$\mathcal{N}$-WL: A NEW HIERARCHY OF EXPRESSIVITY FOR GRAPH NEURAL NETWORKS

Qing Wang, Dillon Chen, Asiri Wijesinghe, Shouheng Li, Muhammad Farhan
School of Computing, Australian National University
{qing.wang,dillon.chen2,asiri.wijesinghe}@anu.edu.au
{shouheng.li,muhammad.farhan}@anu.edu.au

ABSTRACT

The expressive power of Graph Neural Networks (GNNs) is fundamental for understanding their capabilities and limitations, i.e., what graph properties can or cannot be learnt by a GNN. Since standard GNNs have been characterised to be upper-bounded by the Weisfeiler-Lehman (1-WL) algorithm, recent attempts concentrated on developing more expressive GNNs in terms of the k-WL hierarchy, a well-established framework for graph isormorphism tests. In this work we show that, contrary to the widely accepted view, the k-WL hierarchy is not well-suited for measuring expressive GNNs. This is due to limitations that are inherent to high-dimensional WL algorithms such as the lack of a natural interpretation and high computational costs, which makes it difficult to draw any firm conclusions about the expressive power of GNNs beyond 1-WL. Thus, we propose a novel hierarchy of graph isomorphism tests, namely *Neighbourhood WL* ($\mathcal{N}$-WL), and also establish a new theorem on the equivalence of expressivity between induced connected subgraphs and induced subgraphs within this hierarchy. Further, we design a GNN model upon $\mathcal{N}$-WL, *Graph Neighbourhood Neural Network* (G3N), and empirically verify its expressive power on synthetic and real-world benchmarks.

1 INTRODUCTION

Graph-theoretic algorithms are a powerful source of inspiration for Graph Neural Networks (GNNs). The most known is that the expressive power of standard GNNs is upper-bounded by the Weisfeiler-Lehman (1-WL) algorithm (Weisfeiler & Leman, 1968; Xu et al., 2019; Morris et al., 2019). In pursuit of more expressive GNNs, various attempts have been made to leverage existing results in graph theory such as high-dimensional WL algorithms (Azizian & Lelarge, 2021; Maron et al., 2019a; Morris et al., 2020b), substructure counting (Bouritsas et al., 2022; Barceló et al., 2021), and individualisation (Dupty et al., 2022). The expressivity of these GNNs is measured in terms of the k-WL hierarchy, a well-established framework for graph isomorphism testing (Grohe, 2017).

However, the k-WL hierarchy exhibits several theoretical and practical limitations as a measure of expressivity for GNNs. Theoretically, it is a highly non-trivial problem to tell if and when k-WL algorithms can distinguish two particular graphs (Kiefer, 2020). Deciding *which graph properties* are important for distinguishing graphs is even much harder, if not impossible. A complete description of all subgraph patterns whose counts and occurrence are k-WL invariant is only available for $k = 1$ (Arvind et al., 2020). Even bearing high computational costs, the power of k-WL algorithms in recognising graph properties seems still limited and some negative results are known, e.g., 3-WL cannot identify any k-cliques with $k > 3$ (Fürer, 2017). These issues hamper the practical applicability of high-dimensional WL algorithms for solving real-world tasks on graph-structured data (Chen et al., 2020; Garg et al., 2020). A question that arises from this is - *Whether the k-WL hierarchy is a good yardstick for expressivity of GNNs?*

In the search for an answer to this question, we observe several disparities between (standard) GNNs and the k-WL hierarchy. First, GNNs encode structural information into nodes as an efficient and practical way for graph learning. This is however against the spirit of the k-WL hierarchy which increases expressive power by going up to higher order objects, i.e., k-tuples, rather than just nodes (Cai et al., 1992; Grohe, 2017). Second, GNNs are built upon a natural notion of local

neighbourhood, i.e., within a certain distance to a node. In contrast, the k-WL hierarchy defines the neighbourhood of a k-tuple based on "adjacency". This notion of adjacency involves the enumeration of all nodes of a graph in each dimension, which is not local and raises concerns about computational efficiency (Morris et al., 2020b). Last but not least, GNNs learn node representations by aggregating the information from its neighbouring nodes, assuming "*birds of a feather flock together*" from real-world perception (Zhu et al., 2020; McPherson et al., 2001). The k-WL hierarchy updates the representation of a k-tuple by aggregating the information from its adjacent neighbours, which does not have a natural interpretation and thus makes it difficult to understand its real-world implications.

In light of these observations, we explore a hierarchy of expressivity that is grounded on a new class of graph isomorphism algorithms, called *Neighbourhood WL* ($\mathscr{N}$-WL) algorithms. This hierarchy overcomes the aforementioned limits of the k-WL hierarchy. More importantly, it enables a new paradigm for designing expressive GNNs while still remaining intuitive and computational efficient. We integrate the following novel insights into the algorithmic design of this hierarchy: (1) Instead of imposing a rigid condition that both objects and its neighbours use the same structure (e.g., k-tuple and its variants), why can we not separate them by colouring nodes in a lower-dimensional space based on information from induced subgraphs in a high-dimensional space? (2) Can we build a hierarchy of expressivity for GNNs upon a natural choice about neighbourhood, i.e., d-hop neighbourhood? On one hand, this ensures the locality of neighbourhood and thus brings in computational efficiency; on the other hand, it allows high-dimensional neighbours to capture intricate graph properties into node representations for distinguishing graphs. (3) Unlike the k-WL hierarchy which increases expressivity only through one dimension k, the hierarchy in our work enables two independent ways of controlling expressive power: the size t of induced subgraphs and the size d of neighbourhoods, i.e., enumerating all subgraphs of order t within a d-hop neighbourhood. This helps strike a balance between computational complexity and expressivity of algorithms, which is often highly sought by real-world applications.

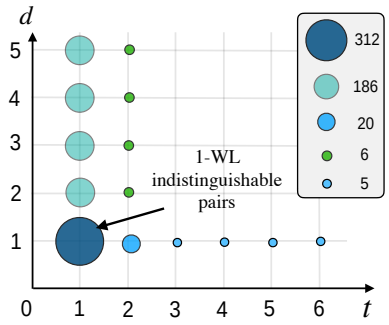


Figure 1: Indistinguishable pairs of simple graphs of eight vertices by 1-WL, which are distinguishable by $\mathscr{N}$-WL under different d and t values.

Figure 1 shows pairs of simple graphs of eight vertices that are indistinguishable by 1-WL but can be distinguished by our proposed hierarchy $\mathscr{N}$-WL under different t and d parameters. By the k-WL hierarchy, we only know that 312 pairs of simple graphs lie between 1-WL and 3-WL as none of them can be distinguished by 1-WL but all of them can be distinguished by 3-WL. Rather than "None" or "All", our $\mathscr{N}$-WL hierarchy can distinguish these graphs in a more refined way under varied t and d values, i.e., each point (t, d) in Figure 1 indicates the number of pairs of simple graphs that remain indistinguishable under these parameters, and all pairs are distinguishable when t 3 and d 2. Further details and example graphs are provided in Appendix A.

With a hierarchy of expressivity, it is natural to ask whether the hierarchy is strict. We thus further explore whether the $\mathscr{N}$-WL hierarchy is strictly more expressive when considering induced subgraphs or neighbourhoods of larger sizes. To show the strictness, we construct counterexample graphs such that, for any fixed d 2 N and every t 2 N, there exist non-isomorphic graphs which $\mathscr{N}$-WL with (t, d) fails to distinguish but can be distinguished by $\mathscr{N}$-WL with $(t{+}1, d)$; on the other hand, for any fixed t 2 N and every d 2 N, there also exist non-isomorphic graphs which $\mathscr{N}$-WL with (t, d) fails to distinguish but can be distinguished by $\mathscr{N}$-WL with $(t, d{+}1)$. Not surprisingly, constructing such counterexample graphs turns out to be difficult, due to the intricate interaction between t and d as well as the combinatorial nature of graph structure. We present such a construction which can produce families of non-isomorphic graph pairs with $O(t)$ or $O(d)$ vertices.

To understand how graph connectivity may affect the expressivity of $\mathscr{N}$-WL, we go on to examine the relation between induced subgraphs and their connectivity. Inspired by the Algebra of Subgraphs (Kocay, 1982), we discover a previously unknown connection between induced subgraphs of size t, for any t 2 N, and induced *connected* subgraphs whose sizes are less than or equal to t. This surprisingly leads to the finding that these two families of subgraphs have equivalent expressive power for distinguishing graphs. Hence, when graphs are sparse, instead of considering all induced subgraphs, we may consider only induced connected subgraphs, improving efficiency considerably.

Table 1: Comparison of k-WL and their variants δ-k-LWL (Morris et al., 2020b), (k, s)-LWL (Morris et al., 2022), and (k, c)( )-SETWL (Zhao et al., 2022a) with our algorithms $\mathscr{N}(t, d)$-WL and $\mathscr{N}^{\mathsf{c}}(t, d)$-WL, where #*Coloured objects* and #*Neighbour objects* refer to the number of coloured objects in a graph and the number of neighbour objects for each coloured object, respectively; Δ*Coloured objects* and Δ*Neighbour objects* refer to the type of coloured objects and the type of neighbour objects, respectively; n is the number of nodes and a is the average node degree in a graph; a^{d} is the average number of nodes in the d-hop neighbourhood of a node. Note that a^{d} n for graphs whose diameters are considerably greater than d.

	k-WL	δ-k-LWL	(k, s)-LWL	(k, c)( )-SETWL	$\mathscr{N}(t, d)$-WL	$\mathscr{N}^{\mathsf{c}}(t, d)$-WL
#Coloured objects	n^{k}	n^{k}	subset(n^{k}, s)	subset($\sum_{q=1}^{k} \binom{n}{q}, c$)	n	n
#Neighbour objects	n k	a k	a k	n q	$\binom{a^{d}}{t}$	subset($\sum_{q=1}^{t} \binom{a^{d}}{q}, 1$)
ΔColoured objects	k-tuples	k-tuples	k-tuples	k-sets	nodes	nodes
ΔNeighbour objects	k-tuples	k-tuples	k-tuples	k-sets	t-sets	t-sets
Sparsity -awareness	✗	✓	✓	✓	✗	✓

Contributions. The main contributions of this work are summarised below:

- We introduce a new class of graph isomorphism algorithms, *Neighbourhood WL* ($\mathscr{N}$-WL), which exhibit a natural and strict hierarchy of expressivity for measuring GNNs in their ability to distinguish graphs (Theorem 3.1, Theorem 3.2, and Theorem 3.3).
- We establish a new theorem on the equivalence of expressivity between induced connected subgraphs and induced subgraphs in general within the hierarchy of $\mathscr{N}$-WL, which enables us to further leverage graph sparsity for improving efficiency (Theorem 3.7).
- We explore how the hierarchy of $\mathscr{N}$-WL relates to the k-WL hierarchy and establish their connections (Theorem 3.8).
- We propose a new GNN model architecture, *Graph Neighbourhood Neural Network* (G3N), which instantiates the ideas of the $\mathscr{N}$-WL algorithms for graph learning (Theorem 4.1).

2 Related Work

Since the advent of Graph Neural Networks (GNNs), a central theme is to understand the expressivity of GNNs. It has been revealed that standard GNNs are at most expressive as the Weisfeiler-Leman (WL) algorithm (Weisfeiler & Leman, 1968; Xu et al., 2019; Morris et al., 2019). Ever since, the WL algorithm has played a crucial role in the theoretical studies on GNNs.

The k-WL hierarchy generalises the WL algorithm (1-WL) to classify k-tuples of vertices (Grohe, 2017; Babai & Kucera, 1979). It is known that k-WL is strictly more powerful than $(k$ $1)$-WL for any k 3. The Cai-Fürer-Immerman (CFI) construction can produce a family of non-isomorphic graph pairs as counterexamples which are indistinguishable by $(k$ $1)$-WL but can be distinguished by k-WL (Cai et al., 1992). Since k-WL is computationally expensive for k 3, k-WL is mostly used as a theoretical tool in graph isomorphism testing and not practically useful.

Various attempts have been made in recent years to develop more expressive GNNs - see a survey by Sato (2020). Until now, the expressivity of GNNs is typically measured in terms of 1-WL. However, the expressivity gap between $k = 1$ and $k = 3$ for k-WL is often too large for applications in practice. On one hand, 1-WL is not expressive enough since it cannot count some simple structures such as cycles or triangles; on the other hand, many applications may not require strong 3-WL power. There is a lack of measure for expressivity of models that lies in between 1-WL and 3-WL, as well as beyond - higher-dimensional WL - in a more refined and natural way.

In this work, we depart from the k-WL hierarchy by proposing a new hierarchy that builds on high-order subgraphs within a neighbourhood aggregation scheme to characterise the expressivity of GNNs. Table 1 shows a comparison between k-WL and their variants and our hierarchy. A detailed discussion on this comparison and other related works are provided in Appendix B.

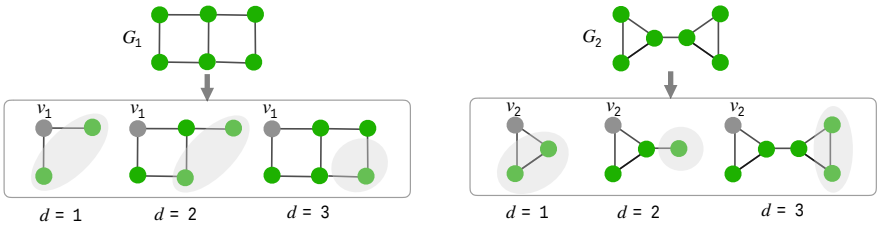


Figure 2: Two graphs distinguishable by $\mathcal{N}(2,1)$-WL and $\mathcal{N}(2,2)$-WL but not $\mathcal{N}(2,3)$-WL.

3 Neighbourhood WL Algorithms

Let $G = (V, E)$ be a simple graph with a set V of nodes and a set E of edges and $|V| = n$. Given a node $u \in V$, its d-hop neighbourhood is defined by $N_d(u) = \{v \in V \mid \rho(u, v) \leq d\}$ where $\rho(\cdot)$ denotes the shortest-path distance between two nodes and $u \in N_d(u)$. The *order* of a (sub)graph is the number of its nodes. We use $G_1 \subseteq G_2$ to denote that G_1 is a subgraph of G_2, and $G_1 \simeq G_2$ that G_1 and G_2 are *isomorphic*, i.e., there exists a bijection between their nodes which induces a correspondence between their edges. Each equivalence class under $\simeq$ is called an *isomorphism type*. All the proofs for lemmas and theorems in this section are provided in Appendix C.

3.1 A Hierarchy of Expressivity

We begin with a simple yet weak hierarchy. This hierarchy shows that expressivity increases with the order of induced subgraphs in a neighbourhood; but counterintuitively, expressivity does not necessarily increase with the size of a neighbourhood.

Weak hierarchy. A *node colouring* in G is a function $\varsigma : V \to \mathbb{N}$ which is refined in iterations, i.e. $\varsigma^l(u)$ for $l = 0, 1, \ldots, m$. Initially, each node u is assigned with some colour $\varsigma^0(u)$. Then, the colour $\varsigma^{l+1}(u)$ is iteratively refined based on the its own colour $\varsigma^l(u)$ and the colours of induced subgraphs in its neighbourhood at the l-th iteration. The colours of induced subgraphs are also defined iteratively, according to the colours of their nodes in the same iteration and connectivity of the nodes. Let $\mathbb{S}_G$ denote the set of all induced subgraphs of G and $f_{iso} : \mathbb{S}_G \to \mathbb{N}$ be a permutation invariant function that encodes the isomorphism types of induced subgraphs. A *subgraph colouring* in G at the l-th iteration is a function $\zeta^l : \mathbb{S}_G \to \mathbb{N}$ such that $\zeta^l(S_1) = \zeta^l(S_2)$ iff $f_{iso}(S_1) = f_{iso}(S_2)$.

Given the set $\mathbb{S}_{(u;t;d)}$ of all induced subgraphs of order t within the d-hop neighbourhood of a node u, $\xi^l_{(u;i)} = \{\!\{\zeta^l(S) \mid S \in \mathbb{S}_{(u;t;d)} \text{ and } f_{iso}(S) = i\}\!\}$, and $I_t = \{f_{iso}(S) | S \in \mathbb{S}_{(u;t;d)}\}$, we have

$$\varsigma^{l+1}(u) = \text{HASH}(\varsigma^l(u), \{\xi^l_{(u;i)}\}_{i \in I_t}). \tag{1}$$

Here, HASH( ) is an injective hash function.

We use $\mathcal{N}(t,d)$-WL to denote the above algorithm with two parameters: t for the order of induced subgraphs and d for the hop of neighbourhood. The following theorem can be proven.

Theorem 3.1. *For any fixed $d \in \mathbb{N}$, $\mathcal{N}(t+1,d)$-WL is strictly more expressive than $\mathcal{N}(t,d)$-WL in distinguishing non-isomorphic graphs, where $t \geq 1$.*

However, $\mathcal{N}(t,d+1)$-WL is not necessarily more expressive than $\mathcal{N}(t,d)$-WL in distinguishing non-isomorphic graphs. Figure 2 illustrates the problem. We observe that the source of the problem is that subgraphs are treated in multisets in which their colours can be distinguished only by their isomorphism types. Thus, aggregating subgraphs from a larger neighbourhood may lose information about subgraphs in the previous smaller neighbourhood.

Strong hierarchy. To alleviate the above problem, we propose to consider not only isomorphism types but also positional types of subgraphs. This enables us to capture relative importance of structures, even of the same isomorphism type, in a neighbourhood as its receptive field increases.

We define a permutation invariant function $f_{pos} : \mathbb{S}_G \to \mathbb{N}$ that encodes the positional types of induced subgraphs in G to satisfy the following condition for any $t \in \mathbb{N}$ and any $d \in \mathbb{N}$:

$$\forall S_i \in \mathbb{S}_{(u;t;d)} \forall S_j \in (\mathbb{S}_{(u;t;d+1)} \setminus \mathbb{S}_{(u;t;d)}) \; (f_{pos}(S_i) \neq f_{pos}(S_j)). \tag{2}$$

We also reformulate the subgraph colouring function as $\zeta^l : S_G \; ! \; \mathbb{N}$ such that $\zeta^l(S_1) = \zeta^l(S_2)$ iff $f_{iso}(S_1) = f_{iso}(S_2)$ and $f_{pos}(S_1) = f_{pos}(S_2)$. Let $\xi^l_{(u;i;j)} = \{\!\{\zeta^l(S) | S \; 2 \; S_{(u;t;d)}, f_{iso}(S)=i,$ and $f_{pos}(S)=j\}\!\}$ and $J_d = \{f_{pos}(S) | S \; 2 \; S_{(u;t;d)}\}$. Then the node colouring in Equation 1 is redefined to integrate the positional types of induced subgraphs as

$$\zeta^{l+1}(u) = \text{HASH}(\zeta^l(u), \{\!\{\xi^l_{(u;i;j)} | i 2 I_t; j 2 J_d\}\!\}). \tag{3}$$

We denote this algorithm with node colouring defined in Equation 3 as $\mathscr{N}(t, d)$-WL, to distinguish it from $\mathscr{N} \; (t, d)$-WL. The following theorem states that the expressivity of $\mathscr{N}(t, d)$-WL increases when considering higher-order subgraphs or larger neighbourhoods.

Theorem 3.2. *For any fixed d 2 $\mathbb{N}$, $\mathscr{N}(t+1, d)$-WL is strictly more expressive than $\mathscr{N}(t, d)$-WL in distinguishing non-isomorphic graphs, where t 1.*

Theorem 3.3. *For any fixed t 2 $\mathbb{N}$, $\mathscr{N}(t, d+1)$-WL is strictly more expressive than $\mathscr{N}(t, d)$-WL in distinguishing non-isomorphic graphs, where d 1.*

Strictness of hierarchies. To prove the strictness of the hierarchies in Theorem 3.1 and Theorem 3.2, we need to construct a family of non-isomorphic graph pairs such that, for any fixed d 2 $\mathbb{N}$, there exist at least two non-isomorphic graphs that are distinguishable by the colours of their nodes that are iteratively refined through counting all $t+1$-order induced subgraphs of different isomorphism and positional types in d-hop neighbourhoods of their nodes, but indistinguishable if considering all t-order induced subgraphs instead. Conversely, Theorem 3.3 requires to construct a family of non-isomorphic graph pairs such that, for any fixed t 2 $\mathbb{N}$, there exist at least two non-isomorphic graphs which can be distinguished by the colours of their nodes through counting all t-order induced subgraphs of different isomorphism and positional types in $d+1$-hop neighbourhoods of their nodes, but cannot be distinguished if considering d-hop neighbourhoods of their nodes instead. We present these constructions in Appendix C with illustrations in Figure 6, Figure 7, and Figure 8.

