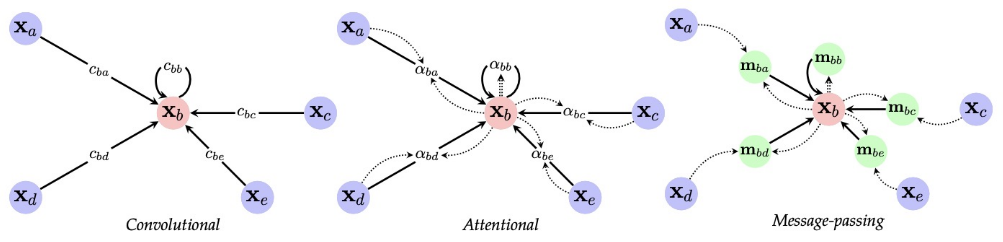


Figure 2: Left-to-right: GCN, where sender node features are multiplied with a constant, c_{uv} ; GAT, where this multiplier is implicitly computed via an attention mechanism of the receiver over the sender: $\alpha_{uv} = a(\mathbf{x_u}, \mathbf{x_v})$; and MPN, where vector-based messages are computed based on both the sender and receiver: $\mathbf{m}_{uv} = \psi(\mathbf{x}_u, \mathbf{x}_v)$. Image from Bronstein et al. (2021).

$$\mathbf{h}_u = \phi(\mathbf{x}_u, \bigoplus_{v \in N(u)} c_{uv}\psi(\mathbf{x}_v)) \tag{1}$$

The amount of aggregation c_{uv} is completely determined by the graph structure, such as the spectral normalized adjacency matrix.

Similar to GCN, a GAT performs local averaging of hidden features. However, instead of only using the graph structure to perform the propagation, it learns to reweight the propagation weight c_{uv} based on the hidden features via a learnable function a that computes the importance coefficients $\alpha_{uv} = a(\mathbf{x_u}, \mathbf{x_v})$.

$$\mathbf{h}_u = \phi(\mathbf{x}_u, \bigoplus_{v \in N(u)} \alpha_{uv}\psi(\mathbf{x}_v)) \tag{2}$$

Message passing networks further generalize the idea of having control over the amount of information aggregated by replacing the hidden feature aggregation with some arbitrary function of hidden features of two nodes.

$$\mathbf{h}_u = \phi(\mathbf{x}_u, \bigoplus_{v \in N(u)} \psi(\mathbf{x}_u, \mathbf{x}_v)) \tag{3}$$

Actually, GAT can represent GCN by an attention mechanism implemented as a look-up table $a(\mathbf{x_u}, \mathbf{x_v}) = c_{uv}$, and both GCN and GAT are special cases of MPN where the messages are only the sender nodes' features: $\psi(\mathbf{x}_u, \mathbf{x}_v) = c_{uv}\psi(\mathbf{x}_v)$ for GCN and $\psi(\mathbf{x}_u, \mathbf{x}_v) = \alpha_{uv}\psi(\mathbf{x}_v)$ for GAT. Thus, the inclusion relation is $GCN \subseteq GAT \subseteq MPN$. Figure 2 illustrates three different categories. Therefore, we can say that most of the popular GNN models such as GCN, GAT, GraphSage(Hamilton et al., 2017), Graph Isomorphism Networks(Xu et al., 2019) etc fall under the category of message passing neural networks.

3.2 Can GNNs Count Substructures?

Chen et al. (2020) showed the ability to detect and count substructures as a study of the expressive power of most popular GNN models. Counting the number of substructures in a graph-structured data is a very intuitive and practical application of GNNs. It is useful in computational chemistry, computational biology and social network studies.

Naturally, the ability to count substructures entails the ability to distinguish between graphs with different counts of a given substructure in the sense that the count output will be different if graphs have different counts of substructures. For the converse argument, with the help of universal approximation theorem, if a function can distinguish between graphs with different counts of a given substructure, we can approximate that function with neural networks. Thus, we build an equivalence between the two abilities and we can now bring in the WL test and talk about the ability to count substructures of the WL test.

There are two types of substructure counting, one is subgraph count and the other is induced subgraph count. The difference is that for an induced subgraph, for any two vertices in the induced subgraph, if there exists an edge connecting the two vertices, it must be included in the induced subgraph. Hence, induced subgraphs are a subset of subgraphs.

