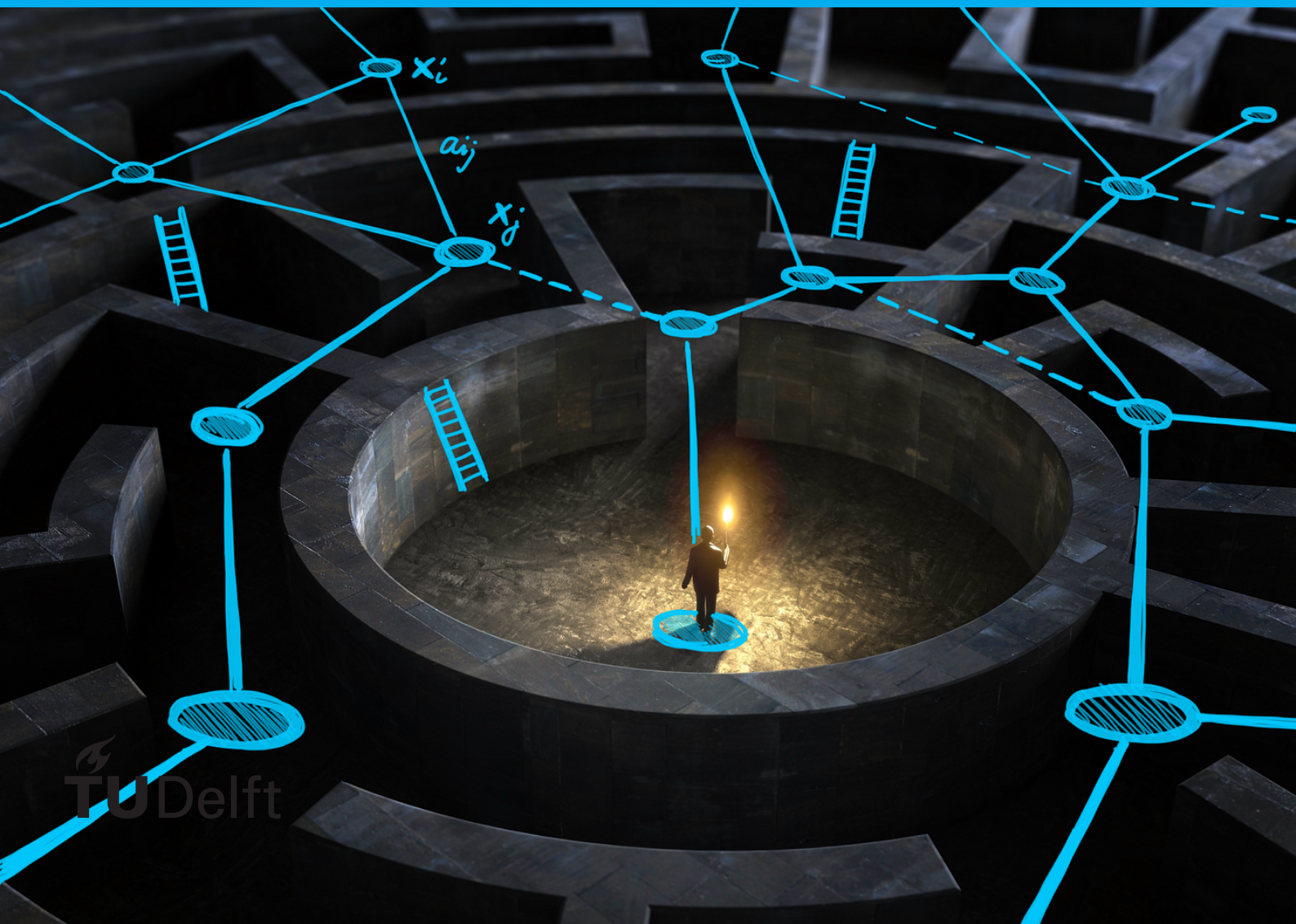
# The Expressive Power of (Multi)Set-based Higher-order Graph Neural Networks

A. Vasileiou




TU Delft

# The Expressive Power of (Multi)Set-based Higher-order Graph Neural Networks

by

A. Vasileiou

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Thursday August 31, 2023 at 13:00.

| | | |
|---|---|---|
| Student number: | 5608414 | |
| Project duration: | January 1, 2023 – August 31, 2023 | |
| Thesis committee: | Prof. Dr. C. Morris, | RWTH University, supervisor |
| | Prof. Dr. J. Söhl, | TU Delft, supervisor |
| | Prof. Dr. D. Kurowicka, | TU Delft |

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

TU Delft Delft University of Technology

# Preface

I want to express my gratitude to all those who supported me throughout this thesis journey. A special thanks goes to Pr. Christopher Morris and Pr. Jakob Söhl for their valuable feedback and assistance. Their contributions have greatly enriched this work, as well as the input from everyone else who provided feedback.

*A. Vasileiou*
*Delft, August 2023*

# Abstract

Graph data is prevalent across various applications, leading to the rapid advancement of graph-based machine learning techniques. However, traditional machine learning algorithms designed for graphs have limitations in capturing complex relationships between nodes and higher-order patterns. Recent findings by Morris et al. [2019] and Chen et al. [2019] have provided valuable insights into the comparison of graph neural network architectures. In their work, Morris et al. [2020b] proposed higher order neural networks that can capture more complex patterns in graph data. However, these models suffer from scalability issues, making them challenging to apply to real-world datasets. In this thesis, we build upon the aforementioned work by conducting a theoretical comparison of the different neural architectures proposed in those studies. Additionally, we propose novel models that aim to strike a balance between capturing higher order patterns and maintaining scalability. Our goal is to improve the modeling capabilities of graph-based algorithms and overcome the limitations of existing methods. Furthermore, we implemented our proposed models on well-known benchmark datasets to evaluate their performance. The results confirmed that our models exhibit significantly better generalization performance compared to traditional graph neural networks. Additionally, we found that our models demonstrate significant improvements in scalability when compared to other higher order graph neural networks. This research contributes to the field of graph machine learning by providing more effective and scalable solutions for capturing higher order patterns in graph data.

# Contents

# 1

# Introduction

Graph-structured data is prevalent across diverse application domains, including molecular structure and bioinformatics [Barabasi and Oltvai, 2004, Stokes et al., 2020, Jumper et al., 2021, Rittig et al., 2022, Schweidtmann et al., 2023], image analysis [Simonovsky and Komodakis, 2017], social network analysis [Easley and Kleinberg, 2010], pharmaceutical drug discovery [Gaudelet et al., 2021], combinatorial optimization [Cappart et al., 2021, Bengio et al., 2021] and material science [Reiser et al., 2022]. Successful machine learning models in these domains require techniques that can effectively manage the information inherent in graph structures, as well as the feature information associated with nodes and edges. Various approaches have been proposed for graph-based machine learning tasks, with notable ones being graph kernels [Morris, 2019, Borgwardt et al., 2020, Morris et al., 2021a] and graph neural networks (GNNs) [Gilmer et al., 2017, Chami et al., 2020, Morris et al., 2021b, Wu et al., 2022]. Graph kernels employ a two-step feature extraction and learning approach, computing a vectorial representation or a positive semi-definite kernel matrix that captures predefined graph features, which are subsequently fed into a learning algorithm such as support vector machines [Hastie et al., 2009]. On the other hand, GNNs are specialized neural architectures designed to operate on graph-structured data. GNNs iteratively update node representations by aggregating the representations of neighboring nodes and their own previous representations. These learned node representations can be further combined with pooling techniques to extract a comprehensive graph representation vector which then is fed to an artificial neural network (ANN).

In this work, our primary objective is to develop novel Graph Neural Network architectures that strike a balance between capturing and leveraging a greater amount of structural information within graphs compared to traditional GNNs architectures, while maintaining computational efficiency. By achieving this equilibrium, our proposed architectures facilitate scalable and efficient graph-based learning, making them well-suited for practical applications in real-world scenarios. To achieve our goals, we focus on the development of GNN architectures that provably offer enhanced expressiveness. These architectures are implemented and evaluated on well-known benchmark datasets [Morris et al., 2020a]. Our experiments focus on graph-based classification tasks that involve two classes. Nevertheless, it is crucial to highlight that the insights and findings derived from our research possess broad applicability and can be easily extended to address either multiclass classification, graph-based regression, or even

node-based classification/regression tasks. This is primarily due to the fact that our developments primarily affect the transformation and the message passing phases of a GNN and not the readout phase, as we will see in Section 4.5.1.

## 1.1. Related work

In this work, two main notions are often referred to: the graph isomorphism problem, which inquires whether two provided graphs have the same underlying structure, and the expressive power of GNNs, which denotes their ability to learn and represent complex graph structures, capturing intricate patterns and relationships within the data they process. The relationship between the graph isomorphism problem and the expressiveness of GNNs has been widely explored by Xu et al. [2019a], Maron et al. [2019a], Maehara and NT [2019], Azizian and Lelarge [2020], Barceló et al. [2020], Geerts [2020], Geerts and Reutter [2022]. The Weisfeiler-Lehman (WL) algorithm, which is a heuristic for the graph isomorphism problem, has been effective in capturing structural patterns in graphs and providing a sufficient condition for distinguishing non-isomorphic graphs. Moreover WL is powerful enough to distinguish almost all pairs of non-isomorphic graphs except from rare counterexamples [Babai and Kucera, 1979, Cai et al., 1992]. In recent research, Morris et al. [2019] established an equivalence between GNN architectures [Gilmer et al., 2017, Scarselli et al., 2009] based on the WL algorithm and the WL heuristic. This finding indicates that these GNN models can only provide distinct outputs for non-isomorphic graphs if the WL algorithm itself distinguishes between them. Additionally, a significant contribution by Chen et al. [2019] reveals a connection between the GNN's ability to discriminate between pairs of non-isomorphic graphs and its capability to approximate arbitrary permutation invariant functions on graphs. These crucial findings regarding the equivalence between GNN architectures based on the WL algorithm and the WL heuristic, and their ability to approximate permutation invariant functions have shed light on the expressive power of GNNs. However, these advancements have also raised awareness of the limitations inherent in the WL algorithm [Kiefer, 2020]. As a result, researchers have been interested in dive deeper into the existing GNN architecture and their constraints and shortcomings in effectively capturing the diverse structural patterns present in data (Chen et al. [2020], Loukas [2020]).

This exploration into the limitations of existing GNNs has sparked the development of novel neural architectures that aim to overcome the shortcomings of the WL algorithm. Researchers have actively pursued innovative approaches to capture higher-order patterns and enhance the expressive power of GNNs. Morris et al. [2020b] introduced higher-order GNN variants , based on an extension of the WL algorithm called the $k$-WL algorithm Babai and Kucera [1979]. These higher-order GNN architectures can capture more global structures of graphs and have been proven to be more expressive than their first-order counterparts. Furthermore, to incorporate graph sparsity into models, Morris et al. [2022] proposed sparsity-aware neural networks. These innovative architectures are specifically designed to accommodate and leverage the sparsity inherent in graphs. By considering the sparse nature of graphs, these models can effectively reduce computational costs while maintaining their expressive power. Other works that have made significant contributions to the development of more expressive GNN architectures are subgraph aggregation networks or approaches utilizing random features [Murphy et al., 2019, Dasoulas et al., 2020, Vignac et al., 2020, NT and Maehara, 2020, Abboud et al., 2021, Sato et al., 2021, Barceló et al., 2021, Balcilar et al., 2021, Bodnar et al., 2021, Tönshoff et al., 2021, Talak et al., 2021, Beaini et al., 2021, Qian et al., 2022, Horn et al., 2022, Bouritsas et al., 2023]. In

addition, recent research has explored the integration of tools from topology to leverage higher-order information in networks ([Bick et al., 2021], [Horn et al., 2022], [Hajij et al., 2023]). By incorporating concepts and techniques from algebraic topology, such as simplicial and cellular complexes, researchers have been able to capture and analyze higher-order structural patterns in network data.

However, despite the promising advancements in capturing higher-order patterns, many of the aforementioned designs suffer from scalability issues due to their increased computational complexity during training. One such challenge is evident in the architectures proposed by Morris et al. [2020b], where the computation of all possible $k$-tuples in the set of graph nodes becomes necessary. This leads to a substantial memory requirement of $n^k$ nodes for a graph with $n$ nodes, posing significant limitations on the scalability of these models. Therefore, while these approaches offer valuable insights into capturing intricate structural information, their computational demands hinder their practical applicability, particularly for large-scale datasets. To address these limitations, Morris et al. [2019] developed a set-based variant of the WL algorithm and a corresponding neural architecture. This set-based variant considers sets of nodes instead of individual nodes, reducing computational complexity while still capturing important structural information. By extending this set-based WL variant, our work in this thesis aims to enhance the expressiveness of GNN architectures while maintaining computational efficiency.

## 1.2. Present work

In this section, we outline the key contributions and findings of our work. To address the described drawbacks, our work introduces multiple heuristics for the graph isomorphism problem, each designed to detect higher-order patterns within a graph. These heuristics are inspired by the ones described in Morris et al. [2020b], but with a key distinction: we consider both sets and multisets of nodes instead of tuples. By disregarding the ordering of nodes, our heuristics offer computational efficiency while maintaining a high level of expressiveness.

In addition to introducing new architectures based on sets and multisets, we also undertake a thorough examination of the mathematical properties of both existing and proposed algorithms. Our investigation begins with the $\delta$-$k$-LWL algorithm, which is a slight modification of the WL algorithm. We delve into its behavior in relation to the parameter $k$ and establish a clear hierarchy of expressiveness within the algorithm. As the value of $k$ increases, the algorithm becomes increasingly powerful, enabling it to capture and represent more complex graph patterns. Furthermore, we conduct a detailed analysis and comparison of our (multi)set-based variants of the $k$-WL algorithm. Through rigorous mathematical proofs, we demonstrate that considering sets alone cannot result in more expressive algorithms than considering multisets. Similarly, considering multisets alone cannot surpass the expressiveness achieved by employing ordered tuples. Moreover, we explore the impact of discriminating between local and global neighbors, as outlined by Morris et al. [2020b], and observe that this discrimination yields strictly more expressive algorithms.

Finally, we validate the performance and effectiveness of our proposed models by conducting experiments on well-known benchmark datasets. By applying the developed architectures to these datasets, we assess their capabilities and provide empirical evidence of their effectiveness in solving graph isomorphism problems. Additionally, we pose and address basic questions, paving the way for further research in this field. In summary, our work introduces novel architectures based on sets and multi-

sets, explores the mathematical properties of existing and proposed algorithms, establishes a hierarchy of expressiveness, highlights the superiority of multiset-based models, develops corresponding neural architectures, and validates their performance through experiments on benchmark datasets.

## 1.3. Document structure

Section 2, (Preliminaries), serves as a foundation by introducing the necessary notation, presenting basic concepts related to graph isomorphism, and discussing existing approaches for solving the problem. Building upon this groundwork, Section 3, (Weisfeiler-Leman Refinement Algorithms), delves into the details of the Weisfeiler-Leman algorithm, its higher-order variants, and their theoretical properties. This section also includes a comparative analysis of these algorithms, establishes a hierarchy result for the local version ($\delta$-$k$-LWL), and explores the development of neural architectures based on these algorithms, demonstrating their equivalence in terms of distinguishing non-isomorphic graphs. In Section 4, titled ((Multi)Set-Based Refinement Algorithms), attention is shifted towards the proposed (multi)set-based WL algorithms. This section involves a comprehensive comparison of these algorithms, considering their expressiveness and computational efficiency. Furthermore, new neural architectures based on these algorithms are developed and their equivalence with the corresponding algorithms is established. Section 5 focuses on experimental evaluation, where the proposed neural architectures are applied to well-known benchmark datasets. This section provides detailed insights into the experimental setup, implementation details, and configuration choices. The obtained results are analyzed, compared, and aligned with the main research questions posed throughout the thesis. Finally, Section 6 serves as the concluding section, summarizing the main findings, contributions, and implications of the research. It also outlines potential future research directions, highlighting areas that can be further explored to advance the field of graph isomorphism and neural architectures.

# 2

# Preliminaries

In this section, we provide an overview of the key notation and mathematical concepts used throughout the work. In addition, we discuss the graph isomorphism problem, which is a fundamental problem in computer science. This problem has important applications in areas such as chemistry, computer vision, and cryptography [Baird and Cho, 1975, Grohe and Schweitzer, 2020, Merkys et al., 2023]. Finally we present the work so far on the computational cost of the graph isomorphism problem, as well as different approaches that have been proposed to solve it.

## 2.1. Basic notation

As usual, let $[n] = \{1, \dots, n\} \subset \mathbb{N}$ for $n \geq 1$. We further define the concept of a multiset as a generalization of that of a set. A multiset is a collection of unordered elements, where every element $x$ occurs a finite number of times. The number of times that each element occurs is called the multiplicity of the element. We denote multisets using double curly braces, as $\{\{x_1, x_2, \dots, x_n\}\}$, where $x_1, x_2, \dots, x_n$ are the elements of the multiset. We say that a multiset $M$ is strictly multiset if M is not a set. For a multiset $M$, we denote $|M|$ as the cardinality of the multiset, which is the sum of the multiplicities of its elements. Moreover, for an arbitary set $\mathcal{X}$ we call a partition $\rho$ of $\mathcal{X}$ a grouping of its elements into non-empty subsets, in such a way that every element is included in exactly one subset. That is, $\rho$ is a family of subsets of $\mathcal{X}$ such that $\emptyset \notin \rho$, $\bigcup_{A \in \rho} A = \mathcal{X}$ and for all $A, B \in \rho$ with $A \neq B$ we have that $A \cap B = \emptyset$. A partition $\alpha$ of a set $\mathcal{X}$ is a refinement of a partition $\rho$ of $\mathcal{X}$ and we say that $\alpha$ is finer than $\rho$ and that $\rho$ is coarser than $\alpha$ if every element of $\alpha$ is a subset of some element of $\rho$. Informally, this means that $\alpha$ is a further fragmentation of $\rho$. In that case, it is written that $\alpha \leq \rho$. Now, as usual for a set $\mathcal{X}$ and $k \in \mathbb{N}$ we denote by $\mathcal{X}^k$ the Cartesian product $X \times \cdots \times X$ and the elements of $\mathcal{X}^k$ are called tuples. In addition we denote the set of all $k$-element subsets of $\mathcal{X}$ as $[\mathcal{X}]^k$, that is $[\mathcal{X}]^k = \{U \subset \mathcal{X} \mid |U| = k\}$. Similarly $[[\mathcal{X}]]^k$ is the set of all $k$-element multisets with elements from $\mathcal{X}$, i.e. $[[\mathcal{X}]]^k = \{U \textit{ multiset} \mid |U| = k \textit{ and } \forall u \in U\ u \in \mathcal{X}\}$. For a tuple $\mathbf{u} = (u_1, \dots, u_k) \in \mathcal{X}^k$ we write *multiset*($\mathbf{u}$) to describe the multiset $\{\{u_1, \dots, u_k\}\}$. Let $\mathbf{u}$ be a $k$-tuple in $\mathcal{X}^k$, i.e, $\mathbf{u} = (u_1, \dots, u_k)$. Then, for $j \in [k]$ and $\omega \in \mathcal{X}$ we denote by $\phi_j(\mathbf{u}, \omega)$ the $k$-tuple obtained by replacing the $j$-th component of $\mathbf{u}$ with $\omega$, i.e. $\phi_j(\mathbf{u}, \omega) = (u_1, \dots, u_{j-i}, \omega, u_{j+1}, \dots, u_k)$. Moreover for a function $f : \mathcal{X}^k \to \Sigma$ and for $\mathbf{u} \in \mathcal{X}^k, w \in \mathcal{X}$ we define the *sift* operator as *sift*$(f, \mathbf{u}, w) = (f(\phi_1(\mathbf{u}, w)), \cdots, f(\phi_k(\mathbf{u}, w))$, where $\Sigma$ is an arbitary codomain for the function $f$.

## 2.2. Graph theory

An undirected graph $G$ is a pair $(V, E)$ with a finite set of vertices $V$ and a set of edges $E \subseteq \{\{u, v\} \subseteq V \mid u \neq v\}$. We denote the set of vertices and the set of edges of $G$ by $V(G)$ and $E(G)$, respectively. The graph structure is characterized by the adjacency matrix $A \in \{0,1\}^{n \times n}$, where $A_{i,j} = 1$ if and only if nodes $i$ and $j$ are connected by an edge. Moreover, $N(v)$ denotes the neighborhood of $v$ in $V(G)$, i.e., $N(v) = \{u \in V(G) \mid \{v, u\} \in E(G)\}$. In the case of directed graphs, $E \subseteq \{(u, v) \in V \times V \mid u \neq v\}$. For the sake of clarity, in this thesis we use the term graph to refer to undirected graphs. When referring to directed graphs, we will make this distinction explicit. A vertex coloring is a function $C \colon V(G) \to \mathbb{S}$, with arbitary codomain $\mathbb{S}$. A (vertex) labeled graph $(G, l)$ is a graph $G$ endowed with a vertex coloring $l \colon V(G) \to \Sigma$, where $\Sigma$ is some finite alphabet. We call $l$ the vertex labeling function of graph $G$. If we replace $\Sigma$ with $\mathbb{R}^d$, we say $l$ is a vertex encoding, and $G$ is a continuously labeled graph. We say that $l(u)$ is a label (or encoding) of $u$ for $u$ in $V(G)$. In addition, we say that a vertex encoding $f \colon V(G) \to \mathbb{R}^d$ is consistent with a vertex labeling $l \colon V(G) \to \Sigma$, if for every pair of vertices $u, v \in V(G)$ we have that $f(u) = f(v)$ if and only if $l(u) = l(v)$. Edge labeled and continuously labeled graphs are defined analogously. We say that a vertex coloring $c$ refines a vertex coloring $d$, written $c \sqsubseteq d$, if $c(v) = c(w)$ implies $d(v) = d(w)$ for every $v$ and $w$ in $V(G)$. If $c \sqsubseteq d$ and $d \sqsubseteq c$, we say that the two colorings are equivalent and we write $c \equiv d$. Finally, we say that $c$ strictly refines $d$ and we write $c \sqsubset d$ if $c \sqsubseteq d$ and $c$, $d$ are not equivalent. A color class $S_c \subseteq V(G)$ of a vertex coloring $c$ is the maximal set of vertices with $c(v) = c(w)$ for every $v$ and $w$ in $S_c$. For a graph $G$, let $S \subseteq V(G)$ then $G[S] = (S, E_S)$ is the subgraph induced by $S$ with $E_S = \{(u, v) \in E(G) \mid u, v \in S\}$. In addition given a tuple $\mathbf{v} = (v_1, \ldots, v_k) \in V(G)^k$ let $G[\mathbf{v}]$ denote the subgraph induced on the set $\{v_1, \ldots, v_k\}$, where, the vertex $v_i$ is labeled with $i$, for $i$ in $[k]$. Moreover, given a multiset $M$ in $[[V(G)]]^k$ let $G[M]$ denote the subgraph induced by the set of all distinct elements of $M$, where, each vertex is labeled with its multiplicity in the multiset $M$. A graph is said to be connected if every pair of vertices in the graph is connected. This means that there is a path (sequence of edges) between every pair of vertices. A graph $G$ is disconnected if $G$ is not connected. We say that a graph $G$ is a tree if $G$ is connected but $G$ would become disconnected if any single edge is removed from $G$. A rooted tree is a directed tree with a designated vertex called root in which the edges are directed in such a way that they point away from the root. Let $p$ be a vertex in a directed tree then we call its out-neighbors children with parent $p$.