Upper bound expressivity. With increasing t and d values, the upper bound expressivity of $\mathscr{N}(t, d)$-WL is reached when both t and d equal to the diameter of a graph. In such cases, whether $\mathscr{N}(t, d)$-WL can distinguish two non-isormorphic graphs is reduced to the classical graph isomorphism (GI) problem (Grohe & Schweitzer, 2020). This is because, given a graph G, each d-hop neighbourhood subgraph is the same as the graph G when d $dia(G)$; induced subgraphs of t vertices are also the same as the graph G when t $dia(G)$, where $dia(G)$ refers to the diameter of G.

Note that as shown by Cai, Fürer, and Immerman in their seminal paper (Cai et al., 1992), the k-WL algorithm is incomplete in the sense that, for any k 2 $\mathbb{N}$, there exists a pair of non-isomorphic graphs that cannot be distinguished by the k-WL algorithm. Similarly, in our work, the construction presented in Appendix C shows that, for any t 2 $\mathbb{N}$ and d 2 $\mathbb{N}$, there exists a pair of non-isomorphic graphs that cannot be distinguished by $\mathscr{N}(t, d)$-WL.

3.2 Connected-Hereditary Subgraphs

The locality of neighbourhood restricts $\mathscr{N}(t, d)$-WL to considering only induced subgraphs that are "close" to a node. This nice property helps reduce the computational complexity, compared with k-WL. Nonetheless, enumerating all induced subgraphs in a neighbourhood can still be inefficient if their order is high. Real-world applications often focus on analysing connectivity of nodes in a graph. For example, highly interacting proteins are more likely to form function modules (Alokshiya et al., 2019). Thus, one natural thought to further reduce complexity is to consider only connected subgraphs. Following this direction, we identify connected-hereditary subgraphs. Surprisingly, it turns out that all connected induced subgraphs with orders no greater than t can capture no less information than all induced subgraphs of order t. This finding leads us to improving the $\mathscr{N}$-WL algorithm by taking graph connectivity into consideration.

Let I^c_t I_t be the set of isomorphism types in I_t for connected subgraphs. Instead of considering all subgraphs in the neighbourhood as in Equation 3, we now restrict node colouring to connected subgraphs but allow the orders of connected subgraphs to range over $[1, t]$:

$$\zeta^{l+1}(u) = \text{HASH}(\zeta^l(u), \bigcup_{k 2 [1;t]} \{\!\{\xi^l_{(u;i;j)} | i 2 I^c_k; j 2 J_d\}\!\}). \tag{4}$$

For this algorithm with node colouring based only on connected subgraphs, we denote it as $\mathcal{N}^c(t,d)$-WL. Although Equation 4 might look rather restricted, we show that if two graphs G_1 and G_2 are distinguishable by $\mathcal{N}(t,d)$-WL, they are also distinguishable by $\mathcal{N}^c(t,d)$-WL. This is due to a property of subgraphs occurring in $\mathcal{N}^c(t,d)$-WL, which we elaborate below.

Let $S_c^{\ t}$ and $S^{=t}$ be the set of all subgraphs in the d-hop neighbourhood of a node u being considered by $\mathcal{N}^c(t,d)$-WL and $\mathcal{N}(t,d)$-WL, respectively, and $S_c^{=t}$ the set of all subgraphs of order t in $S_c^{\ t}$.

Definition 3.4. Let S S_G. Then S is said to be *connected-hereditary* if every S 2 S is a connected induced subgraph and S 2 S implies that every connected induced subgraph of S is also in S.

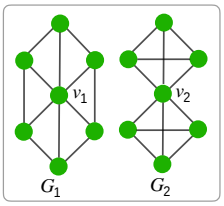
$S_c^{\ t}$ contains all connected induced subgraphs of k vertices with 1 k t in the d-hop neighbourhood of a node. We have the following lemma.

Lemma 3.5. *$S_c^{\ t}$ is connected-hereditary.*

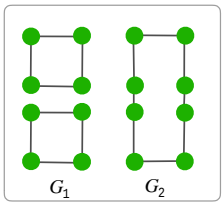
Let S_1 S_2 denote the union of two node-disjoint subgraphs, i.e., $V(S_1) \setminus V(S_2) = ;$, and $\mu(S)$ the number of connected components in S. The lemma below states that any non-connected subgraph in ($S^{=t}$ $S_c^{=t}$) is a union of a number of smaller connected subgraphs in $S_c^{\ t}$ whose nodes are disjoint to each other.

Lemma 3.6. *For each induced subgraph S satisfying S 2 $S^{=t}$ but S 2 $S_c^{=t}$, there exists a set $\{S_1, S_2, \ldots, S_q\}$ $S_c^{\ t}$ such that $S = S_1$ S_2 ... S_q, where $\mu(S) = q$.*

The intuition is that smaller connected induced subgraphs can actually be used as pieces to reconstruct all the disconnected order-t induced subgraphs, without losing expressivity. Hence, based on these lemmas, we can prove the following theorem that the $\mathcal{N}^c(t,d)$-WL algorithm is as powerful as the $\mathcal{N}(t,d)$-WL algorithm. In fact, both algorithms have the same expressivity.



(a) Neighbourhoods of Rook's graphs and the Shrikhande graph.



(b) An 8-cycle and a pair of disconnected 4-cycles.

Figure 3: Pairs of non-isomorphic (sub)graphs.

Theorem 3.7. *For any t 2 N and d 2 N, $\mathcal{N}^c(t,d)$-WL and $\mathcal{N}(t,d)$-WL have the same expressivity in distinguishing non-isomorphic graphs.*

It is worth noting that, by further restricting connected subgraphs to specific isomorphism types, we may obtain different subclasses of algorithms. For example, if limiting I_c to cycles, cliques, and paths, $\mathcal{N}^c(t,d)$-WL is reduced to the neighbourhood WL algorithms $\mathcal{N}^{cycle}(t,d)$-WL, $\mathcal{N}^{clique}(t,d)$-WL, and $\mathcal{N}^{path}(t,d)$-WL, respectively, which encode structures of specific kinds from the node neighbourhoods into node colours.

3.3 Connections to k-WL Hierarchy

One may arise a question with regard to how $\mathcal{N}$-WL relates to the k-WL hierarchy. Since 1-WL corresponds to node labelling based on a multiset of the colours of its neighbours within a 1-hop neighbourhood, it is straightforward to establish the following connection.

Theorem 3.8. *$\mathcal{N}(1,1)$-WL is equivalent to 1-WL in distinguishing non-isomorphic graphs.*

By Theorems 3.2 and 3.3, we know that the expressivity of $\mathcal{N}(t,d)$ increases when increasing either the order of subgraphs or the hop of a neighbourhood. This leads to the proposition below.

Proposition 3.9. *Both $\mathcal{N}(2,1)$-WL and $\mathcal{N}(1,2)$-WL are strictly more powerful than 1-WL in distinguishing non-isomorphic graphs.*

However, in general, the $\mathcal{N}$-WL hierarchy is incomparable with k-WL, as we notice in the nontrivial case that $\mathcal{N}(3,1)$ is incomparable to 3-WL. A canonical example of two non-isomorphic, strongly regular graphs which 3-WL cannot distinguish are Rook's graph and the Shrikhande graph. Despite being strongly regular, $\mathcal{N}(3,1)$-WL can distinguish them by distinguishing 3-order subgraphs in 1-hop node neighbourhoods as in Figure 3(a). Nevertheless, $\mathcal{N}(3,1)$-WL fails to detect 4-cycles as its receptive field is too small in contrast to 3-WL. This is seen where the former is unable to distinguish an 8-cycle from two disconnected 4-cycles in Figure 3(b) whereas 3-WL is able to distinguish them.

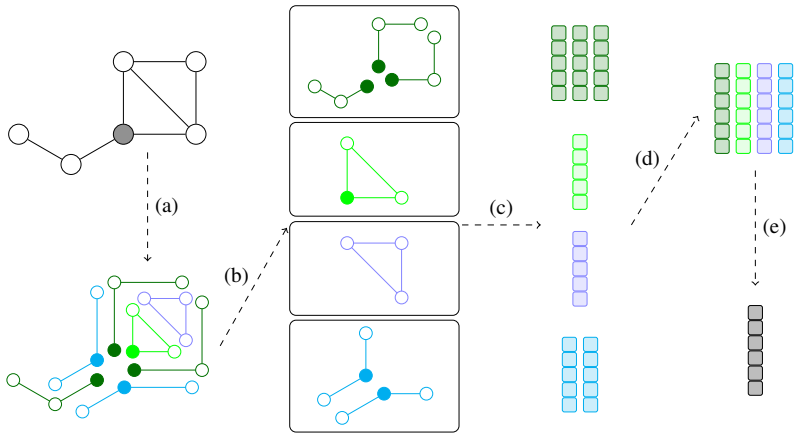


Figure 4: An overview of a G3N layer: (a) t-order subgraphs are extracted from a node's d-hop neighbourhood. (b) The subgraphs are grouped by their positional and isomorphic types. (c) The subgraphs are embedded by a pooling function POOL. (d) The subgraph embeddings are aggregated in their own topological groups by AGG^T. (e) The resulting embedding vectors are further aggregated and combined with AGG^N and COMBINE to form an updated node embedding.

4 GRAPH NEIGHBOURHOOD NEURAL NETWORK

Motivated by the $\mathscr{N}$-WL algorithm, we design *Graph Neighbourhood Neural Network* (G3N) which is able to leverage and learn structural information from neighbourhoods.

Model design. Given a graph $G = (V, E)$, each node $u \in V$ is associated with an f-dimensional feature vector $x_u \in \mathbb{R}^f$ and $h_u^{(0)} = x_u$. Let $S_u^{(l)}(i, j)$ denote the set of all t-order subgraphs within the d-hop neighbourhood of a node u with the isomorphism type i and positional type j at the l-th layer. Then at the $(l+1)$-th layer the node embedding $h_u^{(l+1)}$ of a node u is defined by

$$h_u^{(l+1)} = \text{COMBINE}\left(h_u^{(l)}, \text{AGG}^N_{(i;j)\in I_t \times J_d}\left(\text{AGG}^T_{S\in S_u^{(l)}(i;j)}(\text{POOL}(S))\right)\right). \tag{5}$$

POOL( ) extracts node representations within a subgraph S as a subgraph embedding which can be defined by any graph pooling method. Aggregation proceeds in two steps: $\text{AGG}^T(\,)$ combines subgraph embeddings of the same isomorphism and positional types and $\text{AGG}^N(\,)$ combines the resulting embeddings from all subgraphs in the neighbourhood. COMBINE( ) combines the node embedding of node u at the previous layer with the aggregated embedding of subgraphs. Further details of the G3N model architecture are described in Appendix D.

One can compare Equation 5 to the node colouring of $\mathscr{N}$-WL described in Equation 3 where subgraph pooling corresponds to subgraph colouring, and the aggregation corresponds to the hashing of sets of multisets. We refer to G3N with given t and d by G3N-(t, d).

Expressivity analysis. G3N-(t, d) is at most as expressive as $\mathscr{N}(t, d)$-WL. To match the expressiveness of $\mathscr{N}(t, d)$-WL, one may insert Multi-Layer Perceptrons (MLPs) to approximate injective functions as employed by Xu et al. (2019). However, this comes at higher parameter complexity which may increase expressiveness but could decrease generalisability. The following theorem phrases this formally with the proof available in Appendix C.

Theorem 4.1. *G3N-(t, d) with injective* COMBINE *and* AGG^N *functions, an injective* AGG^T *function w.r.t. multisets of subgraphs with the same isomorphism and positional types, an injective graph readout function, and sufficiently many layers is as powerful as $\mathscr{N}(t, d)$-WL.*

Complexity analysis. Usually, expressivity comes at a cost of computational complexity and this is no exception for G3N. Let a denote the average node degree of a graph. Then ignoring node features embedding dimensions, standard GNNs have the complexity $O(n\ a)$ while G3N has $O(n\ \binom{a^d}{t})$ per layer. Here, t is a very small value, usually less than 6. Note that $a^d \ll n$ is an average size of a local d-hop neighbourhood, different from k-WL which considers all vertices in a graph, i.e., $O(n^k)$.

Table 2: Graph isomorphism tasks on EXP, SR25, graph8c and CSL are evaluated by counting the pairs of graphs which are indistinguishable. Substructure counting tasks are performed on the RandomGraph dataset and evaluated by MSE.

Model	Graph isomorphism				Substructure counting			
	EXP	SR25	graph8c	CSL	Triangle	Tailed Tri.	Star	4-Cycle
MLP	600	105	293K	45	3.13E-1	2.22E-1	**1.0E-4**	1.73E-1
GCN	600	105	4775	45	2.43E-1	1.42E-1	**1.0E-4**	1.14E-1
GAT	600	105	1828	45	2.47E-1	1.44E-1	**1.0E-4**	1.12E-1
GIN	600	105	386	45	2.06E-1	1.18E-1	**1.0E-4**	1.21E-1
PPGN	**0**	105	**0**	**1**	**1.00E-4**	**2.61E-4**	**1.0E-4**	**3.30E-4**
G3N-(2,1)	600	105	5	36	**1.81E-4**	**3.58E-4**	**6.46E-4**	1.03E-1
G3N-(3,1)	600	**0**	5	36	**4.31E-4**	**4.30E-4**	**3.41E-4**	1.14E-1
G3N-(2,2)	**0**	105	6	15	**6.77E-4**	**7.06E-4**	**8.63E-4**	1.61E-2
G3N-(3,2)	**0**	**0**	**0**	10	***1.55E-3***	***1.69E-3***	***4.42E-3***	***7.38E-3***

Table 3: TU datasets evaluated by classification accuracy (%). The first 3 rows are kernel methods and the remaining are GNN models. The first and second best results are highlighted and underlined.

Model	MUTAG	PTC_MR	PROTEINS	NCI1	IMDB-B	IMDB-M
RWK	79.2±2.1	55.9±0.3	59.6±0.1	>3 days	-	-
WL-kernel	90.4±5.7	59.9±4.3	75.0±3.1	**86.0±1.8**	73.8±3.9	50.9±3.8
P-WL	90.5±1.3	64.0±0.8	75.9±0.8	**85.6±0.3**	-	
PATCHY-SAN	**92.6±4.2**	60.0±4.8	75.9±2.8	78.6±1.9	71.0±2.2	45.2±2.8
DCNN	-	-	61.3±1.6	56.6±1.0	49.1±1.4	33.5±1.4
DGCNN	85.8±0.0	58.6±0.0	75.5±0.9	74.4±0.5	70.0±0.9	47.8±0.9
GIN	89.4±5.6	64.6±7.0	**76.2±2.8**	82.7±1.7	**75.1±5.1**	52.3±2.8
PPGN	90.6±8.7	**66.2±6.6**	**77.2±4.7**	83.2±1.1	73.0±5.8	50.5±3.6
G3N-(1,1)	89.9±8.0	63.3±6.3	73.6±5.3	82.3±5.2	73.6±3.5	**51.5±2.9**
G3N-(2,2)	**92.0±4.2**	**65.5±6.3**	74.6±4.1	83.2±1.5	**74.3±3.7**	**52.4±3.4**

Further, we may consider only connected subgraphs when graphs are sparse. Subgraph aggregations can be conducted in parallel across all subgraphs according to their isomorphism and positional types. Computationally, it is much more efficient than expressive GNNs based on k-WL.

5 EXPERIMENTS

We run experiments on synthetic and real-world datasets to empirically validate the theoretical properties of G3N. For synthetic datasets, we focus on two main tasks: the graph isomorphism problem and substructure counting. As for real-world datasets, we focus on graph classification and regression on molecular datasets and social networks. Details about datasets, baselines, and experimental setups, as well as additional experimental results, are available in Appendix E.

5.1 SYNTHETIC DATASETS

Setup. **EXP** (Abboud et al., 2021) consists of 600 pairs of nonisomorphic graphs (G_i, H_i) each encoding a propositional formula which is either satisfiable or not. The graphs are designed to be indistinguishable by 1-WL. **SR25**[1] (Balcilar et al., 2021) contains 15 strongly regular graphs, each with 25 nodes and indistinguishable by 3-WL. **graph8c** (Balcilar et al., 2021) contains all 11,117 connected non-isomorphic 8-node graphs, of which 312 pairs are indistinguishable by 1-WL and all are distinguishable by 3-WL. **CSL** was first introduced by Murphy et al. (2019) and commonly used to test graph expressiveness (Dwivedi et al., 2020). It consists of 10 isomorphism classes of 41 node 4-regular graphs which are almost all distinguishable by 3-WL. For counting graph substructures, we consider **RandomGraph** (Chen et al., 2020). We follow the setup described by Balcilar et al. (2021).

[1]available at `http://users.cecs.anu.edu.au/~bdm/data/graphs.html`

Table 4: Experiments on ZINC and MolHIV, where the performance of the methods marked by * (i.e., DEEP LRP, GSN, and CIN) is based on pre-selected subgraph patterns.

(a) ZINC evaluated by MAE.

Model	ZINC (no edge features)	ZINC (edge features)
GCN	0.459±0.006	0.321±0.009
PPGN	0.407±0.028	-
GIN	0.387±0.015	0.163±0.004
PNA	0.320±0.032	0.188±0.004
DGN	0.219±0.010	0.168±0.003
DEEP LRP*	0.223±0.008	-
GSN*	0.140±0.006	0.115±0.012
CIN*	**0.115±0.003**	**0.079±0.006**
G3N-(2,3)	**0.165±0.018**	**0.128±0.015**

(b) MolHIV evaluated by ROC-AUC.

Model	MolHIV (test)	MolHIV (validation)
GCN	0.7606±0.0097	0.8204±0.0141
GIN	0.7558±0.0140	0.8232±0.0090
GraphSNN	0.7851±0.0170	0.8359±0.0096
PNA	0.7905±0.0132	-
DGN	**0.7970±0.0097**	-
DEEP LRP*	0.7687±0.0180	0.8131±0.0088
GSN*	0.7799±0.0100	**0.8658±0.0084**
CIN*	**0.8094±0.0057**	-
G3N-(2,3)	0.7900±0.0134	**0.8359±0.0061**

Observations. From Table 2 we can see that G3N with $t = 2$ and $d = 1$ alone is already more expressive than 1-WL bounded GNNs, being able to distinguish most graph8c graphs and some CSL graphs. By increasing t to 3, we are able to distinguish all SR25 graphs which the 3-WL bounded PPGN is unable to. Other configurations of d and t also support that $\mathscr{N}$-WL is incomparable with k-WL. Looking at substructure counting tasks, G3N with $t = 2$ are able to trivially detect triangles but not 4-cycles. Setting $t = 3$ and $d = 2$ allows us to detect 4-cycles. Note that G3N with $t = 2$ and $d = 2$ is unable to learn 4-cycles but is able to memorise some configurations, leading to the slightly lower error. We finally observe that for higher d and t, convergence of G3N takes longer due to its higher expressivity and low parameter budget in this setting, resulting in slighty poorer predictions.

5.2 Real-world Datasets

Setup. **ZINC** (Irwin et al., 2012) contains 250K molecules of which 12K are selected for the task of regressing constrained solubility. We follow the experimental setup described in (Dwivedi et al., 2020). **MolHIV** was introduced by Wu et al. (2018) as part of the MoleculeNet benchmark and has been proposed in (Hu et al., 2020) to be graph classification tasks. We follow the train, validation and test split from Hu et al. (2020) and evaluate on the test score corresponding to the best validation score. We further consider the bioinformatics and social network datasets from the TUDataset collection (Morris et al., 2020a). We follow the setup described in (Xu et al., 2019).