2-WL test cannot induced-subgraph-count any connected pattern with 3 or more nodes. Chen et al. (2020) proved that MPNN architectures are at most as powerful as a 2-WL test, so graphs that are indistinguishable for 2-WL test can not be distinguished by MPNN. Therefore, MPNN cannot perform induced-subgraph-count on the same setting. MPNNs can perform subgraph-count on star-shaped patterns. Thanks to the equivalence proved by Xu et al. (2019), 2-WL can do that as well.

3.3 Can GNNs Determine Graph Properties?

Models that fall into the category of MPNN cannot distinguish graphs that have different graph properties such as girth (length of the shortest cycle), circumference (length of the longest cycle), diameter (maximum distance, in terms of shortest path, between any pair of nodes in the graph), radius (minimum node eccentricity, where eccentricity of a node u is defined as the maximum distance from u to other vertices), conjoint cycle (two cycles that share an edge), total number of cycles, and k-clique (a subgraph of at least $k \geq 3$ vertices such that each vertex in the subgraph is connected by an edge to any other vertex in the subgraph) which is equivalent to say that these models cannot compute these graph properties. Garg et al. (2020) proved this by providing counter examples as shown in Figure 3 where the graph with two copies of S4 is indistinguishable from S8 despite having different girth, circumference, diameter, radius, and total number of cycles. Constructed similarly, the graph with two copies of G1, each having a conjoint cycle, cannot be distinguished from G2. The result can be easily extend to k-clique by modifying the graph.

3.4 Can GNNs Compute Graph Algorithms?

Loukas (2020) demonstrates the first impossibility results that explicitly connect GNN properties (depth and width) with graph properties and that go beyond isomorphism by addressing decision, optimization, and estimation graph problems. He proves that many algorithms are possible for GNNs to perform only if the GNN has a large enough receptive field. With

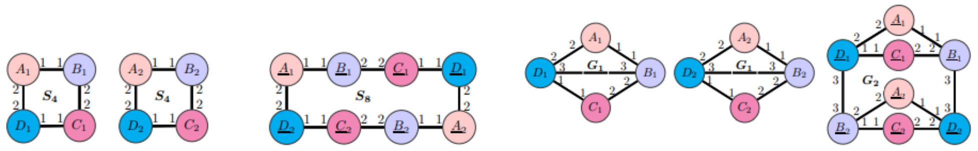
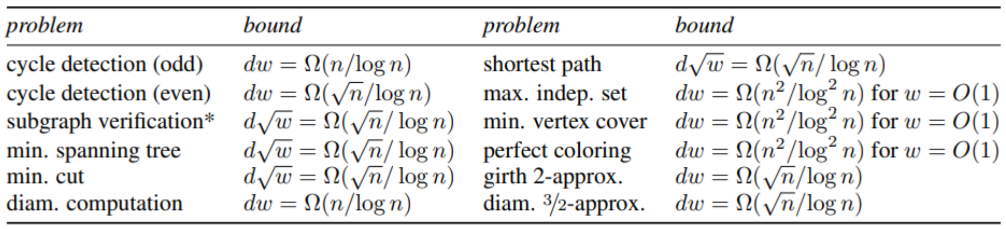


Figure 3: Leftmost are two copies of S_4, they are identical to S_8 from the perspective of WL test, which has the same amount. of nodes as that of two S_4. The same goes for two copies of G_1 and G_2. Image from Garg et al. (2020).

the help of the LOCAL model from a different field of study, distributed computing, the author managed to draw equivalence between the expressive power of GNNs and the LOCAL model and thus transfer the results from LOCAL model to GNNs. He gives the lower bound necessary for GNNs to solve classic algorithms in decision, optimization, and estimation graph problems in terms of n, number of nodes in the input graph, depth, d, the number of layers of the network, and width, w, the largest dimension of state x over all layers and nodes. One thing to note here is that in the original paper, Loukas (2020) studied the problems under the setting that each node in the graph has a unique/random ID which gives extra expressive power to the GNN models while in the vanilla settings, nodes from a graph cannot identify each other. This extra expressive power enables model to perform tasks that are impossible for an anonymous GNN(a GNN that does not use any ID) for example, detecting estimating cycles and girth which we have shown to be impossible in the previous section.