## 2.3. Graph isomorphism problem

We say that two graphs $G$ and $H$ are isomorphic if there exists an edge preserving bijection $\phi \colon V(G) \to V(H)$, i.e., $(u, v) \in E(G)$ if and only if $(\phi(u), \phi(v)) \in E(H)$. If $G$ and $H$ are isomorphic, we write $G \cong H$ and call $\phi$ an isomorphism between $G$ and $H$. Moreover, we call the equivalence classes induced by $\cong$ isomorphism types, and denote the isomorphism type of $G$ by $\tau_G$. In the case of vertex labeled graphs, we additionally require that $l(v) = l(\phi(v))$ for $v \in V(G)$ and for the edge labeled graphs $l((u, v)) = l((\phi(u), \phi(v)))$ for $(u, v) \in E(G)$.

The graph isomorphism problem (GI) is a computational problem that involves determining whether two graphs are isomorphic. The computational complexity of GI is widely recognized as a major unresolved question in computer science, drawing significant research attention. First introduced as an open problem by Karp in his influential work on the NP-completeness of combinatorial problems [Book, 1975], GI continues to be a notable example of a natural problem in NP that has not been definitively

classified as either NP-complete or polynomial-time solvable. However, Babai [2016] made a breakthrough by showing that GI can be solved in quasi-polynomial time, improving upon previous bounds. Research on the Graph Isomorphism Problem (GI) began in the 1960s, with the proposal of heuristic approaches to solve it. For more information on the complexity of the Graph Isomorphism (GI) problem and recent advancements in this field, you can refer to the paper by Grohe and Neuen [2021]. In the following section, we will explore the heuristics introduced by Weisfeiler and Leman [1968] in detail, examining their principles and techniques employed for tackling the GI problem.

# 3

# Weisfeiler-Leman refinement algorithms

This section aims to present an overview of the Weisfeiler-Leman algorithm [Weisfeiler and Leman, 1968], which is a well-known heuristic for the graph isomorphism problem. The basic version of the algorithm, called 1-WL algorithm operates by iteratively coloring the vertices of graph based on the neighborhood of each vertex. In each iteration the different colors define a partition on the set of vertices and for that reason the 1-WL is often reffered as a vertex classification algorithm. The idea behind WL is to give different colors to two vertices whenever it is obvious that neither of them can be mapped to the other one by an isomorphism. As it will be discussed, although 1-WL is a quite powerful tool in distinguishing non isomorphic graphs and has been applied in many areas (e.g. molecule classification and social network analysis [Milo et al., 2002, Clauset et al., 2011], there are several examples of graphs with practical importance where the 1-WL fails to capture the desired structure of graphs. Intuitively, the reason behind this weakness of 1-WL it is its purely local nature. To address these issues, we describe the $k$-WL algorithm analyzed by Cai et al. [1992], Segoufin [2017], Maron et al. [2019a], Grohe [2021], and Kiefer and Schweitzer [2019] which colors tuples of vertices instead of individual vertices and can capture more global, higher-order patterns. Additionally to $k$-WL, Malkin [2014] and Morris et al. [2020b] introduced several variants of the algorithm that differ in the information that is aggregated in each iteration. Finally, we compare the different versions of the WL algorithm, highlighting their strengths and limitations.

## 3.1. Weisfeiler-Leman algorithm (1-WL)

We now describe the 1-WL or Naive vertex classification algorithm for labeled graphs introduced by Weisfeiler and Leman [1968]. Let $(G, l)$ be a labeled graph. In each iteration, $t \geq 0$, the 1-WL computes a vertex coloring $c_l^{(t)} : V(G) \to \Sigma$, which depends on the coloring from the previous iteration. In iteration $0$, we set $c_l^{(0)} \equiv l$. Now for $t > 0$, we set

$$c_l^{(t)}(v) = \mathit{HASH}\left(\left(c_l^{(t-1)}(v), \{\!\{c_l^{(t-1)}(u) \mid u \in N(v)\}\!\}\right)\right)$$

where *HASH* bijectively maps the above pair to a unique value in $\Sigma$, which has not been used on previous iterations. Note that in iteration $t+1$ the color classes of the vertex coloring, induce a partition in the set of vertices which is finer than the one induced in iteration $t$. Therefore, since $V(G)$ is finite, the

termination of the algorithms is guaranteed after at most $|V(G)|$ iterations. In the end, the last vertex coloring function is called the stable coloring of 1-WL and denoted by $c_l^{(\infty)}$. Now, to test if two graphs $G$ and $H$ are isomorphic, we run the above algorithm in parallel on both graphs. If for an iteration $t$ the two graphs have a different number of nodes colored $\sigma$ in $\Sigma$ , the 1-WL concludes that the graphs are not isomorphic. In Figure 3.1 we see two non-isomorphic labeled graphs with the same number of nodes, edges and the same initial color histograms that can be distinguished in the first iteration of the WL algorithm.

**Note** that we have defined the WL algorithm in the context of labeled graphs. In cases where the given graph is unlabeled, we assume it to be a labeled graph with all nodes sharing the same color. Additionally, it is important to observe that in that case, during the first iteration of the algorithm, nodes with the same degree are assigned the same color. A graph G is considered to be identified by the

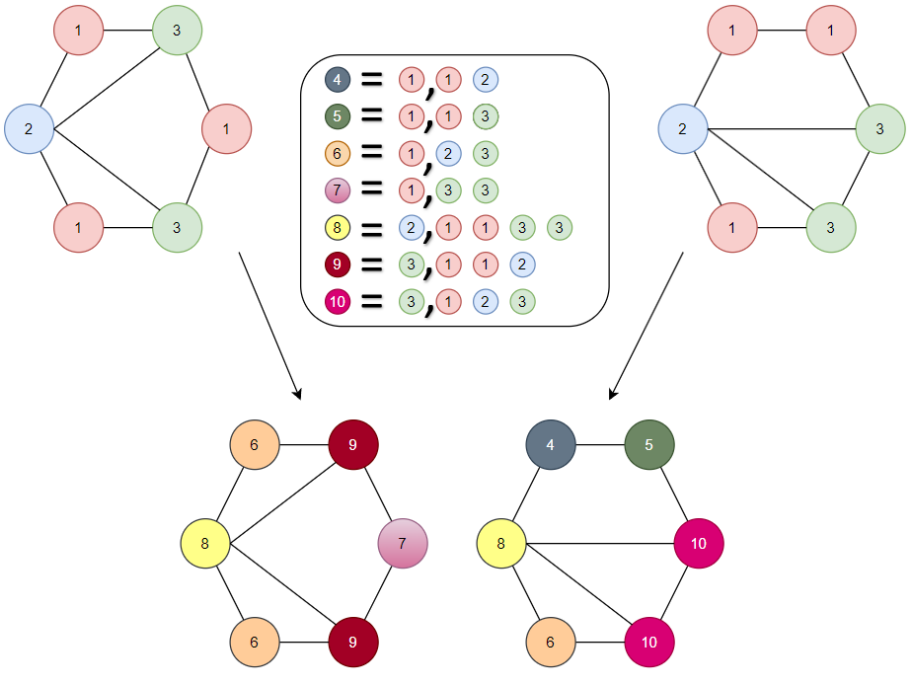



Figure 3.1: One iteration of the WL algorithm in a pair of labeled non isomorphic graphs.

1-WL algorithm if it can distinguish G from any other non-isomorphic graph H. IT has been proved that the 1-WL algorithm is capable of identifying almost all graphs [Babai and Kucera, 1979, Karp, 1979, Czajka and Pandurangan, 2008]. That is, as the number of nodes $n$ in a graph goes to infinity, the probability that the 1-WL algorithm identify a randomly selected graph from the entire set of $n$-node graphs goes to 1. The above result is an indication of how much power this simple algorithm has. Arvind et al. [2015] present a detailed analysis of the effectiveness of the 1-WL algorithm in distinguishing different types of graphs. It describes the class of graphs that can be distinguished by this algorithm and provides insights into the limitations of its capabilities. However, it is also easy to see that 1-WL cannot distinguish all non-isomorphic graphs. For example, Figures 3.2, 3.3 show two pairs of graphs, where 1-WL fails to distinguish. For more example where 1-WL fails see [Cai et al., 1992, Evdokimov and Ponomarenko, 1999, Douglas, 2011].

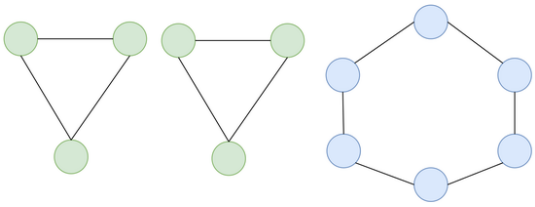


Figure 3.2: An example with triangle counts where 1-WL fails to distinguish. Both graphs are unlabeled. The green and blue colors are only used to visually distinguish the two graphs.

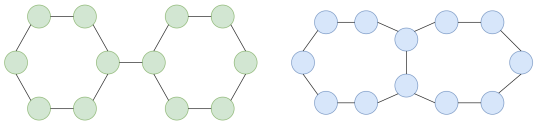


Figure 3.3: An example with cyclic information where 1-WL fails to distinguish. Both graphs are unlabeled. The green and blue colors are only used to visually distinguish the two graphs.

## 3.2. Higher order color refinement algorithms

This section focuses on presenting higher-order variants of the 1-WL algorithm. These variants are designed to address the limitations of the 1-WL algorithm, which were discussed in the previous section. By considering higher-order neighborhoods of nodes, these variants aim to capture more complex structural features of graphs. We will describe the key ideas behind each variant and we will perform a comparative analysis to evaluate their ability to distinguish between non-isomorphic graphs.

### 3.2.1. Notation

We now describe the $k$-dimensional Weisfeiler-Leman algorithm ($k$-WL), for $k \geq 2$, that is a generalization of the 1-WL which iteratively colors tuples from $V(G)^k$ instead of nodes. To begin with, we introduce the notion of the neighbor for a tuple $\mathbf{v}$ in a graph $G$. For an integer $j \in [k]$, a tuple $\mathbf{w} \in V(G)^k$ is called a $j$-neighbor of the tuple $\mathbf{v} \in V(G)^k$ and we write $\mathbf{w} \sim_j \mathbf{v}$, if there exists and $\omega \in V(G)$ such that $\mathbf{w} = \phi_j(\mathbf{v}, \omega)$. If additionally, $\{\omega, v_j\} \in E(G)$ we say that $\mathbf{w}$ is a local $j$-neighbor of $\mathbf{v}$ ($\mathbf{w} \overset{L}{\sim}_j \mathbf{v}$), otherwise we say that $\mathbf{w}$ is a global $j$-neighbor of $\mathbf{v}$ ($\mathbf{w} \overset{G}{\sim}_j \mathbf{v}$). Let also the function *adj*($\mathbf{v}, \mathbf{w}$) evaluate to L or G, depending on whether $\mathbf{w}$ is a local or a global neighbor, respectively, of $\mathbf{v}$.

### 3.2.2. k-WL algorithm

In the literature there exist two variants of the $k$-WL algorithm. They are known as the Folklore Weisfeiler Leman ($k$-FWL) and the Oblivious Weisfeiler-Leman($k$-OWL) algorithms and they differ in the way that they aggregate the colors of the neighbors for each tuple. However, in the next section we will see that the $k$-FWL is equivalent to the $(k+1)$-OWL in the sense that they can distinghuish between the exact same pair of graphs.

In the $k$-OWL algorithm, in each iteration $i \geq 0$, the algorithm computes a coloring function $C^{\mathsf{O}}_{k,i} : V(G)^k \to \Sigma$. In the first iteration ($i = 0$), two tuples $\mathbf{v}$ and $\mathbf{u}$ in $V(G)^k$ get the same color from $C^{\mathsf{O}}_{k,0}$ if the isomorphism types $\tau_{\mathbf{v}}$ and $\tau_{\mathbf{u}}$ of the graphs $G[\mathbf{v}]$ and $G[\mathbf{u}]$, respectively are equal. That is, the map $v_i \mapsto u_i$ induces an isomorphism between the induced subgraphs $G[\mathbf{v}]$ and $G[\mathbf{u}]$. For $i > 0$, $C^{\mathsf{O}}_{k,i+1}$ is defined by the following equation:

$$C^{\mathsf{O}}_{k,i+1}(\mathbf{v}) = \left(C^{\mathsf{O}}_{k,i}(\mathbf{v}), M^{\mathsf{O}}_i(\mathbf{v})\right),$$

where

$$M^{\mathsf{O}}_i(\mathbf{v}) = \left(\{\!\{C^{\mathsf{O}}_{k,i}(\phi_1(\mathbf{v}, \omega) \mid \omega \in V(G)\}\!\}, \ldots, \{\!\{C^{\mathsf{O}}_{k,i}(\phi_k(\mathbf{v}, \omega) \mid \omega \in V(G)\}\!\}\right).$$

In Figures 3.4,3.5, we see an illustration of the transformation of a graph through the 2-WL algorithm

For the $k$-FWL algorithm, in each iteration we similarly compute a coloring function $C^{\mathsf{F}}_{k,i} : V(G)^k \to \Sigma$

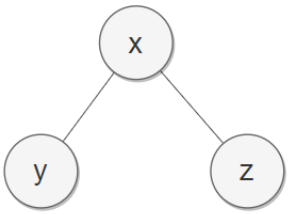



Figure 3.4: A simple unlabeled graph with 3 nodes.

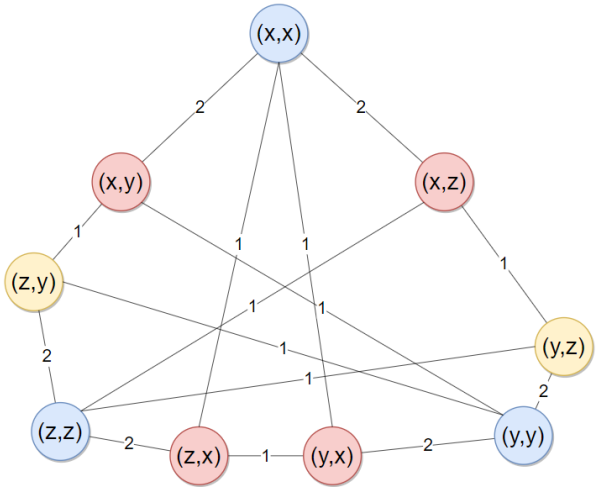



Figure 3.5: The 2-WL transformed graph for the graph shown in Figure 3.4, which incorporates the different types of neighbors and the initial colorings based on the isomorphism types.

with initial coloring $C^{\mathsf{F}}_{k,0} \equiv C^{\mathsf{O}}_{k,0}$ and by replacing the $M^{\mathsf{O}}_i$ with $M^F_i$ as follows:

$$M^{\mathsf{F}}_i(\mathbf{v}) = \{\{\mathit{sift}\left(C^{\mathsf{F}}_{k,i}, \mathbf{v}, \omega\right) \mid \omega \in V(G)\}\}.$$

From now on, we will use the notation $k$-WL to refer to the $k$-OWL variant described earlier. If we are referring to the $k$-FWL variant, we will explicitly state so.

Similarly to 1-WL we can see that the $k$-WL algorithm terminates in a finite number of iterations [Morris et al., 2019] and therefore, we can define the stable coloring function as the coloring profile in which the next iteration of the algorithm does not create a new partition (refinement) between the tuples.

We use the $k$-WL algorithm in order to check if two graphs $G$ and $H$ are isomorphic as following. We run the algorithm separately on $G$ and $H$ by iteratively coloring tuples from $V(G)^k$ and $V(H)^k$ and we end up with two partitions $\rho_G$ and $\rho_H$ of $V(G)^k$ and $V(H)^k$, respectively. Every set in $\rho_G$ contains tuples having the same color under the stable coloring function. Therefore, each set in $\rho_G$ is uniquely characterized by a color $c$ in the stable coloring function and thus we denote it as $S_c$. Similarly for the the partition $\rho_H$. Finally, we say that the $k$-WL distinguishes $G$ and $H$ if there exists a color $c$ in the stable coloring such that the corresponding color class $S_c$ satisfies $|V(G)^k \cap S_c| \neq |V(H)^k \cap S_c|$, i.e., there exist an unequal number of $c$-colored tuples in $V(G)^k$ and $V(H)^k$ or equivalently their color histograms differ Morris et al. [2021b]).

We call $d$-regular graphs those where all nodes have the same degree $d$. It is easy to see that 1-WL fails to distinguish any two regular graphs. On the other hand, Bollobás [1982] showed that 2-FWL identifies almost all $d$-regular graphs for every degree $d$. However there are also cases (Grohe and Neuen [2021]) where 2-FWL fails to distinguish. An example where 2-FWL fails can be found in Grohe and Neuen [2021] with the line graph of $K_{4,4}$ and the Shrikhande graph. Increasing $k$ $(k \geq 3)$ it becomes much more harder to find an example of non-isomorphic graphs that cannot be distinguished by $k$-FWL.

However, Cai et al. [1992] constructed a family of pair of graphs $G_k$, $H_k$ which are not distinguishable by $k$-FWL but they are by $(k+1)$-FWL resulting in an interesting hierarchy result which will be discussed later.

### 3.2.3. Higher-order WL variants

In addition to the $k$-WL algorithm described above, we present three different variants of the algorithm that operate similarly to $k$-WL. In many cases, these variants result in better performance in terms of distinguishing non-isomorphic graphs. These variants are $\delta$-$k$-WL, $\delta$-$k$-LWL and $\delta$-$k$-LWL$^+$. While for all variants the initial coloring function remains the same as in the $k$-WL variant, the corresponding coloring functions along with their aggregation neighborhoods which are now denoted by $C^{\delta}_{k,i}, M^{\delta}_i$, $C^{L}_{k,i}, M^{L}_i$ and $C^{+}_{k,i}, M^{+}_i$ for $\delta$-$k$-WL, $\delta$-$k$-LWL and $\delta$-$k$-LWL$^+$, respectively, are now different.

The $\delta$-$k$-WL variant of $k$-WL takes also into account the type of neighbors and therefore, discriminates the aggregation between local and global neighbors. Formally, the $\delta$-$k$-WL algorithm refines a coloring via the aggregation function:

$$M^{\delta}_i(\mathbf{v}) = (\{\!\{\big(C^{\delta}_{k,i}(\phi_1(\mathbf{v},\omega), \mathit{adj}(\mathbf{v},\phi_1(\mathbf{v},\omega)\big) \mid \omega \in V(G)\}\!\}, \dots,$$
$$\{\!\{\big(C^{\delta}_{k,i}(\phi_k(\mathbf{v},\omega), \mathit{adj}(\mathbf{v},\phi_k(\mathbf{v},\omega)\big) \mid \omega \in V(G)\}\!\}).$$

In Figure 3.6, we see an illustration of the $\delta$-2-WL transformation of the graph in Figure 3.4.

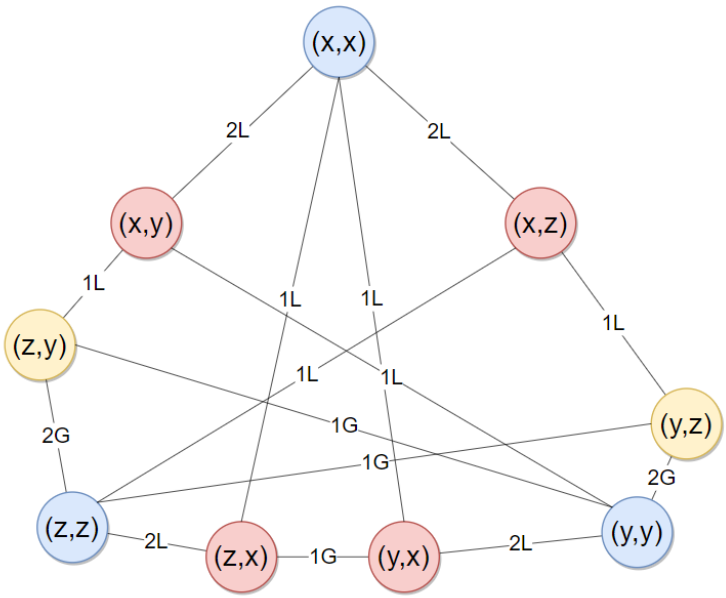



Figure 3.6: The $\delta$-2-WL transformed graph for the graph shown in Figure 3.4, which incorporates the different types of neighbors and the initial colorings based on the isomorphism types. The edge labels 1L, 1G, 2L, 2G, corresponds to 1 local, 1 global, 2 local and 2 global type of neighbors, respectively.

In [Morris et al., 2019], it is proven that the discrimination between local and global neighbors in the $\delta$-$k$-WL algorithm results in a more powerful algorithm. This means that the pairs of graphs that can be distinguished by the $k$-WL algorithm, are a strictly subset of the pairs of graphs that can be distinguished by the $\delta$-$k$-WL algorithm. A more explicit comparison between color refinement algorithms will be described in Section 3.3. However, it is easy to observe that implementing the $\delta$-$k$-WL algorithm is

computationally more expensive, since we also need to inspect the types of tuple neighbors in each iteration.

A less computationally expensive variant is the $\delta$-$k$-LWL algorithm, which considers only local neighbors during the neighborhood aggregation process and discards any information about global neighbors. This results in graphs with a significantly smaller number of edges. Formally, the $\delta$-$k$-LWL algorithm refines a coloring via the aggregation function:

$$M_i^L(\mathbf{v}) = \left(\{\!\{C_{k,i}^L(\phi_1(\mathbf{v},\omega) \mid \omega \in N(v_1)\}\!\}, \ldots, \{\!\{C_{k,i}^L(\phi_k(\mathbf{v},\omega) \mid \omega \in N(v_k)\}\!\}\right).$$

In Figure 3.7, we see an illustration of the $\delta$-2-LWL transformation of the graph in Figure 3.4.