Observations. From the results collated in Tables 3 and 4, G3N performs well compared with the other baselines. On ZINC, G3N performs better on learning topological features over PPGN. There exist stronger performing models on molecular datasets such as CIN (Bodnar et al., 2021a) and GSN (Bouritsas et al., 2022); however, these methods explicitly compute cycle information which is known to be correlated with molecular classes and attributes. For example, cycle counts are used in the calculation of the regressed constrained solubility attribute of ZINC datasets (Irwin et al., 2012). Among the baselines that learn topological features without pre-selecting application-dependent subgraph patterns, our G3N performs best on ZINC both with edge features and without edge features.

6 Conclusions

We proposed a new class of graph isomorphism algorithms known as $\mathscr{N}$-WL algorithms. This class of graph isomorphism algorithms exhibits a more refined hierarchy of expressivity for distinguishing non-isomorphic graphs by focusing on approximating graph neighbourhood structures as best as possible. We further explored how to narrow our focus down to only connected subgraphs to improve efficiency while still preserving expressiveness. Motivated by $\mathscr{N}$-WL, we proposed the G3N architecture which is provably more expressive than vanilla message-passing neural networks and is able to leverage graph structure more effectively for prediction tasks.

ACKNOWLEDGMENTS

We are thankful to Professor Brendan Mckay for helpful discussions and feedback. We gratefully acknowledge the support of the Australian Research Council under Discovery Project DP210102273. We also would like to thank anonymous reviewers for their comments and suggestions which helped improve the quality of the paper.

REPRODUCIBILITY STATEMENT

To ensure reproducibility of the theoretical and empirical results included in this work, we have made the following efforts: (1) For theoretical results, the complete proofs for lemmas and theorems are provided in Appendix C; (2) Details of the model architecture and the implementation are included in Appendix D; (3) Experimental details, including datasets, baseline methods, hyperparameter selection, and additional experimental results, are provided in Appendix E; (4) The code is available at `https://github.com/seanli3/G3N`.

REFERENCES

Ralph Abboud, Ismail Ilkan Ceylan, Martin Grohe, and Thomas Lukasiewicz. The surprising power of graph neural networks with random node initialization. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2021.

Sami Abu-El-Haija, Bryan Perozzi, Amol Kapoor, Nazanin Alipourfard, Kristina Lerman, Hrayr Harutyunyan, Greg Ver Steeg, and Aram Galstyan. Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing. In *International Conference on Machine Learning (ICML)*, pp. 21–29, 2019.

Mohammed Alokshiya, Saeed Salem, and Fidaa Abed. A linear delay algorithm for enumerating all connected induced subgraphs. *BMC bioinformatics*, 20(12):1–11, 2019.

Vikraman Arvind, Frank Fuhlbrück, Johannes Köbler, and Oleg Verbitsky. On weisfeiler-leman invariance: Subgraph counts and related graph properties. *Journal of Computer and System Sciences*, 113:42–59, 2020.

James Atwood and Don Towsley. Diffusion-convolutional neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 1993–2001, 2016.

Waïss Azizian and Marc Lelarge. Expressive power of invariant and equivariant graph neural networks. In *International Conference on Learning Representations (ICLR)*, 2021.

László Babai and Ludik Kucera. Canonical labelling of graphs in linear average time. In *Annual Symposium on Foundations of Computer Science (SFCS)*, pp. 39–46, 1979.

Muhammet Balcilar, Pierre Héroux, Benoit Gaüzère, Pascal Vasseur, Sébastien Adam, and Paul Honeine. Breaking the limits of message passing graph neural networks. In *International Conference on Machine Learning (ICML)*, volume 139, pp. 599–608, 2021.

Pablo Barceló, Floris Geerts, Juan Reutter, and Maksimilian Ryschkov. Graph neural networks with local graph parameters. *Neural Information Processing Systems (NeurIPS)*, 34, 2021.

Dominique Beaini, Saro Passaro, Vincent Létourneau, William L. Hamilton, Gabriele Corso, and Pietro Lió. Directional graph networks. In *International Conference on Machine Learning*, volume 139, pp. 748–758, 2021.

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

Cristian Bodnar, Fabrizio Frasca, Nina Otter, Yuguang Wang, Pietro Lio, Guido F Montufar, and Michael Bronstein. Weisfeiler and lehman go cellular: Cw networks. *Advances in Neural Information Processing Systems*, 34:2625–2640, 2021a.

Cristian Bodnar, Fabrizio Frasca, Yuguang Wang, Nina Otter, Guido F Montufar, Pietro Lio, and Michael Bronstein. Weisfeiler and lehman go topological: Message passing simplicial networks. In *International Conference on Machine Learning (ICML)*, pp. 1026–1037, 2021b.

Giorgos Bouritsas, Fabrizio Frasca, Stefanos P Zafeiriou, and Michael Bronstein. Improving graph neural network expressivity via subgraph isomorphism counting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

Robert Brijder, Floris Geerts, Jan Van Den Bussche, and Timmy Weerwag. On the expressive power of query languages for matrices. *ACM Transactions on Database Systems (TODS)*, 44(4):1–31, 2019.

Rémy Brossard, Oriel Frigo, and David Dehaene. Graph convolutions that can finally model local structure. *arXiv preprint arXiv:2011.15069*, 2020.

Jin-Yi Cai, Martin Fürer, and Neil Immerman. An optimal lower bound on the number of variables for graph identification. *Combinatorica*, 12(4):389–410, 1992.

Zhengdao Chen, Lei Chen, Soledad Villar, and Joan Bruna. Can graph neural networks count substructures? *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Gabriele Corso, Luca Cavalleri, Dominique Beaini, Pietro Liò, and Petar Velickovic. Principal neighbourhood aggregation for graph nets. *Advances in Neural Information Processing Systems (NeurIPS)*, 33, 2020.

Mohammed Haroon Dupty, Yanfei Dong, and Wee Sun Lee. Pf-gnn: Differentiable particle filtering based approximation of universal graph representations. In *International Conference on Learning Representations*, 2022.

Vijay Prakash Dwivedi, Chaitanya K Joshi, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking graph neural networks. *arXiv preprint arXiv:2003.00982*, 2020.

Jiarui Feng, Yixin Chen, Fuhai Li, Anindya Sarkar, and Muhan Zhang. How powerful are k-hop message passing graph neural networks. *Advances in Neural Information Processing Systems*, 2022.

Fabrizio Frasca, Beatrice Bevilacqua, Michael M Bronstein, and Haggai Maron. Understanding and extending subgraph gnns by rethinking their symmetries. *arXiv preprint arXiv:2206.11140*, 2022.

Frank Fuhlbrück, Johannes Köbler, Ilia Ponomarenko, and Oleg Verbitsky. The weisfeiler-leman algorithm and recognition of graph properties. *Theoretical Computer Science*, 895:96–114, 2021.

Martin Fürer. On the combinatorial power of the weisfeiler-lehman algorithm. In *International Conference on Algorithms and Complexity*, pp. 260–271. Springer, 2017.

Vikas Garg, Stefanie Jegelka, and Tommi Jaakkola. Generalization and representational limits of graph neural networks. In *International Conference on Machine Learning (ICML)*, pp. 3419–3430, 2020.

Thomas Gärtner, Peter Flach, and Stefan Wrobel. On graph kernels: Hardness results and efficient alternatives. In *Learning theory and kernel machines*, pp. 129–143. 2003.

Floris Geerts and Juan L Reutter. Expressiveness and approximation properties of graph neural networks. In *International Conference on Learning Representations (ICLR)*, 2022.

Martin Grohe. *Descriptive complexity, canonisation, and definable graph structure theory*, volume 47. Cambridge University Press, 2017.

Martin Grohe and Pascal Schweitzer. The graph isomorphism problem. *Communications of the ACM*, 63(11):128–134, 2020.

Max Horn, Edward De Brouwer, Michael Moor, Yves Moreau, Bastian Rieck, and Karsten Borgwardt. Topological graph neural networks. In *International Conference on Learning Representations (ICLR)*, 2021.

Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *Neural Information Processing Systems (NeurIPS)*, 2020.

John J Irwin, Teague Sterling, Michael M Mysinger, Erin S Bolstad, and Ryan G Coleman. Zinc: a free tool to discover chemistry for biology. *Journal of chemical information and modeling*, 52(7): 1757–1768, 2012.

Sandra Kiefer. *Power and limits of the Weisfeiler-Leman algorithm*. PhD thesis, Dissertation, RWTH Aachen University, 2020.

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

William L Kocay. Some new methods in reconstruction theory. In *Combinatorial Mathematics IX*, pp. 89–114. Springer, 1982.

Andreas Loukas. What graph neural networks cannot learn: depth vs width. In *International Conference on Learning Representations (ICLR)*, 2020.

Haggai Maron, Heli Ben-Hamu, Nadav Shamir, and Yaron Lipman. Invariant and equivariant graph networks. In *International Conference on Learning Representations (ICLR)*, 2018.

Haggai Maron, Heli Ben-Hamu, Hadar Serviansky, and Yaron Lipman. Provably powerful graph networks. *Advances in Neural Information Processing Systems (NeurIPS)*, 32, 2019a.

Haggai Maron, Ethan Fetaya, Nimrod Segol, and Yaron Lipman. On the universality of invariant networks. In *International Conference on Machine Learning (ICML)*, pp. 4363–4371, 2019b.

Brendan D McKay. Practical graph isomorphism. *Congressus Numeranitum*, 30:45–87, 1981.

Brendan D Mckay and Adolfo Piperno. Practical graph isomorphism, ii. *Journal of Symbolic Computation*, 60:94–112, 2014.

Miller McPherson, Lynn Smith-Lovin, and James M Cook. Birds of a feather: Homophily in social networks. *Annual review of sociology*, 27(1):415–444, 2001.

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 *AAAI Conference on Artificial Intelligence*, volume 33, pp. 4602–4609, 2019.

Christopher Morris, Nils M Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. Tudataset: A collection of benchmark datasets for learning with graphs. *ICML Graph Representation Learning and Beyond (GRL+) Workhop*, 2020a.

Christopher Morris, Gaurav Rattan, and Petra Mutzel. Weisfeiler and leman go sparse: Towards scalable higher-order graph embeddings. *Neural Information Processing Systems (NeurIPS)*, 33, 2020b.

Christopher Morris, Matthias Fey, and Nils M Kriege. The power of the weisfeiler-leman algorithm for machine learning with graphs. *International Joint Conference on Artificial Intelligence (IJCAI)*, 2021.

Christopher Morris, Gaurav Rattan, Sandra Kiefer, and Siamak Ravanbakhsh. Speqnets: Sparsity-aware permutation-equivariant graph networks. In *ICML*, 2022.

Ryan Murphy, Balasubramaniam Srinivasan, Vinayak Rao, and Bruno Ribeiro. Relational pooling for graph representations. In *International Conference on Machine Learning (ICLR)*, pp. 4663–4673, 2019.

Hoang Nguyen and Takanori Maehara. Graph homomorphism convolution. In *International Conference on Machine Learning*, pp. 7306–7316. PMLR, 2020.

Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. Learning convolutional neural networks for graphs. In *International Conference on Machine Learning (ICML)*, pp. 2014–2023, 2016.

Giannis Nikolentzos, George Dasoulas, and Michalis Vazirgiannis. k-hop graph neural networks. *Neural Networks*, 130:195–205, 2020.

Bastian Rieck, Christian Bock, and Karsten Borgwardt. A persistent weisfeiler-lehman procedure for graph classification. In *International Conference on Machine Learning (ICML)*, pp. 5448–5458, 2019.

Ryoma Sato. A survey on the expressive power of graph neural networks. *arXiv preprint arXiv:2003.04078*, 2020.

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

Nino Shervashidze, Pascal Schweitzer, Erik Jan Van Leeuwen, Kurt Mehlhorn, and Karsten M Borgwardt. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, 12(9), 2011.

Erik Thiede, Wenda Zhou, and Risi Kondor. Autobahn: Automorphism-based graph neural nets. *Advances in Neural Information Processing Systems*, 34, 2021.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *International Conference on Learning Representations (ICLR)*, 2017.

Clément Vignac, Andreas Loukas, and Pascal Frossard. Building powerful and equivariant graph neural networks with structural message-passing. In *Neural Information Processing Systems (NeurIPS)*, 2020.

Guangtao Wang, Rex Ying, Jing Huang, and Jure Leskovec. Multi-hop attention graph neural network. *arXiv preprint arXiv:2009.14332*, 2020.

Boris Weisfeiler and Andrei Leman. The reduction of a graph to canonical form and the algebra which appears therein. *NTI, Series*, 2(9):12–16, 1968.

Asiri Wijesinghe and Qing Wang. A new perspective on" how graph neural networks go beyond weisfeiler-lehman?". In *International Conference on Learning Representations*, 2022.

Zhenqin Wu, Bharath Ramsundar, Evan N Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S Pappu, Karl Leswing, and Vijay Pande. Moleculenet: a benchmark for molecular machine learning. *Chemical science*, 9(2):513–530, 2018.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations (ICLR)*, 2019.

Jiaxuan You, Jonathan Gomes-Selman, Rex Ying, and Jure Leskovec. Identity-aware graph neural networks. In *Conference on Artificial Intelligence (AAAI)*, 2021.

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

Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. An end-to-end deep learning architecture for graph classification. In *AAAI Conference on Artificial Intelligence*, 2018.

Lingxiao Zhao, Louis Härtel, Neil Shah, and Leman Akoglu. A practical, progressively-expressive gnn. *Advances in Neural Information Processing Systems*, 2022a.

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

Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. Beyond homophily in graph neural networks: Current limitations and effective designs. *Advances in Neural Information Processing Systems*, 33:7793–7804, 2020.

CONTENTS (APPENDIX)

A INDISTINGUISHABLE PAIRS - k-WL VS $\mathscr{N}$-WL

Figure 5 presents two pairs of non-isomorphic simple graphs of eight vertices from the 312 pairs of non-isomorphic simple graphs reported in Figure 1.

In terms of the k-WL hierarchy, both pairs of these non-isomorphic simple graphs cannot be distinguished by 1-WL but can be distinguished by 3-WL. Beyond these, there is no further information about why these pairs of non-isomorphic graphs can or cannot be distinguished and what graph properties are important in distinguishing them.

Nonetheless, if looking into these pairs of non-isomorphic graphs in terms of the $\mathscr{N}$-WL hierarchy, we have the following observations:

(1) Although both pairs of graphs are among the 312 pairs of non-isomorphic simple graphs of eight vertices that cannot be distinguished by 1-WL, they look very different, exhibiting different properties.

(2) The reason why G_1^0 and G_2^0 in Figure 5(a) cannot be distinguished by 1-WL is due to the fact that these two graphs have the same number of vertices in their 1-hop neighbourhoods. Indeed, they also have the same number of edges in their 1-hop neighbourhoods. However, they do have different numbers of triangles in their 1-hop neighbourhoods, which explains why they can only be distinguished by $\mathscr{N}(t, d)$-WL if t 3.

(3) The reason why G_1 and G_2 in Figure 5(b) cannot be distinguished by 1-WL is different from the case why G_1^0 and G_2^0 in Figure 5(a) cannot be distinguished by 1-WL. If $d = 1$, then no matter how t is increased, it is always insufficient to distinguish these two graphs. On the other hand, even if $t = 1$, these two graphs can be distinguished as long as we consider their 2-hop neighbourhoods rather than 1-hop neighbourhoods, i.e., d 2.

Table 5 presents that, among all simple graphs of eight vertices, $\mathscr{N}(t, d)$-WL with varying t and d values have different expressive powers in distinguishing pairs of these simple graphs. For example $\mathscr{N}(1, 1)$-WL fails to distinguish 312 pairs of simple graphs from all pairs of simple graphs of eight

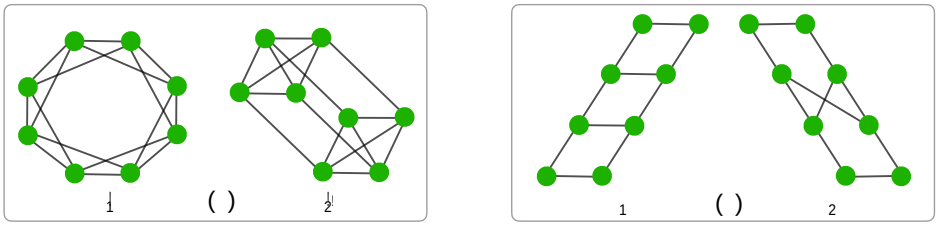


Figure 5: Two pairs of non-isomorphic simple graphs of eight vertices: (a) G_1^0 and G_2^0 cannot be distinguished by $\mathscr{N}(t,d)$-WL unless t 3, regardless how large d is; (b) G_1 and G_2 cannot be distinguished by $\mathscr{N}(t,d)$-WL unless d 2, regardless how large t is.

vertices and $\mathscr{N}(1,d)$-WL with $d = 2,3,4,5,6$ fails to distinguish 186 pairs of simple graphs. On the other hand, $\mathscr{N}(t,1)$-WL with $t = 3,4,5,6$ fails to distinguish 5 pairs of simple graphs.

In summary, for all simple graphs of eight vertices, $\mathscr{N}(1,1)$-WL with t 3 and d 2 is sufficient to distinguish any pair of them.

Table 5: Indistinguishable pairs by $\mathscr{N}(t,d)$-WL on all simple graphs of eight vertices.

	$d=1$	$d=2$	$d=3$	$d=4$	$d=5$	$d=6$
$t=1$	312	186	186	186	186	186
$t=2$	20	6	6	6	6	6
$t=3$	5	**0**	**0**	**0**	**0**	**0**
$t=4$	5	**0**	**0**	**0**	**0**	**0**
$t=5$	5	**0**	**0**	**0**	**0**	**0**
$t=6$	5	**0**	**0**	**0**	**0**	**0**

B Related Work

B.1 Expressive GNNs Beyond 1-WL Test

Below, we summarise the main directions of research for designing expressive GNNs that go beyond 1-WL test.

High-order GNNs. Following k-WL, a natural way of increasing the power of GNNs is to go higher dimensions (Azizian & Lelarge, 2021; Morris et al., 2021). Morris et al. (2019) introduced k-GNN based on a set variant of k-WL, which learns features over subgraphs on k nodes and is strictly weaker than k-WL. Maron et al. (2018; 2019b;a) developed k-order GNNs that are as expressive as k-WL and showed that a reduced 2-order GNN is as powerful as 3-WL. Morris et al. (2020b; 2022) and Zhao et al. (2022a) proposed high-order GNNs by only considering a subset of all k-tuples, i.e., k-tuples that are local or correspond to inducing subgraphs with certain connected components. These GNN models are all provably stronger than 1-WL; however, they still suffer from the shortcomings of k-WL - non-intuitive design and high computational costs - and are thus impractical for real-world tasks, particularly when graphs are large.

Unlike these works, our GNN architecture is not built upon the classical k-WL algorithms. At its core, the $\mathscr{N}$-WL algorithm proposed in this work is a node (1-dimensional) refinement algorithm based on high-order induced subgraphs in its d-hop neighbourhood, i.e., incorporating subgraph colouring into node colouring instead of propagating k-tuple colouring based on k-tuple colouring of the same dimension. Further, the $\mathscr{N}$-WL algorithm has the same locality as in standard GNNs, i.e., considering local structures in the neighbourhood within d-hop away from each node.

Injecting structures. Substructure counting has been leveraged by several GNNs to inject structural information into node or edge features of a graph (Bouritsas et al., 2022; Barceló et al., 2021). Typically, isormorphism counts (Bouritsas et al., 2022) or homomorphism counts (Barceló et al.,

2021) of small subgraph patterns (e.g., cycles and cliques) are precomputed as application-specific inductive biases. In a similar spirit, Thiede et al. (2021) applied convolutions on automorphism groups of subgraph patterns; Nguyen & Maehara (2020) used homomorphism counts of subgraph pattens as graph invariants. A recent work (Wijesinghe & Wang, 2022) introduced a way of injecting structural coefficients from local structures into neighbour aggregation. Despite being more expressive than 1-WL, all these models require preprocessing to extract structural information from subgraphs.