problem	*bound*	*problem*	*bound*
cycle detection (odd)	$dw = \Omega(n/\log n)$	shortest path	$d\sqrt{w} = \Omega(\sqrt{n}/\log n)$
cycle detection (even)	$dw = \Omega(\sqrt{n}/\log n)$	max. indep. set	$dw = \Omega(n^2/\log^2 n)$ for $w = O(1)$
subgraph verification*	$d\sqrt{w} = \Omega(\sqrt{n}/\log n)$	min. vertex cover	$dw = \Omega(n^2/\log^2 n)$ for $w = O(1)$
min. spanning tree	$d\sqrt{w} = \Omega(\sqrt{n}/\log n)$	perfect coloring	$dw = \Omega(n^2/\log^2 n)$ for $w = O(1)$
min. cut	$d\sqrt{w} = \Omega(\sqrt{n}/\log n)$	girth 2-approx.	$dw = \Omega(\sqrt{n}/\log n)$
diam. computation	$dw = \Omega(n/\log n)$	diam. $3/2$-approx.	$dw = \Omega(\sqrt{n}/\log n)$

Figure 4: A summary of the results. Algorithms for decision problems: Subgraph detection, subgraph verification. Algorithms for optimization problems: the minimum cut problem, the shortest s-t path problem, the minimum spanning tree problem. Algorithms for NP-hard optimization problems: the minimum vertex cover problem, the maximum independent set problem, the perfect coloring problem. Algorithms for estimation problem: approximate the girth and graph diameter within a factor of 2 and 3/2, respectively. Image from Loukas (2020)

The result are shown in Figure 4. Loukas (2020) provides a lower bound for each problem. Note that the result comes from a setting where each node can identify each other, the lower bound will still hold if we remove this setting, though there might be tighter bounds.

4. Beyond 1-WL

There have been many attempts to increase the expressive power of message passing GNNs in distinguishing non-isomorphic graphs from 1-WL to k-WL. These higher order GNN architectures based on the k-WL hierarchy operate on $k-$tuples of nodes which require higher dimensional tensors. While these architectures are strictly more powerful than traditional message passing GNNs, two major drawbacks of these approaches is that they often require $O(n^k)$ memory and are non-local. This makes these architectures impractical to use at scale.

Morris et al. (2019) introduced higher order GNNs called $k-$GNNs that for a given k operate on all k element subsets of $[V(G)]^k$ over $V(G)$. When k is an integer larger than 2, $k-$GNN's expressive power is strictly stronger than $(k-1)$-GNN and is equal to k-WL test. However, the computational time and memory costs of $k-$GNNs also increases exponentially with k as we recursively consider all possible subsets of G which may include subsets that break locality. There have been many other architectures which can increase the expressive power and possibly beyond the $k-$WL bounds.

4.1 Graph Reconstruction

Graph reconstruction derives from the reconstruction conjecture, which states that a graph can be decided by the multi-set of its induced subgraphs. Then two graphs are isomorphic if they have same induced subgraphs, which can be a powerful method provided that the reconstruction conjecture holds.

The full reconstruction neural network build the induced subgraph representations with smaller induced subgraphs, thus recursively calculate the representation of the whole graph. It has exponential computational complexity since a graph has exponential subgraphs. Furthermore, we can build a k-Reconstruction GNN (Cotta et al., 2021) which only compute the subgraphs with no more than k nodes as a trade-off between the expressive power and the efficiency.

The expressive power of the k-Reconstruction GNN varies with k. Since two graphs definitely have the same k-deck if they have the same $(k+1)$-deck, we can conclude that expressive power increases with k. Therefore, the $(n-1)$-Reconstruction GNN reaches the maximum expressive power of graph reconstruction and the 3-Reconstruction GNN is the least expressive since 2-reconstruction subgraphs would be degraded to the number of edges.

When compared with standard GNN, k-Reconstruction GNN can distinguish regular graphs like CSL(circular skip links) with large enough k unlike standard GNNs. An example of how this happens is shown in Figure 5. However, when k is small there are graphs that GNN can distinguish but k-Reconstruction GNN cannot therefore k-Reconstruction GNNs do not follow the same expressive hierarchy as $k-$GNNs.