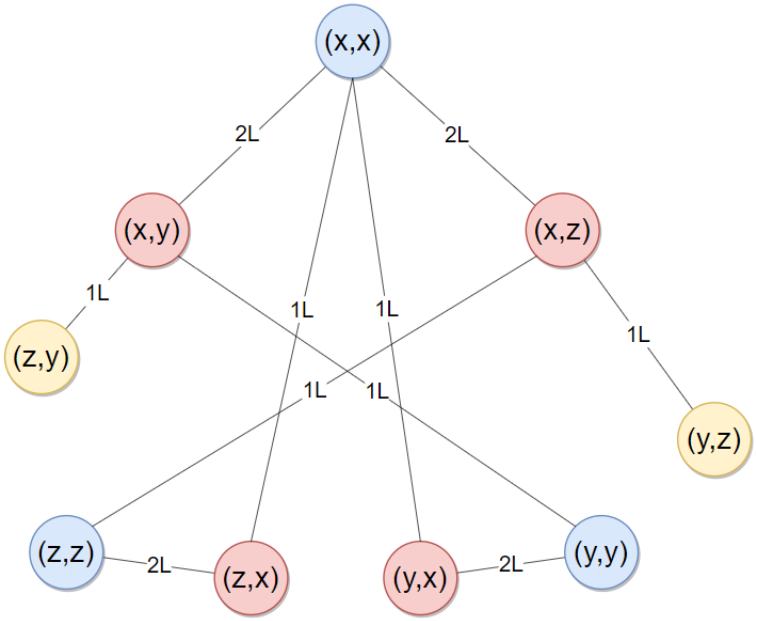



Figure 3.7: The $\delta$-2-LWL transformed graph for the graph shown in Figure 3.4, which incorporates the different types of neighbors and the initial colorings based on the isomorphism types.

Morris et al. [2020b] proved that $\delta$-$k$-LWL is not comparable to $k$-WL. Which means that there are pairs of graphs that can be distinguished by $\delta$-$k$-LWL and not by $k$-WL (Cai et al. [1992]) but there are also pairs of graphs that can be distinguished by $k$-WL and not by $\delta$-$k$-LWL (ref: Example?).

Finally, we describe $\delta$-$k$-LWL$^+$, a minor variation of the $\delta$-$k$-LWL. Morris et al. [2020b] proved that the $\delta$-$k$-LWL$^+$ variant is equivalent in power to the $\delta$-$k$-WL for all connected graphs. Formally, the $\delta$-$k$-LWL$^+$ algorithm refines a coloring via the aggregation function,

$$M_i^+(\mathbf{v}) = (\{\!\{\left(C_{k,i}^+(\phi_1(\mathbf{v},\omega), \#_i^1(\mathbf{v},\phi_1(\mathbf{v},\omega))\right) \mid \omega \in V(G)\}\!\}, \ldots,$$
$$\{\!\{\left(C_{k,i}^+(\phi_k(\mathbf{v},\omega), \#_i^k(\mathbf{v},\phi_k(\mathbf{v},\omega))\right) \mid \omega \in V(G)\}\!\}),$$

where

$$\#_i^j(\mathbf{v},\mathbf{x}) := |\{\mathbf{w} : \mathbf{w} \sim_j \mathbf{v}, C_{k,i}^+(\mathbf{w}) = C_{k,i}^+(\mathbf{x})\}|.$$

Informally, $\#_i^j(\mathbf{v}, \mathbf{x})$ counts the number of $j$-neighbors (local or global) of **v** which have the same color as **x** after $i$ rounds. Note that the $\delta$-1-LWL is exactly the 1-WL algorithm described in Section 3.1

## 3.3. Algorithms comparison

In this section we compare the above described algorithms in terms of their ability to distinguish non isomorphic graphs. Formally, for two color refinement algorithms $A$ and $B$ we say that A refines B and we denote $A \sqsubseteq B$ if every pair of graphs $G, H$ that can be distinguished by algorithm B, can also be distinguished by algorithm A. Moreover we say that A strictly refines B (denote $A \sqsubset B$) if $A \sqsubseteq B$ and there exists a pair of graphs $G, H$ that can be distinguished by algorithm $A$ but not by algorithm $B$. Finally, we say that A is equivalent to B (denote $A \equiv B$) if $A \sqsubseteq B$ and $B \sqsubseteq A$

As it has been mentioned, it is extremely difficult to find non-isomorphic graphs that are not distinguished by the $k$-FWL algorithm for $k \geq 2$. Cai et al. [1992], using tools from Mathematical Logic [Grohe and Otto, 2015], constructed a family of graphs called Cai-Furer-Immermman (CFI) which are not distinguishable by the $k$-WL algorithm. We describe the above mentioned construction which will be used in order to show that increasing $k$ in $k$-WL algorithm leads to strictly more powerful algorithms.

**CFI Construction [Cai et al., 1992, Morris et al., 2020b]** Let $K$ denote the complete graph on $k+1$ vertices (there are no loops in $K$). The vertices of $K$ are numbered from 0 to $k$. Let $E(v)$ denote the set of edges incident to $v$ in $K$: clearly, $|E(v)| = k$ for all $v \in V(K)$. Define the graph $G_k$ as follows:

1. For the vertex set $V(G)$, we add:
   (a) $(v, S)$ for each $v \in V(K)$ and for each even subset $S$ of $E(v)$,
   (b) Two vertices $e_1$ and $e_0$ for each edge $e \in E(K)$.
2. For the edge set $E(G)$, we add:
   (a) An edge $e_0, e_1$ for each $e \in E(K)$,
   (b) An edge between $(v, S)$ and $e_1$ if $v \in e$ and $e \in S$,
   (c) An edge between $(v, S)$ and $e_0$ if $v \in e$ and $e \notin S$.

Define a companion graph $H_k$, in a similar manner to $G_k$, with the following exception: in Step 1(a), for the vertex $0 \in V(K)$, we choose all odd subsets of $E(0)$. Additionally, we label the vertices of the graph as follows:

- For each $v \in V(K)$ all vertices of the form $(v, S)$ have the same label
- If $u \neq v$, then for all $S_1 \in E(u), S_2 \in E(v)$ the vertices $(u, S_1)$ and $(v, S_2)$ have different labels.
- All the other vertices have the same label but different from all the labels that have been used for $(u, S)$-type of vertices.

In Figures 3.8, 3.9, 3.10, 3.11 we can see the CFI construction graphs $G$ and $H$ for the cases $k = 2$ and $k = 3$ as described above.

To begin, we present a lemma that helps to establish the hierarchical structure of $k$-WL algorithms. That is, increasing the value of $k$ yields more powerful algorithms for distinguishing non-isomorphic graphs.

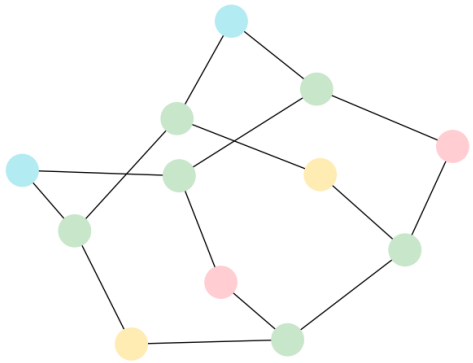


Figure 3.8: The $G_2$ graph from CFI constructions.

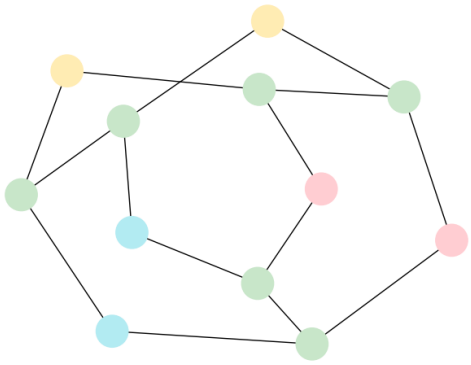


Figure 3.9: The $H_2$ graph from CFI constructions.

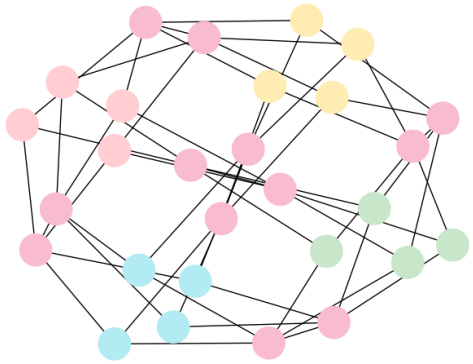


Figure 3.10: The $G_3$ graph from CFI constructions.

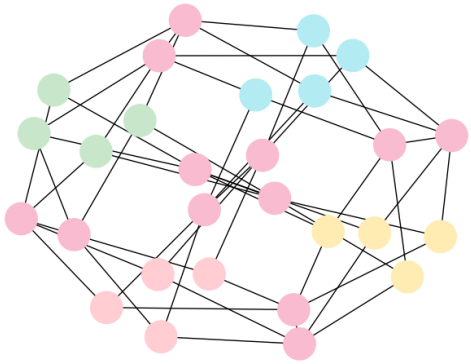


Figure 3.11: The $H_3$ graph from CFI constructions.

**Lemma 3.3.1.** *Let $G$ be a graph. We define the function $m\colon V(G)^{k+1} \to V(G)^k$ as $m(u_1,\dots,u_k,u_{k+1}) = (u_1,\dots,u_k)$. If $C_l^k$ is the coloring function induced by the $k$-WL algorithm, we have the following implication:*

$$C_l^{k+1}(\boldsymbol{u}) = C_l^{k+1}(\boldsymbol{v}) \Rightarrow C_l^k(m(\boldsymbol{u})) = C_l^k(m(\boldsymbol{v})).$$

*Proof.* For $l = 0$, it is easy to see that if $\tau_{\mathbf{u}} = \tau_{\mathbf{v}}$ we can remove from both subgraphs $G_{\mathbf{u}}$ and $G_{\mathbf{v}}$ the node labeled $k$ (i.e. the last component of each tuple) and verify that the remained subgraphs $G_{m(\mathbf{u})}$ and $G_{m(\mathbf{v})}$ still have the same isomorphism type. Therefore, $C_l^0(m(\mathbf{u})) = C_l^0(m(\mathbf{v}))$.

Assuming that the induction hypothesis is true for some integer $l \in \mathbb{N}$ and that $C_{l+1}^{k+1}(\mathbf{u}) = C_{l+1}^{k+1}(\mathbf{v})$ then by definition we have that $C_l^{k+1}(\mathbf{u}) = C_l^{k+1}(\mathbf{v})$. Therefore, by the inductive hypothesis we have:

$$C_l^k(m(\mathbf{u})) = C_l^k(m(\mathbf{v})) \tag{3.1}$$

Additionally,

$$\{\!\{C_l^{k+1}(\phi_j(\mathbf{u},\omega)) \mid \omega \in V(G)\}\!\} = \{\!\{C_l^{k+1}(\phi_j(\mathbf{v},\omega)) \mid \omega \in V(G)\}\!\},\ \forall j \leq k+1. \tag{3.2}$$

Applying the inductive hypothesis for $j \leq k$ in Equation 3.2 we get the following:

$$\{\!\{C_l^k(m(\phi_j(\mathbf{u},\omega))) \mid \omega \in V(G)\}\!\} = \{\!\{C_l^k(m(\phi_j(\mathbf{v},\omega))) \mid \omega \in V(G)\}\!\},\ \forall j \leq k. \tag{3.3}$$

But, by definition of the function $m$ we have that $m(\phi_j(\mathbf{u}, \omega)) = \phi_j(m(\mathbf{u}), \omega)$ for all $j \leq k$ and therefore,

$$\{\{C_l^k(\phi_j(m(\mathbf{u}), \omega)) \mid \omega \in V(G)\}\} = \{\{C_l^k(\phi_j(m(\mathbf{u}), \omega)) \mid \omega \in V(G)\}\}, \ \forall j \leq k. \tag{3.4}$$

Equations 3.1, 3.4 implies that $C_{l+1}^k(m(\mathbf{u})) = C_{l+1}^k(m(\mathbf{v}))$ and hence complete the induction. ■

Now by simulating all $k$-tuples with $k+1$-tuples and using Lemma 3.3.1 we have the following result.

**Proposition 3.3.2.** *$(k+1)$-WL $\sqsubseteq$ $k$-WL.*

*Proof.* If the $(k+1)$-WL algorithm fails to distinguish the pair of graphs $G$ and $H$, then there exists a bijection $f : V(G)^{k+1} \to V(H)^{k+1}$ such that $C_l^{k+1}(\mathbf{u}) = C_l^{k+1}(f(\mathbf{u}))$ for all $\mathbf{u} \in V(G)^{k+1}$, where $C_l^k$ is the coloring function induced by the $k$-WL algorithm. We define a new coloring function $C_l' : V(G)^k \to \Sigma$ as $C_l'(\mathbf{u}) = C_l^{k+1}(\mathbf{u}_+)$ where $\mathbf{u}_+ = (u_1, \dots, u_k, u_k)$ for every $u = (u_1, \dots, u_k) \in V(G)^k$.

**Claim 1.** *The coloring function $C_l'$ fails to distinguish $G$ and $H$.*

It suffices to show that there exists a bijection $g : V(G)^k \to V(H)^k$ such that $C_l'(\mathbf{u}) = C_l'(g(\mathbf{u}))$ for all $\mathbf{u} \in V(G)^k$. We define $A_G = \{\mathbf{u}_+ \mid \mathbf{u} \in V(G)^k\}$ and $A_H = \{\mathbf{u}_+ \mid \mathbf{u} \in V(H)^k\}$. Due to isomorphism types we have that $f(A_G) = A_H$. Additionally, we define the functions $\phi_G : A_G \to V(G)^k$ and $\phi_H : A_H \to V(H)^k$ with $\phi_G(\mathbf{u}_+) = \mathbf{u}$ for all $u \in A_G$ and $\phi_G(\mathbf{v}_+) = \mathbf{v}$ for all $v \in A_H$, respectively. It is easy to verify that $\phi_G, \phi_H$ are bijections. Now if $g \equiv \phi_H \circ f \circ \phi_G^{-1} : V(G)^k \to V(H)^k$, we have that $g$ is a bijection as a composition of bijections and additionally $C_l'(\mathbf{u}) = C_l'(g(\mathbf{u}))$ since,

$$\begin{aligned} C_l'(\mathbf{u}) &= C_l^{k+1}(\mathbf{u}_+) \\ &= C_l^{k+1}(\phi_G^{-1}(\mathbf{u})) \\ &= C_l^{k+1}(f(\phi_G^{-1}(\mathbf{u}))) \\ &= C_l'(\phi_H(f(\phi_G^{-1}(\mathbf{u})))) \\ &= C_l'(g(\mathbf{u})). \end{aligned}$$

**Claim 2.** $C_l' \sqsubseteq C_l^k$

Claim 2 is a special case of Lemma 3.3.1

Claims 1, 2 complete the proof. ■

The following result indicates the relationship between the two different variants (folklore and oblivious) of $k$-WL described in Section 3.2.2. This theorem follows directly from the logical characterisation of $k$-WL in Cai et al. [1992], Immerman and Lander [1990]. However, for a direct proof using explicitly graph theory tools see Grohe [2021] (Theorem V.6.).

**Theorem 3.3.3.** *For $k \geq 2$, $k$-FWL $\equiv$ $(k+1)$-OWL*

Intuitively, the distinction between the folklore and oblivious variants is that, in oblivious WL, when a vertex $w$ is substituted for the $i$-th entry of a tuple $\mathbf{v}$, the context is disregarded. That is, the entry $v_i$ that is replaced, as well as the colors obtained by replacing other entries of $\mathbf{v}$ with $w$, are forgotten. In contrast, $k$-FWL considers all of these colors jointly. This advantage of $k$-FWL allows the algorithm to capture possible edges between the entries of the tuple and any other vertex in the graph. Therefore, it is reasonable to expect that $k$-FWL transfers more information than $\delta$-$k$-LWL. However, it is noteworthy

that this result is first stated and rigorously proved in this work, demonstrating the following significant theorem.

**Theorem 3.3.4.** *For $k \geq 2$, $k$-FWL $\sqsubseteq$ $\delta$-$k$-LWL*

*Proof.* If $C_l^F$ and $C_l^L$ are the coloring functions in iteration $l$ defined by $k$-FWL and $\delta$-$k$-LWL, respectively, then it suffices to show that $C_l^F \sqsubseteq C_l^L$. By induction on $l$, for $l = 0$ it is clear by comparing isomorphism types.
(IH): $C_l^F(u) = C_l^F(v) \implies C_l^L(u) = C_l^L(v)$ , $\forall u, v \in V(G)^k$
If $C_{l+1}^F(u) = C_{l+1}^F(v)$ then by definition we have that $C_l^F(u) = C_l^F(v)$ and by (I.H)

$$C_l^L(u) = C_l^L(v) \tag{3.5}$$

In addition we have that

$$\{\{\text{sift}(C_l^F, u, \omega) \mid \omega \in V(G)\}\} = \{\{\text{sift}(C_l^F, v, \omega) \mid \omega \in V(G)\}\}$$

Therefore, we can find a bijection $\sigma$ between these multisets such that for every $\omega$ in $V(G)$ we have:

$$\text{sift}(C_l^F, u, \omega) = \text{sift}(C_l^F, v, \sigma(\omega))$$

or equivalently,

$$C_l^F(\phi_j(u, \omega)) = C_l^F(\phi_j(v, \sigma(\omega))) \; \forall j \in [n] \tag{3.6}$$

What is more about the bijection $\sigma$, is that if $\omega$ is adjacent to $u_j$ then $\sigma(\omega)$ is adjacent $v_j$. To see that, we consider an $\omega \in N(u_{j_0})$ for some $j_0 \in [n]$. Then we pick an arbitary $m_0 \in [n] \setminus \{j_0\}$ and from Equation 3.6 (for $j = m_0$) we have that $C_l^F(\phi_{m_0}(u, \omega)) = C_l^F(\phi_{m_0}(v, \sigma(\omega)))$. Therefore, $\phi_{m_0}(u, \omega)$ and $\phi_{m_0}(v, \sigma(\omega))$ have the same isomorphism type and since for the subgraph $G_{\left[\phi_{m_0}(u,\omega)\right]}$ we know that the node labeled $m_0$ is adjacent to the node labeled $j_0$ we imply that $\sigma(\omega) \in N(v_{j_0})$. Therefore, the bijection $\sigma$ is also a bijection between the $j$-local neighbors of $u$ and $v$ for all $j \in [k]$ which means that:

$$\{\{C_l^F(\phi_j(u, \omega)) \mid \omega \in N(u_j)\}\} = \{\{C_l^F(\phi_j(v, \omega)) \mid \omega \in N(v_j)\}\}, \; \forall j \in [k]$$

and by (I.H)

$$\{\{C_l^L(\phi_j(u, \omega)) \mid \omega \in N(u_j)\}\} = \{\{C_l^L(\phi_j(v, \omega)) \mid \omega \in N(v_j)\}\}, \; \forall j \in [k] \tag{3.7}$$

Equations 3.5,3.7 complete the induction. ■

As previously mentioned, finding a pair of non-isomorphic graphs $G_k$ and $H_k$ that cannot be distinguished by the $k$-WL algorithm for $k > 2$ is exceedingly difficult. Therefore, in the subsequent results, we utilize the CFI construction described in Section 3.3 from the seminal paper Cai et al. [1992] to provide strictly refinement comparison results between the above described algorithms.

The following proposition demonstrates that, despite the inability of $k$-WL to distinguish the CFI construction graphs, the local variant $\delta$-$k$-LWL can differentiate between them. It is worth noting, however, that this result does not imply that $\delta$-$k$-LWL is necessarily more powerful than $k$-WL. The proof of Proposition 3.3.5 is omitted here, as Proposition 4.3.9 in Chapter 4 provides a proof using multisets instead of tuples. However, for a detailed proof of Proposition 3.3.5, one may refer to Morris et al. [2020b] (Lemma 9).

**Proposition 3.3.5.** *$\delta$-$k$-LWL distinguish the CFI construction graphs described in 3.3.*

In the following theorem we prove that the $\delta$-$k$-LWL forms a strict hierarchy of algorithms.

**Theorem 3.3.6.** *For $k \geq 2$, $\delta$-$k$-LWL $\sqsubset$ $\delta$-$(k-1)$-LWL*

*Proof.* Following the proof of Proposition 3.3.2 it is easy to see that $\delta$-$k$-LWL $\sqsubseteq$ $\delta$-$(k-1)$-LWL for all $k \geq 2$. Therefore, it suffices to show an infinite family of graphs $(G_k, H_k)$, $k \in \mathbb{N}$, such that (a) $\delta$-$(k-1)$-LWL does not distinguish $G_k$ and $H_k$, although (b) $\delta$-$k$-LWL distinguishes $g_k$ and $H_k$.

This family is exactly the CFI construction 3.3. From Cai et al. [1992] we know that $k$-OWL (or equivalently $(k-1)$-FWL) fails to distinguish $G_k$ and $H_k$. Now by Proposition 3.3.4 we have that $\delta$-$(k-1)$-LWL also fails to distinguish $G_k$ and $H_k$. Finally from Proposition 3.3.5 we have tha $\delta$-$k$-LWL distinguishes between $G_k$ and $H_k$. ■

Again using the CFI construction and Proposition 3.3.5 it is easy to prove the following result.

**Proposition 3.3.7** (Morris et al. [2020b])**.** *For $k \geq 2$, $\delta$-$k$-WL $\sqsubset$ $k$-WL*

For completeness, we state the following theorem from Morris et al. [2020b] (Lemma 5), which indicates that a slightly modified version of $\delta$-$k$-LWL is equivalent to $\delta$-$k$-WL for the class of connected graphs, which is a strictly more powerful algorithm than $k$-WL, as stated in Proposition 3.3.7.

**Proposition 3.3.8** (Morris et al. [2020b])**.** *$\delta$-$k$-LWL$^+$ $\equiv$ $\delta$-$k$-WL for connected graphs.*

Finally, we show that the $k$-WL algorithm also forms a hierarchy. That is, increasing $k$ the $k$-WL algorithm becomes strictly more powerful.

**Proposition 3.3.9.** *$(k+1)$-WL $\sqsubset$ $k$-WL*

*Proof.* By Proposition 3.3.2 we only need to find a pair of graphs $G_k$, $H_k$ which are not distinguishable by $k$-WL and they are by $(k+1)$-WL. These are the CFI construction graphs. In Cai et al. [1992] it is proved that $k$-WL fails to distinguish CFI construction and by Propositions 3.3.3, 3.3.4, 3.3.5 we have that $(k+1)$-WL distinguishes the CFI graphs. ■

## 3.4. Neural architectures based on vertex refinement algorithms

This section presents the fundamental concepts of Graph Neural Networks (GNNs), a machine learning algorithm designed to tackle graph representation learning tasks (Hamilton et al. [2017]). We describe the basic idea behind GNNs, which revolves around learning node embeddings through message passing between connected nodes in a graph. Additionally, we present the core structure of Graph Neural Networks, which typically consists of multiple layers of computation, each of which performs a set of graph operations. By comprehending the core structure and underlying principles of Graph Neural Networks, we can develop novel GNN architectures based on vertex refinement algorithms, as those described in previous sections ($k$-WL, $\delta$-$k$-LWL etc.), that are applicable to various graph-based machine learning tasks. Finally, we demonstrate that the expressive power of neural architectures increases with the strength of the vertex refinement algorithm upon which it is based.