Although considering subgraphs, our work has some important distinctions from the above works. We consider subgraphs from a combinatorial perspective which does not require hand-choosing subgraph patterns, while the above GNN models consider specific subgraphs that are pre-defined. As noted by Barceló et al. (2021), knowing which subgraph patterns work well and which do not is critical for the power of the GNN models that reply on specific pre-defined subgraph patterns. However, this question is not easy to answer because determining which subgraph patterns work well is often application-dependent.

Subgraph-based GNNs. Several recent studies proposed to apply a base GNN on subgraphs of a graph rather than on the whole graph directly (Zhao et al., 2022b; Zhang & Li, 2021; Bevilacqua et al., 2022; Frasca et al., 2022). The way of representing a graph in terms of subgraphs may vary in models. Bevilacqua et al. (2022) represents each graph as a multiset of subgraphs chosen according to a predefined policy, e.g., removing one edge or one node. Zhao et al. (2022b) and Zhang & Li (2021) represent a graph with a multiset of induced subgraphs, each rooted at one node in the graph, which generalises rooted subtrees used in the traditional setting. It has been shown that, by applying a base GNN that is expressive as 1-WL such as GIN (Xu et al., 2019) on subgraphs, the expressive power of these models goes beyond 1-WL.

Our work is different from these subgraph-based GNNs because we do not apply a GNN on subgraphs. Instead, our GNN architecture aims to incorporate structural information (i.e., isomorphism types) and positional information (i.e., positional types) of subgraphs in the neighbourhood of each node into the node's embedding. To this end, our GNN architecture is similar to standard GNNs except that the neighbourhood information for aggregation may include the information from high-order subgraphs in a neighbourhood rather than merely neighbouring nodes.

K-hop message-passing GNNs. Recently, some attempts have been made to extend GNNs from 1-hop message passing aggregation to K-hop message passing aggregation (Abu-El-Haija et al., 2019; Nikolentzos et al., 2020; Brossard et al., 2020; Wang et al., 2020; Feng et al., 2022). The key idea is to aggregate information from not only direct neighbours (1-hop neighbours) of a node but also neighbours within K hops of a node. Feng et al. (2022) showed that standard K-hop message-passing GNNs are strictly more powerful than 1-WL test when $K > 1$, but their expressive power is bounded by 3-WL test whose proof is based on the work in (Frasca et al., 2022). To further improve the expressive power, Feng et al. (2022) proposed a new GNN model, called KP-GNN, which leverages the information of peripheral subgraphs into K-hop message-passing GNNs.

Our work provides a comprehensive theoretical framework for characterising the expressive power of different K-hop message-passing GNNs. For example, standard K-hop message-passing GNNs with the shortest path distance kernel (Feng et al., 2022) can be characterised as corresponding to our $\mathscr{N}(1, d)$-WL with $d = K$, where each node has its shortest-path distance to the target node as a positional type. Further, depending on the isomorphism types of peripheral subgraphs, KP-GNN can be characterised as an instance of $\mathscr{N}(t, d)$-WL in our work with $d = K$ and t referring to the size of peripheral subgraphs, where isomrphism types are restricted to pre-selected isormorphism types by KP-GNN.

Node identity and individualisation. Several works augmented node identifiers (Loukas, 2020; You et al., 2021) and random features (Vignac et al., 2020; Sato et al., 2021; Abboud et al., 2021) or applied individualisation and refinement (Dupty et al., 2022) to improve the expressive power of GNNs. Specifically, You et al. (2021) proposed ID-GNNs which add identity information for each rooted node during message passing. Vignac et al. (2020) manipulated node identifiers in a permutation-invariant way to learn a local context around each node. Murphy et al. (2019) introduced RP-GNN which runs a permutation-sensitive GNN by adding node identifiers and then sums over all permutations of node identifiers to obtain permutation-invariant representations. Dupty et al. (2022) individualised nodes using individualisation and refinement for canonical colouring. These GNNs are permutation invariant in expectation.

In our work, we do not assign identifiers and random features to nodes, nor individualise and refine the colours of nodes. Our GNN architecture can preserve permutation invariance.

Tensor languages. Several works have studied the expressive power of GNNs from the perspective of matrix query languages. Balcilar et al. (2021) obtained the upper bounds of the expressive power of GNNs in terms of 1-WL and 2-WL based on the MATLANG matrix query language (Brijder et al., 2019). Geerts & Reutter (2022) explored a tensor language specifically designed for specifying GNNs and discovered the connections between a guarded fragment of the MATLANG matrix query language and k-WL. The key idea is to first reduce the expressivity problem of GNNs to the problem of specifying GNNs in a tensor language and then analyse their expressive powers in terms of the number of indices and the summation depth used in tensor language expressions.

In addition to the above, Bodnar et al. (2021a;b) proposed a message passing scheme operating on topological objects such as simplicial complexes or cell complexes. Horn et al. (2021) proposed a topological layer for GNNs that can leverage topological information of a graph using persistent homology - a technique for calculating topological features such as connected components and cycles of a graph. These works also lead to more expressive GNNs than 1-WL.

B.2 Connections to k-WL and its Variants

Recently, several heuristic algorithms for the graph isomorphism problem have been developed based on the classical k-WL algorithms, which take into account the sparsity of graphs (Morris et al., 2020b; 2022; Zhao et al., 2022a). These works were motivated by addressing the issue of high computational cost associated with the classical k-WL algorithms.

In the following, we conduct a comparison of the k-WL and its variants, and our $\mathscr{N}$-WL algorithms. To obtain a better understanding of the underlying differences between these two families of algorithms (k-WL vs $\mathscr{N}$-WL), we separate the comparison of these algorithms into two aspects:

(1) Objects to be coloured in a graph (i.e., #*Coloured objects* referring to the number of coloured objects and Δ*Coloured objects* referring to the type of coloured objects);

(2) Neighbour objects when colouring each object (i.e., #*Neighbour objects* referring to the number of neighbour objects and Δ*Neighbour objects* referring to the type of neighbour objects).

Let n denote the number of nodes in a graph, a be the average number of nodes adjacent to a node in a graph (i.e., average node degree), a^d be the average number of nodes in the d-hop neighbourhood of a node, and $dia(G)$ be the diameter of a graph. We denote sets with exactly k nodes as k-sets and sets with at most k nodes as k-sets. Usually, $a^d \ll n$ when $d \ll dia(G)$. Further, subset(n^k, s) refers to the number of k-tuples in a subset of all n^k k-tuples whose induced subgraphs have at most s connected components, while subset$(\sum_{i=1}^{k} \binom{n}{i}, c)$ refers to the number of k-sets in a subset of all $\sum_{i=1}^{k} \binom{n}{i}$ sets whose induced subgraphs have at most c connected components.

Table 6 compares k-WL, $\mathscr{N}$-WL, and their variants in terms of #*Coloured objects*, #*Neighbour objects*, Δ*Coloured objects*, Δ*Neighbour objects*, and *sparsity-awareness*. We have the following observations:

- k-WL and its variants focus on colouring k-tuples, subsets of k-tuples or their corresponding set forms such as k-sets and k-sets by aggregating information from their neighbours of the same types. $\mathscr{N}$-WL aims to colour *nodes* by aggregating information from neighbouring subgraphs. Here, subgraphs may be considered as corresponding to t-sets or t-sets. In regard to their connections to GNN architectures, both k-WL and $\mathscr{N}$-WL can be easily implemented for graph-level learning tasks. However, for node-level learning tasks, as discussed in Morris et al. (2022), k-WL and its variants need an additional pooling operation to combine features of k-tuples, k-sets, or k-sets that contain a node for computing a representation of the node. In contrast, $\mathscr{N}$-WL serves as a natural paradigm that computes node representations as first-class citizens.
- k-WL has a more global nature than $\mathscr{N}$-WL in the sense that its coloured objects are grounded on all possible k-tuples of *an entire graph*. The variants of k-WL impose some

Table 6: Comparison of k-WL and their variants δ-k-LWL (Morris et al., 2020b), (k, s)-LWL (Morris et al., 2022), and (k, c)( )-SETWL (Zhao et al., 2022a) with our $\mathscr{N}(t, d)$-WL and $\mathscr{N}^{\mathsf{c}}(t, d)$-WL, where #*Coloured objects* and #*Neighbour objects* refer to the number of coloured objects in a graph and the number of neighbour objects for each coloured object, respectively; Δ*Coloured objects* and Δ*Neighbour objects* refer to the type of coloured objects and the type of neighbour objects, respectively; n is the number of nodes and a is the average node degree in a graph; a^{d} is the average number of nodes in the d-hop neighbourhood of a node. Note that a^{d} n for graphs whose diameters are considerably greater than d.

	k-WL	δ-k-LWL	(k, s)-LWL	(k, c)( )-SETWL	$\mathscr{N}(t, d)$-WL	$\mathscr{N}^{\mathsf{c}}(t, d)$-WL
#Coloured objects	n^{k}	n^{k}	subset(n^{k}, s)	subset($\sum_{q=1}^{k} \binom{n}{q}, c$)	n	n
#Neighbour objects	n k	a k	a k	n q	$\binom{a^{\mathsf{d}}}{t}$	subset($\sum_{q=1}^{t} \binom{a^{\mathsf{d}}}{q}, 1$)
ΔColoured objects	k-tuples	k-tuples	k-tuples	k-sets	nodes	nodes
ΔNeighbour objects	k-tuples	k-tuples	k-tuples	k-sets	t-sets	t-sets
Sparsity -awareness	7	3	3	3	7	3

additional conditions on all k-tuples with an aim to reduce all k-tuples to a subset of all k-tuples or k-tuples. Two commonly used conditions by the variants of k-WL are:

- reducing k-tuples or k-tuples to a set form such as k-sets and k-sets, and
- restricting coloured objects or neighbour objects to satisfying a certain connectivity conditions, e.g., containing no more than a specified number of connected components.

In contrast, $\mathscr{N}$-WL is designed to be local. This difference can be observed from Table 6. Specifically, for coloured objects, k-WL and its variants deal with the number n^{k} of all k-tuples, or a subset of all k-tuples, or k-sets from an entire graph, while $\mathscr{N}$-WL only deals with the number n of all nodes in a graph. For neighbour objects, k-WL considers the number n k of neighbour objects for each coloured object; δ-k-LWL (Morris et al., 2020b) and (k, s)-LWL (Morris et al., 2022) consider the number a k of neighbour objects, where a refers to the average node degree, since only local neighbours of a k-tuple are taken into consideration; and (k, c)( )-SETWL (Zhao et al., 2022a) considers the number n q of neighbour objects for each coloured object, where q 2 $[k]$. For $\mathscr{N}(t, d)$-WL, it considers the number $\binom{a^{\mathsf{d}}}{t}$ of subgraphs of order t from the d-hop neighbourhood of a node while $\mathscr{N}^{\mathsf{c}}(t, d)$-WL considers only connected subgraphs with t nodes. When a^{d} n, $\mathscr{N}$-WL performs much more efficient than k-WL and its variants.

- As depicted in Table 6, the variants δ-k-LWL, (k, s)-LWL, (k, c)( )-SETWL, and our $\mathscr{N}^{\mathsf{c}}(t, d)$-WL are sparsity-aware (i.e., adapt to the sparsity of a graph), while the classical k-WL and our $\mathscr{N}(t, d)$-WL algorithms are not sparsity-aware. Nonetheless, different from these k-WL variants which trade off expressivity for computational efficiency, our $\mathscr{N}^{\mathsf{c}}(t, d)$-WL is provably as powerful as $\mathscr{N}(t, d)$-WL without compromising expressive power while improving efficiency on sparse graphs (see Theorem 3.7).

Remark B.1. *It is worthy to note that, when learning a node representation, our $\mathscr{N}$-WL algorithms consider all or connected subgraphs bounded within the d-hop neighbourhood of a node, but these subgraphs may not necessarily be incident to the node. This is different from rooted subgraphs considered by the local variants of k-WL as well as subgraph-based GNNs in the literature.*

C Proofs of Theorems

C.1 Proofs for Weak Hierarchy (Theorem 3.1)

To prove Theorem 3.1, we begin with the following lemma.

Lemma 1. For any fixed $d \in \mathbb{N}$, $\mathcal{N}$ $(t{+}1, d)$-WL is at least as expressive as $\mathcal{N}$ (t, d)-WL in distinguishing non-isomorphic graphs, where $t \geq 1$.

Proof. We prove this lemma by contradiction. Assume that there exist two non-isomorphic graphs G_1 and G_2 which can be distinguished by $\mathcal{N}$ (t, d)-WL, but cannot be distinguished by $\mathcal{N}$ $(t{+}1, d)$-WL after k iterations. This implies that, for any l-th iteration where $l = 0, 1, \ldots, k-1$, $\mathcal{N}$ $(t{+}1, d)$-WL must have the same multiset of node colours for G_1 and G_2.

Now it suffices to show that, for any iteration l, if the colours of any two nodes in G_1 and G_2 are the same by $\mathcal{N}$ $(t{+}1, d)$-WL, then their node colours by $\mathcal{N}$ (t, d)-WL must also be the same. We prove this by induction.

When $l = 0$, the initial node colours are the same for $\mathcal{N}$ $(t{+}1, d)$-WL and $\mathcal{N}$ (t, d)-WL; hence, the above statement holds. When $l > 0$, we assume that this statement holds for $l-1 \geq 1$. Then, if the colours of any two nodes in G_1 and G_2 are the same at the l-th iteration by $\mathcal{N}$ $(t{+}1, d)$-WL, i.e., $\varsigma^l(u_1) = \varsigma^l(u_2)$, by Equation 1, we have:

$$(\varsigma^{l-1}(u_1), \{\!\{\xi^{l-1}_{(u_1;i)} g_{i \in I_{t+1}}\}\!\}) = (\varsigma^{l-1}(u_2), \{\!\{\xi^{l-1}_{(u_2;i)} g_{i \in I_{t+1}}\}\!\}).$$

Since u_1 and u_2 have the same colour in the $l-1$-th iteration, this gives us the following equation which must hold

$$\{\!\{\xi^{l-1}_{(u_1;i)} g_{i \in I_{t+1}}\}\!\} = \{\!\{\xi^{l-1}_{(u_2;i)} g_{i \in I_{t+1}}\}\!\}.$$

By the definition of the $\mathcal{N}$ -WL algorithm, $\{\!\{\xi^{l-1}_{(u;i)} g_{i \in I_{t+1}}\}\!\}$ represents the set of multisets of colours for subgraphs of $t{+}1$ order from the d-hop neighbourhood of a node u at the $l-1$-th iteration of applying $\mathcal{N}$ $(t{+}1, d)$-WL. This corresponds to $\binom{m}{t+1}$ induced subgraphs of order $t{+}1$ where m refers to the number of vertices in the d-hop neighbourhood of node u and w.l.o.g. we assume $t{+}1 \leq m$.

We show that $\{\!\{\xi^{l}_{(u;i)} g_{i \in I_{t+1}}\}\!\}$ determines $\{\!\{\xi^{l}_{(u;i)} g_{i \in I_{t}}\}\!\}$. Here, $\{\!\{\xi^{l}_{(u;i)} g_{i \in I_{t}}\}\!\}$ corresponds to $\binom{m}{t}$ induced subgraphs of order t in the d-hop neighbourhood of node u at the l-th iteration of $\mathcal{N}$ (t, d)-WL. Since each $\xi^{l}_{(u;i)}$ encodes the count of induced subgraphs with respect to an isomorphism type i, we just need to show that the isomorphism counts of induced subgraphs of order t are determined by the isomorphism counts of induced subgraphs of order $t{+}1$.

Let S be an induced subgraph of order $t{+}1$. We denote the deck of S as $\mathit{deck}(S)$ which is the set of induced subgraphs of order t that are obtained by deleting one vertex in every possible way from S. That is, $\mathit{deck}(S) = \{\!\{S^0 | v \in V(S), V(S^0) = V(S) \setminus \{\!\{v\}\!\}, S^0 \subseteq S\}\!\}$.

By the results in (Kocay, 1982), we know that counting induced subgraphs and counting all subgraphs have the same power. Note that each subgraph of t vertices lies in $m-t$ subgraphs of $t{+}1$ vertices. So there is a total number $(m-t)$ of subgraphs of $t{+}1$ vertices, for which each subgraph S^0 of t vertices sits in their decks. Thus, to determine the number of times a subgraph S^0 of t vertices occurs in a graph, we can first count the occurrence of S^0 in all of subgraphs of $t{+}1$ vertices, and then divide the count by $(m-t)$. Therefore, the counts of subgraphs of vertices $t{+}1$ implies the counts of subgraphs of vertices t. In doing so, we are able to infer the isomorphism counts of induced subgraphs of order t from the isomorphism counts of induced subgraphs of order $t{+}1$.

Thus, we have the following equation:

$$\{\!\{\xi^{l-1}_{(u_1;i)} g_{i \in I_{t}}\}\!\} = \{\!\{\xi^{l-1}_{(u_2;i)} g_{i \in I_{t}}\}\!\}.$$

Accordingly, the following must hold:

$$(\varsigma^{l-1}(u_1), \{\!\{\xi^{l-1}_{(u_1;i)} g_{i \in I_{t}}\}\!\}) = (\varsigma^{l-1}(u_2), \{\!\{\xi^{l-1}_{(u_2;i)} g_{i \in I_{t}}\}\!\}).$$

Hence, by Equation 1 again, we know that the node colours of u_1 and u_2 by $\mathcal{N}$ (t, d)-WL must also be the same at the l-th iteration.

This implies that there exists an injective function between colours of the nodes by $\mathcal{N}$ $(t+1, d)$-WL and by $\mathcal{N}$ (t, d)-WL at any l-th iteration. Since for any l-th iteration where $l = 0, 1, \ldots, k$ 1 $\mathcal{N}$ $(t+1, d)$-WL has the same multiset of node colours for G_1 and G_2, $\mathcal{N}$ (t, d)-WL must also have the same multiset of node colours for G_1 and G_2. This means that $\mathcal{N}$ (t, d)-WL cannot distinguish G_1 and G_2 after k iterations, which contradicts with the assumption. The proof is done. □

Theorem 3.1. For any fixed d 2 N, $\mathcal{N}$ $(t+1, d)$-WL is strictly more expressive than $\mathcal{N}$ (t, d)-WL in distinguishing non-isomorphic graphs, where t 1.

Proof. By Lemma 1, we know that $\mathcal{N}$ $(t+1, d)$-WL is as expressive as $\mathcal{N}$ (t, d)-WL. Thus, we just need to show that, for any fixed but arbitrary d 2 N, there exists at least a pair of non-isomorphic graphs $(\hat{G}_1, \hat{G}_2)$ that cannot be distinguished by $\mathcal{N}$ (t, d)-WL but can be distinguished by $\mathcal{N}$ $(t+1, d)$-WL.

Let $V(G)$ and $E(G)$ denote the set of nodes and the set of edges in G, respectively. In general, there are three cases when constructing such a pair of non-isomorphic graphs $(\hat{G}_1, \hat{G}_2)$:

- When $t = 1$, we construct $\hat{G}_1$ to be two cycles of length $2d + 1$ and $\hat{G}_2$ to be one cycle of length $4d + 2$.
- When $t = 2$, we first construct a pair (G_1, G_2) of graphs where G_1 consists of two cycles of length 4 and G_2 consists of one cycle of length 8. Let $\overline{G}$ denote the complement of a graph G. Then, we construct $\hat{G}_r = \overline{G}_r$ to be the complement of G_r for $r = 1, 2$.
- When t 3, we first construct a pair (G_1, G_2) of graphs where G_1 consists of two cycles of length ℓ and G_2 consists of one cycle of length 2ℓ, where r is defined as follows:

$$\ell = \begin{cases} (3t \quad 1)/2 \text{ or } (3t+1)/2 & \text{if } t \text{ is odd;} \\ 3t/2 & \text{if } t \text{ is even.} \end{cases} \tag{6}$$

  Then, for each of G_1 and G_2, we add a single vertex that connects to all vertices in a graph. More precisely, given two new vertices v_1 2 $V(G_1)$ and v_2 2 $V(G_2)$, we define $\hat{G}_1$ and $\hat{G}_2$ as $V(\hat{G}_r) = V(G_r)$ [fv_rg and $E(\hat{G}_r) = E(G_r)$ [f(v_r, u)ju 2 $V(G_r)$g for $r = 1, 2$.