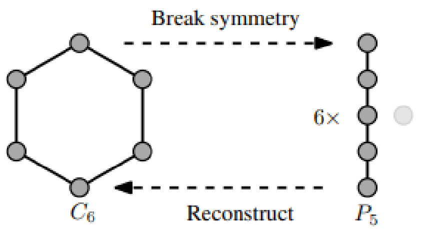


Figure 5: A cycle, undistinguishable by GNNs, and how reconstruction makes it distinguishable.

4.2 Dropout

A new framework called Dropout Graph Neural Networks (DropGNN) was proposed in Papp et al. (2021) to overcome some limitation of the standard GNN. DropGNN can be considered as an ensemble model of standard GNN where it executes multiple runs of GNN each with certain nodes drop out and then aggregate the results from all the runs as the final result. In each run every node will be dropped with a certain probability p and it will not send or receive message from its neighbors. It is proved in the paper that DropGNN is able to recognize various graph neighborhoods that standard GNN is not able to by message passing, thus greatly improved the expressiveness of the GNN. Basically what it does is that in each round, by dropping certain nodes, it introduces perturbation to the nodes neighborhood and the embedding calculation will be slightly different. In this way the neighborhood difference can be learnt by the dropout. In the aggregation step, it applies a non-linear transformation (sigmoid or step function) of the final embedding from each GNN run and on top of it further applies a permutation-invariant function, for example *max*, *mean* or *sum* for final aggregation.

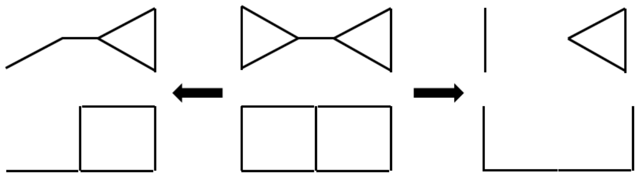


Figure 6: Two graphs become easy to distinguish after dropping out a node.

The number of runs required for the dropout result to stabilize has positive correlation with the size of the neighbor for each node u, denoted as Γ. In order to ensure that we execute enough number of runs (denoted as r) so that 1-dropout is observed a few times, we need to set the probability of dropout to be $*p = \frac{1}{1+\gamma}$ where γ is the size of Γ. With this choice of p, the number of runs required is bounded by $r > e(\gamma + 1) = \Omega(\gamma)$.

It would be natural to ask about the expressive limits of DropGNN as it can succeed in cases where WL-tests fails. The analysis is based on what aggregation function we choose.

For *sum*, it is proved that if $r \geq \Omega(\gamma \log(\gamma t))$ and $p = \frac{1}{t}$, then a dropGNN with port numbers can distinguish any two non-isomorphic d-hop neighborhoods. The general expressive power of dropGNN is closely related to the graph reconstruction network. For *mean* aggregation function, let $S1$ and $S2$ denote two multisets of feature vectors that have the same size, then dropGNN is able to distinguish $S1$ and $S2$. However, if $|S1| \neq |S2|$, there is no proof yet that dropGNN can distinguish them. For *max* aggregation function, however, it does not work well with the dropout method.

4.3 Graph Substructure Network

We have seen in section 3.2 that normal GNNs have limitation on detecting and counting substructures while on the other hand, substructures are often intimately related to downstream tasks in network science and bioinformatics. Therefore, Bouritsas et al. (2020) proposed a network, Graph Substructure Network that pre-computes and encodes the count of substructures related to each node/edge and train these new nodes and edges with a slightly modified MPNN which can take the additional structural information. As shown in Figure 7 [2], two graphs that are ambiguous from the perspective of WL test are now distinguishable with structural information encoded. This implies that with the proper choice of substructure, GSN will be more powerful than the basic WL test or even higher order k-WL test.

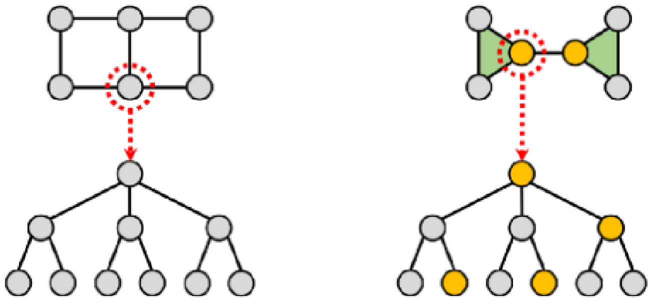


Figure 7: If we look at the node circled in red, two nodes have the same state after two steps of aggregation from their neighbors as illustrated below the graphs. After recoloring the nodes with one triangle, two graphs are not ambiguous anymore. Image from Michael Bronstein.