Before diving into the details of different GNN structures, it is important to highlight some significant findings related to the expressive power of graph neural networks. These are graph separation power

and function approximation which are two **equivalent** key notions that are used to describe the expressive power of GNNs [Chen et al., 2019]. **Graph separation power** refers to the ability of a model class to differentiate between non-isomorphic graphs by providing different outputs for every pair of different (non-isomorphic) graphs. This notion plays a critical role in understanding how well a GNN can distinguish between graphs with subtle differences. **Function approximation**, on the other hand, refers to the capability of a model class to effectively approximate permutation-invariant functions on graphs. This property bears resemblance to the capability exhibited by Multilayer Feedforward Neural Networks (Hornik et al. [1989]).

Understanding these two notions is essential for designing and developing novel GNN architectures that can perform effectively on graph-based machine learning tasks. Therefore, a deeper understanding of these concepts could help us to evaluate and compare the expressive power of different GNN models and select the appropriate model for each specific task.

Let $G$ be a graph with $n$ nodes, $[n] := \{1, \dots, n\}$. We allow additional information to be stored in the form of node and edge features, which we assume to belong to a compact set $\mathcal{X} \subset \mathbb{R}$, and we define $\mathcal{G} := \mathcal{X}^{n \times n}$. With an abuse of notation, we also regard $G$ as an $n \times n$ matrix that belongs to $\mathcal{G}$, where $\forall i \in [n]$, $G_{i,i}$ is the feature of the $i$-th node, and $\forall i, j \in [n]$, $G_{i,j}$ is the feature of the edge from node $i$ to node $j$ (and equal to some predefined null value if $i$ and $j$ are not connected). For example, an undirected graph is represented by a symmetric matrix $G$. If there are no node and edge features, $G$ can be viewed as identical to the adjacency matrix. Thus, modulo the permutation of the node orders, $\mathcal{G}$ is the space of discrete labeled graphs with $n$ nodes. For two graphs $G, G' \in \mathcal{G}$, we say they are isomorphic (and write $G \cong G'$) if $\exists \Pi \in S_n$ such that $\Pi^\top \cdot G \cdot \Pi = G'$, where $S_n$ denotes the set of all $n \times n$ permutation matrices. A function $f$ on $\mathcal{G}$ is called permutation-invariant if $f(\Pi^\top \cdot G \cdot \Pi) = f(G)$, $\forall G \in \mathcal{G}$, $\forall \Pi \in S_n$.

A GNN with a graph-level scalar output represents a parameterized collection $\mathcal{C}$ of functions from $\mathcal{G}$ to $\mathbb{R}$, which are typically permutation-invariant by design. Given such a collection, we will mathematically describe the approximation power and the separation power of the GNN model class $\mathcal{C}$

The above two notions are formally described through the Definitions 1,2 as they were given in Chen et al. [2019].

**Definition 1** (Chen et al. [2019])**.** *We say $\mathcal{C}$ is GIso-discriminating if $\forall G_1, G_2 \in \mathcal{G}$ such that $G_1 \ncong G_2$, $\exists h \in \mathcal{C}$ such that $h(G_1) \neq h(G_2)$.*

**Definition 2** (Chen et al. [2019])**.** *We say $\mathcal{C}$ is universally approximating if for all continuous permutation-invariant functions $f : \mathcal{G} \to \mathbb{R}$, and $\forall \epsilon > 0$, $\exists h \in \mathcal{C}$ such that $||f - h||_\infty := \sup_{G \in \mathcal{G}} |f(G) - h(G)| < \epsilon$.*

The equivalence of the two aforementioned notions, Graph separation power and Function approximation, is demonstrated analytically by Chen et al. [2019] and Azizian and Lelarge [2020].

### 3.4.1. 1-GNNs

We now present the 1-GNN architecture as it is described in Morris et al. [2019] and Scarselli et al. [2009]. Intuitively, 1-GNNs compute a vectorial representation, i.e., a $d$-dimensional real vector, representing each node in a graph by aggregating information from neighboring nodes. Formally, let $(G, l)$ be a labeled graph with an initial vertex encoding $f^{(0)} : V(G) \to \mathbb{R}^{1 \times d}$ that is consistent with $l$. That is, two nodes have the same initial encoding through $f^{(0)}$ if and only if they have the same initial label through

$l$. These initial node features may represent continuous atomic properties in chemoinformatic applications where nodes correspond to atoms, or vector representations of text in social network applications. A typical 1-GNN model consists of a stack of neural network layers, where each layer aggregates local neighborhood information, i.e., features of neighbors, around each node and then passes this aggregated information on to the next layer. In full generality a new feature $f^{(t)}$ for a node $v$ is computed as:

$$f_{\text{merge}}^{W_1}\left(f^{(t-1)}(v), f_{\text{aggr}}^{W_2}\left(\{\!\{f^{(t-1)}(w) \mid w \in N(v)\}\!\}\right)\right), \tag{3.8}$$

where $f_{\text{aggr}}^{W_2}$ aggregates over the set of neighborhood features and $f_{\text{merge}}^{W_1}$ merges the node's representations from the previous layer with the computed neighborhood features. Both $f_{\text{merge}}^{W_1}$, $f_{\text{aggr}}^{W_2}$ may be arbitrary differentiable, permutation-invariant functions (e.g., neural networks) and we denote their parameters as $W_1$, $W_2$, respectively. After computing a vector representation for each node in a graph, we can obtain a vector representation (fingerprint) $f_{\text{GNN}}$ for the entire graph using pooling layers (e.g. sum, mean and set2set [Vinyals et al., 2016]). See the work by Schweidtmann et al. [2023] for a more detailed analysis about the importance of pooling layer in graph representation learning tasks. For example in the following equation we use sum pooling to derive a vector representation for the entire graph.

$$f_{\text{GNN}}(G) = \sum_{v \in V(G)} f^{(T)}(v), \tag{3.9}$$

where $T > 0$ denotes the last layer.

Upon the computation of the fingerprint-vector, it is subsequently fed to a feedforward artificial neural network. The output is a probability distribution representing the likelihood of the graph belonging to each respective class in the case of a classification task, or a real number for a regression task. See Figure 3.12 for an illustration of the traditional GNN architecture.

A simple example of a GNN's architecture is described in Hamilton et al. [2017] with the following network:

$$f^{(t)}(v) = \sigma\left(f^{(t-1)}(v) \cdot W_1^{(t)} + \sum_{w \in N(v)} f^{(t-1)}(w) \cdot W_2^{(t)}\right) \tag{3.10}$$

For more complicated architectures using LSTM layers see Gilmer et al. [2017]

### 3.4.2. Equivalence between 1-WL and 1-GNN

In the following we present the results from Morris et al. [2019] demonstrating that, under certain conditions, 1-GNNs have the same expressive power as the 1-WL algorithm described in the previous section in terms of distinguishing between non-isomorphic graphs.

Let $(G, l)$ be a labeled graph, and let $W^{(t)} = (W_1^{(t')}, W_2^{(t')})_{t' \leq t}$ denote the GNN parameters given by Equation 3.10 or Equation 3.8 up to layer (iteration) $t$. We encode the initial labels $l(v)$ by vectors $f^{(0)}(v) \in \mathbb{R}^{1 \times d}$ such that the encoding function $f^{(0)}$ is consistent with the labels $l$ (example one hot encoding). The first theoretical result from Morris et al. [2019] states that for any two functions $f_{\text{merge}}^{W_1}$ and $f_{\text{aggr}}^{W_2}$ chosen in Equation 3.8, for every choice of initial encoding $f^{(0)}$ that is consistent with $l$, and for every choice of parameters $W^{(t)}$, we have that the coloring $C_l^{(t)}$ of 1-WL always refines the coloring $f^{(t)}$ induced by Equation 3.8. Formally we have the following theorem.

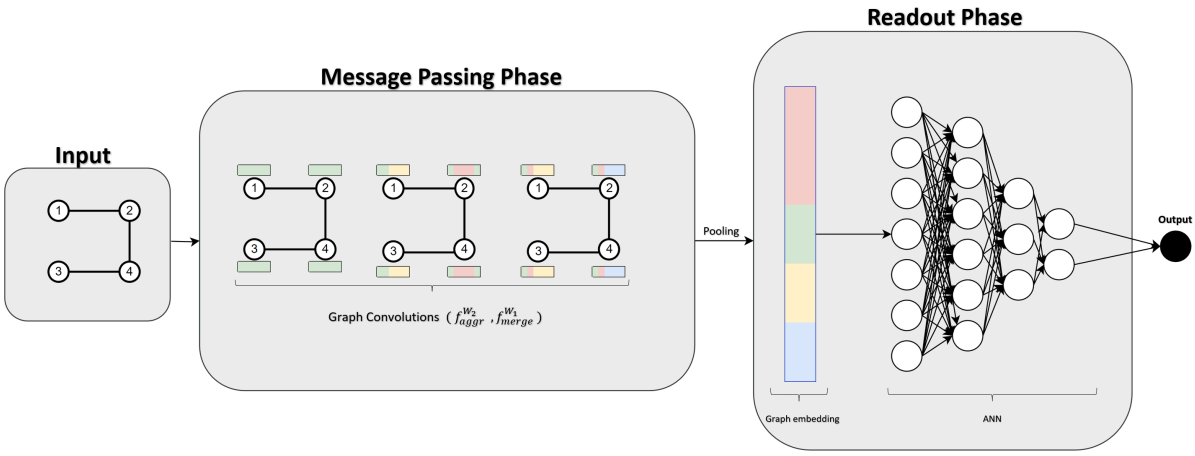



Figure 3.12: Illustration of the $\delta$-2-LWL$\{\cdot\}$-GNN structure highlighting the message passing, pooling and the ANN phases.

**Theorem 3.4.1.** *Let $(G, l)$ be a labeled graph. Then for all $t \geq 0$ and for all choices of initial colorings $f^{(0)}$ consistent with l, and weights $W^{(t)}$ we have that:*

$$C_l^{(t)} \sqsubseteq f^{(t)}.$$

Which means that if two nodes have the same color after $t$ iterations of the 1-WL algorithm, then they will also have the same vector encoding in layer $t$ of the 1-GNN architecture described by Equation 3.8. Hence, any aggregation-based 1-GNN is at most as powerful as the 1-WL test in distinguishing non-isomorphic graphs. Therefore, a natural follow-up question is if there exist GNNs that are exactly as powerful as the 1-WL test. The answer is yes and a sufficient condition for a neural architecture to be as powerful as the 1-WL test is given in Xu et al. [2019b]. More specifically for the architecture of Equation 3.10, Morris et al. [2019] proved the following theorem:

**Theorem 3.4.2.** *Let $(G, l)$ be a labeled graph. Then for all $t \geq 0$, there exists a sequence of weights $W^{(t)}$, and a 1-GNN architecture such that $C_l^{(t)} \equiv f^{(t)}$.*

The above results suggest that 1-GNNs can be considered an extension of the 1-WL algorithm, as they have the same expressive power but are more flexible in adapting to different learning tasks. Furthermore, 1-GNNs are able to handle continuous node features via vector representations, which makes them more versatile than 1-WL in this respect. See Figure 3.13 for an illustration of the equivalence between the WL heuristic and GNN architectures

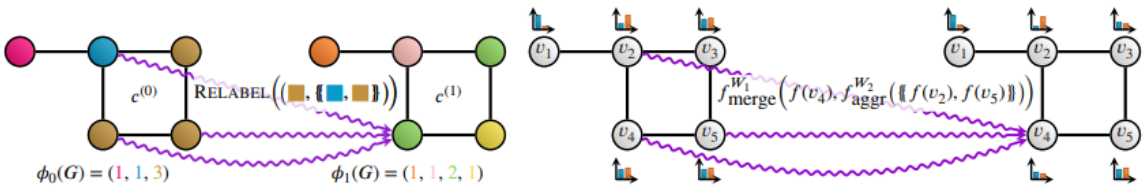



Figure 3.13: Illustration of the update rules of the 1-WL and 1-GNNs. Figure adapted from Morris et al. [2021b]

### 3.4.3. Higher order neural architectures

In the previous section, we discussed the architecture of 1-GNNs, a simple GNN architecture based on node neighborhood aggregation message passing, and its equivalence with the 1-WL algorithm in terms of distinguishing non-isomorphic graphs. In this section, we introduce GNN variants based on the $k$-WL algorithm and its variants, which overcome the limitations of 1-GNNs.

As discussed in Section 3.2 the $k$-WL and its variants are introduced to overcome the shortcomings of the 1-WL algorithm. Similarly, the GNN variants described in this section which are based on the corresponding $k$-WL variants are able to capture more global information about a graph. This enables higher-order GNNs to overcome the limitations of 1-GNNs.

To begin with, we introduce the $k$-GNN architecture based on the $k$-WL algorithm. For a given graph $G$ and an integer $k \geq 2$, $k$-GNN computes vector representations for all $k$-element tuples in $V(G)^k$ and then similarly to Equation 3.9, it computes a new vector representation for the whole graph. Initially, two tuples $\mathbf{u}$, $\mathbf{v}$ are assigned with the same vector if and only if the induced subgraphs $G_{\mathbf{u}}$, $G_{\mathbf{v}}$ are isomorphic (e.g. one hot encoding). Additionally, for a given tuple $\mathbf{v} \in V(G)^k$ we define the $j$-neighborhood of $\mathbf{v}$ as

$$N_j(\mathbf{u}) = \{\mathbf{u} \in V(G)^k \mid \mathbf{u} \sim_j \mathbf{v}\}.$$

In each $k$-WL-GNN layer $t \geq 0$, we compute a feature vector $f_k^{(t)}(\mathbf{v})$ for each tuple $\mathbf{v}$ by:

$$f_{\text{merge}}^{W_1}\left(f^{(t-1)}(\mathbf{v}), f_{\text{aggr}}^{W_2}\left(\{\{f^{(t-1)}(\mathbf{w}) \mid \mathbf{w} \in N_1(\mathbf{v})\}\}, \ldots, \{\{f^{(t-1)}(\mathbf{w}) \mid \mathbf{w} \in N_k(\mathbf{v})\}\}\right)\right).$$

where $f_{\text{merge}}^{W_1}$,$f_{\text{aggr}}^{W_2}$,$W_1$,$W_2$ are the aggregation functions and their parameters as described in Equation 3.8.

We further define the set of all $j$-local neighbors and the set of all $j$-global neighbors of a tuple $\mathbf{v}$ as $N_{j,L}(\mathbf{v})$, $N_{j,G}(\mathbf{v})$, respectively. Where:

$$N_{j,L}(\mathbf{v}) = \{\mathbf{u} \in V(G)^k \mid \mathbf{u} \overset{L}{\sim}_j \mathbf{v}\}$$

and

$$N_{j,G}(\mathbf{v}) = \{\mathbf{u} \in V(G)^k \mid \mathbf{u} \overset{G}{\sim}_j \mathbf{v}\}.$$

Therefore, for the $\delta$-$k$-WL-GNN we have the following architecture:

$$f_{\text{merge}}^{W_1}\big(f^{(t-1)}(\mathbf{v}), f_{\text{aggr}}^{W_2}(\{\{f^{(t-1)}(\mathbf{w}) \mid \mathbf{w} \in N_{1,L}(\mathbf{v})\}\}, \{\{f^{(t-1)}(\mathbf{w}) \mid \mathbf{w} \in N_{1,G}(\mathbf{v})\}\}, \ldots, \{\{f^{(t-1)}(\mathbf{w}) \mid \mathbf{w} \in N_{k,L}(\mathbf{v})\}\}, \{\{f^{(t-1)}(\mathbf{w}) \mid \mathbf{w} \in N_{k,G}(\mathbf{v})\}\})\big).$$

For the $\delta$-$k$-LWL algorithm we have the $\delta$-$k$-LGNN architecture, respectively given below:

$$f_{\text{merge}}^{W_1}\left(f^{(t-1)}(\mathbf{v}), f_{\text{aggr}}^{W_2}\left(\{\{f^{(t-1)}(\mathbf{w}) \mid \mathbf{w} \in N_{1,L}(\mathbf{v})\}\}, \ldots, \{\{f^{(t-1)}(\mathbf{w}) \mid \mathbf{w} \in N_{k,L}(\mathbf{v})\}\}\right)\right).$$

Similarly we can define the $\delta$-$k$-LGNN$^+$ architecture based on $\delta$-$k$-LWL$^+$ algorithm.

The architectures described above have similar results to Theorem 3.4.1, which demonstrates the

expressive power of the neural architectures in terms of distinguishing non-isomorphic graphs. For example, in Morris et al. [2019], the following theorem is proven:

**Theorem 3.4.3** (Morris et al. [2019])**.** *Let $(G, l)$ be a labeled graph. Then for all $t \geq 0$ there exists a sequence of weights $\boldsymbol{W}^{(t)}$ such that:*

$$C^L_{k,t}(\boldsymbol{v}) = C^L_{k,t}(\boldsymbol{w}) \Rightarrow f^{(t)}(\boldsymbol{v}) = f^{(t)}(\boldsymbol{w}).$$

*Hence, $\delta$-$k$-LGNN $\equiv$ $\delta$-$k$-LWL.*

Figure 3.14 illustrates an overview of the power of different $k$-WL variants and therefore, for their corresponding Neural Architectures in terms of distinguishing between non-isomorphic graphs.

**Generalization power of neural architectures**: Garg et al. [2020] studied the generalization abilities of a neural architecture for binary classification tasks. They found that local, sparsity-aware, higher-order variants, such as the $\delta$-$k$-LWL-GNN, exhibit smaller generalization errors compared to dense, global variants like the $k$-WL-GNN. In Section 5, we will conduct experiments to verify the aforementioned result to some extent. Our experimental findings demonstrate that the local variants perform better on unseen data compared to the global variants. However, it is important to note that this result does not hold true in terms of the expressive power of the models. For instance, the $k$-WL algorithm and the $\delta$-$k$-LWL algorithm are incomparable. This means that there exist graphs that can be distinguished by the $k$-WL algorithm but not by the $\delta$-$k$-LWL algorithm, and vice versa.

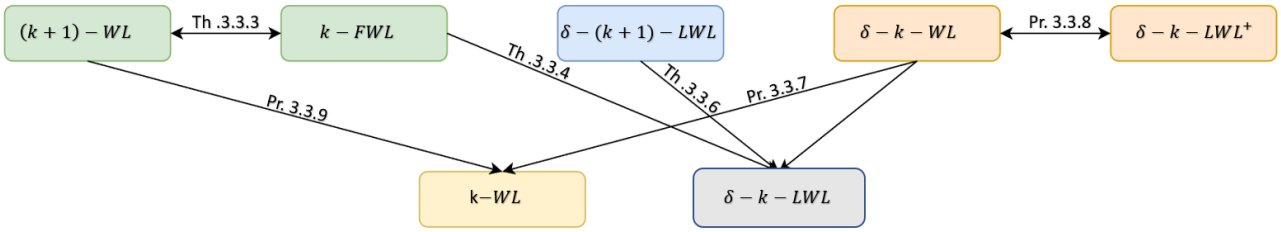



Figure 3.14: The double arrows indicate that two boxes are equivalent. One-directional arrows indicate that the edge of the arrow points to the less powerful algorithm

# 4

# (Multi)Set based refinement algorithms

In the previous section, we analyzed refinement algorithms that iteratively color tuples of vertices instead of individual vertices to address the limitations of the simple 1-WL algorithm. We also described various higher-order variants of the WL algorithm for the Graph Isomorphism problem, along with their proposed neural architectures for graph representation learning tasks. While these algorithms exhibit an enhanced ability to capture higher-order patterns in the structure of graphs and distinguish between non-isomorphic graphs, they suffer from scalability issues.

Specifically, the $k$-WL algorithm considers all $n^k$ $k$-tuples of an $n$-node graph, which results in a prohibitive computational cost for large real-world graphs. It is worth mentioning that, besides the neural architecture described in Section 3.4.3, there are also other architectures that possess the same power as $k$-WL in terms of distinguishing non-isomorphic graphs (see Azizian and Lelarge [2020], Geerts [2020], Maron et al. [2019b] ). However, for all of these architectures, the memory requirement is lower-bounded by $n^k$ for an $n$-node graph and they have to resort to dense matrix multiplication. Another limitation of the $k$-WL structure is that it always considers neighborhoods of size $k \cdot n$ without taking into account the sparsity of the graph. Local variants that we described earlier address this problem by incorporating only a subset of the neighborhoods, namely the local neighbors, in $k$-WL. This approach partially integrates the sparsity of the graph, but the number of all $k$-tuples that need to be computed remains $n^k$ while the computational costs for the (reduced) neighborhoods remains relatively high for large graphs. Advanced $k$-WL algorithms have been developed in Morris et al. [2022], which operate on a subset of all possible $k$-tuples and leverage the sparsity of graphs to a greater extent, resulting in significantly reduced computational costs.

In the following, we introduce two different variants of the $k$-WL algorithm that aim to tackle above mentioned issues. First, we present a $k$-WL algorithm that iteratively colors multisets of nodes instead of tuples, reducing the lower bound for the memory requirement to $\binom{n+k-1}{k}$ for an $n$-node graph. Next, we describe a similar variant that does not allow for copies and considers only sets of nodes, leading to a further reduction in the memory requirement since it computes $\binom{n}{k}$ different sets for an $n$-node graph. However, we prove that transitioning from a tuple-based refinement algorithm to a multiset-based refinement algorithm can only results in a loss of expressive power. Similar results hold, when

transitioning from a multiset-based to a set-based algorithm. This result indicates a tradeoff between the expressive power of the algorithm and the computational costs required for its implementation.

## 4.1. Notation

For two multisets $M_1$ and $M_2$, let $S_1$ and $S_2$ be the sets containing only the distinct elements of $M_1$ and $M_2$, respectively. The intersection of $M_1$ and $M_2$, denoted as $M_1 \cap M_2$, is defined as the multiset with all the elements from $S_1 \cap S_2$, where the multiplicity of each element is equal to the minimum of its multiplicity in $M_1$ and $M_2$. Analogously, we define $M_1 \cup M_2$ as the multiset with all the elements from $S_1 \cup S_2$ where the multiplicity of each element is equal to the maximum of its multiplicity in $M_1$ and $M_2$. Finally, the difference of two multisets $M_1$ and $M_2$, is a multiset such that the multiplicity of an element is equal to the multiplicity of the element in $M_1$ minus the multiplicity of the element in $M_2$ if this difference is positive, and is equal to $0$ if this difference is $0$ or negative. Now, let $G$ be a graph, then for $S, T \in [[V(G)]]^k$ we say that $S$ is a neighbor of $T$ and we write $S \sim T$ if $|S \cap T| = k - 1$. In addition we say that $S$ is a local neighbor of $T$ and we write $S \overset{L}{\sim} T$ if $S \sim T$ and $S \triangle T := (S \setminus T) \cup (T \setminus S) \in E(G)$ and we say that $S$ is a global neighbor of $T$ if $S \sim T$ and $S$ is not a local neighbor of $T$ (we write $S \overset{G}{\sim} T$). Same definitions hold for $S, T \in [V(G)]^k$. Let again the function *adj*$(S, T)$ evaluate to L or G, depending on whether S is a local or a global neighbor, respectively, of T.