Figure 8.(a) demonstrates the aforementioned pairs of non-isomorphic graphs for the case of $t = 1$. Figure 6 shows the pair of non-isomorphic graphs for the case of $t = 2$. Figure 7.(a)-(c) shows the pairs of non-isomorphic graphs for the cases of $t = 3$ and $t = 4$.

The proof is done. □

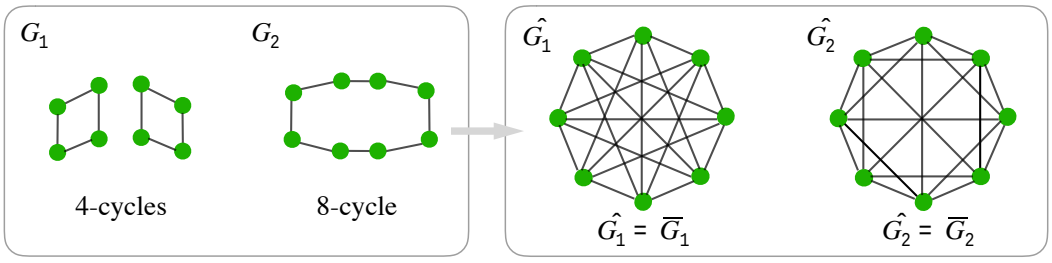
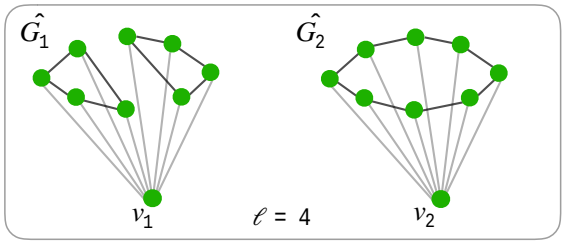
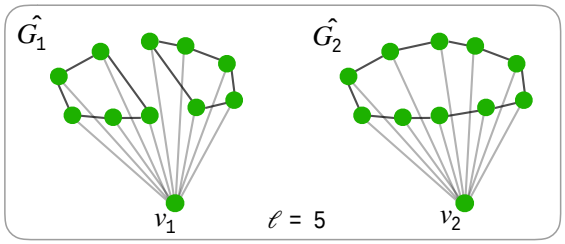


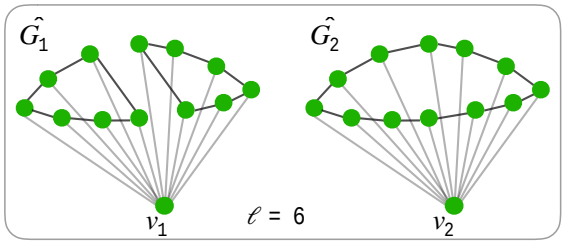
Figure 6: (**Left**) Two graphs G_1 and G_2 where G_1 consists of two 4-cycles and G_2 is one 8-cycle; (**Right**) Two graphs $\hat{G}_1$ and $\hat{G}_2$ that can be distinguished by $\mathcal{N}$ $(t+1, d)$-WL but cannot be distinguished by $\mathcal{N}$ (t, d)-WL for $t = 2$ and d 1, where $\hat{G}_1$ and $\hat{G}_2$ are the complements of G_1 and G_2, respectively.



(a)



(b)



(c)

Figure 7: Three pairs of non-isomorphic graphs $(\hat{G}_1, \hat{G}_2)$ that can be distinguished by $\mathcal{N}$ $(t+1, d)$-WL but cannot be distinguished by $\mathcal{N}$ (t, d)-WL: (a) $t = 3$ and $\ell = 4$, (b) $t = 3$ and $\ell = 5$, and (c) $t = 4$ and $\ell = 6$, where d 1.

C.2 Proofs for Strong Hierarchy (Theorem 3.2 and Theorem 3.3)

Lemma 2. For any fixed d 2 N, $\mathcal{N}(t+1, d)$-WL is at least as expressive as $\mathcal{N}(t, d)$-WL in distinguishing non-isomorphic graphs, where t 1.

Proof. The proof can be carried out in the same way as in the proof for Lemma 1. Specifically, to prove this lemma, it suffices to show that, for any two non-isomorphic graphs G_1 and G_2, if they can be distinguished by $\mathcal{N}(t, d)$-WL at the l-th iteration, they can also be distinguished by $\mathcal{N}(t+1, d)$-WL at the l-th iteration. By the definition of the $\mathcal{N}$-WL algroithm, the set of multisets of colours for subgraphs of $t+1$ order from the d-hop neighbourhood of a node u at the l-th iteration of applying $\mathcal{N}$ $(t+1, d)$-WL is represented as $f\xi^l_{(u;i;j)} g_{i2l_{t+1};j2J_d}$. This still corresponds to $\binom{m}{t+1}$ induced subgraphs of order $t+1$, where m refers to the number of vertices in the d-hop neighbourhood of node u and w.l.o.g. we assume $t+1$ m.

Now, we need to show that $f\xi^l_{(u;i;j)} g_{i2l_{t+1};j2J_d}$ determines $f\xi^l_{(u;i;j)} g_{i2l_t;j2J_d}$. $f\xi^l_{(u;i;j)} g_{i2l_t;j2J_d}$ corresponds to $\binom{m}{t}$ induced subgraphs of order t in the d-hop neighbourhood of node u at the l-th iteration of $\mathcal{N}(t, d)$-WL. Different from $\xi^l_{(u;i)}$ that encodes the count of induced subgraphs with respect to only an isomorphism type i, $\xi^l_{(u;i;j)}$ encodes the count of induced subgraphs with respect to both an isomorphism type i and a positional type j. Accordingly, we need to show that the counts

of induced subgraphs of order t with respect to both isomorphism types and positional types are determined by the counts of induced subgraphs of order $t{+}1$ with respect to both isomorphism types and positional types.

The fact that each subgraph of t vertices lies in $m \quad t$ subgraphs of $t{+}1$ vertices remains. To take both isomorphism types and positional types into account, when determining the number of times a subgraph S^0 of t vertices occurs in a graph, we can first count the occurrence of S^0 in all subgraphs of $t{+}1$ vertices with respect to both the isomorphism and positional types, and then divide the count by $(m \quad t)$. Thus, the counts of induced subgraphs of vertices $t{+}1$ imply the counts of induced subgraphs of vertices t respect to both isomorphism types and positional types. □

Theorem 3.2. For any fixed d 2 N, $\mathscr{N}(t{+}1, d)$-WL is strictly more expressive than $\mathscr{N}(t, d)$-WL in distinguishing non-isomorphic graphs, where $t \quad 1$.

Proof. By Lemma 2, we know that $\mathscr{N}(t{+}1, d)$-WL is at least as expressive as $\mathscr{N}(t, d)$-WL in distinguishing non-isomorphic graphs. For the strictness of expressivity between $\mathscr{N}(t{+}1, d)$-WL and $\mathscr{N}(t, d)$-WL, we can see that the pairs of non-isomorphic graphs $(\hat{G}_1, \hat{G}_2)$ that cannot be distinguished by $\mathscr{N} \quad (t, d)$-WL but can be distinguished by $\mathscr{N} \quad (t{+}1, d)$-WL in the proof of Theorem 3.1 still hold when taking positional types of induced subgraphs into account. Thus, we can prove this strictness in a similar way as for the case that $\mathscr{N} \quad (t{+}1, d)$-WL is strictly more expressive than $\mathscr{N} \quad (t, d)$-WL, i.e. based on the same pairs of non-isomorphic graphs $(\hat{G}_1, \hat{G}_2)$ described in the proof of Theorem 3.1. □

Before proving Theorem 3.3, we first prove the following lemma.

Lemma 3. For any fixed t 2 N, $\mathscr{N}(t, d{+}1)$-WL is at least as expressive as $\mathscr{N}(t, d)$-WL in distinguishing non-isomorphic graphs, where $d \quad 1$.

Proof. We can proceed with the proof as follows, in a way similar to Lemma 1. Assume that there exist two non-isomorphic graphs G_1 and G_2 which can be distinguished by $\mathscr{N}(t, d)$-WL, but cannot be distinguished by $\mathscr{N}(t, d{+}1)$-WL after k iterations. This implies that, for any l-th iteration where $l = 0, 1, \ldots, k \quad 1$, $\mathscr{N}(t, d{+}1)$-WL must have the same multiset of node colours for G_1 and G_2.

Then we show that, for any iteration l, if the colours of any two nodes in G_1 and G_2 are the same by $\mathscr{N}(t, d{+}1)$-WL, then their node colours by $\mathscr{N}(t, d)$-WL must also be the same. Again, we prove this by induction.

When $l = 0$, the initial node colours are the same for $\mathscr{N}(t, d{+}1)$-WL and $\mathscr{N}(t, d)$-WL. Hence, the above statement holds. When $l > 0$, we assume that this statement holds for the $l \quad 1$-th iteration. Then, if the colours of any two nodes in G_1 and G_2 are the same at the l-th iteration by $\mathscr{N}(t, d{+}1)$-WL, i.e., $\varsigma^l(u_1) = \varsigma^l(u_2)$, by Equation 3, we have:

$$(\varsigma^{l \quad 1}(u_1), f\xi^{l \quad 1}_{(u_1;i;j)} g_{i2I_t;j2J_{d+1}}) = (\varsigma^{l \quad 1}(u_2), f\xi^{l \quad 1}_{(u_2;i;j)} g_{i2I_t;j2J_{d+1}}).$$

Since u_1 and u_2 have the same colour in the $l \quad 1$-th iteration, this gives us the following equation which must hold

$$f\xi^{l \quad 1}_{(u_1;i;j)} g_{i2I_t;j2J_{d+1}} = f\xi^{l \quad 1}_{(u_2;i;j)} g_{i2I_t;j2J_{d+1}}.$$

By the definition of the permutation invariant function $f_{pos} : S_G \quad ! \quad \mathbb{N}$ that encodes the positional types of induced subgraphs and satisfies the condition: $8S_i \; 2 \; S_{(u;t;d)} 8S_j \; 2 \; (S_{(u;t;d+1)} \quad S_{(u;t;d)}) \, (f_{pos}(S_i) \quad 6= \quad f_{pos}(S_j))$, we know that $J_d \qquad J_{d+1}$ holds for induced subgraphs in neighbourhoods and any fixed but arbitrary t 2 N, where $J_d = f f_{pos}(S) | S \; 2 \; S_{(u;t;d)} g$ and $J_{d+1} = f f_{pos}(S) | S \; 2 \; S_{(u;t;d+1)} g$. Accordingly, we have the following for any node u in G_1 and G_2:

$$f\xi^{l \quad 1}_{(u;i;j)} g_{i2I_t;j2J_d} \qquad f\xi^{l \quad 1}_{(u;i;j)} g_{i2I_t;j2J_{d+1}}.$$

Since $\zeta^l : S_G \to \mathbb{N}$ such that $\zeta^l(S_1) = \zeta^l(S_2)$ iff $f_{iso}(S_1) = f_{iso}(S_2)$ and $f_{pos}(S_1) = f_{pos}(S_2)$ and $\xi^l_{(u;i;j)} = \{\!\{\zeta^l(S) | S \in S_{(u;t;d)}, f_{iso}(S)=i, \text{ and } f_{pos}(S)=j\}\!\}$, by Equation 3 again and the fact that HASH( ) in Equation 3 is an injective function, we further have

$$\{\!\{\xi^{l-1}_{(u_1;i;j)}\}\!\}_{i\in I_t;j\in J_d} = \{\!\{\xi^{l-1}_{(u_2;i;j)}\}\!\}_{i\in I_t;j\in J_d}.$$

We know $\varsigma^{l-1}(u_1) = \varsigma^{l-1}(u_2)$ by the assumption for the $l-1$-th iteration. Thus, the following must hold:

$$(\varsigma^{l-1}(u_1), \{\!\{\xi^{l-1}_{(u_1;i;j)}\}\!\}_{i\in I_t;j\in J_d}) = (\varsigma^{l-1}(u_2), \{\!\{\xi^{l-1}_{(u_2;i;j)}\}\!\}_{i\in I_t;j\in J_d}).$$

According to Equation 3, we know that the node colours of u_1 and u_2 by $\mathscr{N}(t, d)$-WL must also be the same at the l-th iteration.

This implies that there exists an injective function between the colours of nodes by $\mathscr{N}(t, d+1)$-WL and $\mathscr{N}(t, d)$-WL at any l-th iteration. Since for any l-th iteration where $l = 0, 1, \ldots, k-1$ $\mathscr{N}(t, d+1)$-WL has the same multiset of node colours for G_1 and G_2, $\mathscr{N}(t, d)$-WL must also have the same multiset of node colours for G_1 and G_2. This means that $\mathscr{N}(t, d)$-WL cannot distinguish G_1 and G_2 after k iterations, which contradicts with the assumption. Hence, $\mathscr{N}(t, d+1)$-WL is at least as expressive as $\mathscr{N}(t, d)$-WL in distinguishing non-isomorphic graphs. □

Theorem 3.3. For any fixed $t \in \mathbb{N}$, $\mathscr{N}(t, d+1)$-WL is strictly more expressive than $\mathscr{N}(t, d)$-WL in distinguishing non-isomorphic graphs, where $d \geq 1$.

Proof. Firstly, we show that, for a fixed but arbitrary $t \in \mathbb{N}$, there exists at least one pair of non-isomorphic graphs $(\hat{G}_1, \hat{G}_2)$ that cannot be distinguished by $\mathscr{N}(t, d)$-WL but can be distinguished by $\mathscr{N}(t, d+1)$-WL. Generally speaking, there are also three cases to consider when constructing such a pair of non-isomorphic graphs $(\hat{G}_1, \hat{G}_2)$:

- When $t = 1$, we construct $\hat{G}_1$ to be two cycles of length $2d + 1$ and $\hat{G}_2$ to be one cycle of length $4d + 2$.
- When $t = 2$, we construct $\hat{G}_1$ to be two cycles of length $2d + 3$ and $\hat{G}_2$ to be one cycle of length $4d + 6$.
- When $t \geq 3$, we construct $\hat{G}_1$ to be two cycles of length $2d + 2$ and $\hat{G}_2$ to be one cycle of length $4d + 4$.

Figure 8.(a)-(c) depicts the pairs of non-isomorphic graphs being constructed for the cases of any $d \geq 1$ and $t = 1$, $t = 2$, and $t = 3$, respectively.

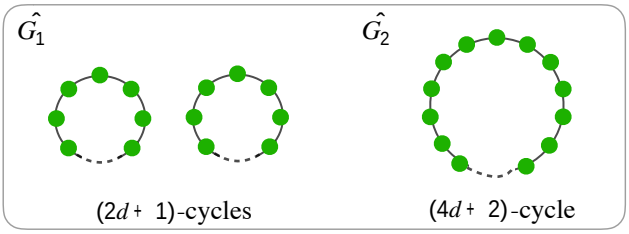
Then, by Lemma 3, we also know that $\mathscr{N}(t, d+1)$-WL is as expressive as $\mathscr{N}(t, d)$-WL. Based on the above two aspects, it can thus be concluded that $\mathscr{N}(t, d+1)$-WL is strictly more expressive than $\mathscr{N}(t, d)$-WL in distinguishing non-isomorphic graphs. □

C.3 Proofs for Connected-Hereditary Subgraphs (Theorem 3.7)

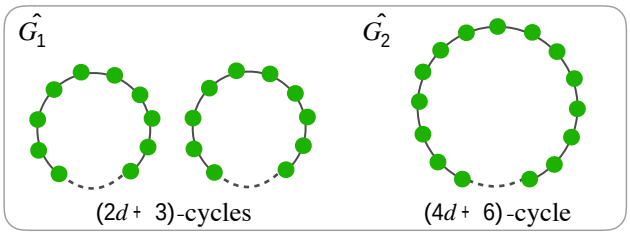
Lemma 3.5. $\mathbb{S}_c^t$ is connected-hereditary.

Proof. By the definition of $\mathbb{S}_c^t$, every $S \in \mathbb{S}_c^t$ is a connected induced subgraph. Further, if $S \in \mathbb{S}_c^t$, then every connected induced subgraph of S is also in $\mathbb{S}_c^t$. By Theorem 3.4, $\mathbb{S}_c^t$ is thus connected-hereditary. □

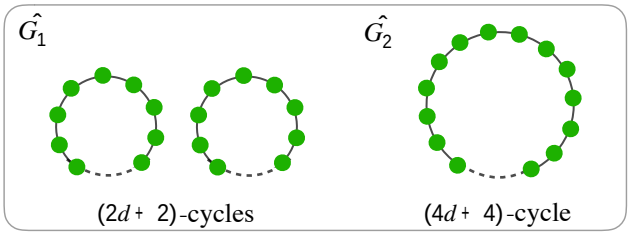
Lemma 3.6. For each induced subgraph S satisfying $S \in \mathbb{S}^{=t}$ but $S \notin \mathbb{S}_c^{=t}$, there exists a set $\{\!\{S_1, S_2, \ldots, S_q\}\!\} \subseteq \mathbb{S}_c^t$ such that $S = S_1 \diamond S_2 \diamond \ldots \diamond S_q$, where $\mu(S) = q$.



(a)



(b)



(c)

Figure 8: Three families of pairs of graphs that can be distinguished by $\mathcal{N}\ (t, d+1)$-WL but cannot be distinguished by $\mathcal{N}\ (t, d)$-WL: (a) $t = 1$ and $d\ \ 1$, (b) $t = 2$ and $d\ \ 1$, and (c) $t\ \ 3$ and $d\ \ 1$.

Proof. Since S 2 $S^{=t}$ but S 2 $S_c^{=t}$, such an induced subgraph S must have more than one connected component. Then, for each connected component in S, it corresponds to one connected induced subgraph in $S_c{}^t$. Thus, it is straightforward to show that $S = S_1\ \ S_2\ \ \ldots\ \ S_q$ where $fS_1, S_2, \ldots, S_qg$ corresponds to the set of components of S and $fS_1, S_2, \ldots, S_qg\ \ S_c{}^t$ is such a set of connected induced subgraphs. □

The proof of Theorem 3.7 depends on two lemmas - Lemma 4 and Lemma 5. In the following, we prove these two lemmas first.

Lemma 4. For any t 2 N and d 2 N, $\mathcal{N}^c(t, d)$-WL is at least as expressive as $\mathcal{N}(t, d)$-WL in distinguishing non-isomorphic graphs.

Proof. For clarity, we use $\varsigma_c^l(u)$ and $\varsigma^l(u)$ to refer to the colour of a node u by $\mathcal{N}^c(t, d)$-WL and $\mathcal{N}(t, d)$-WL, respectively, in the following of this proof.

We prove this lemma by contradiction. Assume that there exist two non-isomorphic graphs G_1 and G_2 which can be distinguished by $\mathcal{N}(t, d)$-WL, but cannot be distinguished by $\mathcal{N}^c(t, d)$-WL after k iterations. This implies that, for any l-th iteration where $l = 0, 1, \ldots, k\ \ 1$, $\mathcal{N}^c(t, d)$-WL must have the same multiset of node colours for G_1 and G_2, i.e., $ff\varsigma_c^l(u_1)gg_{u_1 2 V(G_1)} = ff\varsigma_c^l(u_2)gg_{u_2 2 V(G_2)}$.

Below, we need to prove the following statement:

(A1). For any iteration l, if the colours of any two nodes in G_1 and G_2 are the same by $\mathcal{N}^c(t, d)$-WL, i.e., $\varsigma_c^l(u_1) = \varsigma_c^l(u_2)$, then their node colours by $\mathcal{N}(t, d)$-WL must also be the same, i.e., $\varsigma^l(u_1) = \varsigma^l(u_2)$.

We prove Statement A1 by induction.