The advantage of GSN is that it is essentially still MPNN with additional edge/node features which preserves locallity and has linear training, inference and memory complexity. The extra computation cost occurs in the pre-processing step which would only need to happen once for many applications making the $O(n^k)$ computation cost(counting substructures of size k in a graph with n nodes) somewhat tolerable.

2. https://towardsdatascience.com/using-subgraphs-for-more-expressive-gnns-8d06418d5ab

The expressive power of GSN is dependent on the type of substructure information we encode in the node/edge feature. Generally speaking, using structures more complex than stars which MPNN is capable of detecting, GSN can be strictly more powerful than 1-WL test or equivalent standard MPNN. However, the expressive power of GSN in principle is still unsolved.

5. A New Frontier: GNNs based on Algebraic Topology

Previously, we saw how message passing GNNs were upper-bounded by the expressiveness of 1-WL and to overcome these representation limits, higher order GNNs were introduced but at a cost of non-locality and high computational complexity. Authors of a new generation of graph neural networks (Bodnar et al., 2021a,b) based on algebraic topology propose a principled way to enrich the graph representations with higher order structure and less severe drawbacks.

5.1 Cellular Graph Neural Networks

We consider the higher order structures called cellular complexes which can be created from normal graphs via lifting transformations (Figure 8). A regular cell complex is a topological space X built from cells. Cells are topological objects where 0-cells represent nodes, 1-cells represents edges, and 2-cells represents faces. Of course these cells can be abstracted to n-dimensions.

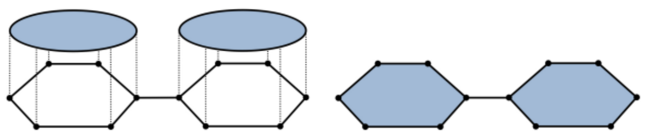


Figure 8: A lifting transformation that attaches disks to the boundary of the rings of a graph.

Cells σ are adjacent to one another via four boundary relations which form the basis for cellular message passing:

- Boundary adjacent cells $B(\sigma) = \{\tau | \tau \prec \sigma\}$ e.g., boundary cells of edges are vertices.
- Co-boundary adjacent cells $C(\sigma) = \{\tau | \sigma \prec \tau\}$ e.g., co-boundary cells of a vertex are the edges it is part of.
- Lower adjacent cells $\mathcal{N}_{\uparrow} = \{\tau | \exists \delta \text{ such that } \delta \prec \tau \text{ and } \delta \prec \sigma\}$. These are cells of the same dimension as σ but they share a lower dimensional cell on their boundary. e.g., edges in a ring where vertices are on the boundary between the edges.
- Upper adjacent cells $\mathcal{N}_{\downarrow} = \{\tau | \exists \delta \text{ such that } \tau \prec \delta \text{ and } \sigma \prec \delta\}$ These are cells of the same dimension as σ but they share a higher dimensional cell on their boundary. e.g., vertices in a ring where edges are on the boundary between the vertices.

Then a cellular graph neural network (i.e. CW Network or CWN) performs message passing in a similar manner to normal MPNNs where instead there is communication between cells not just nodes:

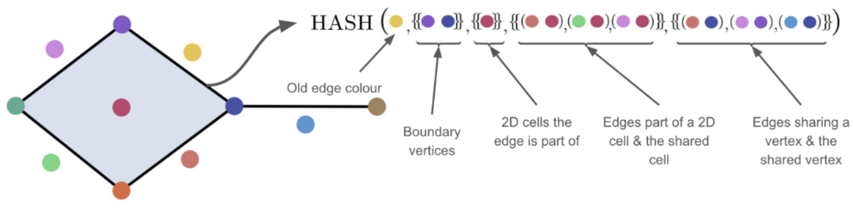
$$
\begin{aligned}
m_B^{t+1}(\sigma) &= \mathrm{AGG}_{\tau \in B(\sigma)}(M_B(h_\sigma^t, h_\tau^t)) \\
m_C^{t+1}(\sigma) &= \mathrm{AGG}_{\tau \in B(\sigma)}(M_C(h_\sigma^t, h_\tau^t)) \\
m_\uparrow^{t+1}(\sigma) &= \mathrm{AGG}_{\tau \in \mathcal{N}_\uparrow(\sigma), \delta \in \mathcal{C}(\sigma,\tau)}(M_\uparrow(h_\sigma^t, h_\tau^t, h_\delta^t)) \\
m_\downarrow^{t+1}(\sigma) &= \mathrm{AGG}_{\tau \in \mathcal{N}_\downarrow(\sigma), \delta \in \mathcal{C}(\sigma,\tau)}(M_\downarrow(h_\sigma^t, h_\tau^t, h_\delta^t)) \\
h_\sigma^{t+1} &= U(h_\sigma^t, m_B^t(\sigma), m_C^t(\sigma), m_\uparrow^t(\sigma), m_\downarrow^t(\sigma))
\end{aligned}
$$