## 4.2. (Multi)Set based k-WL and variants

We now describe the multiset and the set based Weisfeiler-Leman algorithms denoted by $k$-WL$\{\{\cdot\}\}$ and $k$-WL$\{\cdot\}$, respectively. We define the $k$-WL$\{\{\cdot\}\}$ refinement algorithm as following: for $i > 0$ and for $S \in [[V(G)]]^k$, we define the coloring function $C^m_{k,i}$ on $[[V(G)]]^k$ as:

$$C^m_{k,i}(S) = \left(C^m_{k,i-1}(S), M^m_{i-1}(S)\right)$$

where,

$$M^m_i(S) = \{\{C^m_{k,i}(T) \mid T \sim S\}\}$$

For $i = 0$ we define the initial coloring $C^m_{k,0}(S)$ of a mutliset $S$ to be equal with the isomorphism type of the induced graph $G_S$. Additionally, we have the local variant of the multiset based algorithm denoted by $\delta$-$k$-LWL$\{\{\cdot\}\}$ defined as following:

$$C^{L_m}_{k,i}(S) = \left(C^{L_m}_{k,i-1}(S), M^{L_m}_{i-1}(S)\right)$$

where:

$$M^{L_m}_i(S) = \{\{C^{L_m}_{k,i}(T) \mid T \overset{L}{\sim} S\}\}$$

and the $\delta$-variant denoted by $\delta$-$k$-WL$\{\{\cdot\}\}$ as:

$$C^{\delta_m}_{k,i}(S) = \left(C^{\delta_m}_{k,i-1}(S), M^{\delta_m}_{i-1}(S)\right)$$

where:

$$M^{\delta_m}_i(S) = \left\{\!\!\left\{\left(C^{\delta_m}_{k,i}(T), \textit{adj}(S,T)\right) \mid T \sim S\right\}\!\!\right\}.$$

Or equivalently,

$$M^{\delta_m}_i(S) = \left(\{\{C^{\delta_m}_{k,i}(T) \mid T \overset{L}{\sim} S\}\}, \{\{C^{\delta_m}_{k,i}(T) \mid T \overset{G}{\sim} S\}\}\right).$$

Similar definitions hold for the set based algorithms $k$-WL$\{\cdot\}$, $\delta$-$k$-LWL$\{\cdot\}$, $\delta$-$k$-WL$\{\cdot\}$.

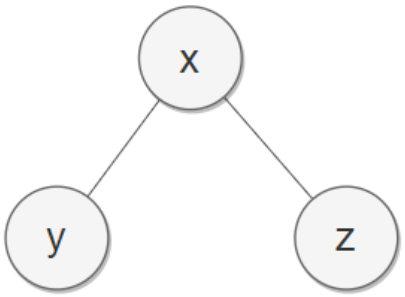



Figure 4.1: A simple unlabeled graph with 3 nodes and 2 edges.

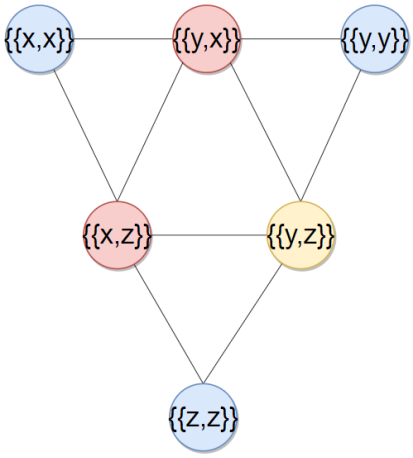



Figure 4.2: The 2-WL$\{\{\cdot\}\}$ transformed graph for the graph shown in Figure 4.1, which incorporates the edges and the initial colorings based on the isomorphism types.

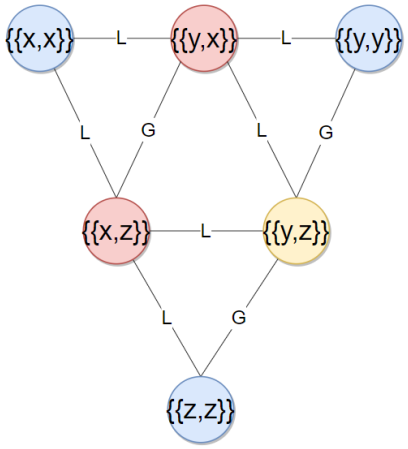



Figure 4.3: The $\delta$-2-WL$\{\{\cdot\}\}$ transformed graph for the graph shown in Figure 4.1, which incorporates the different types of neighbors and the initial colorings based on the isomorphism types.

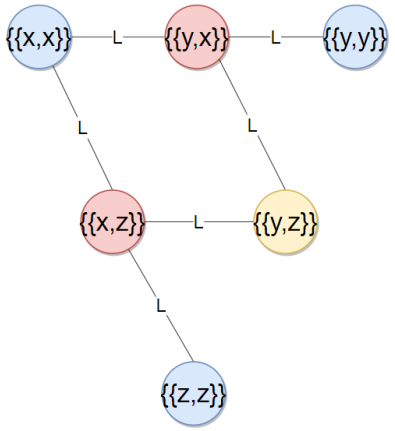



Figure 4.4: The $\delta$-2-LWL$\{\{\cdot\}\}$ transformed graph for the graph shown in Figure 4.1, which incorporates the local type of neighbors and the initial colorings based on the isomorphism types.

**Unrollings**

Given a graph $G$, a multiset $S$ in $[[V(G)]]^k$ and an integer $l \geq 0$, the unrolling UNR$[G, S, l]$ is a rooted, directed tree with vertex labels, defined recursively as follows.

- for $l = 0$, the unrolling UNR$[G, S, l]$ is defined to be a single vertex labeled with the isomorphism type of $G_S$. This lone vertex is also the root vertex
- For $l > 0$, UNR$[G, S, l]$ is defined as follows. First, introduce a root vertex $r$, labeled with the isomorphism type of $G_S$. Next, for each neighbor $\tilde{S}$ of $S$, append the rooted subtree UNR$[G, \tilde{S}, l-1]$ below the root $r$.

We refer to UNR$[G, S, l]$ as the unrolling of the graph G at $S$ of depth $l$.

Analogously, we can define the unrolling trees $\delta$-UNR, L-UNR, for the refinement algorithms $\delta$-$k$-WL$\{\{\cdot\}\}$ and $\delta$-$k$-LWL$\{\{\cdot\}\}$, respectively. The minor differences lie in the recursive step above, since the unrolling construction needs to faithfully represent the aggregation process.

- For $\delta$-UNR, we label the directed edge with $L$ or $G$, depending on whether the neighborhood is local or global.
- For L-UNR, we consider only the subtrees L-UNR$[G, \tilde{S}, l-1]$ for local neighbors $\tilde{S}$.

The usefulness and power of unrolling techniques will become clear later in Lemma 4.3.5

## 4.3. Theoretical results

In this section, we conduct a comprehensive comparison of the algorithms described above in terms of their ability to distinguish non-isomorphic graphs. As expected, the multiset-based variant proves to be at least as powerful as the set-based variant, while the tuple-based variant of $k$-WL demonstrates at least the same power as the multiset-based variant. To further strengthen our findings, we employ the CFI constructions (3.3) and demonstrate that the $\delta$-$k$-WL$\{\{\cdot\}\}$ algorithm is strictly more powerful than the $k$-WL$\{\{\cdot\}\}$ algorithm. Finally, we describe the challenges in establishing a hierarchical result for the multiset-based $k$-WL algorithm. As a solution, we propose a slight modification to the $k$-WL$\{\{\cdot\}\}$ algorithms, which not only achieves comparable performance but also yields results similar to those established in Proposition 3.3.2.

**Proposition 4.3.1.** *$k$-FWL $\sqsubseteq$ $k$-WL$\{\{\cdot\}\}$ .*

*Proof.* Denote $C_l^m$ and $C_l^{\mathsf{F}}$ the coloring functions of the $k$-WL$\{\{\cdot\}\}$ and $k$-FWL, respectively. We further define the coloring function $C_l' : V(G)^k \to \Sigma$ as follows:

$$C_l'(u) = C_l^m(\textit{multiset}(\mathbf{u})), \textit{ for } \mathbf{u} \in V(G)^k.$$

**Claim 3.** *For two graphs $G, H$, if $k$-WL$\{\{\cdot\}\}$ distinghuishes $G, H$, then the coloring function $C_l'$ also distinghuishes $G$ and $H$.*

If $k$-WL$\{\{\cdot\}\}$ distinghuishes $G$ and $H$, then there exists a color $c$ and a color class $\hat{S_c} = \{A_1, \dots, A_p\}$ with $A_1, \dots, A_p \in [[V(G)]]^k \dot\cup [[V(H)]]^k$, such that,

$$|\hat{S_c} \cap [[V(G)]]^k| \neq |\hat{S_c} \cap [[V(H)]]^k|$$

in some iteration $l$. We denote $m_1 = |\hat{S_c} \cap [[V(G)]]^k|$ and $m_2 = |\hat{S_c} \cap [[V(H)]]^k|$. Now, by definition of $C_l'$ we know that the coloring function $C_l'$ uses the same colors with the coloring function $C_l^m$. Therefore, we define the color class of the color $c$ as the set of all tuples colored $c$ by $C_l'$. That is, $S_c = \{\mathbf{u} \in V(G)^k \dot\cup V(H)^k \mid C_l'(\mathbf{u}) = c\}$. Additionally note that, a tuple $\mathbf{u} \in V(G)^k$ if and only if there exists a multiset $M \in [[V(G)]]^k$ such that $M = \textit{multiset}(\mathbf{u})$. Now, by comparing isomorphism types, we imply that all multisets in $\hat{S_c}$ have the same multiplicity profiles. Therefore, there exists a $q \in \mathbb{N}$ such that for each multiset $M \in \hat{S_c} \cap [[V(G)]]^k$ we can find exactly $q$ different tuples $\mathbf{u}_1, \dots, \mathbf{u}_q \in V(G)^k$ such that

$$\textit{multiset}(\mathbf{u}_1) = \cdots = \textit{multiset}(\mathbf{u}_q) = M$$

and by definition of $C_l'$ we have,

$$C_l'(\mathbf{u}_1) = \cdots = C_l'(\mathbf{u}_q) = c.$$

Hence, we have found $m_1 \cdot q$ different tuples colored $c$ in $V(G)^k$ and we prove that there are not other tuples with this property. If there exists a $w \in V(G)^k$ with $C'_l(w) = c$ and *multiset*$(w) \neq M$ for all $M \in [[V(G)]]^k$, then $|\hat{S_c} \cap [[V(G)]]^k| > m_1$ which is a condradiction. Using the same arguments we can prove that there are exactly $m_2 \cdot q$ tuples colored $c$ in $V(H)^k$. Finally,

$$m_1 q = |S_c \cap [[V(G)]]^k| \neq |S_c \cap [[V(H)]]^k| = m_2 q$$

which completes the proof of Claim 3.

**Claim 4.** $C_l^F \sqsubseteq C'_l$

We prove the Claim 4 by induction on $l$. For $l = 0$, it is clear by comparing isomorphism types.
(I.H): $C_l^{\mathsf{F}}(\mathbf{u}) = C_l^{\mathsf{F}}(\mathbf{v}) \implies C'_l(\mathbf{u}) = C'_l(\mathbf{v})$.
Now, if $C_{l+1}^{\mathsf{F}}(\mathbf{u}) = C_{l+1}^{\mathsf{F}}(\mathbf{v})$ by definition we have that $C_l^{\mathsf{F}}(\mathbf{u}) = C_l^{\mathsf{F}}(\mathbf{v})$ and by the inductive hypothesis (I.H)

$$C'_l(\mathbf{u}) = C'_l(\mathbf{v})$$

or

$$C_l^m(\textit{multiset}(\mathbf{u})) = C_l^m(\textit{multiset}(\mathbf{v})) \tag{4.1}$$

In addition we have that

$$\{\!\{\textit{sift}(C_l^{\mathsf{F}}, \mathbf{u}, \omega) \mid \omega \in V(G)\}\!\} = \{\!\{\textit{sift}(C_l^{\mathsf{F}}, \mathbf{v}, \omega) \mid \omega \in V(G)\}\!\}$$

Therefore, there exists a bijection $\sigma : V(G) \to V(G)$ such that:

$$C_l(\phi_j(\mathbf{u}, \omega) = C_l(\phi_j(\mathbf{v}, \sigma(\omega)),\ \forall j \in [k]$$

We now, define a bijection $f$ from the set of neighbors of *multiset*$(\mathbf{u})$ to the set of neighbors of *multiset*$(\mathbf{v})$ such that $C_l^m(A) = C_l^m(f(A))$ for every neighbor $A$ of *multiset*$(\mathbf{u})$ which completes the proof of Claim 4. For a neighbor A of *multiset*$(\mathbf{u})$ we denote

$$\omega_0 := A \setminus \textit{multiset}(\mathbf{u}) \text{ and } j_0 := \min\{j \in [k] \mid A = \textit{multiset}(\phi_j(\mathbf{u}, \omega_0))\}$$

Then, we define the function $f$ as: $f(A) := \textit{multiset}(\phi_{j_0}(\mathbf{v}, \sigma(\omega_0)))$. We have that $f$ is a well defined function with the desired properties. To see that, first note that comparing the isomorphism types of *multiset*$(\mathbf{u})$ and *multiset*$(\mathbf{v})$ we have that:

$$|\{\textit{neighbors of multiset}(\mathbf{u})\}| = |\{\textit{neighbors of multiset}(\mathbf{v})\}|$$

Therefore, it suffices to show that $f$ is an injection. Indeed if for two neighbors $A_1, A_2$ of *multiset*$(\mathbf{u})$ we have $f(A_1) = f(A_2)$ or equivalently

$$\textit{multiset}(\phi_{j_1}(\mathbf{v}, \sigma(\omega_1))) = \textit{multiset}(\phi_{j_2}(\mathbf{v}, \sigma(\omega_2))),$$

then by the equality of the multisets we have that

$$\sigma(\omega_1) = \sigma(\omega_2) \textit{ and } v_{j_1} = v_{j_2}$$

Therefore, since $\sigma$ is a bijection and **u** has the same isomorphism type with **v**, we have that

$$\omega_1 = \omega_2 \textit{ and } u_{j_1} = u_{j_2}$$

which implies

$$\underbrace{\textit{multiset}(\phi_{j_1}(\mathbf{u}, \omega_1))}_{A_1} = \underbrace{\textit{multiset}(\phi_{j_2}(\mathbf{u}, \omega_2))}_{A_2}$$

■

**Proposition 4.3.2.** $k$-*OWL* $\sqsubseteq$ $k$-*WL*$\{\!\{\cdot\}\!\}$

*Proof.* Denote $C_l^m$ and $C_l^{\mathsf{O}}$ the coloring functions of the $k$-WL$\{\!\{\cdot\}\!\}$ and $k$-OWL, respectively. We further define the coloring function $C_l' : V(G)^k \to \Sigma$ as follows:

$$C_l'(\mathbf{u}) = C_l^m(\textit{multiset}(\mathbf{u}))$$

By Claim 3 of Proposition 4.3.1 we know that: For two graphs $G, H$, if $k$-WL$\{\!\{\cdot\}\!\}$ distinguishes $G, H$, then the coloring function $C_l'$ also distinguishes $G$ and $H$. We complete the proof with the following Claim which is similar to Claim 4 of Proposition 4.3.1.

**Claim 5.** $C_l^{\mathsf{O}} \sqsubseteq C_l'$

We prove Claim 5 by induction on $l$. For $l = 0$, it is clear by comparing isomorphism types.
(I.H): $C_l^{\mathsf{O}}(\mathbf{u}) = C_l^{\mathsf{O}}(\mathbf{v}) \implies C_l'(\mathbf{u}) = C_l'(\mathbf{v})$
Now, if $C_{l+1}(\mathbf{u}) = C_{l+1}(\mathbf{v})$ by definition we have that $C_l^{\mathsf{O}}(\mathbf{u}) = C_l^{\mathsf{O}}(\mathbf{v})$ and by the inductive hypothesis (I.H)

$$C_l'(\mathbf{u}) = C_l'(\mathbf{v})$$

or

$$C_l^m(\textit{multiset}(\mathbf{u})) = C_l^m(\textit{multiset}(\mathbf{v})) \tag{4.2}$$

In addition we have that

$$\{\!\{C_l^{\mathsf{O}}(\phi_j(\mathbf{u}, \omega)) \mid \omega \in V(\Gamma)\}\!\} = \{\!\{C_l^{\mathsf{O}}(\phi_j(\mathbf{v}, \omega)) \mid \omega \in V(G)\}\!\},\ \forall j \in [k]$$

Therefore, for each $j \in [k]$ there exists a bijection $\sigma_j : V(G) \to V(G)$ such that:

$$C_l^{\mathsf{O}}(\phi_j(\mathbf{u}, \omega)) = C_l^{\mathsf{O}}(\phi_j(\mathbf{v}, \sigma_j(\omega))),\ \forall \omega \in V(G)$$

We now define a bijection $f$ from the set of neighbors of *multiset*(**u**) to the set of neighbors of *multiset*(**v**) such that $C_l^m(A) = C_l^m(f(A))$ for every neighbor $A$ of *multiset*(**u**). For a neighbor A of *multiset*(**u**) we denote

$$\omega_0 := A \setminus \textit{multiset}(\mathbf{u}) \text{ and } j_0 := \min\{j \in [k] \mid A = \textit{multiset}(\phi_j(\mathbf{u}, \omega_0))\}$$

Then, we define the function $f$ as: $f(A) := \textit{multiset}(\phi_{j_0}(\mathbf{v}, \sigma_{j_0}(\omega_0)))$. We have that $f$ is a well defined function with the desired properties. To see that, first note that comparing the isomorphism types of *multiset*(**u**) and *multiset*(**v**) we have that:

$$|\{\textit{neighbors of multiset}(\mathbf{u})\}| = |\{\textit{neighbors of multiset}(\mathbf{v})\}|$$

Therefore, it suffices to show that $f$ is an injection. Indeed if for two neighbors $A_1, A_2$ of *multiset*(**u**) we have that $f(A_1) = f(A_2)$ or equivalently

$$\textit{multiset}(\phi_{j_1}(\mathbf{v}, \sigma_{j_1}(\omega_1))) = \textit{multiset}(\phi_{j_2}(\mathbf{v}, \sigma_{j_2}(\omega_2))),$$

then by the equality of the multisets we have that

$$\sigma_{j_1}(\omega_1) = \sigma_{j_2}(\omega_2) \textit{ and } v_{j_1} = v_{j_2}$$

But if $v_{j_1} = v_{j_2}$ by the isomorphism types of **u** and **v** we have that also $u_{j_1} = u_{j_2}$. Now, by definition of $j_1$,$j_2$ we conclude that $j_1 = j_2$, since if $j_1 < j_2$ (wlog), then by definition of $j_2$ we would have that $u_{j_1} \neq u_{j_2}$ Therefore, the equality $\sigma_{j_1}(\omega_1) = \sigma_{j_2}(\omega_2)$ becomes $\sigma_{j_1}(\omega_1) = \sigma_{j_1}(\omega_2)$ and since $\sigma_{j_1}$ is a bijection we have that $\omega_1 = \omega_2$ and hence $A_1 = A_2$. ■

By combining Proposition 4.3.2 with Proposition 3.3.3, we obtain the following result.

**Corollary 4.3.3.** $k$-*FWL* $\sqsubset$ $k$-*WL*$\{\!\{\cdot\}\!\}$

It is reasonable to anticipate that the set-based variant of the $k$-WL algorithm is less powerful than the multiset-based variant. The following result precisely demonstrates this by simulating sets in $[V(G)]^k$ with the corresponding sets in $[[V(G)]]^k$.

**Proposition 4.3.4.** $k$-*WL*$\{\!\{\cdot\}\!\} \sqsubseteq k$-*WL*$\{\cdot\}$

*Proof.* Denote $C_l^m$ and $C_l^s$ the coloring functions induced by $k$-WL$\{\!\{\cdot\}\!\}$ and $k$-WL$\{\cdot\}$, respectively on each iteration $l$. It is easy to see that $C_l^m$ induces a coloring function $C_l'$ on $[V(G)]^k$ as:

$$C_l'(S) = C_l^m(S)$$

What is more, $C_l^m$ defines a partition in $[V(G)]^k$ because due to the isomorphism types of the sets, there are not initial color classes in $[[V(G)]]^k$ containing both sets and strictly multisets. So, since in each step the coloring function $C_l^m$ defines a finer partion, we know that $C_l^m$ defines a partition on $[V(G)]^k$. Therefore, if $k$-WL$\{\!\{\cdot\}\!\}$ fails to distinghuish between two graphs $G$ and $H$, then the coloring function $C_l'$ also fails to distinghuish $G$ and $H$. Hence, it suffices to show that $C_l' \sqsubseteq C_l^S$. We prove this by induction on $l$. For $l = 0$ it is clear comparing the isomorphism types. Now, if $C_{l+1}'(A) = C_{l+1}'(B)$ by definition of $C_l'$ and the inductive hypothesis we have that

$$C_l^s(A) = C_l^s(B) \tag{4.3}$$

In addition, we have that $\{\!\{C_l^m(T) \mid T \sim A\}\!\} = \{\!\{C_l^m(T) \mid T \sim B\}\!\}$. Therefore, we can find a bijection $\sigma$ between these two multisets such that $C_l^m(T) = C_l^m(\sigma(T))$, $\forall T \sim A$. Similarly comparing isomorphism types, we can see that $\sigma$ also maps sets to sets. Therefore, $\{\!\{C_l^m(T) \mid T \text{ is set and } T \sim A\}\!\} = \{\!\{C_l^m(T) \mid T \text{ is set and } T \sim B\}\!\}$ and by the inductive hypothesis:

$$\{\!\{C_l^s(T) \mid T \sim A\}\!\} = \{\!\{C_l^s(T) \mid T \sim B\}\!\} \tag{4.4}$$

Equations 4.3,4.4 complete the proof. ■

Up to this point, our comparison findings are limited in showing refinement relations between algorithms and not strictly. We have not seen yet examples with strictly refinement relation between two

algorithms as the one discussed in Proposition 3.3.7 between $\delta$-$k$-WL and $k$-WL. The challenge lies in constructing pairs of graphs that cannot be distinguished by higher-order algorithms. In this section, we will utilize the construction introduced in Section 3.3 and we introduce the unrolling tools defined in Section 4.2 to prove that the $\delta$-$k$-WL$\{\{\cdot\}\}$ refinement algorithm strictly refines the $k$-WL$\{\{\cdot\}\}$ algorithm.

**Lemma 4.3.5** (**Unrollings characterization**). *If $A, B \in [[V(G)]]^k$ and $l \geq 0$ then,*

$$C_l(A) = C_l(B) \Leftrightarrow \textit{UNR}[G, A, l] \cong \textit{UNR}[G, B, l]$$

*Where $C_l$ and UNR$[G, A, l]$ are the coloring function and the unrolling tree induced by $k$-WL$\{\{\cdot\}\}$, respectively.*

*Proof.* We prove separately the two directions of this lemma by induction on $l$. For $l = 0$ it is easy to see that both directions are true due the isomorphism types.