- For $l = 0$, Statement A1 holds since the initial node colours are the same for $\mathscr{N}^c(t,d)$-WL and $\mathscr{N}(t,d)$-WL.
- Assume that Statement A1 holds for $l \quad 1$. Then if $\varsigma_c^l(u_1) = \varsigma_c^l(u_2)$, by Equation 4 and the fact that HASH( ) is injective, we have:

$$\Big(\varsigma_c^{l\ 1}(u_1), \Big[_{k2[1;t]} f_{\xi^{l\ 1}_{(u_1;i;j)}} g_{i2I_k^c;j2J_d}\Big) = \Big(\varsigma_c^{l\ 1}(u_2), \Big[_{k2[1;t]} f_{\xi^{l\ 1}_{(u_2;i;j)}} g_{i2I_k^c;j2J_d}\Big).$$

  This leads to the following equation:

$$\Big[_{k2[1;t]} f_{\xi^{l\ 1}_{(u_1;i;j)}} g_{i2I_k^c;j2J_d} = \Big[_{k2[1;t]} f_{\xi^{l\ 1}_{(u_2;i;j)}} g_{i2I_k^c;j2J_d}.$$

  Now, we need to show that, with respect to both isomorphism types and positional types, the counts of connected subgraphs of vertices k for $1 \quad k \quad t$ determine the counts of all subgraphs of t vertices.

  We first ignore positional types and focus on proving that the counts of connected subgraphs of k vertices for $1 \quad k \quad t$ determine the counts of all subgraphs of t vertices with respect to isomorphism types. The main theorem upon which our proof will be based is the Vertex Theorem of Kocay (1982).

  Let $G = (V, E)$ be a graph, $G[V^0]$ for $V^0 \quad V$ be an induced subgraph of G whose vertex set is V^0 and whose edge set consists of all of the edges in E that have both endpoints in V^0, and $c(S, G) = \mathit{f}V^0 \quad V(G) : G[V^0] \ ' \ S\mathit{gj}$ denote the number of induced subgraphs of G which are isomorphic to S, i.e., isomorphism counts of S in G. In the following, we omit the symbol G in $c(S, G)$ unless the graph G needs to be specified.

  We start with introducing Kocay's Vertex Theorem.

  Vertex Theorem. Let $\mathit{hi}_1, \ldots, i_b\mathit{i}$ be a list of all isomorphism types of graphs, and S_1 and S_2 be any two graphs. Then

$$c(S_1)c(S_2) \quad \sum_{1 \quad m \quad b} a_m c(i_m), \tag{7}$$

  where the coefficient a_m is the number of decompositions of $V(i_m)$ into $V_1 \ [\ V_2$ such that $i_m[V_1] \ ' \ S_1$ and $i_m[V_2] \ ' \ S_2$, and V_1 and V_2 may be overlapping.

  Below, we show how to use Kocay's Vertex Theorem to prove that, given the counts of connected subgraphs of k vertices with $1 \quad k \quad t$ with respect to their isomorphism types, the counts of subgraphs of t vertices with respect to all isomorphism types can be determined.

  If an isomorphism type i corresponds to connected subgraphs, i.e., $i \ 2 \ I_t^c$, then it is trivial to prove because the count of connected subgraphs with respect to such an isomorphism type is already available. Thus, we only need to discuss the case for $i \ 2 \ (I_t \quad I_t^c)$ in the following. We prove this by induction:

  - For $k = 1$, there is only one isomorphism type i_{node} that is in I_k^c. Thus, the count of nodes with respect to i_{node} is already available.
  - For $k = 2$, there are two isomorphism types - one is in I_k^c (i.e., i_{edge}) and the other is in $(I_k \quad I_k^c)$ (i.e., i_{noedge}). By the Vertex Theorem, we can determine the count of subgraphs for i_{noedge} easily through the following equation by applying the Vertex Theorem:

$$c(i_{\mathit{node}})c(i_{\mathit{node}}) \quad a_1 c(i_{\mathit{noedge}}) + a_2 c(i_{\mathit{edge}}) + a_3 c(i_{\mathit{node}}). \tag{8}$$

Here, a_1, a_2, and a_3 are the coefficients. Following the Vertex Theorem, each a_m where $m = 1, 2, 3$ represents the number of decompositions of $V(i_m)$, where $i_m = i_{noedge}, i_{edge}, i_{node}$, respectively, into $V_1 \subseteq V_2$ such that $i_m[V_1] \simeq i_{node}$ and $i_m[V_2] \simeq i_{node}$. Note that, it is possible that $V_1 \cap V_2 \neq \emptyset$. Based on this, we have $a_1 = 2$, $a_2 = 2$, and $a_3 = 1$. That is,

$$c(i_{node})c(i_{node}) = 2c(i_{noedge}) + 2c(i_{edge}) + c(i_{node}). \tag{9}$$

Since $i_{node} \in I_k^c$ and $i_{edge} \in I_k^c$, the values of $c(i_{node})$ and $c(i_{edge})$ are already available. Hence, we can calculate the value of $c(i_{noedge})$ according to the above equation.

– Assume that this holds for $k = t - 1$, we now prove the case $k = t$. For each isomorphism type $i \in (I_t - I_t^c)$, we apply Equation 7 of the Vertex Theorem as follows:

  * The LHS of Equation 7 corresponds to two subgraphs S_1 and S_2 of i which satisfy the condition: $V(S_1) \neq \emptyset$, $V(S_2) \neq \emptyset$, and $S_1 \cup S_2 = i$, i.e., i is the union of two node-disjoint subgraphs S_1 and S_2;
  * The RHS of Equation 7 corresponds to the set of all isomorphism types in $\bigcup_{1 \le k \le t} I_k$, including the isomorphism type i.

  Note that the order of considering isomorphism types in $(I_t - I_t^c)$ is non-trivial. Specifically, given two isomorphism types i and i' from I_t with $V(i) = V(i')$, we say that i *subsume* i', denoted as $i \succeq i'$, if and only if there exists at least one decomposition of $V(i)$ into $V_1^0, \ldots, V_q^0$ such that $\bigcup_{1 \le b \le q} V_b^0 = V(i)$ and $\{i[V_1^0], \ldots, i[V_q^0]\} \simeq \{S_1^0, S_2^0, \ldots, S_q^0\}$ where $\{S_1^0, S_2^0, \ldots, S_q^0\}$ is the set of all connected components of i'. For any two isomorphism types from $(I_t - I_t^c)$, one isomorphism type should be calculated before the other isomorphism type if the former subsumes the latter. Since all isomorphism types in I_t have the same number of vertices (i.e., t vertices) and are distinct from each other, we have the following properties:

  P1. $(I_t, \succeq)$ is a poset, i.e., $\succeq$ is a partial order on isomorphism types in I_t.
  P2. Coefficients for isomorphism types in $I_t - \{i\}$ are zero if they are subsumed by i.

  According to these two properties, we know that, given any isomorphism type $i \in (I_t - I_t^c)$, the values of $c(i')$ for isomorphism types $i' \in (I_t - I_t^c)$ that satisfy $i' \succeq i$ are already known, while the values of $c(i'')$ for isomorphism types $i'' \in (I_t - I_t^c)$ that satisfy $i \succeq i''$ are unknown but also not needed since their corresponding coefficients are zeros. Moreover, by the assumption that the case $k = t - 1$ holds, the values of $c(i''')$ for isomorphism types $i''' \in I_k$ where $1 \le k \le t$ are known, including $c(S_1)$ and $C(S_2)$. Hence, the value of $c(i)$ can be determined by applying Equation 7 of the Vertex Theorem as described above.

Since positional types provide additional information to distinguish subgraphs, we may view them as one additional feature for subgraphs and handle it as an extension to isomorphism types to prove that the counts of connected subgraphs of vertices k for $1 \le k \le t$ determine the counts of all subgraphs of t vertices with respect to both isomorphism types and positional types. This thus leads to the following:

$$\{\xi^{l-1}_{(u_1;i;j)}\}_{i \in I_t; j \in J_d} = \{\xi^{l-1}_{(u_2;i;j)}\}_{i \in I_t; j \in J_d}.$$

Since we assume that Statement A1 holds for the $(l-1)$-th iteration, i.e., if $\varsigma_c^{l-1}(u_1) = \varsigma_c^{l-1}(u_2)$, then $\varsigma^{l-1}(u_1) = \varsigma^{l-1}(u_2)$, we further have the following:

$$\varsigma^{l-1}(u_1), \{(\xi^{l-1}_{(u_1;i;j)}, i, j)\}_{i \in I_t; j \in J_d}) = (\varsigma^{l-1}(u_2), \{(\xi^{l-1}_{(u_2;i;j)}, i, j)\}_{i \in I_t; j \in J_d}).$$

Hence, from the above equation and by the definition of node colouring in Equation 3, we know that $\varsigma^l(u_1) = \varsigma^l(u_2)$. Statement A1 thus holds for the l-th iteration.

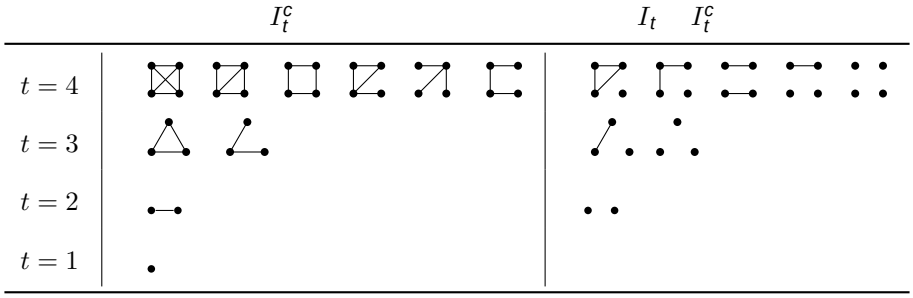


Figure 9: Isomorphism types for graphs of t vertices, where $1 \quad t \quad 4$, I_t^c refers to a set of isomorphism types for connected subgraphs of order less than or equal to t, and $I_t \quad I_t^c$ refers to a set of isomorphism types for disconnected subgraphs (i.e., containing at least two connected components) of order less than or equal to t.

According to Statement A1, there must exist an injective function f such that $\varsigma^l(u) = f(\varsigma_c^l(u))$ for any vertex in G_1 and G_2. Then, since for any l-th iteration where $l = 0, 1, \ldots, k \quad 1$ $\mathscr{N}^c(t,d)$-WL has the same multiset of node colours for G_1 and G_2, i.e., $\{\!\{\varsigma_c^l(u_1) | u_1 \in V(G_1)\}\!\} = \{\!\{\varsigma_c^l(u_2) | u_2 \in V(G_2)\}\!\}$, $\mathscr{N}(t,d)$-WL must also have the same multiset of node colours for G_1 and G_2, i.e., $\{\!\{f(\varsigma_c^l(u_1)) | u_1 \in V(G_1)\}\!\} = \{\!\{f(\varsigma_c^l(u_2)) | u_2 \in V(G_2)\}\!\}$. This means that $\mathscr{N}(t,d)$-WL cannot distinguish G_1 and G_2 after k iterations, which contradicts with the assumption. □

Example 1. Figure 9 depicts all isomorphism types for graphs with up to 4 vertices. Below, we illustrate how to calculate $c(\cdot\!-\!\cdot\ \ \cdot)$ on graphs by applying the Vertex Theorem, where $\cdot\!-\!\cdot\ \ \cdot$ is an isomorphism type in $(I_t \quad I_t^c)$ for $t = 4$.

– **Step 1:** We calculate $c(\cdot\ \cdot\ \ \cdot)$ using the following equation, where $c(\ \cdot\ )$ and $c(\cdot\!-\!\cdot)$ are known because both $\cdot$ and $\cdot\!-\!\cdot$ are isomorphism types in $_{1\ \ k\ \ 2} I_k^c$:

$$c(\ \cdot\ )c(\ \cdot\ ) \quad 2c(\cdot\ \cdot\ ) + 2c(\cdot\!-\!\cdot) + c(\ \cdot\ ). \tag{10}$$

– **Step 2:** We calculate $c(\cdot\!-\!\cdot\ \ \cdot)$ by decomposing $\cdot\!-\!\cdot\ \ \cdot$ into two subgraphs $S_1 = \cdot\ \ \cdot$ and $S_2 = \cdot\!-\!\cdot$. Since the coefficients that correspond to the following isomorphism types are zeros, we omit these isomorphism types in the equation:

Then, we have the equation below:

$$\begin{aligned} c(\cdot\!-\!\cdot)c(\cdot\ \cdot\ ) \quad & a_1 c(\cdot\!-\!\cdot\ \ \cdot) + a_2 c(\cdot\!-\!\cdot\ \ \cdot) + a_3 c(\cdot\!-\!\cdot\ \ \cdot) + a_4 c(\cdot\!-\!\cdot\ \ \cdot) + a_5 c(\cdot\!-\!\cdot\ \ \cdot) + a_6 c(\cdot\!-\!\cdot\ \ \cdot) + \\ & a_7 c(\cdot\!-\!\cdot\ \ \cdot) + a_8 c(\cdot\!-\!\cdot\ \ \cdot) + a_9 c(\cdot\!-\!\cdot\ \ \cdot) + a_{10} c(\cdot\!-\!\cdot\ \ \cdot) + \\ & a_{11} c(\cdot\!-\!\cdot\ \ \cdot) + a_{12} c(\cdot\!-\!\cdot\ \ \cdot) + \\ & a_{13} c(\cdot\!-\!\cdot\ \ \cdot) + \\ & a_{14} c(\cdot\!-\!\cdot) \end{aligned} \tag{11}$$

Following the Vertex Theorem, we have the following coefficients for Equation 11:

a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}	a_{12}	a_{13}	a_{14}
0	1	0	2	3	1	3	2	0	1	0	2	2	0

In the RHS of Equation 11, the isomorphism types in the first, third and fifth lines belong to I_k^c for $k = 4, 3, 2$, respectively. Hence, the counts of subgraphs with respect to these isomorphism types are already available. The isomorphism types in the second line belong to $I_k \quad I_k^c$ for $k = 4$, By the property $P1$, we know that the counts of subgraphs with respect to these isomorphism types  ,  and  have been calculated before calculating the count with respect to  . The isomorphism types in the fourth line belong to $I_k \quad I_k^c$ for $k = 3$, and by the induction, the count of subgraphs with respect to the isomorphism type  is available. Therefore, only the value of $c($  $)$ is unknown in the RHS of Equation 11. On the other hand, the values for $c($  $)$ and $c($  $)$ are both known because and belong to I_k for $k = 2$. Putting things together, we can determine the value of $c($$)$ using Equation 11.

In the following, we illustrate how to apply Equation 10 and Equation 11 in the above steps to calculate $c($  $)$ for two graphs: and  .

(1) Calculating $c($  $)$ for  : We hereby detail how $c($$)$ can be derived for the graph  . We know $c($  $) = 5$ and $c($$) = 6$ for the graph  . By applying Equation 10 in Step 1, we have

$$5 \quad 5 \quad 2c(\text{}) + 2 \quad 6 + 5. \tag{12}$$

Hence, we obtain $c($$) = 4$.

Now we apply Step 2. Because  ,  and  all subsume , we have calculated $c($$) = c($$) = 0$ and $c($$) = 2$. Further, we know the counts of isomorphism types that belong to I_k^c for $k = 4, 3, 2$, i.e. we have $c($  $) = c($$) = 0$, $c($$) = c($$) = 1$, $c($$) = c($$) = 2$ and $c($$) = c($$) = 6$. By the induction, we have also calculated $c($$) = 3$ at $k = 3$.

By applying Equation 11, we thus have

$$\begin{aligned} 6 \quad 4 \quad & 0 + 0 + 0 + 2 \quad 2 + 0 + 1 \quad 2 + \\ & 0 + 0 + 0 + c(\text{  }) + \\ & 0 + 2 \quad 6 + \\ & 2 \quad 3 + \\ & 0 \end{aligned}$$

By solving it we have $c($$) = 0$ for the graph .

(2) Calculating $c($$)$ for : We show another example of calculating $c($$)$ for the graph . We know $c($$) = 5$ and $c($$) = 4$ for the graph . By applying Equation 10 in Step 1, we have

$$5 \quad 5 \quad 2c(\text{}) + 2 \quad 4 + 5. \tag{13}$$

Hence, we know $c($$) = 6$.

Now we apply Step 2. Because , and all subsume , we have calculated $c($$) = c($$) = 0$ and $c($$) = 1$. Further, we know the counts of isomorphism types that belong to I_k^c for $k = 4, 3, 2$, i.e. we have $c($$) = c($$) = c($$) = c($$) = 0$, $c($$) = 1$, $c($$) = 2$ and $c($$) = c($$) = 4$. By the induction, we have also calculated $c($$) = 4$ at $k = 3$.

By applying Equation 11, we thus have

$$
\begin{aligned}
4 \quad 6 \quad & 0+0+0+0+3 \quad 1+1 \quad 2+ \\
& 0+2 \quad 1+0+c(\bullet\!\!-\!\!\bullet)+ \\
& 0+2 \quad 4+ \\
& 2 \quad 4+ \\
& 0
\end{aligned}
$$

By solving it we have $c(\bullet\!\!-\!\!\bullet) = 1$ for the graph  .

Remark C.1. *It is worthy to notice that, when calculating $c(i)$ for an isomorphism type i 2 $(I_t \quad I_t^c)$, the ways of applying Equation 7 of the Vertex Theorem may not be unique.*

Taking the isomorphism type $c(\bullet\!\!-\!\!\bullet)$ for example, we can calculate it in two different ways:

(1) As illustrated in Example 1, we may first calculate $c(\bullet\ \bullet)$ via treating $S_1 =$ $\bullet$ and $S_2 =$ $\bullet$, and then calculate $c(\bullet\!\!-\!\!\bullet)$ via treating $S_1 =$ $\bullet\!-\!\bullet$ and $S_2 =$ $\bullet\ \bullet$.

(2) Alternatively, we may first calculate $c(\diagup\!\bullet\ \bullet)$ via treating $S_1 =$ $\bullet\!-\!\bullet$ and $S_2 =$ $\bullet$, and then calculate $c(\bullet\!\!-\!\!\bullet)$ via treating $S_1 =$ $\diagup\!\bullet\ \bullet$ and $S_2 =$ $\bullet$.

Lemma 5. For any t 2 N and d 2 N, $\mathscr{N}(t,d)$-WL is at least as expressive as $\mathscr{N}^{\mathsf{C}}(t,d)$-WL in distinguishing non-isomorphic graphs.

Proof. The proof follows a similar structure as used in the previous lemmas. Specifically, we assume that there exist two non-isomorphic graphs G_1 and G_2 which can be distinguished by $\mathscr{N}^{\mathsf{C}}(t,d)$-WL, but cannot be distinguished by $\mathscr{N}(t,d)$-WL after k iterations. This implies that, for any l-th iteration where $l = 0, 1, \ldots, k \quad 1$, $\mathscr{N}(t,d)$-WL must have the same multiset of node colours for G_1 and G_2, i.e., $\mathsf{ff}\varsigma^l(u_1)\mathsf{gg}_{u_1 2 V(G_1)} = \mathsf{ff}\varsigma^l(u_2)\mathsf{gg}_{u_2 2 V(G_2)}$.

We need to prove the following statement:

(A2). For any iteration l, if the colours of any two nodes in G_1 and G_2 are the same by $\mathscr{N}(t,d)$-WL, i.e., $\varsigma^l(u_1) = \varsigma^l(u_2)$, then their node colours by $\mathscr{N}^{\mathsf{C}}(t,d)$-WL must also be the same, i.e., $\varsigma_{\mathsf{C}}^l(u_1) = \varsigma_{\mathsf{C}}^l(u_2)$.