5.2 Cellular Weisfeiler Lehman

CW Networks are as powerful as their corresponding graph isomorphism test called CWL. The coloring procedure for CWL is as follows:

- Given a regular cell complex, initialize all cells σ with the same color.
- Denote c_σ^t as the color of cell σ at iteration t, $c_\sigma^{t+1} = HASH(c_\sigma^t, c_B^t(\sigma), c_C^t(\sigma), c_\downarrow^t(\sigma), c_\uparrow^t(\sigma))$ where $HASH$ is a perfect hash function and $c^t(\sigma), c_C^t(\sigma), c_\downarrow^t(\sigma), c_\uparrow^t(\sigma)$ are the colors of the adjacent cells of σ.
- When the coloring stables, the algorithm stops. Two cell complexes are considered non-isomorphic if the colour histograms are different. Otherwise, the test is inconclusive.

Below is a visualization of the above procedure, taken from the original paper (Bodnar et al., 2021a)



HASH
Old edge colour
Boundary vertices
2D cells the edge is part of
Edges part of a 2D cell & the shared cell
Edges sharing a vertex & the shared vertex

Figure 9: Cellular WL

We can see from the CWL algorithm that it is very similar to WL with the difference being in that it assigns color to each cell type (nodes, edges and faces) not just nodes. k-WL ($k \geq 3$) works on k-tuples of nodes but not edges and faces. The definition of neighborhood in k-WL is also different than the definition of adjacent cells in CWL. The elements in the neighborhood in k-WL simply means that one element of the nodes triplets is different.

Note that CW complex is a generalization of simplicial complex in the sense that simplicial complex is a convex combination of vertices, edges, triangles and its higher order counterpart while CW complex extends to arbitrary structures as long as the inductive step of constructing the CW complex is joining on the boundary. So when the above procedure is applied to simplicial complex, it is the same as SWL.

5.3 Expressiveness of CWL

Since CWL is a generalization of Simplicial WL (SWL) and WL tests, we start by looking into the theorems and definitions for expressiveness of the SWL because it helps us to extend to CWL by generalizing those definitions and theorems.

Bodnar et al. (2021a) introduced the *clique complex lifting map*. It is an injective transformation that maps non-isomorphic graphs to non-isomorphic simplicial complexes. By first preprocessing the graph with this clique transformation before input into SWL, the authors proved the following theorem.

Theorem 2. *SWL with a clique complex lifting is strictly more powerful than WL.*

One example of the above theorem is shown in the graph below. WL cannot distinguish the graphs below but SWL is able to because it contains a 2-simplices (triangles).

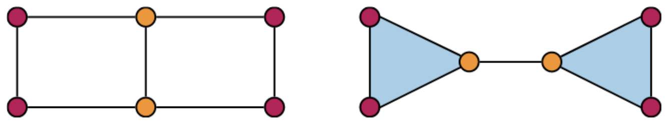


Figure 10: SWL vs WL

We extend the definition of **complex lifting mapping** to **cellular lifting map** below.