($\Rightarrow$) For the inductive hypothesis we assume that for $S, T \in [[V(G)]]^k$ we have that $C_l(S) = C_l(T) \Rightarrow$ UNR$[G, S, l] \cong$ UNR$[G, T, l]$. Now, if $C_{l+1}(S) = C_{l+1}(T)$ we have that $C_l(S) = C_l(T)$ and $\{\{C_l(\tilde{S} \mid \tilde{S} \sim S\}\} = \{\{C_l(\tilde{T} \mid \tilde{T} \sim T\}\}$. Therefore, there exists a bijection $\sigma$ between the above multisets such that:

$$C_l(\tilde{S}) = C_l(\sigma(\tilde{S})),\ \forall \tilde{S} \sim S.$$

Hence, by the inductive hypothesis we have that UNR$[G, \tilde{S}, l] \cong$ UNR$[G, \sigma(\tilde{S}), l]$. Therefore, for each $\tilde{S} \sim S$ there exists an isomorphism $\theta_{\tilde{S}}$ between UNR$[G, \tilde{S}, l]$ and UNR$[G, \sigma(\tilde{S}), l]$. If we define $\theta$ as $\theta := \dot{\bigcup}_{\tilde{S} \sim S} \theta_{\tilde{S}}$ and extend $\theta$ to map the root of UNR$[G, S, l+1]$ to the root of UNR$[G, T, l+1]$, we have that $\theta$ is an isomorphism between UNR$[G, S, l+1]$ and UNR$[G, T, l+1]$

($\Leftarrow$) For the inductive hypothesis we assume that for $S, T \in [[V(G)]]^k$ we have that UNR$[G, S, l] \cong$ UNR$[G, T, l] \Rightarrow C_l(S) = C_l(T)$. Now, if UNR$[G, S, l+1] \cong$ UNR$[G, T, l+1]$. We define a partition between all the subtrees appended to the root of UNR$[G, S, l+1]$ according to their isomorphism types. Say $A_1 \dot{\cup} A_2 \dot{\cup} \ldots \dot{\cup} A_p$ the partion of these subtrees. Similarly, since UNR$[G, S, l+1] \cong$ UNR$[G, T, l+1]$ all the subtrees appended to the root of UNR$[G, T, l+1]$ are partioned according to their isomorphism types to $p$ sets denoted by $B_1 \dot{\cup} B_2 \ldots \dot{\cup} B_p$ such that $|A_i| = |B_i|$ and for all subtrees, (with depth $l$), $x \in A_i$ and $y \in B_i$ we have that $x \cong y$. Therefore, by the inductive hypothesis, the color profile of the neighbors of $S$ is the same with the one of the neighbors of $T$. Finally, UNR$[G, S, l+1] \cong$ UNR$[G, T, l+1] \Rightarrow$ UNR$[G, S, l] \cong$ UNR$[G, T, l]$ (by definition of the unrollings) and hence by the inductive hypothesis $C_l(S) = C_l(T)$. Therefore, $S, T$ have the same color in iteration $l$ and their neighbors have the same color profile. ■

Below, we present an extended version of the above lemma for all the different variants of set and multiset-based $k$-WL.

**Lemma 4.3.6.** *Consider the following cases:*

1. *Let $C_l$ and $\delta - UNR[G, A, l]$ denote the coloring function and the unrolling tree induced by $\delta - k - WL\ \{\{\cdot\}\}$, respectively, where $A, B \in [[V(G)]]^k$.*
2. *Let $C_l$ and $L - UNR[G, A, l]$ denote the coloring function and the unrolling tree induced by $\delta - k - LWL\ \{\{\cdot\}\}$, respectively, where $A, B \in [[V(G)]]^k$.*
3. *Let $C_l$ and UNR$[G, A, l]$ denote the coloring function and the unrolling tree induced by $k - WL\ \{\cdot\}$, respectively, where $A, B \in [V(G)]^k$.*

4. *Let $C_l$ and $\delta - UNR[G, A, l]$ denote the coloring function and the unrolling tree induced by $\delta - k - WL\ \{\cdot\}$, respectively, where $A, B \in [V(G)]^k$.*

5. *Let $C_l$ and $L - UNR[G, A, l]$ denote the coloring function and the unrolling tree induced by $\delta - k - LWL\ \{\cdot\}$, respectively, where $A, B \in [V(G)]^k$.*

*For all the above cases, the following holds:*

$$C_l(A) = C_l(B) \Leftrightarrow UNR[G, A, l] = UNR[G, B, l]$$

*This implies that the coloring function outputs the same color for sets $A$ and $B$, if and only if the unrolling tree induced by $G$ and $A$ at layer $l$ is equal to the unrolling tree induced by $G$ and $B$ at layer $l$.*

*Proof.* We omit the proof as it follows a similar structure to the proof of Lemma 4.3.5. ■

**Definition 3.** *In a graph $G = (V, E)$, a set $S$ of vertices is said to form a distance-two-clique if the distance between any two vertices in S is exactly two.*

**Lemma 4.3.7** (Morris et al. [2020b])**.** *The following holds for the CFI construction (desribed in 3.3 ) graphs $G_k$ and $H_k$ defined above.*

- *There exists a distance-two-clique of size $(k+1)$ inside $G_k$ with the following form $(0, S_0), \ldots, (k, S_k)$*
- *There does not exist a distance-two-clique of size $(k+1)$ inside $H_k$ with the following form $(0, S_0), \ldots, (k, S_k)$*

*Proof.* In $G$, consider the vertex subset $S$ where $S = \{(0, \emptyset), \ldots, (k, \emptyset)\}$.

**Claim 6.** *S forms a distance-two-clique of size $(k+1)$.*

Let $K$ be the complete graph on $k+1$ nodes. Then for all $i, j \in V(K)$ we know that $(i, \emptyset)$ is not adjacent to $(j, \emptyset)$. Which means that $(i, \emptyset), (j, \emptyset)$ have distance at least two. On the other hand, the node $\{i, j\}_0$ is adjacent to both $(i, \emptyset)$ and $(j, \emptyset)$ since $i, j \in \{i, j\}$ and $\{i, j\} \notin \emptyset$. Therefore, $(i, \emptyset)$ and $(j, \emptyset)$ are in distance two for all $i, j \in V(K)$

**Claim 7.** *There does not exists a distance-two-clique of size $(k+1)$ inside $H_k$ with the following form $(0, S_0), \ldots, (k, S_k)$*

We assume for contradiction that there exist $k+1$ vertices in $V(H)$, say $(0, S_0), \ldots (k, S_k)$, which forms a distance-two-clique. Then we compute the sum $|S_0| + \cdots + |S_k| \pmod 2$ with two different ways. First note that since $|S_0|$ is odd and $|S_i|$ is even for all $0 < i \leq k$ we have $|S_0| + \cdots + |S_k| \equiv 1, \pmod 2$. On the other hand, for all $i, j \in V(K)$ since $(i, S_i)$ and $(j, S_j)$ are in distance two, we have that either $\{i, j\} \in (S_i \cap S_j)$ or $\{i, j\} \in (V(K) \setminus S_i) \cap (V(K) \setminus S_j)$. Hence, the sum contribution of $S_i$ and $S_j$ in $\pmod 2$ to the term corresponding to $\{i, j\}$ is zero. Since the contribution of each edge to the total sum is 0 in $\pmod 2$, the total sum must be zero in $\pmod 2$. This is a contradiction, and hence, there does not exist a distance-two-clique in $H$. ■

By Construction 3.3 since for $u_1 \neq u_2$ the nodes $(u_1, S_1)$ and $(u_2, S_2)$ have different labels we know that if $\phi$ was an isomorphism between $G$ and $H$ then $\phi(\{(0, \emptyset), \ldots, (k, \emptyset)\}) = \{(0, S_0), \ldots, (k, S_k)\}$. But from Lemma 4.3.7 it is impossible to find such a function $\phi$ therefore, we have the following result.

**Corollary 4.3.8.** *$G_k$ and $H_k$ are not isomorphic graphs.*

**Proposition 4.3.9.** *$\delta$-$k$-LWL$\{\{\cdot\}\}$ distinguishes $G_k$,$H_k$*

*Proof.* We denote $P = \{\{(1,\emptyset),\dots,(k,\emptyset)\}\} \in [[V(G_k)]]^k$. We will show that there is not $Q \in [[V(H_k)]]^k$ such that $C_{l_\infty}(P) = C_{l_\infty}(Q)$ in the stable coloring of the $\delta$-$k$-LWL$\{\{\cdot\}\}$ algorithm. Equivalently, by Lemma 4.3.6 we will show that there is not $Q \in [[V(H_k)]]^k$ such that $L-\mathsf{UNR}[G_k,P,l_\infty] \cong L-\mathsf{UNR}[H_k,Q,l_\infty]$. If there exists a $Q \in [[V(H_k)]]^k$ such that $L-\mathsf{UNR}[G_k,P,l_\infty] \cong L-\mathsf{UNR}[H_k,Q,l_\infty]$ throught an isomorphism called $\phi$, then comparing isomorphism types we know that $Q$ must be of the following form:

$$Q = \phi(P) = \{\{(1,S_1),\dots,(k,S_k)\}\}.$$

If we consider the unrolling in depth two, for $P_1 = \{\{(2,\emptyset),(2,\emptyset),\dots,(k,\emptyset)\}\}$, by construction we have that $P$ is not adjacent to $P_1$, but both $P$ and $P_1$ are adjacent to $\bar{P} = \{\{\{1,2\}^0,(2,\emptyset),\dots,(k,\emptyset)\}\}$. Therefore, $P$ and $P_1$ are in distance two. Therefore, $Q = \phi(P)$ and $\phi(P_1)$ are also in distance two. But comparing the isomorphism types of $P_1$ and $\phi(P_1)$ we know that:

$$\phi(P_1) = \{\{(2,S_2),(2,S_2),\dots,(k,S_k)\}\}$$

and hence $(1,S_1)$ and $(2,S_2)$ are in distance two. Repeating this process we get that the set $\{(1,S_1),(2,S_2),\dots,(k,S_k)\}$ forms a distance two clique of size $k$ in $H_k$. If we consider the unrolling in depth four and

$$R = \{\{(0,\emptyset),(2,\emptyset),\dots,(k,\emptyset)\}\},$$

then similarly $P$ and $R$ are in distance two and there exists an $S_0 \subset E(0)$ such that

$$\phi(R) = \{\{(0,S_0),(2,S_2),\dots,(k,S_k)\}\}$$

hence $\phi(R), Q$ are also in distance two which implies that $(0,S_0),(1,S_1)$ are in distance two. Denote:

$$R_j = \{\{(0,\emptyset),(2,\emptyset),\dots,(j-1,\emptyset),(0,\emptyset),(j+1,\emptyset),\dots,(k,\emptyset)\}\}.$$

Following the same argumentation we have that $R, R_j$ are in distance two, $\phi(R),\phi(R_j)$ are in distance two and $(0,S_0),(j,S_j)$ are in distance two for all $j \geq 2$. Therefore, $\{(0,S_0),(1,S_1),\dots,(k,S_k)\}$ forms a distance two clique of size $(k+1)$ in $V(H_k)$ which contradicts Lemma 4.3.7.

■

**Proposition 4.3.10.** *$\delta$-$k$-WL$\{\{\cdot\}\} \sqsubset k$-WL$\{\{\cdot\}\}$*

*Proof.* It is easy to see that $\delta$-$k$-WL$\{\{\cdot\}\}$ is at least as powerful as both $k$-WL$\{\{\cdot\}\}$ and $\delta$-$k$-LWL$\{\{\cdot\}\}$. Now, from Proposition 4.3.2 we know that $k$-OWL $\sqsubseteq$ $k$-WL$\{\{\cdot\}\}$ and by Cai et al. [1992] $k$-OWL does not distinguish between the CFI constructions $G_k, H_k$. Therefore, $\delta$-$k$-WL$\{\{\cdot\}\}$ distinguishes $G_k, H_k$ (since $\delta$-$k$-LWL$\{\{\cdot\}\}$ does by Proposition 4.3.9) and $k$-WL$\{\{\cdot\}\}$ does not. ■

We now introduce the $k$-WL$\{\{\cdot\}\}^*$ which is a slightly extended modification of $k$-WL$\{\{\cdot\}\}$ algorithm that also forms a hierarchy. That is increasing the value of $k$ leads to more powerful algorithms in terms of distinguishing non-isomorphic graphs. As in the proof of Proposition 3.3.2, in order two compare two color refinement algorithms which partition different spaces, we first need to simulate all $k$-element multisets with $(k+1)$-element multisets. The absence of node ordering in a multiset does not allow us to simulate $k$-element multisets just by copying an element of the multiset as we did in the tuple version. Therefore, to overcome this, we introduce a special node on each graph and we define an

extended version of a graph.

For a labeled graph $(G, l)$ with $l : V(G) \to \Sigma$ we define the star-extension of graph $(G, l)$ as a new labeled graph $(G^*, l^*)$ defined as follows: $V(G^*) = V(G) \dot\cup \{*\}$, $E(G^*) = E(G)$, $l^*(u) = l(u)$ , $\forall u \in V(G)$ and $l^*(*) = \sigma^*$ where $\sigma^*$ is a special color such that $\sigma^* \notin l(V(G))$.

For a pair of graphs $(G, l_G)$ and $(H, l_H)$ we define the star-extension pair as $(G^*, l_G^*)$ and $(H^*, l_H^*)$ where in both graphs we have added the $(*)$ node with degree zero and a special color $\sigma^* \notin l_G(V(G)) \cup l_H(V(H))$. It is easy to see that $G$ and $H$ are isomorphic if and only if $G^*$ and $H^*$ are isomorphic too. In addition, we have the following lemma.

**Lemma 4.3.11.** *For a pair of labeled graphs $(G, l_G)$, $(H, l_H)$, if the $k$-WL$\{\{\cdot\}\}$ algorithm distinguishes the star extension pair of graphs $(G^*, l_G^*)$, $(H^*, l_H^*)$, then it also distinguishes $(G, l_G)$ and $(H, l_H)$.*

*Proof.* Let $G, H$ be two finite graphs. If $k$-WL$\{\{\cdot\}\}$ fails to distinguish the star extension pair $G^*, H^*$ of $G$ and $H$, respectively then there exists a bijection $f : [[V(G^*)]]^k \to [[V(H^*)]]^k$ such that:

$$C_l^k(S) = C_l^k(f(S)),\ \forall S \in [[V(G^*)]]^k,$$

where $C_l^k$ is the coloring function of $k$-WL$\{\{\cdot\}\}$ in iteration $l$. Now for each $S \in [[V(G^*)]]^k$ such that $(*) \notin S$ (i.e., $S \in [[V(G)]]^k$) if we denote $T_1 = \text{UNR}[G^*, S, l]$ and $T_2 = \text{UNR}[H^*, f(S), l]$ by Lemma 4.3.5 we have that $T_1 \cong T_2$. Now note that every isomorphism between $T_1$ and $T_2$ maps nodes with isomorphic types containing the $(*)$ node to nodes with isomorphic types containing the $(*)$ node. Therefore, if we remove every node from $T_1$ containing the $(*)$ node in its isomorphic type along with its children, and similarly for $T_2$, we derive the trees $\tilde{T_1}$ and $\tilde{T_2}$, respectively where $\tilde{T_1} \cong \tilde{T_2}$. Finally it is easy to note that $\tilde{T_1} = \text{UNR}[G, S, l]$ and $\tilde{T_2} = \text{UNR}[H, f(S), l]$. Therefore, for the function $f$ constrained on $[[V(G)]]^k$ we have that is a bijection between $[[V(G)]]^k$ and $[[V(H)]]^k$ such that $C_l^k(S) = C_l^k(f(S))$ for all $S \in [[V(G)]]^k$. Hence $k$-WL$\{\{\cdot\}\}$ fails to distinguish graphs $G, H$. ■

We now define the modification of the $k$-WL$\{\{\cdot\}\}$ algorithm named $k$-WL$\{\{\cdot\}\}^*$. The $k$-WL$\{\{\cdot\}\}^*$ algorithm is exactly the $k$-WL$\{\{\cdot\}\}$ implemented to the star extension pair of the graphs. From Lemma 4.3.11 we already have that $k$-WL$\{\{\cdot\}\}^* \sqsubseteq k$-WL$\{\{\cdot\}\}$. We proceed to the proof that this new algorithm forms a hierarchy.

**Proposition 4.3.12.** $(k+1)$-*WL*$\{\{\cdot\}\}^* \sqsubseteq k$-*WL*$\{\{\cdot\}\}^*$

*Proof.* We denote $C_l^{k+1}$,$C_l^k$ the coloring functions in iteration $l$ of the algortihms $(k+1)$-WL$\{\{\cdot\}\}^*$ and $k$-WL$\{\{\cdot\}\}^*$, respectively. We further define the coloring function $C_l' : [[V(G^*)]]^k \dot\cup [[V(H^*)]]^k \to \Sigma$ as $C_l'(S) = C_l^{k+1}(S \cup \{*\})$

**Claim 8.** *If $C_l^{k+1}$ fails to distinguish the pair of graphs $G^*$ and $H^*$, then $C_l'$ also fails to distinguish $G^*$ and $H^*$.*

By applying the algorithm $(k+1)$-WL$\{\{\cdot\}\}$ in $G^*$, $H^*$ (or equivalently the $k$-WL$\{\{\cdot\}\}^*$ algorithm in $G$ and $H$), we define a partition on the set $\mathcal{M}_G = \{M \in [[V(G^*)]]^{k+1} \mid (*) \in M\}$ and similarly on $\mathcal{M}_H = \{M \in [[V(H^*)]]^{k+1} \mid (*) \in M\}$, where $(*) \in M$ means that the multiset $M$ contains the $(*)$ node. This happens because due the isomorphism types, every multiset which contains the special node $(*)$ cannot be in the same color class with a multiset that does not contain the special node $(*)$ in the initial coloring. Additionaly, $\mathcal{M}_G$ and $\mathcal{M}_H$ are bijectively mapped to $[[V(G^*)]]^k$ and $[[V(H^*)]]^k$, respectively

throught the function $S \mapsto S \setminus \{(*)\}$. Therefore, since $C_l^{k+1}$ fails to distinguish $G^*$ and $H^*$, we know that $|S_c \cap \mathcal{M}_G| = |S_c \cap \mathcal{M}_H|$ for every color class $S_c$ induced by $C_l^{k+1}$. Hence, by the definition of $C_l'$ we imply that $C_l'$ also fails to distinghuish between $G^*$ and $H^*$.

**Claim 9.** $C_l' \sqsubseteq C_l^k$.

We prove Claim 9 by induction on $l$. For $l = 0$ it is clear. We assume that $C_l'(A) = C_l'(B)$ implies $C_l^k(A) = C_l^k(B)$. Now, if $C_{l+1}'(A) = C_{l+1}'(B)$, i.e. $C_{l+1}^{k+1}(A \cup \{(*)\}) = C_{l+1}^{k+1}(B \cup \{(*)\})$ by definition we have that

$$C_l^{k+1}(A \cup \{(*)\}) = C_{l+1}^{k+1}(B \cup \{(*)\}) \tag{4.5}$$

and

$$\{\!\{C_l^{k+1}(S) \mid S \sim A \cup \{(*)\}\}\!\} = \{\!\{C_l^{k+1}(T) \mid T \sim B \cup \{(*)\}\}\!\}. \tag{4.6}$$

By the inductive hypothesis Equation 4.5 becomes

$$C_l^k(A) = C_l^k(B). \tag{4.7}$$

Now, by Equation 4.6, we can find a bijection $\sigma$ from $\{S \mid S \sim A \cup \{(*)\}\}$ to $\{T \mid S \sim B \cup \{(*)\}\}$ such that, $C_l^{k+1}(S) = C_l^{k+1}(\sigma(S))$ for all $S \in \{S \mid S \sim A \cup \{(*)\}\}$. Now, by comparing isomorphism types we have that $\sigma$ maps multisets containing the $(*)$ node to multisets containing the $(*)$ node which means that

$$\sigma(\{S \mid S \sim A \cup \{(*)\}, (*) \in S\}) = \{T \mid T \sim B \cup \{(*)\}, (*) \in T\}. \tag{4.8}$$

Now, it is easy to verify that

$$\{S \mid S \sim A\} = \{S \setminus \{(*)\} \mid S \sim A \cup \{(*)\}, (*) \in S\}. \tag{4.9}$$

By Equation 4.8 we have that

$$\{\!\{C_l^{k+1}(S) \mid S \sim A \cup \{(*)\}, (*) \in S\}\!\} = \{\!\{C_l^{k+1}(T) \mid T \sim B \cup \{(*)\}, (*) \in T\}\!\}$$

and by definition of $C_l'$ this implies,

$$\{\!\{C_l'(S \setminus \{(*)\}) \mid S \sim A \cup \{(*)\}, (*) \in S\}\!\} = \{\!\{C_l'(T \setminus \{(*)\}) \mid T \sim B \cup \{(*)\}, (*) \in T\}\!\}.$$

Therefore, by Equation 4.9

$$\{\!\{C_l'(S) \mid S \sim A\}\!\} = \{\!\{C_l'(T) \mid T \sim B\}\!\}$$

and by the inductive hypothesis

$$\{\!\{C_l^k(S) \mid S \sim A\}\!\} = \{\!\{C_l^k(T) \mid T \sim B\}\!\} \tag{4.10}$$

Equations 4.7, 4.10 completes the proof of Claim 9.

Finally, we have shown that if $(k+1)$-WL$\{\!\{\cdot\}\!\}^*$ fails to distinguish $G$ and $H$, by Claim 8 $C_l'$ also fails to distinguish $G^*$ and $H^*$ and by Claim 9 $k$-WL$\{\!\{\cdot\}\!\}^*$ fails to distinguish $G$ and $H$. ■

## 4.4. Experimental results

This section focuses on the implementation and analysis of graph isomorphism algorithms to gain a comprehensive understanding of their comparative capabilities. Throughout our exploration, we have observed that certain algorithms exhibit superiority over others. For instance, our investigations have revealed that $\delta$-2-WL$\{\cdot\}$ is at least as powerful as 2-WL$\{\cdot\}$, indicating that any graphs distinguishable by 2-WL$\{\cdot\}$ can also be distinguished by $\delta$-2-WL$\{\cdot\}$. However, a fundamental question arises: Do there exist pairs of graphs that can be distinguished by $\delta$-2-WL$\{\cdot\}$ but not by 2-WL$\{\cdot\}$, or are these two algorithms are equivalent in their discriminative abilities?