We prove Statement A2 by induction:

- For $l = 0$, Statement A2 holds since the initial node colours are the same for $\mathscr{N}^{\mathsf{C}}(t,d)$-WL and $\mathscr{N}(t,d)$-WL.
- Assume that Statement A2 holds for $l \quad 1$. Then if $\varsigma^l(u_1) = \varsigma^l(u_2)$, by Equation 3, we have

$$\varsigma^{l \quad 1}(u_1), \mathsf{f}(\xi^{l \quad 1}_{(u_1;i;j)}, i, j)\mathsf{g}_{i2I_t;j2J_d}) = (\varsigma^{l \quad 1}(u_2), \mathsf{f}(\xi^{l \quad 1}_{(u_2;i;j)}, i, j)\mathsf{g}_{i2I_t;j2J_d} \quad .$$

  This gives us the following

$$\mathsf{f}\xi^{l \quad 1}_{(u_1;i;j)}\mathsf{g}_{i2I_t;j2J_d} = \mathsf{f}\xi^{l \quad 1}_{(u_2;i;j)}\mathsf{g}_{i2I_t;j2J_d}.$$

  Then, we need to further prove that, with respect to both isomorphism types and positional types, the counts of all subgraphs of t vertices can determine the counts of connected subgraphs of k vertices for $1 \quad k \quad t$. The proof may follow a similar counting argument from Lemma 1 by noticing that each subgraph of t vertices uniquely determines the set of all subgraphs of k vertices contained within it for $1 \quad k \quad t$. Thus, to determine the

number of times a subgraph S of k vertices occurs in a graph where $1 \quad k \quad t$, we can check all subgraphs of t vertices which contain it and then divide the count by $\binom{m-t}{t-k}$, i.e., the number of subgraphs of vertices t which contain S where $m = |N_d(u)|$ is the size of the neighbourhood of a node u. Thus, we have

$$\left[f_{\xi^{l-1}_{(u_1;i;j)}} g_{i2I^c_k;j2J_d} = \left[f_{\xi^{l-1}_{(u_2;i;j)}} g_{i2I^c_k;j2J_d}. \right. \right.$$

Similar to the previous direction, by assumption Statement A2 holds for the $(l-1)$-th iteration, i.e., if $\varsigma^{l-1}(u_1) = \varsigma^{l-1}(u_2)$, then $\varsigma_c^{l-1}(u_1) = \varsigma_c^{l-1}(u_2)$. We obtain the following:

$$\left(\varsigma_c^{l-1}(u_1), \left[f_{\xi^{l-1}_{(u_1;i;j)}} g_{i2I^c_k;j2J_d} \right) = \left(\varsigma_c^{l-1}(u_2), \left[f_{\xi^{l-1}_{(u_2;i;j)}} g_{i2I^c_k;j2J_d} \right). \right. \right.$$

By the definition of node colouring in Equation 4 for $\mathscr{N}^c(t,d)$-WL, we conclude $\varsigma_c^l(u_1) = \varsigma_c^l(u_2)$. Hence, Statement A2 also holds for the l-th iteration.

According to Statement A2, there must exist an injective function f^0 such that $\varsigma_c^l(u) = f^0(\varsigma^l(u))$ for any vertex in G_1 and G_2. Then, since for any l-th iteration where $l = 0, 1, \ldots, k-1$ $\mathscr{N}(t,d)$-WL has the same multiset of node colours for G_1 and G_2, i.e., $\{\!\{\varsigma^l(u_1) | u_1 2 V(G_1)\}\!\} = \{\!\{\varsigma^l(u_2) | u_2 2 V(G_2)\}\!\}$, $\mathscr{N}_c(t,d)$-WL must also have the same multiset of node colours for G_1 and G_2, i.e.,

$$\{\!\{f^0(\varsigma^l(u_1)) | u_1 2 V(G_1)\}\!\} = \{\!\{f^0(\varsigma^l(u_2)) | u_2 2 V(G_2)\}\!\}.$$

This means that $\mathscr{N}^c(t,d)$-WL cannot distinguish G_1 and G_2 after k iterations, which contradicts with the assumption. The proof is done. □

Theorem 3.7. For any $t \in \mathbb{N}$ and $d \in \mathbb{N}$, $\mathscr{N}^c(t,d)$-WL and $\mathscr{N}(t,d)$-WL have the same expresivity in distinguishing non-isomorphic graphs.

Proof. Lemma 4 and Lemma 5 together prove this theorem. □

C.4 Proofs for Connections to k-WL Hierarchy (Theorem 3.8)

Theorem 3.8. $\mathscr{N}(1,1)$-WL is equivalent to 1-WL in distinguishing non-isomorphic graphs.

Proof. $\mathscr{N}(1,1)$-WL has $t = 1$ and $d = 1$. Accordingly, the node colouring function in Equation 3 for $\mathscr{N}(1,1)$-WL can be expressed as

$$\varsigma^{l+1}(u) = \text{HASH}\left(\varsigma^l(u), \quad \{\!\{\varsigma^l(u) | v \in N_1(u)\}\!\} \quad .\right.$$

We may remove $\{\!\{\varsigma^l(u)\}\!\}$ since it occurs twice in the input of the above hash function and does not add additionl information. This leads to the following simplified expression:

$$\varsigma^{l+1}(u) = \text{HASH}(\varsigma^l(u), \{\!\{\varsigma^l(v) | v \in N(u)\}\!\}),$$

which is exactly the same as the node colouring function of 1-WL. □

C.5 PROOFS FOR GRAPH NEIGHBOURHOOD NEURAL NETWORK (THEOREM 4.1)

Theorem 4.1. G3N-(t, d) with injective COMBINE and AGG^{N} functions, an injective AGG^{T} function w.r.t. multisets of subgraphs with the same isomorphism and positional types, an injective graph readout function, and sufficiently many layers is as powerful as $\mathscr{N}(t, d)$-WL.

Proof. The proof can proceed similarly to the proof that GIN is as expressive as 1-WL (Xu et al., 2019). Let G_1 and G_2 be any two non-isomorphic graphs which can be distinguished by $\mathscr{N}(t, d)$-WL at the k-th iteration. It suffices to show that G3N's neighbourhood aggregation scheme described in Equation 5 with the above assumptions is able to map G_1 and G_2 into different multisets of node features. The G3N's neighbourhood aggregation scheme is restated as

$$h_u^{(l)} = \text{COMBINE}\left(h_u^{(l-1)}, \text{AGG}^{N}_{(i;j)\in I_t \times J_d}\left(\text{AGG}^{T}_{S\in S_u^{(l-1)}(i;j)}(\text{POOL}(S))\right)\right).$$

According to Equation 3, $\mathscr{N}(t, d)$-WL applies an injective hash function $\text{HASH}(\,)$ to update the node colours at the l-th iteration:

$$\varsigma^{l}(u) = \text{HASH}(\varsigma^{l-1}(u), \{\!\{\xi^{l-1}_{(u;i;j)} \mid i \in I_t; j \in J_d\}\!\}),$$

where $\xi^{l-1}_{(u;i;j)} = \{\!\{\varsigma^{l-1}(S) \mid S \in S_{(u;t;d)}, f_{iso}(S)=i, \text{ and } f_{pos}(S)=j\}\!\}$.

Below, we need to show that there always exists an injective function f such that $h_u^{(l)} = f(\varsigma^{l}(u))$, for any iteration l. We prove this by induction.

- When $l = 0$, both $h_u^{(l)}$ and $\varsigma^{l}(u)$ are the input node feature of a node u. Thus, $h_u^{(l)} = \varsigma^{l}(u)$ holds for any node u in G_1 and G_2.
- Assume that $h_u^{(l-1)} = f(\varsigma^{l-1}(u))$ holds, we now need to show that $h_u^{(l)} = f(\varsigma^{l}(u))$ holds. Firstly, we have:

$$h_u^{(l)} = \text{COMBINE}\left(f(\varsigma^{l-1}(u)), \text{AGG}^{N}_{(i;j)\in I_t \times J_d}\left(\text{AGG}^{T}_{S\in S_u^{(l-1)}(i;j)}(\text{POOL}(S))\right)\right).$$

  Since we assume that both $\text{COMBINE}(\,)$ and $\text{AGG}^{N}(\,)$ are injective functions and the composition of injective functions is injective, there exists some injective function g such that

$$h_u^{(l)} = g\left(\varsigma^{l-1}(u), \left\{\!\!\left\{\text{AGG}^{T}_{S\in S_u^{(l-1)}(i;j)}(\text{POOL}(S))\right\}\!\!\right\}_{i\in I_t; j\in J_d}\right).$$

  Because $S_u^{(l-1)}(i, j)$ denotes the set of t-order subgraphs within the d-hop neighbourhood of a node u with the isomorphism type i and the positional type j at the $(l-1)$-th layer, we have $\xi^{l-1}_{(u;i;j)} = \{\!\{\varsigma^{l-1}(S) \mid S \in S_u^{(l-1)}(i, j)\}\!\}$. By the definition of subgraph colouring, i.e., $\zeta^{l} : S_G \to \mathbb{N}$ such that $\zeta^{l}(S_1) = \zeta^{l}(S_2)$ iff $f_{iso}(S_1) = f_{iso}(S_2)$ and $f_{pos}(S_1) = f_{pos}(S_2)$, and the assumption that $\text{AGG}^{T}(\,)$ is an injective function with respect to multisets of subgraphs with the same isomorphism and positional types, we further have the following:

$$h_u^{(l)} = g'\left(\varsigma^{l-1}(u), \{\!\{\xi^{l-1}_{(u;i;j)} \mid i \in I_t; j \in J_d\}\!\}\right),$$

  where g' is also an injective function. Thus, we have $h_u^{(l-1)} = g' \circ \text{HASH}^{-1}(\varsigma^{l}(u))$ and $f = g' \circ \text{HASH}^{-1}$. Since g' and HASH are injective, and the composition of injective functions is injective, f is injective.

At the k-th iteration, if $\mathscr{N}(t,d)$-WL distinguishes G_1 and G_2 to be non-isomorphic, this implies that G_1 and G_2 differ in $\{\!\{\varsigma^k(u) | u \in V(G_1)\}\!\}$ and $\{\!\{\varsigma^k(u) | u \in V(G_2)\}\!\}$. Then, G_1 and G_2 must also have different G3N's node embeddings at the k-th iteration due to the injectivity of f. The proof is done. □

D G3N MODEL ARCHITECTURE

Here we describe the specific G3N layer in more detail. An instance of Equation 5 we employ is as follows:

$$h_u^{(l+1)} = \sigma\left(h_u^{(l)} W^{(l)} + \sum_{(i,j)\in I_t \times J_d} \alpha_{(i,j)}^{(l)} \left(\sum_{S \in \mathcal{S}_u^{(l)}(i,j)} \sigma\left(\sum_{v \in S} h_v^{(l)} W_{(i,j)}^{(l)} \right) \right)\right) \tag{14}$$

where $h_u^{(l)} \in \mathbb{R}^{d_1}$ and $h_u^{(l+1)} \in \mathbb{R}^{d_2}$ are learned node embeddings at each layer, and excluding sub- and superscript notations, $W \in \mathbb{R}^{d_1 \times d_2}$ are learnable linear transformations and α are learnable scalars corresponding to different isomorphism and positional types of subgraphs. Furthermore, σ is any nonlinear activation function such as sigmoid or ReLU.

Matching the notation described earlier in Equation 5, the pooling function for subgraphs can be defined by any permutation invariant pooling function. Here, given a subgraph $S \in \mathcal{S}_u^{(l)}(i,j)$, we have

$$\textsc{Pool}(S) = \sigma\left(\sum_{v \in S} h_v^{(l)} W_{(i,j)}^{(l)} \right). \tag{15}$$

In practice, we may also consider component-wise product over node embeddings in a subgraph: $\textsc{Pool}(S) = \prod_{v \in S} \sigma\left(h_v^{(l)} W_{(i,j)}^{(l)} \right)$. In this case, σ has to be a bounded activation function such as sigmoid in order to ensure numerical stability. The inner aggregation step $\textsc{Agg}^T$ simply sums up all the subgraph embeddings of the same isomorphism and positional types, while the outer aggregation step $\textsc{Agg}^N$ applies a weighted sum with learnable weights on the aggregated groups of subgraph embeddings of the same isomorphism and positional types. To match expressivity of $\mathscr{N}$-WL, one may replace the outer activation function σ with an MLP and replace $W^{(l)}$ with $1 + \varepsilon^{(l)}$, where $\varepsilon^{(l)}$ is a learnable scalar parameter, as seen in GIN (Xu et al., 2019).

However, we found that using different learnable linear transformations for all possible isomorphism and positional types leads to much slower training. For example, if we had a number $|J_d|$ of positional types and a number $|I_t|$ of isomorphism types, this would lead to $|I_t| \cdot |J_d|$ possible combinations of linear transformations, resulting in longer training and higher generalisation gap due to the large number of parameters. To remedy this problem, we instead only use different weight matrices for positional types only, and inject the isomorphism type information as additional features to node embeddings. For example, we change Equation 14 to Equation 16 below

$$h_u^{(l+1)} = \sigma\left(h_u^{(l)} W^{(l)} + \sum_{(i,j)\in I_t \times J_d} \alpha_{j}^{(l)} \left(\sum_{S \in \mathcal{S}_u^{(l)}(i,j)} \sigma\left(\sum_{v \in S} (h_v^{(l)} \| e_i) W_{j}^{(l)} \right) \right)\right) \tag{16}$$

where e_i denotes a vector with entries all zero except one at the i-th component. Thus, we have a different pool function for subgraph embeddings but keep all other components the same:

$$\textsc{Pool}(S) = \sigma\left(\sum_{v \in S} (h_v^{(l)} \| e_i) W_{j}^{(l)} \right). \tag{17}$$

In the implementation, we apply a pre-processing step to compute all indices of induced neighbourhood subgraphs with respect to their isomorphism and positional types and store all these indices, which are used by G3N layers described in Equation 5. This pre-processing step is only required to perform once on each dataset.

For isomorphism types, we can compute them generally for any t-order subgraphs using the McKay's nauty algorithm (McKay, 1981; Mckay & Piperno, 2014). When $t \quad 3$, for efficiency, we consider a simple way to determine isomorphism types which count edges and nodes of subgraphs. In order to preserve structural information, initial colours of nodes are treated as being the same in the computation of isomorphism types.

For positional types, we implement f_{pos} as follows. Let S be an induced subgraph in the d-hop neighbourhood of a node u. Then $f_{pos}(S)$ is the set of shortest-path distances of the nodes in S to the node u, i.e., $f_{pos}(S) = \{\!\{\rho(u,v) | v \in V(S)\}\!\}$, where ρ refers to the shortest-path distance between nodes u and v. For example, if S has three nodes $V(S) = \{\!\{v_1, v_2, v_3\}\!\}$ with $\rho(u, v_1) = 1$, $\rho(u, v_2) = 1$ and $\rho(u, v_3) = 2$, then the positional type of S is $f_{pos}(S) = \{\!\{1, 1, 2\}\!\}$ in this case.

Remark D.1. *In our work, the definition of positional types for induced subgraphs (via the function f_{pos}) is intended to characterise two general conditions that are required at the minimum to establish the proposed strong hierarchy: (1) permutation invariant; (2) the condition specified by Equation 2. Thus, in general, there are many different ways to instantiate the function f_{pos} for positional types in implementation, as long as f_{pos} is permutation invariant and satisfies the condition specified in Equation 2.*

There are two model variants in our G3N implementation: one is to aggregate all connected-hereditary subgraphs of order up to t, corresponding to the node colouring function defined by Equation 4 as discussed in Section 3.2, and the other is to aggregate all t-order subgraphs regardless of connectivity, corresponding to the node colouring function defined by Equation 3. In our experiments, we consider the former one as the default model, unless otherwise stated.

E Experimental Details and Results

In the following, we provide the further information about the datasets, baseline methods, and parameter selection considered in our experiments as well as additional experimental results.

E.1 Datasets

Table 7 summarises the dataset tasks and the statistics of the datasets used in our experiments.

Table 7: Dataset statistics.

Dataset	Task type	#Classes	#Graphs	Avg. #nodes	Avg. #edges	Avg. diameter
graph8c	Isomorphism	11117	11117	8	28.8	2.7
EXP (iso ver.)	Isomorphism	1200	1200	44.4	110.2	7.1
SR25 (iso ver.)	Isomorphism	15	15	25	300	2
CSL (iso ver.)	Isomorphism	10	10	41	164	6
RandomGraph	Regression	4	5000	18.8	62.6	4.2
MUTAG	Classification	2	188	17.9	19.8	8.2
PTC-MR	Classification	2	344	14.3	14.7	7.5
PROTEINS	Classification	2	1113	39.0	72.8	11.3
NCI1	Classification	2	4110	29.9	32.3	11.5
IMDB-B	Classification	2	1000	19.8	96.5	1.9
IMDB-M	Classification	3	1500	13.0	65.9	1.5
ZINC	Regression	1	12000	23.1	49.8	12.5
MolHIV	Classification	2	41127	25.5	27.5	12.0
MolTOX21	Classification	2	7831	18.6	19.3	9.6

E.2 BASELINE METHODS

In our experiments on synthetic datasets, we compared the performance of G3N against the following baseline methods: MLP, GCN (Kipf & Welling, 2017), GAT (Velickovic et al., 2017), GIN (Xu et al., 2019), and PPGN (Maron et al., 2019a).

In our experiments for real-world datasets, we considered the following baseline methods:

- For the TU datasets, we compared G3N against (1) three kernel methods - RWK (Gärtner et al., 2003), WL-kernel (Shervashidze et al., 2011), and P-WL (Rieck et al., 2019); (2) five GNN models - PATCHY-SAN (Niepert et al., 2016), DCNN (Atwood & Towsley, 2016), DGCNN (Zhang et al., 2018), GIN (Xu et al., 2019), and PPGN (Maron et al., 2019a).
- For the ZINC and MolHIV datasets, we compared G3N against GCN (Kipf & Welling, 2017), PPGN (Maron et al., 2019a), GIN (Xu et al., 2019), PNA (Corso et al., 2020), DGN (Beaini et al., 2021), DEEP LPR (Chen et al., 2020), GSN (Bouritsas et al., 2022) and CIN (Bodnar et al., 2021a).

E.3 PARAMETER SELECTION

Synthetic datasets. We follow the experimental setup described by Balcilar et al. (2021).

- For the EXP, SR25, graph8c, and CSL datasets, we run 100 randomly initialised weights on the models and record pairs of graphs to be similar if the L1 distance of the length 10 graph representations is less than 0.001 on any of the runs. All models are constrained to a 30K parameter budget, consisting of 4 convolutional layers, sum readout, and one final linear layer.
- For the RandomGraph dataset, all models are also restricted to a 30K parameter budget, consisting of 4 convolutional layers, sum readout, and a further 2 fully connected layers trained for up to 200 iterations with a fixed learning rate of 0.001. Learning terminates when error goes below 10^{-4}.

TU datasets. Following the setup from Xu et al. (2019), model evaluation and selection are done by collecting the accuracy from the single epoch with the best cross-validation accuracy averaged over the 10 folds. We fix $t = 2, d = 2$, hidden units of 128 and learning rate of 0.001 which is halved every 50 steps. Table 8 summarises the hyperparameters used in our experiments for these datasets.

ZINC. We follow the setup described in Dwivedi et al. (2020) with batch size 128 and 1000 epochs with initial learning rate of 10^{-3} which is halved when validation does not improve over after 20 steps and training is halted when the learning rate goes under 10^{-5}. We fix $t = 2, d = 3$ and by adhering to the 100k parameter budget, we use 4 message passing layers with hidden size 80, followed by a final MLP readout.

MolHIV. We follow the train, validation and test split from Hu et al. (2020) and evaluate on the test score corresponding to the best validation score. We fix $t = 2, d = 3$, and set the batch size to be 128, hidden dimension 128, number of message passing layers 3, 100 epochs with a fixed learning rate 10^{-3} and dropout ratio 0.5.

MolTOX21. We follow the train, validation and test split from Hu et al. (2020) and evaluate on the test score corresponding to the best validation score. We set the batch size to be 64, hidden dimension 300, number of message passing layers 4, 100 epochs with a fixed learning rate 10^{-3} and dropout ratio 0.5.

Table 8: Hyperparameters for TU datasets.