Definition 3. *A* ***cellular lifting map*** *is a function $f\colon G \to X$ from the space of graphs G to the space of regular cell complexes X with the property that two graphs G_1, G_2 are isomorphic iff the cell complexes $f(G_1)$, $f(G_2)$ are isomorphic.*

The cellular lifting map is injective. One would wonder what cellular lifting map can help us generalize Theorem 2 to CWL. If a lifting map f preserves the isomorphic relation between the 1-*skeleton* of $f(G)$ and G for any graph G, then CWL with this **skeleton-preserving** lifting transformation f (denoted as CWL(f)) is at least as powerful as WL in distinguishing non-isomorphic graphs. There are other kinds of lifting maps that can make CWL strictly more powerful than WL. Examples of such kind are the lifting maps that attach cells to all the cliques, induced cycles and simple cycles of size at most k, denoted as k-CL, k-IC and k-C respectively. The researcher proposed the following theorem:

Theorem 4. *For all $k \geq 3$, CWL(k-CL), CWL(k-IC) and CWL(k-C) are strictly more powerful than WL.*

Proof
It is sufficient to find one such example. CWL is able to count the number of rings based on ring-based lifting.

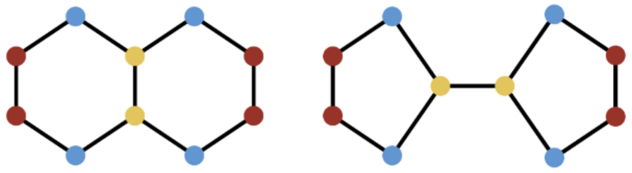


Figure 11: CWL vs WL

■

Theorem 5. *SWL is not less powerful than 3-WL.*

The researchers give an example below which is a pair of SR(16,6,2,2) non-isomorphic graphs. Non-isomorphic graphs that has different clique complexes can be distinguished by SWL. It cannot be distinguished by 3-WL but it can be distinguished by SWL.

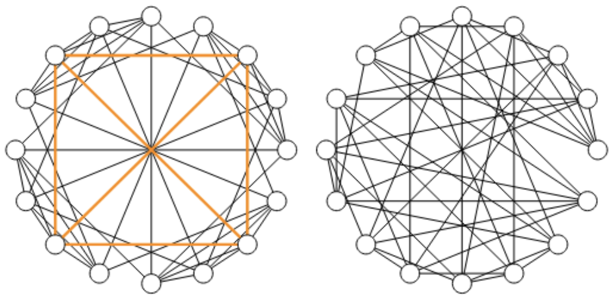


Figure 12: SWL vs 3-WL

What is the counterpart theorem for CWL? It turns out that combinations of *k-CL*, *k-IC* and *k-C* helps us link CWL with 3-WL.

Theorem 6. *There exists a pair of graphs indistinguishable by 3-WL but distinguishable by CWL(k-CL) ($k \geq 4$), CWL(k-IC) ($k \geq 4$) and CWL(k-C) ($k \geq 8$).*

The SR(16,6,2,2) non-isomorphic graphs example above can be reused in proving this theorem. The number of the 4-rings is different for the two graphs, so 3-WL cannot distinguish these two graph but CWL(4-IC) is able to by counting the induced 4-rings thus able to distinguish them.

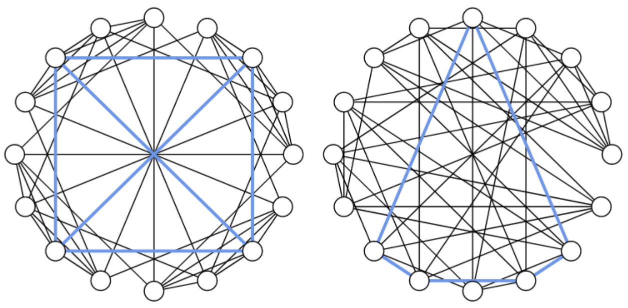


Figure 13: CWL vs 3-WL

6. Conclusion

The new GNN architectures based on algebraic topology are more than just a subset of higher order GNNs. Although it is true that with high enough k, higher order GNNs can represent cellular complexes since they consider all possible combinations of nodes, higher order GNNs are often computationally intractable and break locality precisely because they consider all possible tuples of nodes including non-local subsets. Therefore, cellular graph neural networks are built to preserve locality and are much more efficient provided a suitable lifting transformation. In fact, the computational complexity can be made to scale linearly with respect to the input graph and similar to message passing GNNs. However, this is not always the case. It is also possible for the number of cells in a graph to grow exponentially with the number of nodes.