So far, Theorem 4.3.10 is the only result that establishes a strict hierarchy among (multi)set-based algorithms. To rigorously investigate this question, we implement the $\delta$-$k$-LWL$\{\cdot\}$, $\delta$-$k$-WL$\{\cdot\}$, $\delta$-$k$-WL$\{\{\cdot\}\}$, and $\delta$-$k$-WL$\{\{\cdot\}\}$ algorithms in Python and conduct extensive experimentation. For computational efficiency, we restrict our consideration to algorithms for $k = 2$ or $k = 3$, as graphs with 2 or 3 nodes can be uniquely identified by their total number of edges. This crucial observation leads to the development of rapid algorithms for computing the initial coloring function. Our objective is to meticulously analyze the outcomes and identify concrete examples that clearly illustrate the advantages and disadvantages of the proposed algorithms. The source code for implementing these algorithms can be found at `https://github.com/antvas98/set_based_GNNs`.

The pair of graphs depicted in Figure 4.5 serves as an illustrative example. It can be distinguished by $\delta$-2-LWL$\{\cdot\}$ and 3-WL$\{\{\cdot\}\}$, while remaining indistinguishable by 2-WL$\{\cdot\}$ and 2-WL$\{\{\cdot\}\}^*$. Thus, Figure 4.5, in conjunction with Proposition 4.3.12, Lemma 4.3.11 and Proposition 4.3.4 prove that: $\delta$-2-WL$\{\cdot\}$ $\sqsubset$ 2-WL$\{\cdot\}$, 3-WL$\{\{\cdot\}\}^*$ $\sqsubset$ 2-WL$\{\cdot\}$, and 3-WL$\{\{\cdot\}\}^*$ $\sqsubset$ 2-WL$\{\{\cdot\}\}^*$.

Figure 4.6 presents another pair of graphs that can be distinguished by $\delta$-3-LWL$\{\{\cdot\}\}^*$ but not by $\delta$-2-LWL$\{\{\cdot\}\}^*$. This observation, combined with the process followed in the proof of Proposition 4.3.12, establishes the hierarchical relationship: $\delta$-3-LWL$\{\{\cdot\}\}^*$ $\sqsubset$ $\delta$-2-LWL$\{\{\cdot\}\}^*$.

Finally, Figure 4.7 showcases an example of graphs distinguishable by 2-WL but not by 2-WL$\{\{\cdot\}\}^*$, thereby demonstrating that: 2-WL strictly refines 2-WL$\{\{\cdot\}\}^*$, 2-WL$\{\{\cdot\}\}$, and 2-WL$\{\cdot\}$.

These findings provide important insights into the comparative capabilities of these algorithms.

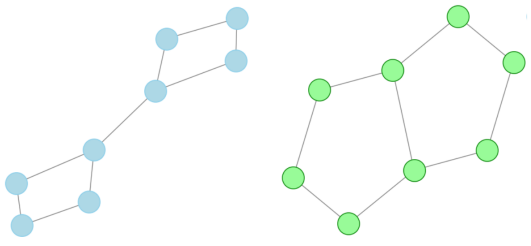


Figure 4.5: A pair of graphs that can be distinghuished by $\delta$-2-LWL$\{\cdot\}$, 3-WL$\{\{\cdot\}\}^*$ and 3-WL$\{\{\cdot\}\}$ and not by 2-WL$\{\cdot\}$, 2-WL$\{\{\cdot\}\}$ and 2-WL$\{\{\cdot\}\}^*$

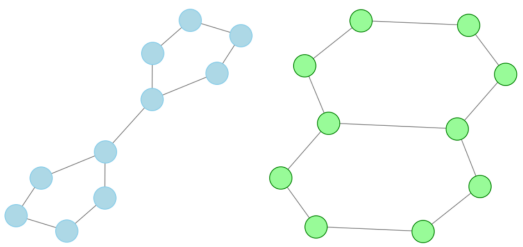


Figure 4.6: A pair of graphs that can be distinghuished by $\delta$-3-LWL$\{\{\cdot\}\}^*$ and not by $\delta$-2-LWL$\{\{\cdot\}\}^*$

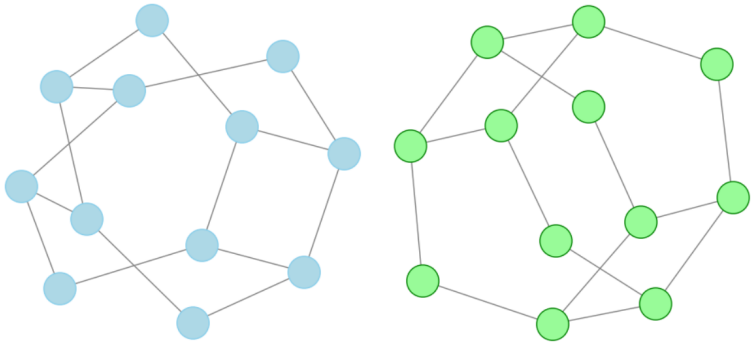


Figure 4.7: A pair of graphs that can be distinguished by 2-WL and not by 2-WL$\{\!\{\cdot\}\!\}^*$, 2-WL$\{\!\{\cdot\}\!\}$ and 2-WL$\{\cdot\}$

## 4.5. Neural architectures based on multi-set variants

By following the process outlined in Section 3.4.3 for developing neural architectures equivalent to various $k$-WL variants, we introduce neural architectures based on the (multi)set-based $k$-WL algorithms discussed previously. While these architectures are considerably less computationally expensive, as Propositions 4.3.2 and 4.3.4 indicate, they correspond to less expressive GNN models.

### 4.5.1. Proposed architectures

For the $k$-WL$\{\!\{\cdot\}\!\}$-GNN induced by the $k$-WL$\{\!\{\cdot\}\!\}$ algorithm, we use Equation 4.11 to update the vector representation of a node (mutliset) $S$ based on its current vector representation as well as on its neighbors vector representations.

$$f_{\text{merge}}^{W_1}\left(f^{(t-1)}(S), f_{\text{aggr}}^{W_2}\left(\{\!\{f^{(t-1)}(T) \mid T \sim S\}\!\}\right)\right). \tag{4.11}$$

Equation 4.12 provides the update rule for the $\delta$-$k$-LWL$\{\!\{\cdot\}\!\}$-GNN based on $\delta$-$k$-LWL$\{\!\{\cdot\}\!\}$ algorithm by considering only information from local neighbors.

$$f_{\text{merge}}^{W_1}\left(f^{(t-1)}(S), f_{\text{aggr}}^{W_2}\left(\{\!\{f^{(t-1)}(T) \mid T \overset{L}{\sim} S\}\!\}\right)\right). \tag{4.12}$$

Finall, Equation 4.12 describes the $\delta$-$k$-WL$\{\!\{\cdot\}\!\}$-GNN models, which incorporate a discrimination in the aggregation function based on whether the information originates from a local or global neighbor.

$$f_{\text{merge}}^{W_1}\left(f^{(t-1)}(S), f_{\text{aggr}}^{W_2}\left(\{\!\{f^{(t-1)}(T) \mid T \overset{G}{\sim} S\}\!\}, \{\!\{f^{(t-1)}(T) \mid T \overset{L}{\sim} S\}\!\}\right)\right). \tag{4.13}$$

In Figure 4.8, we present a comprehensive illustration of the proposed models. The architecture closely resembles Figure 3.12, with the addition of a transformation phase. This inclusion facilitates the creation of more complex graphs and enables the propagation of higher-order information.

### 4.5.2. Equivalence with refinement algorithms

In this section, we establish the equivalence between neural architectures based on (multi)set variants of the $k$-WL algorithm and their corresponding algorithms. Using Theorem 3.4.1 and Theorem 3.4.2, we demonstrate that these architectures possess the same expressive power. We focus on proving the equivalence between the $k$-WL$\{\!\{\cdot\}\!\}$-GNN and the $k$-WL$\{\!\{\cdot\}\!\}$ algorithm, while noting that similar results hold for the $\delta$-$k$-WL$\{\!\{\cdot\}\!\}$, $\delta$-$k$-LWL$\{\!\{\cdot\}\!\}$, $\delta$-$k$-WL$\{\cdot\}$, $\delta$-$k$-LWL$\{\cdot\}$ variants. This equivalence connects graph isomorphism algorithms with their neural network counterparts, providing a theoretical foundation for

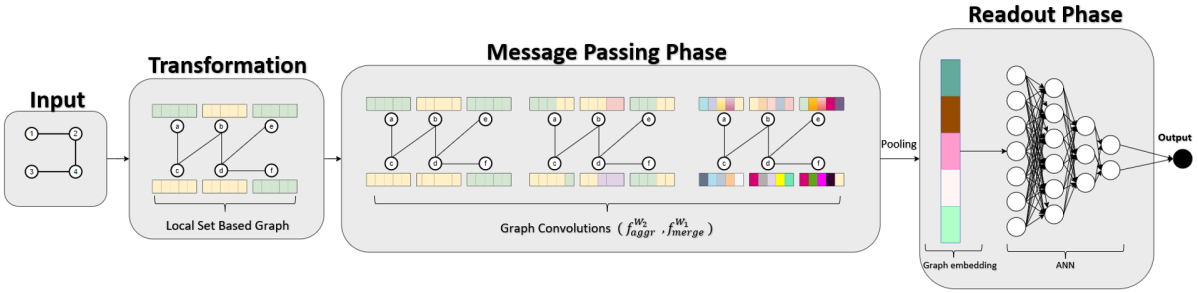



Figure 4.8: Illustration of the $\delta$-2-LWL$\{\cdot\}$-GNN structure highlighting the transformation, message passing, pooling and the ANN phases. Nodes a,b,c,d,e,f corresponds to the sets $\{1,2\},\{1,3\},\{1,4\},\{2,3\},\{2,4\}$ and $\{3,4\}$ respectively. The above architecture consist of 2 layers.

designing and analyzing novel neural architectures.

**Theorem 4.5.1.** *Let $(G,l)$ be a labeled graph and let $k \geq 2$. If $C^m_{k,t}$ is the coloring function induced by the $k$-WL$\{\{\cdot\}\}$ algorithm in iteration $t$, then for all architectures $f_{k,t}$ given by Equation 4 and for all choices of initial colorings $f_{k,0}$ consistent with the isomorphism types of multisets (i.e. if two multisets have the same isomorphism type they also have the same initial coloring through $f_{k,0}$ we have that*

$$C^m_{k,t} \sqsubseteq f_{k,t}$$

*Proof.* We prove our theorem by induction on the iteration $t$. For $t = 0$, the statement holds because the initial coloring $f_{k,0}$ is chosen to be consistent with the isomorphism of the multisets. For the inductive step, we assume that the statement holds until iteration $t$. Consider two multisets $S$ and $T$ with $C^m_{k,t+1}(S) = C^m_{k,t+1}(T)$. Therefore, by definition of the $k$-WL$\{\{\cdot\}\}$ we have:

$$C^m_{k,t}(S) = C^m_{k,t}(T)$$

and

$$\{\{C^m_{k,t}(A) \mid A \sim S\}\} = \{\{C^m_{k,t}(B) \mid B \sim T\}\}.$$

Therefore, by the inductive hypothesis we get

$$f_{k,t}(S) = f_{k,t}(T)$$

and

$$\{\{f_{k,t}(A) \mid A \sim S\}\} = \{\{f_{k,t}(B) \mid B \sim T\}\}.$$

Hence, since the $f_{k,t+1}$ is given by Equation 4, we have that $f_{k,t+1}(S) = f_{k,t+1}(T)$ for every choice of the functions $f^{W_1}_{\text{merge}}$ and $f^{W_2}_{\text{aggr}}$. ■

**Theorem 4.5.2.** *Let $(G,l)$ be a labeled graph and let $k \geq 2$. If $C^m_{k,t}$ is the coloring function induced by the $k$-WL$\{\{\cdot\}\}$ algorithm in iteration $t$. Then for all $t \geq 0$, there exists a sequence of $f^{W_1}_{\text{merge}}$, $f^{W_2}_{\text{aggr}}$ functions from Equation 4.11 such that:*

$$C^m_{k,t} \equiv f_{k,t}$$

*Proof.* From Theorem 3.4.2 we know that for the architecture described by Equation 3.10 there is a sequence of weights $\mathbf{W}^{(t)}$ such that the corresponding GNN architecture is equivalent with the 1-WL algorithm in terms distinguishing between non-isomorphic graphs. It therefore, suffices to simulate the $k$-WL$\{\{\cdot\}\}$ algorithm on the labeled graph $(G,l)$ via a 1-WL on a modified graph $(H,l_H)$. We define the

set of vertices of $H$ as $V(H) = [[V(G)]]^k$ and the edge set of $H$ is defined as follows: two multisets $S,T \in V(H)$ are connected in $H$ if $S \sim T$ according to the definition given in Section 4.1. The labeling $l_H$ of the graph H is determined as follows: For every $S \in V(H)$ the label of $S$ is its ismorphism type, i.e, $l_H(S) = \tau_S$. From the above construction, it immediately follows that performing the 1-WL on the graph $(H, l_H)$ yields the same coloring, as the one obtained by performing the $k$-WL$\{\{\cdot\}\}$ for the graph $(G, l)$. Hence, the sequence of weights $\mathbf{W}^{(t)}$ induced by Theorem 3.4.2, can be directly used in the $k$-WL$\{\{\cdot\}\}$-GNN to simulated the $k$-WL$\{\{\cdot\}\}$. ■

# 5

# Experimental evaluation

Our primary focus is to empirically investigate the performance of the novel (multi)set based neural architectures that were developed in the previous sections. These architectures aim to effectively capture higher order patterns existing in graphs while ensuring computational efficiency. We compare the above architectures with the local tuple based neural architectures described in Morris et al. [2020b] as well as with the 1-GNNs architectures described in Kipf and Welling [2016] and Xu et al. [2019a]. Concretely, we aim to answer the following questions.

**Q1** Do the neural architectures based on (multi)set-based algorithms and their variants, lead to improved classification scores on real world, graph-level benchmark datasets compared to 1-WL based architectures?

**Q2** Do we have significant improvement by considering multi-set based variants instead of set-based as Morris et al. [2019] proposed?

**Q3** To what extent do our proposed architectures reduce computational and memory requirements compared with architectures induced by the $k$-WL?

**Q4** Is there an alignment between the expressive power of a model and its generalization capabilities? Specifically, do models with mathematically proven higher expressive power exhibit improved generalization properties?

The source code of all methods and evaluation procedures is available at `https://github.com/antvas98/set_based_GNNs`.

**Datasets** To evaluate neural architectures, we use the following well known, datasets: MUTAG, PTC FM, PTC MR, IMDB-BINARY (Morris et al. [2020a], Helma et al. [2001], Yanardag and Vishwanathan [2015]). A detailed overview of the above mentioned dataset is illustrated in Table 5.1.

**Neural Architectures** We employed the Graph Isomorphism Network (GIN) architecture (Xu et al.

[2019a]) and the Graph Convolutional Network (GCN) (Kipf and Welling [2016]) as neural baselines to represent the functions $f_{\text{merge}}^{W_1}$ and $f_{\text{agg}}^{W_2}$ discussed in Section 4.5.1. For both architectures, we disregarded edge features. The implementation of the models was carried out using PyTorch Geometric (Fey and Lenssen [2019]). In the subsequent section, we provide a comprehensive description of the evaluation protocols and procedures for selecting hyperparameters.

Table 5.1: Dataset statistics and properties for graph-level prediction tasks

| Dataset | Number of Graphs | Number of classes | Number of nodes | Number of edges | Node labels |
|---|---|---|---|---|---|
| MUTAG | 188 | 2 | 17.9 | 19.8 | ✓ |
| PTC MR | 344 | 2 | 14.3 | 14.7 | ✓ |
| PTC FM | 349 | 2 | 14.1 | 14.5 | ✓ |
| IMDB-BINARY | 1000 | 2 | 19.8 | 96.5 | ✗ |

## 5.1. Experimental protocol and model configuration

We conducted two separate experiments, one for the GIN architecture [Xu et al., 2019a] and another for the GCN architecture [Kipf and Welling, 2016]. According to Garg et al. [2020] both architectures are unable to capture important graph properties such as longest or shortest cycle, diameter, or clique information and therefore, higher order variants are necessary. For both experiments we followed a common setup. We present the general procedure:

**Dataset Split**: We divided the dataset into 10 folds using stratified sampling, ensuring a balanced distribution of classes across the folds.

**Training and Evaluation**: For each experiment, our training process involved using 9 out of the 10 available folds. Within this training phase, we further divided the 9 folds into a $90\%$ training set and a $10\%$ validation set. This split allowed us to perform hyperparameter tuning and select the optimal configuration. Our focus was on tuning the number of hidden units for the models, specifically exploring models with 8, 16, or 32 hidden units. During the training process, we evaluated each configuration on the validation set of $10\%$ and selected the one that exhibited the best performance. Once the hyperparameters were selected, we proceeded to evaluate the trained model on the remaining fold, which served as the test set. We repeated this evaluation process for all 10 folds, resulting in 10 different test sets. For each test set, we measured the accuracy of the model, representing the proportion of correctly classified instances. The average accuracy across the 10 test sets, along with the corresponding standard deviation, are reported in Tables 5.2 and 5.3 for the GCN and GIN architectures, respectively.

It is important to note that due to the nature of the experimental setup, different hyperparameter configurations may have been selected for each test set. Therefore, the described process is suitable for evaluating the overall architecture's performance but not for determining the optimal hyperparameter value.

**Common Model Configurations**: In both the GIN and GCN architectures, we employed two layers for the message passing phase. For the aggregation phase, we utilized a global add pooling method to combine the node vectors and derive the final graph representation. The Adam optimizer with a learning rate of 0.001 and a weight decay of 0.00007 was utilized for both architectures.

**GCN Architecture**: For the GCN layers, we updated the node vectors using the following equation:

$$\mathbf{x}_S^{(t+1)} = \Theta^\intercal \sum_{T \sim S} \frac{e_{j,i}}{\sqrt{\hat{d}_j \hat{d}_i}} \mathbf{x}_T^{(t)}, \tag{5.1}$$

where $t = 0, 1$. Here, $\Theta^\intercal$ denotes the transpose of the parameter matrix $\Theta$.

**GIN Architecture**: For the GIN layers, we updated the node vectors using the following equation:

$$\mathbf{x}_S^{(t+1)} = \text{MLP}^{(t+1)} \left( \left(1 + \epsilon^{(t+1)}\right) \mathbf{x}_S^{(t)} + \sum_{T \sim S} \mathbf{x}_T^{(t)} \right), \tag{5.2}$$

where $t = 0, 1, 2$. This equation represents the transformation of node vectors through a multilayer perceptron (MLP). We can make $\epsilon$ a learnable parameter or a fixed scalar. In the first iteration, we do not need MLPs before summation if input features are one-hot encodings as their summation alone is injective.

For the architectures described in Equation 4.13, we employed distinct layers for the local and global neighbors. The vector representations from these layers were then merged using a Multi-Layer Perceptron (MLP) with a Rectified Linear Unit (ReLU) activation function. This merging process combines the local and global information to derive the final graph representation. For a visual illustration of this architecture, please refer to Figure 5.1.

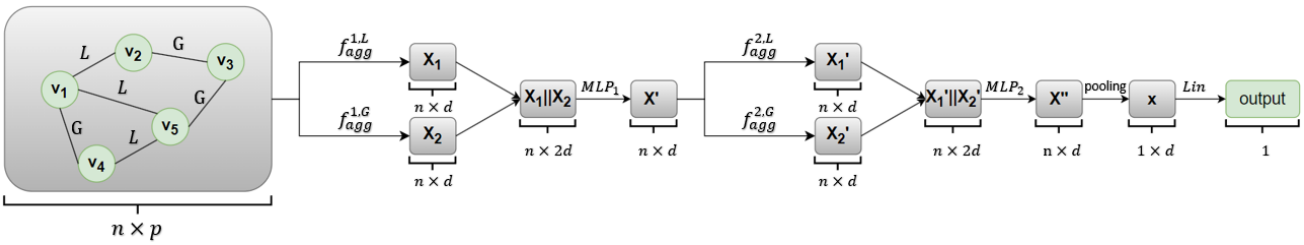



Figure 5.1: The neural architecture diagram illustrating the aggregation process between local and global neighbors. The aggregation between local neighbors is denoted as $f_{agg}^{L}$, while the layers responsible for aggregation between global neighbors are denoted as $f_{agg}^{G}$. Each layer as well as the MLPs transformations have $d$ hidden units, and the initial features of the transformed graph have a dimension of $p$. The transformed graph consists of $n$ nodes.

## 5.2. Results and discussion

Table 5.2 and Table 5.3 display the accuracy results of the GCN convolution architectures and GIN convolution architectures, respectively, for each dataset. The tables also include the corresponding standard deviations.

In the following, we answer questions **Q1** to **Q4**.