Dataset	MUTAG	PTC_MR	PROTEINS	NCI1	IMDB-B	IMDB-M
#Layers	2	4	2	5	2	2
Drop ratio	0.1	0.5	0.75	0.1	0.5	0.5
batch size	64	32	64	32	64	64
#Epochs	200	300	200	200	200	200

E.4 ABLATION STUDY

Setup. We analyse the effect of t and d on the expressivity and generalisability of G3N. We run all different configurations of $t \in \{1, 2, 3\}$, $d \in \{1, 2, 3\}$ and number of layers $\in \{3, 4, 5\}$ on the datasets ZINC and MolTOX21. An initial learning rate is set to 10^{-3} and is halved when validation does not improve over 20 patience steps and training is halted when the learning rate goes under 10^{-5}. The results are reported in Table 9 and Table 10.

Table 9: Test MAE (average test score) and generalisation gap (test score - train score) on ZINC for different configurations of t and d. The results show that d is more important for performance on ZINC.

		Test MAE			Generalisation Gap		
#Layers		3	4	5	3	4	5
$t=1$	$d=1$	0.353±0.006	0.324±0.017	0.292±0.019	0.131±0.026	0.139±0.027	0.147±0.019
	$d=2$	0.305±0.020	0.258±0.012	0.240±0.008	0.161±0.013	0.142±0.018	0.140±0.007
	$d=3$	**0.187±0.004**	**0.191±0.007**	**0.194±0.006**	0.083±0.014	0.101±0.014	0.094±0.016
$t=2$	$d=1$	0.328±0.022	0.361±0.004	0.377±0.005	0.102±0.021	0.094±0.010	0.129±0.013
	$d=2$	0.337±0.008	0.282±0.019	0.258±0.021	0.132±0.024	0.148±0.008	0.133±0.014
	$d=3$	**0.182±0.003**	**0.176±0.005**	**0.185±0.008**	0.076±0.003	0.088±0.006	0.090±0.014
$t=3$	$d=1$	0.370±0.005	0.362±0.009	0.357±0.023	0.127±0.020	0.106±0.010	0.131±0.021
	$d=2$	0.341±0.011	0.328±0.018	0.312±0.028	0.133±0.019	0.130±0.017	0.119±0.010
	$d=3$	**0.176±0.007**	**0.175±0.007**	**0.182±0.010**	0.072±0.011	0.074±0.006	0.077±0.009

Table 10: Test ROC-AUC (average test score) and generalisation gap (test score - train score) on MolTOX21 for different configurations of t and d. The results show that t is more important for performance on MolTOX21.

		Test ROC-AUC (%)			Generalisation Gap		
#Layers		3	4	5	3	4	5
$d=1$	$t=1$	0.7363±0.006	0.7438±0.008	0.7398±0.009	0.1476±0.003	0.1303±0.013	0.1603±0.011
	$t=2$	0.7402±0.003	0.7565±0.006	0.7482±0.006	0.1339±0.002	0.1146±0.016	0.1350±0.006
	$t=3$	**0.7487±0.005**	**0.7649±0.007**	**0.7531±0.005**	0.1171±0.001	0.0734±0.009	0.0934±0.003
$d=2$	$t=1$	0.7365±0.013	0.7400±0.009	0.7553±0.011	0.1364±0.002	0.1390±0.008	0.1260±0.010
	$t=2$	0.7426±0.009	0.7514±0.008	0.7603±0.009	0.1352±0.014	0.0994±0.003	0.0965±0.002
	$t=3$	**0.7496±0.019**	**0.7636±0.009**	**0.7657±0.006**	0.1296±0.006	0.0773±0.016	0.0900±0.016
$d=3$	$t=1$	0.7443±0.017	0.7511±0.007	0.7494±0.001	0.1364±0.003	0.1264±0.003	0.1390±0.004
	$t=2$	0.7531±0.005	0.7593±0.018	0.7577±0.008	0.1203±0.002	0.0934±0.009	0.0996±0.007
	$t=3$	**0.7586±0.006**	**0.7695±0.007**	**0.7608±0.007**	0.1006±0.005	0.0656±0.013	0.0831±0.013

Observations. We observe that increasing the size of subgraphs t or the size of the receptive field d generally increase expressive power and reduces generalisation gap. This supports the theoretical results with the higher expressive power of $\mathcal{N}$-WL for higher t or higher d. One reason the generalisation gap is lower is because graph structure becomes more important for learning than node features with G3N which prevents overfitting. Further, according to the Table 9, a higher number of layers is required for expressivity for lower values of d and vice versa. This is because the total receptive field of each node with L layers of G3N is $d \quad L$, thereby requiring a balance of d and L to receive enough information from far away nodes in a graph. Moreover, according to Table 10, we can obtain higher expressive power by increasing the size t of subgraphs since this can lead to capturing more local structural information into node embeddings.

E.5 RUNTIME AND MEMORY ANALYSIS FOR t AND d

Setup. Here we present an empirical analysis of the runtime and memory usage of G3N as the neighbourhood size d and the order of subgraphs t increase. We measure resource usage by varying d and t while fixing all other hyperparameters. All model configurations include 4 message passing

Table 11: Runtime measured by seconds per epoch on ZINC (molecular dataset) and IMDB-B (social network dataset) with a parameter budget of approximately 30k. A message passing neural network (MPNN) is modelled by G3N with $d = t = 1$. The IMDB-B graphs have maximum diameter 2, meaning that $d = 2, 3$ use the same resources. OOM represents out of memory.

	ZINC		IMDB-B	
	runtime (s/epoch)	#parameters	runtime (s/epoch)	#parameters
MPNN	0.4	32607	0.1	33758
GNNML3	4.3	33309	0.4	34537
PPGN	3.7	32417	0.6	32009
G3N-(2,1)	0.6	31631	0.2	32938
G3N-(2,2)	1.1	31759	0.4	33018
G3N-(2,3)	1.6	31051	-	-
G3N-(3,1)	0.5	32911	1.6	33578
G3N-(3,2)	1.6	31601	OOM	OOM
G3N-(3,3)	2.5	32079	-	-

Table 12: Runtime and number of parameters for G3N on ZINC (molecular dataset) and IMDB-B (social network dataset) with 4 number of layers and hidden dimension of 64. The IMDB-B graphs have maximum diameter 2, meaning that $d = 2, 3$ use the same resources. OOM represents out of memory.

	ZINC		IMDB-B	
	runtime (s/epoch)	#parameters	runtime (s/epoch)	#parameters
MPNN	0.4	23619	0.1	53198
GNNML3	4.4	120077	0.4	122601
PPGN	3.9	91329	0.8	138753
G3N-(2,1)	0.7	32581	0.2	71122
G3N-(2,2)	1.2	50505	0.5	106970
G3N-(2,3)	1.7	68429	-	-
G3N-(3,1)	0.6	23875	1.7	72146
G3N-(3,2)	1.7	60747	OOM	OOM
G3N-(3,3)	2.5	97619	-	-

layers and a hidden dimension adhering to a 30k parameter budget for runtime analysis, and a hidden dimension of 64 for memory analysis. Furthermore, we run experiments on two different datasets ZINC and IMDB-B which have different settings and graph densities: ZINC consists of molecular graphs with average graph density 0.195 and diameter 12.47 which are sparser than the IMDB-B social networks with average graph density 0.520 and diameter 1.86, and max diameter 2. We further compare against GNN models with 3WL expressivity: GNNML3 (Balcilar et al., 2021) and PPGN (Maron et al., 2019a). The implementations are taken from Balcilar et al. (2021). The experiments for this section were run on an RTX 3090 GPU.

Observations. Viewing the results in Table 11, we notice that for sparse graphs the order of magnitude for runtime stays the same even when d and t are increased up to 3. On the other extreme when graphs are small and dense, runtime increases at a greater rate as d and t increase, given that the number of connected components in a neighbourhood grows large as can be seen with $d = 2$ on the IMDB-B dataset which results in global aggregation. For $t = 3$, this results in memory issues from computing all 3-order subgraph indices. Note that the runtime for $(t, d) = (2, 1)$ is greater than for $(t, d) = (3, 1)$. This is because all 3-order subgraphs in 1-hop neighbourhoods have the same positional type, whereas there are more positional types of 2-order subgraphs in the given datasets. We further note that runtime performance is comparable with the higher order GNN methods with the exception on the IMDB-B dataset where our model performs aggregation on whole graphs.

On the other hand in Table 12, we note that the parameter usage increases more moderately regardless of dataset structure, as the parameters are bounded by up to 5 times the parameters used in the message passing version of G3N. This is because the number of parameters scales with the number of isomorphism types, which are modest for connected components and small d and t.

We further note from both tables that the main computation effort arises from varying d and t and not the number of parameters of the model as expected.

E.6 Runtime Analysis for Connected Variants

Table 13: Runtime measured by seconds per epoch and MAE on ZINC with a parameter budget of approximately 100k on G3N with aggregation variants: connected t-subgraphs only, connected-hereditary subgraphs, and all t-subgraph.

	connected	connected-hereditary	all
runtime(s/epoch)	2.5	4.2	5.5
MAE	0.165±0.006	0.174±0.002	0.226±0.009

Setup. We further present an empirical analysis of the runtime and performance of G3N on the ZINC dataset with various aggregation variants depending on which subgraph types it aggregates. Specifically, we consider the original G3N which aggregates only connected t-order subgraphs, G3N which aggregates connected-hereditary subgraphs discussed in Section 3.2, and G3N which aggregates all t-order subgraphs regardless of connectivity. The message G3N layer equations given in Equation 14 and Equation 16 remain unchanged for all connectivity variants with the except of the definition of $\mathsf{S}_u^{(l)}(i,j)$ which consists of the set of all induced subgraphs from the neighbourhood of a node u being restricted to connectivity variants discussed above.

The parameter setup for all three variants consists of the same setup with the ZINC experiments above. Specifically, we have a batch size 128 and an initial learning rate of 10^{-3} which is halved when validation does not improve over after 25 steps and training is halted when the learning rate goes under 10^{-5}. We fix $t = 3, d = 3$ with 2 message passing layers and fix the number of hidden units such that we adhere to a 100k parameter budget. We use an MLP readout. The experiments for this section were run on an RTX 3090 GPU.

Observations. From Table 13, we observe that the runtime of all the G3N variants have the same order of magnitude. By considering connected-hereditary subgraphs, we achieve lower runtime as there are fewer isomorphism types to consider for the given dataset. Note however, that this does not hold in general and that there exist graphs where considering connected-hereditary subgraphs is less efficient, such as with cliques. We further note that the connected-hereditary variant provides better performance which may be attributed to the fact that by considering all possible isomorphism types regardless of connectivity, we may lose information in the training process as to which structures are actually helpful for prediction. This is given by the intuition that connected components should have a greater contribution for predictions. Finally, we note that considering connected t-order subgraphs only as opposed to connected-hereditary subgraphs (all k-order subgraphs for $1 \quad k \quad t$) obviously provides faster runtime but more significantly provides similar performance. This may be attributed to the fact that larger t-order subgraphs in the given dataset generally subsume smaller subgraphs in providing structural information for prediction.

E.7 Complexity Analysis - k-WL vs $\mathscr{N}$-WL

Setup. We conduct an experiment to empirically analyse the time complexity of our $\mathscr{N}$-WL algorithms, in comparison with the classical k-WL algorithms. In the experiment, four datasets ZINC, MolHIV, NCI1, and IMDB-B are selected, which have varying graph structural information, such as the sparsity of a graph, the diameter of a graph, etc. For each dataset, we randomly select 10 graphs.

We compute the average complexity of k-WL and $\mathscr{N}$-WL as follows: (1) for k-WL, we compute $n^t \quad (t \quad n)$ where $t = k$. This is because k-WL considers the number n^k of k-tuples in a graph G with $|V(G)| = n$ and there are further $(k \quad n)$ neighbouring k-tuples considered for colouring each of such k-tuples. (2) for $\mathscr{N}$-WL, we compute $n \quad \binom{d}{t}$ where n refers to the number of nodes to be

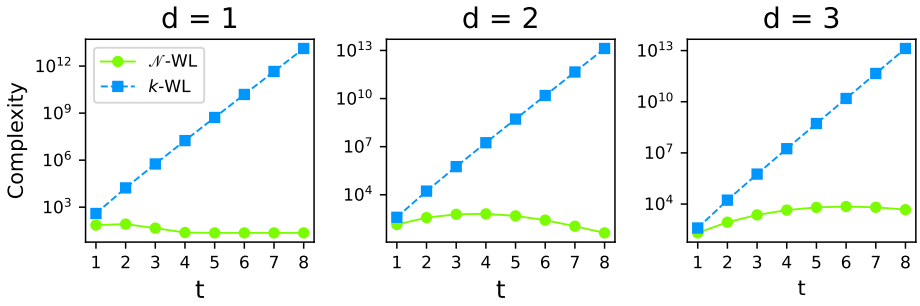


Figure 10: ZINC

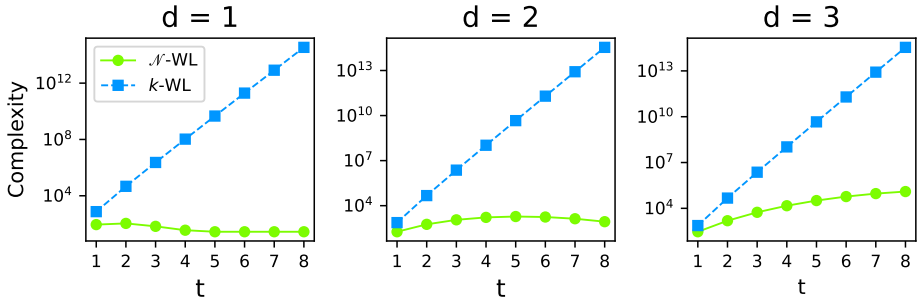


Figure 11: MolHIV

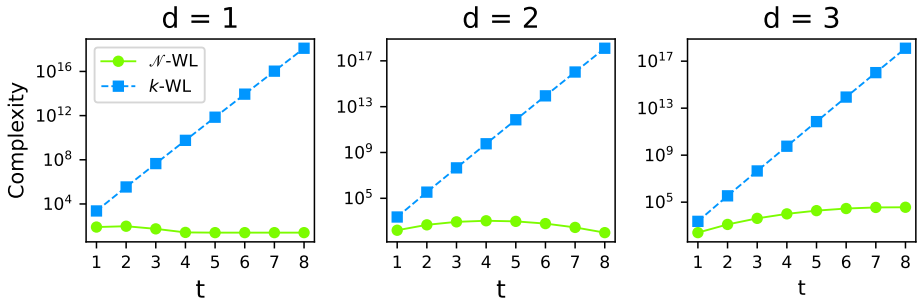


Figure 12: NCI1

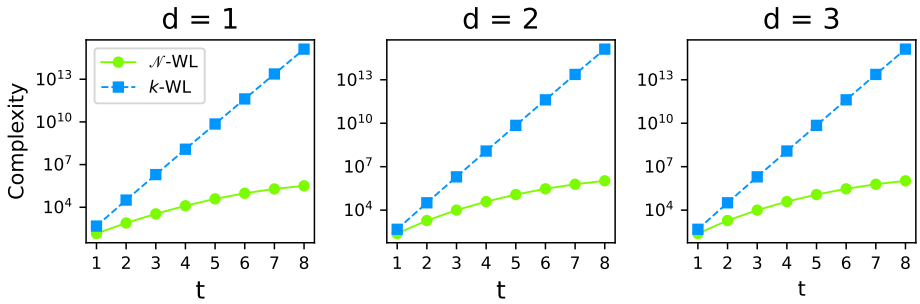


Figure 13: IMDB-B

coloured in a graph and $\binom{a^d}{t}$ approximates the number of neighbouring subgraphs of order t within the local d-hop neighbourhood of a node. The results are reported in Figures 10, 11, 12 and 13.

Observations. From these figures, we observe the following:

- When $d = 1$, for graphs with relatively large diameters such as ZINC, MolHIV, and NCI1, as shown in Figures 10, 11, and 12, our $\mathscr{N}$-WL algorithm is several orders of magnitude faster than the k-WL algorithm. We can see that the performance gap between k-WL and $\mathscr{N}$-WL becomes more and more significant when t increases and becomes stable after $t = 4$. This is because the number of 1-hop neighbours a^1 is much smaller than the total number of nodes n in a graph (i.e., $a^1 \quad n$). For graphs with very small diameters such as IMDB-B, all the nodes are almost within 2-hop of a node; therefore, the number of 1-hop neighbours a^1 is relatively closer to the number of nodes n in such graphs. In this case, as depicted in Figure 13, the performance gap between k-WL and $\mathscr{N}$-WL becomes small.
- When $d = 2$, for graphs with relatively large diameters such as ZINC, MolHIV, and NCI1, as shown in Figures 10, 11, and 12, $\mathscr{N}$-WL still shows a significant performance improvement over k-WL and the performance gap starts to increase after $t = 4$. The increase in the complexity of $\mathscr{N}$-WL for $t = [1, 2, 3]$ is due to the effect of including 2-hop neighbours along with 1-hop neighbours; after $t = 4$, the size of 2-hop neighbours $|a^2|$ and parameter t start to approach each other which results in an increased performance gap. For IMDB-B dataset with a very small diameter, as shown in Figure 13, almost all the nodes in the graphs are within 2-hop (i.e., $a^2 \quad n$) and the time complexity of our algorithm increases for initial values of t and then start to stabilizes or decreases as compared to k-WL.
- When $d = 3$, the value of a^d gradually approaches the total number of nodes in a graph. In such cases, $\mathscr{N}$-WL remains significantly faster on graphs with relatively large diameters, as shown in Figures 10, 11, and 12, and also shows to outperform well for graphs with small diameters with k-WL, as shown in Figure 13. The performance gap still becomes larger with the increased values of parameter t.

Overall, due to the local nature of the $\mathscr{N}$-WL algorithm in contrast to the global nature of the k-WL algorithm, the $\mathscr{N}$-WL algorithm is much more efficient than the k-WL algorithm when performing on graphs with reasonably large diameters, particularly if diameters are greater than d values. The performance gap between the $\mathscr{N}$-WL algorithm and the k-WL algorithm is reduced when increasing d or graphs have very small diameters, e.g., IMDB-B has an average diameter 1.9.

Remark E.1. *The complexity of $\mathscr{N}$-WL can be reduced when the gap between t and a^d gets smaller. This is because the number of induced subgraphs $\binom{a^d}{t}$ may decrease in this case. For instance, we can see from Figure 10 that when $d = 1$ and $t = 4$, the complexity is lower than the one at $d = 1$ and $t = 3$; then it stabilises when $d = 1$ and $t \quad 5$. Similar cases can be observed from Figure 11 and Figure 12.*

F Limitations and Future Work

In this work, the design of G3N still has some limitations. Firstly, as with most expressive GNN models going beyond 1-WL, G3N suffers from inevitably increased runtime per training step and parameter complexity. Although seen to be feasible for sparse molecular graphs, this may still pose an issue with much larger datasets. To solve this issue, one may have to consider sampling methods. Another issue is with an unclear explanation on the gap between expressivity and generalisability. Finally, although G3N is able to detect graph substructures contrary to methods which precompute and inject them as additional features for learning, the parameters d and t are still domain dependent and will have to be hand chosen to balance computational resources and performance. Due to these limitations, one interesting future research direction is to explore the underlying design principles of expressive GNN models for a better understanding of the interaction between expressivity, generalisability, explanability, and training efficiency of GNN models.

From the algorithmic perspective, for both k-WL and $\mathscr{N}$-WL, a possible future research direction is to analyse their connections to node-level, link-level, or more generally subgraph-level properties and accordingly build connections between subgraph properties and expressivity of GNN models. In the literature of the k-WL algorithms, despite their fruitful connections to logic and descriptive

complexity theory, there is no clear understanding for characterisations of subgraph patterns whose counts and occurrence are higher-order k-WL invariant (Kiefer, 2020). In fact, traditionally, the k-WL algorithm is mainly used as a combinatorial tool in graph isomorphism tests and little work has been done to study the applicability of k-WL to recognition of graph properties rather than to testing isomorphism (Fuhlbrück et al., 2021). Further, a study on the formal properties of the $\mathscr{N}$-WL algorithms by exploring the connections between the $\mathscr{N}$-WL hierarchy and logic is needed.