References

Cristian Bodnar, Fabrizio Frasca, Nina Otter, Yu Guang Wang, Pietro Liò, Guido Montúfar, and Michael M. Bronstein. Weisfeiler and lehman go cellular: CW networks. *CoRR*, abs/2106.12575, 2021a. URL `https://arxiv.org/abs/2106.12575`.

Cristian Bodnar, Fabrizio Frasca, Yuguang Wang, Nina Otter, Guido F. Montúfar, Pietro Lió, and Michael M. Bronstein. Weisfeiler and lehman go topological: Message passing simplicial networks. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 1026–1037. PMLR, 2021b. URL `http://proceedings.mlr.press/v139/bodnar21a.html`.

Giorgos Bouritsas, Fabrizio Frasca, Stefanos Zafeiriou, and Michael M. Bronstein. Improving graph neural network expressivity via subgraph isomorphism counting. *CoRR*, abs/2006.09252, 2020. URL `https://arxiv.org/abs/2006.09252`.

Michael M. Bronstein, Joan Bruna, Taco Cohen, and Petar Velickovic. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. 2021.

Zhengdao Chen, Xiang Li, and Joan Bruna. Supervised community detection with line graph neural networks, 2017. URL `https://arxiv.org/abs/1705.08415`.

Zhengdao Chen, Soledad Villar, Lei Chen, and Joan Bruna. On the equivalence between graph isomorphism testing and function approximation with gnns. *Advances in neural information processing systems*, 32, 2019.

Zhengdao Chen, Lei Chen, Soledad Villar, and Joan Bruna. Can graph neural networks count substructures? In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 10383–10395. Curran Associates, Inc., 2020.

Leonardo Cotta, Christopher Morris, and Bruno Ribeiro. Reconstruction for powerful graph representations. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021.

Alex Fout, Jonathon Byrd, Basir Shariat, and Asa Ben-Hur. Protein interface prediction using graph convolutional networks. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 6530–6539, 2017. URL `https://proceedings.neurips.cc/paper/2017/hash/f507783927f2ec2737ba40afbd17efb5-Abstract.html`.

Vikas K. Garg, Stefanie Jegelka, and Tommi Jaakkola. Generalization and representational limits of graph neural networks, 2020. URL `https://arxiv.org/abs/2002.06157`.

Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. *CoRR*, abs/1704.01212, 2017. URL `http://arxiv.org/abs/1704.01212`.

William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. *CoRR*, abs/1706.02216, 2017. URL `http://arxiv.org/abs/1706.02216`.

Nicolas Keriven and Gabriel Peyré. Universal invariant and equivariant graph neural networks. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 7090–7099, 2019. URL `https://proceedings.neurips.cc/paper/2019/hash/ea9268cb43f55d1d12380fb6ea5bf572-Abstract.html`.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *CoRR*, abs/1609.02907, 2016. URL `http://arxiv.org/abs/1609.02907`.

Adrien Leman. The reduction of a graph to canonical form and the algebra which appears therein. 2018.

Andreas Loukas. What graph neural networks cannot learn: depth vs width. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL `https://openreview.net/forum?id=B1l2bp4YwS`.

Haggai Maron, Ethan Fetaya, Nimrod Segol, and Yaron Lipman. On the universality of invariant networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 4363–4371. PMLR, 2019. URL `http://proceedings.mlr.press/v97/maron19a.html`.

Christopher Morris, Martin Ritzert, Matthias Fey, William L. Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 4602–4609. AAAI Press, 2019. doi: 10.1609/aaai.v33i01.33014602.

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

Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter W. Battaglia. Learning to simulate complex physics with graph networks. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 8459–8468. PMLR, 2020. URL `http://proceedings.mlr.press/v119/sanchez-gonzalez20a.html`.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. *arXiv e-prints*, art. arXiv:1710.10903, October 2017.

Gang Wang, Ziyi Guo, Xiang Li, Dawei Yin, and Shuai Ma. Scenerec: Scene-based graph neural networks for recommender systems. In Yannis Velegrakis, Demetris Zeinalipour-Yazti, Panos K. Chrysanthis, and Francesco Guerra, editors, *Proceedings of the 24th International Conference on Extending Database Technology, EDBT 2021, Nicosia, Cyprus, March 23 - 26, 2021*, pages 397–402. OpenProceedings.org, 2021. doi: 10.5441/002/edbt.2021.41. URL `https://doi.org/10.5441/002/edbt.2021.41`.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.