**A1** It is clear that certain neural architectures based on (multi)set-based algorithms consistently outperform the performance of 1-GNN models in terms of classification scores on real-world, graph-level benchmark datasets. Specifically, for GCN architectures (Table 5.2) where the 1-WL algorithm exhibits relatively poor performance, almost all proposed models demonstrate significantly better results. Conversely, for GIN architectures (Table 5.3), which are known for their enhanced expressive power

Table 5.2: Performance comparison of the different models on MUTAG, PTC MR, and PTC FM datasets using GCN architecture. ”Out of time” (OOT) indicates that computation did not finish within one day, and ”Out of memory” (OOM) indicates that our system was unable to allocate the required memory for this model. The horizontal lines separate the tuple-based models, the set-based models, and the multiset-based models.

| model (GCN) | MUTAG | PTC MR | PTC FM | IMDB-BINARY |
|---|---|---|---|---|
| 1-GNN | 0.7815 (±0.077) | 0.6557 (±0.054) | 0.6465 (±0.040) | 0.5720 (±0.035) |
| $\delta$-2-LWL-GNN | 0.8725 (±0.074) | 0.6916 (±0.040) | 0.6505 (±0.037) | 0.6479 (±0.063) |
| $\delta$-3-LWL-GNN | 0.8620 (±0.044) | OOT | OOT | OOT |
| 2-WL{·}-GNN | 0.8564 (±0.062) | 0.7109 (±0.073) | 0.6664 (±0.038) | 0.5380 (±0.026) |
| 2-WL{·}-GNN* | 0.8728 (±0.078) | 0.7112 (±0.057) | 0.6637 (±0.36) | 0.5530 (±0.013) |
| 3-WL{·}-GNN | 0.8567(±0.061) | 0.6794(±0.044) | 0.6794(±0.051) | OOM |
| 3-WL{·}-GNN* | 0.8780 (±0.070) | 0.6676 (±0.037) | 0.6815 (±0.047) | OOM |
| $\delta$-2-LWL{·}-GNN | 0.8669 (±0.075) | 0.6966 (±0.055) | 0.6750(±0.036) | 0.6599 (±0.024) |
| $\delta$-2-LWL{·}-GNN* | 0.8725 (±0.078) | 0.6791 (±0.047) | 0.6552(±0.024) | 0.6530 (±0.026) |
| $\delta$-3-LWL{·}-GNN | 0.8666(±0.072) | 0.6705(±0.048) | 0.6735(±0.024) | OOM |
| $\delta$-3-LWL{·}-GNN* | 0.8611 (±0.074) | 0.6588 (±0.032) | 0.6667(±0.052) | OOM |
| $\delta$-2-WL{·}-GNN | 0.8669(±0.075) | 0.7139(±0.073) | 0.6693(±0.031) | 0.6559 (±0.023) |
| $\delta$-3-WL{·}-GNN | 0.8672 (±0.071) | 0.6823(±0.048) | **0.7029**(±0.038) | OOM |
| 2-WL{{·}}-GNN | 0.8669 (±0.071) | 0.6879 (±0.029) | 0.6808(±0.040) | 0.5790 (±0.034) |
| 2-WL{{·}}-GNN* | 0.8780 (±0.052) | 0.6792 (±0.047) | 0.6521(±0.038) | 0.6050 (±0.027) |
| 3-WL{{·}}-GNN | 0.8777 (±0.062) | 0.6995(±0.070) | 0.6664(±0.046) | OOM |
| 3-WL{{·}}-GNN* | 0.8728 (±0.066) | 0.7054(±0.073) | 0.6838(±0.040) | OOM |
| $\delta$-2-LWL{{·}}-GNN | 0.8669 (±0.075) | 0.6732 (±0.045) | 0.6780(±0.048) | 0.6320 (±0.035) |
| $\delta$-2-LWL{{·}}-GNN* | 0.8775 (±0.071) | 0.6687 (±0.040) | 0.6614(±0.045) | 0.6190 (±0.033) |
| $\delta$-3-LWL{{·}}-GNN | 0.8672 (±0.071) | **0.7197**(±0.061) | 0.6981 (±0.041) | OOM |
| $\delta$-3-LWL{{·}}-GNN* | 0.8728 (±0.070) | 0.6831 (±0.053) | 0.6790 (±0.069) | OOM |
| $\delta$-2-WL{{·}}-GNN | **0.8830** (±0.065) | 0.6822(±0.048) | 0.6750 (±0.054) | **0.6639** (±0.041) |
| $\delta$-3-WL{{·}}-GNN | 0.8514 (±0.065) | 0.7052(±0.062) | 0.6865 (±0.050) | OOM |

Table 5.3: Performance comparison of the different models on MUTAG, PTC MR, and PTC FM datasets using GIN architecture. ”Out of time” (OOT) indicates that computation did not finish within one day, and ”Out of memory” (OOM) indicates that our system was unable to allocate the required memory for this model. The horizontal lines separate the tuple-based models, the set-based models, and the multiset-based models.

| model (GIN) | MUTAG | PTC MR | PTC FM | IMDB-BINARY |
|---|---|---|---|---|
| 1-GNN | 0.9142 (±0.049) | 0.6736 (±0.055) | 0.7009 (±0.044) | 0.7510 (±0.038) |
| $\delta$-2-LWL-GNN | 0.9251 (±0.049) | 0.7239 (±0.040) | 0.7078 (±0.057) | 0.7450 (±0.035) |
| $\delta$-3-LWL-GNN | 0.9036 (±0.021) | OOT | OOT | OOT |
| 2-WL{·}-GNN | 0.8991 (±0.049) | 0.6935 (±0.069) | 0.6549 (±0.035) | 0.7446 (±0.042) |
| 2-WL{·}-GNN* | 0.9099 (±0.046) | 0.6994 (±0.046) | 0.6520 (±0.043) | 0.7459 (±0.037) |
| 3-WL{·}-GNN | 0.8830 (±0.043) | OOM | OOM | OOM |
| 3-WL{·}-GNN* | 0.9046 (±0.060) | OOM | OOM | OOM |
| $\delta$-2-LWL{·}-GNN | 0.9464 (±0.033) | 0.7230 (±0.048) | 0.6952 (±0.051) | 0.7480 (±0.038) |
| $\delta$-2-LWL{·}-GNN* | 0.9201 (±0.042) | 0.7402 (±0.045) | 0.7010 (±0.052) | **0.7529** (±0.040) |
| $\delta$-3-LWL{·}-GNN | 0.9149 (±0.058) | 0.7058 (±0.052) | 0.7058 (±0.047) | 0.6950 (±0.038) |
| $\delta$-3-LWL{·}-GNN* | 0.9254 (±0.035) | 0.7000 (±0.050) | 0.6698 (±0.041) | 0.6990 (±0.038) |
| $\delta$-2-WL{·}-GNN | 0.9415(±0.043) | **0.7431**(±0.040) | 0.7066(±0.053) | 0.7520 (±0.042) |
| $\delta$-3-WL{·}-GNN | 0.9359(±0.032) | OOM | OOM | OOM |
| 2-WL{{·}}-GNN | 0.8994 (±0.063) | 0.7026 (±0.055) | 0.6492(±0.043) | 0.7480 (±0.036) |
| 2-WL{{·}}-GNN* | 0.9201 (±0.048) | 0.7171 (±0.057) | 0.6779(±0.032) | 0.7470 (±0.041) |
| 3-WL{{·}}-GNN | 0.9043 (±0.061) | OOM | OOM | OOM |
| 3-WL{{·}}-GNN* | 0.8885 (±0.063) | OOM | OOM | OOM |
| $\delta$-2-LWL{{·}}-GNN | **0.9467**(±0.041) | 0.7111 (±0.065) | 0.6896(±0.052) | 0.7460 (±0.039) |
| $\delta$-2-LWL{{·}}-GNN* | 0.9359 (±0.046) | 0.7091 (±0.045) | 0.6990 (±0.043) | 0.7520 (±0.032) |
| $\delta$-3-LWL{{·}}-GNN | 0.8889 (±0.049) | 0.7025 (±0.081) | 0.7042 (±0.047) | 0.6799 (±0.040) |
| $\delta$-3-LWL{{·}}-GNN* | 0.9204 (±0.048) | 0.6772 (±0.049) | **0.7194** (±0.050) | 0.6920 (±0.030) |
| $\delta$-2-WL{{·}}-GNN | 0.9309 (±0.033) | 0.7257 (±0.051) | 0.7184 (±0.050) | 0.7470 (±0.036) |
| $\delta$-3-WL{{·}}-GNN | 0.9310 (±0.023) | OOM | OOM | OOM |

according to Xu et al. [2019a], the 1-GNNs shows excellent performance, surpassing other models. However, some local or delta variants of the (multi)set-based algorithms achieve even better perfor-

mance, underscoring the importance of considering sparsity for improved generalization as described in Garg et al. [2020].

**A2** Based on Tables 5.2, 5.3, it is evident that the multiset-based variants, do not consistently demonstrate significant improvements compared to the set-based models. While the multiset-based models are theoretically more expressive, the experimental results reveal that the performance of multiset-based and set-based models is remarkably close, with negligible differences. In some cases, the set-based models even exhibit slightly better performance than their corresponding multiset counterparts. However, it is important to note that these outcomes might be influenced by the specific configurations of the models and the chosen hyperparameters. It is possible that alternative model configurations and hyperparameter tuning could yield better results for the multiset-based models.

**A3** Our proposed architectures offer significant reductions in computational and memory requirements compared to architectures induced by the $k$-WL. When considering the $3$-tuple transformations of the graphs, the computational task was infeasible (did not run within a day) for all datasets except for MUTAG. On the other hand, the set and multiset-based transformations for $k = 3$ were calculated relatively quickly. This showcases the efficiency of our (multi)set-based algorithms in capturing higher-order patterns in graph structures while demanding significantly less computational resources and memory. Consequently, our proposed architectures demonstrate improved efficiency without compromising their ability to effectively capture complex patterns in the data.

**A4** There is not a clear alignment between the expressive power of a model, as described in Section 4, and its generalization capabilities. The performance of the models, as shown in Tables 5.2 and 5.3, does not consistently correlate with their mathematically proven expressive power. For instance, in the case of the PTC MR dataset (Table 5.2), the $2$-WL$\{\cdot\}$-GNN outperforms the $2$-WL$\{\!\{\cdot\}\!\}$-GNN, despite the latter being theoretically more expressive according to Proposition 4.3.4. Similarly, for the PTC FM dataset (Table 5.2), the $\delta$-$2$-WL$\{\!\{\cdot\}\!\}$-GNN and $2$-WL$\{\!\{\cdot\}\!\}$-GNN do not align with the strict expressiveness hierarchy suggested by Proposition 4.3.10. However, it is important to note that this observation does not contradict our theoretical results. While there is a mathematical proof that certain models possess higher expressive power, there is no direct guarantee that this translates to improved prediction performance on unseen data. Additionally, Theorem 4.5.2 establishes the existence of parameters that can achieve the desired expressive power, but it is not clear whether the parameter choices and architecture used in our experiments have indeed achieved this ideal selection indicated by the theorem. Therefore, the relationship between expressive power and generalization capabilities is influenced by various factors beyond the theoretical framework and further research is necessary in order to understand it deeper.

## 5.3. Future work

In order to gain a deeper understanding of Graph Neural Network (GNN) models and how well they can adapt to new, unseen data, there are a few important areas to explore. One key aspect is to investigate how GNN models generalize to real-world applications, where unseen data plays a critical role. Previous research, like the work done by Morris et al. [2023] or Scarselli et al. [2018], can serve as a starting point for further exploration in this direction.

Another important area of study is to delve deeper into the algorithms proposed for GNN models. Although Theorem 4.5.2 suggests that there are ideal parameters, it remains a challenge to practically determine what those parameters are. This calls for a more thorough investigation and the development of new optimization techniques or algorithmic approaches that can effectively find and identify these optimal parameter settings.

Additionally, it would be valuable to propose methods that allow us to assess the practical significance of using computationally expensive GNN models. For example, while 3-multiset based models may be theoretically superior to 3-set based models, it's important to determine if this theoretical difference translates into tangible improvements on specific datasets. By doing so, we can make informed decisions about the trade-offs between model complexity and performance.

By focusing on these research areas, we can gain deeper insights into GNN models, enhance their ability to generalize, and provide practical guidance for choosing the right models and tuning their parameters in real-world applications.

# 6

# Conclusion

In conclusion, this study represents a comprehensive investigation into the expressive power and generalization capabilities of graph neural network (GNN) architectures. We conducted an extensive analysis and comparison of various existing Weisfeiler-Leman (WL) variants, with a particular focus on the $\delta$-$k$-LWL algorithms. Moreover, we proposed and evaluated a range of innovative multiset and set-based GNN architectures. By conducting thorough experiments on diverse benchmark datasets for classification tasks, we consistently observed superior performance of our proposed models compared to classical $1$-GNN approaches, along with enhanced scalability in comparison to the traditional $k$-WL based methods. We firmly believe that this principled approach opens up the way for the design of permutation-equivariant architectures, addressing the limitations of current graph neural networks. By leveraging the power of multiset- and set-based algorithms, we can capture higher-order patterns and exploit the structural information encoded in graphs more effectively. This framework not only improves the expressiveness and scalability of GNNs but also provides a foundation for developing novel architectures that overcome the current limitations.

In summary, this study provides valuable insights into the expressive power, generalization capabilities, and scalability of graph neural network architectures. The proposed multiset and set-based approaches showcase their potential to push the boundaries of GNN design and enable further advancements. There are several opportunities for future research to explore the full potential of these principled approaches and develop GNN architectures with even more generalization capabilities.

# Bibliography


Ralph Abboud, İsmail İlkan Ceylan, Martin Grohe, and Thomas Lukasiewicz. The surprising power of graph neural networks with random node initialization. In Zhi-Hua Zhou, editor, *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021*, pages 2112–2118. ijcai.org, 2021.

Vikraman Arvind, Johannes Köbler, Gaurav Rattan, and Oleg Verbitsky. On the power of color refinement. In Adrian Kosowski and Igor Walukiewicz, editors, *Fundamentals of Computation Theory - 20th International Symposium, FCT 2015, Gdańsk, Poland, August 17-19, 2015, Proceedings*, volume 9210 of *Lecture Notes in Computer Science*, pages 339–350. Springer, 2015.

Waïss Azizian and Marc Lelarge. Characterizing the expressive power of invariant and equivariant graph neural networks. *CoRR*, 2020.

László Babai. Graph isomorphism in quasipolynomial time [extended abstract]. In Daniel Wichs and Yishay Mansour, editors, *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 684–697. ACM, 2016.

Laszlo Babai and Ludik Kucera. Canonical labelling of graphs in linear average time. 1979.

H. S. Baird and Y. E. Cho. Artwork design verification system. 1975.

Muhammet Balcilar, Pierre Héroux, Benoit Gaüzère, Pascal Vasseur, Sébastien Adam, and Paul Honeine. Breaking the limits of message passing graph neural networks. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 599–608. PMLR, 2021.

Albert-Laszlo Barabasi and Zoltan Oltvai. Network biology: Understanding the cell's functional organization. *Nature reviews. Genetics*, pages 101–13, 03 2004.

Pablo Barceló, Egor V. Kostylev, Mikaël Monet, Jorge Pérez, Juan L. Reutter, and Juan Pablo Silva. The logical expressiveness of graph neural networks. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

Pablo Barceló, Floris Geerts, Juan L. Reutter, and Maksimilian Ryschkov. Graph neural networks with local graph parameters. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 25280–25293, 2021.

Dominique Beaini, Saro Passaro, Vincent Létourneau, William L. Hamilton, Gabriele Corso, and Pietro Lió. Directional graph networks. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 748–758. PMLR, 2021.

Yoshua Bengio, Andrea Lodi, and Antoine Prouvost. Machine learning for combinatorial optimization: A methodological tour d'horizon. *Eur. J. Oper. Res.*, 290(2):405–421, 2021.

Christian Bick, Elizabeth Gross, Heather A. Harrington, and Michael T. Schaub. What are higher-order networks? *CoRR*, abs/2104.11329, 2021.

Cristian Bodnar, Fabrizio Frasca, Nina Otter, Yuguang Wang, Pietro Liò, Guido F. Montúfar, and Michael M. Bronstein. Weisfeiler and lehman go cellular: CW networks. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 2625–2640, 2021.

Béla Bollobás. Distinguishing vertices of random graphs. In Béla Bollobás, editor, *Graph Theory*, volume 62 of *North-Holland Mathematics Studies*, pages 33–49. North-Holland, 1982.

Ronald V. Book. Richard m. karp. reducibility among combinatorial problems. complexity of computer computations, proceedings of a symposium on the complexity of computer computations, held march 20-22, 1972, at the ibm thomas j. watson center, yorktown heights, new york, edited by raymond e. miller and james w. thatcher, plenum press, new york and london 1972, pp. 85–103. *Journal of Symbolic Logic*, 40, 1975. ISSN 0022-4812. doi: 10.2307/2271828.

Karsten M. Borgwardt, M. Elisabetta Ghisu, Felipe Llinares-López, Leslie O'Bray, and Bastian Rieck. Graph kernels: State-of-the-art and future challenges. *Found. Trends Mach. Learn.*, 13(5-6), 2020.

Giorgos Bouritsas, Fabrizio Frasca, Stefanos Zafeiriou, and Michael M. Bronstein. Improving graph neural network expressivity via subgraph isomorphism counting. *IEEE Trans. Pattern Anal. Mach. Intell.*, 45(1):657–668, 2023.

Jin Yi Cai, Martin Fürer, and Neil Immerman. An optimal lower bound on the number of variables for graph identification. *Combinatorica*, 1992.

Quentin Cappart, Didier Chételat, Elias B. Khalil, Andrea Lodi, Christopher Morris, and Petar Velickovic. Combinatorial optimization and reasoning with graph neural networks. In Zhi-Hua Zhou, editor, *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021*, pages 4348–4355. ijcai.org, 2021.

Ines Chami, Sami Abu-El-Haija, Bryan Perozzi, Christopher Ré, and Kevin Murphy. Machine learning on graphs: A model and comprehensive taxonomy. *CoRR*, abs/2005.03675, 2020.

Zhengdao Chen, Soledad Villar, Lei Chen, and Joan Bruna. On the equivalence between graph isomorphism testing and function approximation with gnns. 2019.

Zhengdao Chen, Lei Chen, Soledad Villar, and Joan Bruna. Can graph neural networks count substructures? In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Aaron Clauset, M. E J Newman, Cristopher Moore, Luciano Da F. Costa, Francisco a. Rodrigues, Gonzalo Travieso, P. R. Villas Boas, Darren P. Croft, Joah R. Madden, Daniel W. Franks, Richard James, Camila I. Donatti, Paulo R. Guimarães, Mauro Galetti, Marco Aurélio Pizo, Flávia M D Marquitti, Rodolfo Dirzo, Linton C Freeman, San Diego, Stuart H. Hurlbert, San Diego, Jens Krause,

Richard James, Darren P. Croft, Alexander D M Wilson, Stefan Krause, Niels J. Dingemanse, Jens Krause, John Creighton Campbell, Author M E J Newman, Source Siam Review, No Jun, M. E J Newman, Stephen R. Proulx, Daniel E L Promislow, and Patrick C. Phillips. The structure and function of complex networks the structure and function of complex networks *. *Trends in Ecology and Evolution*, 2011.

Tomek Czajka and Gopal Pandurangan. Improved random graph isomorphism. *Journal of Discrete Algorithms*, 2008.

George Dasoulas, Ludovic Dos Santos, Kevin Scaman, and Aladin Virmaux. Coloring graph neural networks for node disambiguation. In Christian Bessiere, editor, *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, pages 2126–2132. ijcai.org, 2020.

B. L. Douglas. The Weisfeiler-Lehman method and graph isomorphism testing, 2011.

David A. Easley and Jon M. Kleinberg. *Networks, Crowds, and Markets - Reasoning About a Highly Connected World*. Cambridge University Press, 2010.

Sergei Evdokimov and Ilia Ponomarenko. Isomorphism of coloured graphs with slowly increasing multiplicity of jordan blocks. *Combinatorica*, 1999.

Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with pytorch geometric. *CoRR*, abs/1903.02428, 2019.

Vikas K. Garg, Stefanie Jegelka, and Tommi Jaakkola. Generalization and representational limits of graph neural networks. 2020.

Thomas Gaudelet, Ben Day, Arian R. Jamasb, Jyothish Soman, Cristian Regep, Gertrude Liu, Jeremy B. R. Hayter, Richard Vickers, Charles Roberts, Jian Tang, David Roblin, Tom L. Blundell, Michael M. Bronstein, and Jake P. Taylor-King. Utilizing graph machine learning within drug discovery and development. *Briefings Bioinform.*, 22(6), 2021.

Floris Geerts. The expressive power of kth-order invariant graph networks. *CoRR*, 2020.

Floris Geerts and Juan L. Reutter. Expressiveness and approximation properties of graph neural networks. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. 2017.

Martin Grohe. The logic of graph neural networks. 2021.

Martin Grohe and Daniel Neuen. *Recent advances on the graph isomorphism problem*. 2021.

Martin Grohe and Martin Otto. Pebble games and linear equations. *Journal of Symbolic Logic*, 2015.

Martin Grohe and Pascal Schweitzer. The graph isomorphism problem. *Commun. ACM*, pages 128–134, 2020.

Mustafa Hajij, Ghada Zamzmi, Theodore Papamarkou, Nina Miolane, Aldo Guzmán-Sáenz, Karthikeyan Natesan Ramamurthy, Tolga Birdal, Tamal K. Dey, Soham Mukherjee, Shreyas N. Samaga, Neal Livesay, Robin Walters, Paul Rosen, and Michael T. Schaub. Topological deep learning: Going beyond graph data, 2023.

William Hamilton, Rex Ying, and Jure Leskovec. Representation learning on graphs: Methods and applications. 09 2017.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer, 2nd edition, 2009.

Christoph Helma, Ross D. King, Stefan Kramer, and Ashwin Srinivasan. The predictive toxicology challenge 2000-2001. *Bioinform.*, 17(1):107–108, 2001.

Max Horn, Edward De Brouwer, Michael Moor, Yves Moreau, Bastian Rieck, and Karsten M. Borgwardt. Topological graph neural networks. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

Kurt Hornik, Maxwell B. Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.

Neil Immerman and Eric Lander. *Describing Graphs: A First-Order Approach to Graph Canonization*, pages 59–81. Springer New York, New York, NY, 1990.

John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, Alex Bridgland, Clemens Meyer, Simon A.A. Kohl, Andrew J. Ballard, Andrew Cowie, Bernardino Romera-Paredes, Stanislav Nikolov, Rishub Jain, Jonas Adler, Trevor Back, Stig Petersen, David Reiman, Ellen Clancy, Michal Zielinski, Martin Steinegger, Michalina Pacholska, Tamas Berghammer, Sebastian Bodenstein, David Silver, Oriol Vinyals, Andrew W. Senior, Koray Kavukcuoglu, Pushmeet Kohli, and Demis Hassabis. Highly accurate protein structure prediction with alphafold. *Nature*, 2021.

Richard M. Karp. *Probabilistic analysis of a canonical numbering algorithm for graphs*. 1979.

Sandra Kiefer. The weisfeiler-leman algorithm: an exploration of its power. *ACM SIGLOG News*, 7(3): 5–27, 2020.

Sandra Kiefer and Pascal Schweitzer. Upper bounds on the quantifier depth for graph differentiation in first-order logic. *Logical Methods in Computer Science*, 2019.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *CoRR*, abs/1609.02907, 2016.

Andreas Loukas. What graph neural networks cannot learn: depth vs width. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

Takanori Maehara and Hoang NT. A simple proof of the universality of invariant/equivariant graph neural networks. *CoRR*, abs/1910.03802, 2019.

Peter N. Malkin. Sherali-adams relaxations of graph isomorphism polytopes. *Discrete Optimization*, 2014.

Haggai Maron, Heli Ben-Hamu, Hadar Serviansky, and Yaron Lipman. Provably powerful graph networks. 2019a.

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

Andrius Merkys, Antanas Vaitkus, Algirdas Grybauskas, Aleksandras Konovalovas, Miguel Quirós, and Saulius Gražulis. Graph isomorphism-based algorithm for cross-checking chemical and crystallographic descriptions. *Journal of Cheminformatics*, 2023.

R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon. Network motifs: Simple building blocks of complex networks. *Science*, 2002.

Christopher Morris. *Learning with graphs: kernel and neural approaches*. PhD thesis, Technical University of Dortmund, Germany, 2019.

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. 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. In *ICML 2020 Workshop on Graph Representation Learning and Beyond (GRL+ 2020)*, 2020a.

Christopher Morris, Gaurav Rattan, and Petra Mutzel. Weisfeiler and leman go sparse: Towards scalable higher-order graph embeddings. 2020b.

Christopher Morris, Matthias Fey, and Nils M. Kriege. The power of the weisfeiler-leman algorithm for machine learning with graphs. In Zhi-Hua Zhou, editor, *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021*, pages 4543–4550. ijcai.org, 2021a.

Christopher Morris, Yaron Lipman, Haggai Maron, Bastian Rieck, Nils M. Kriege ands Martin Grohe, Matthias Fey, and Karsten M. Borgwardt. Weisfeiler and leman go machine learning: The story so far. *CoRR*, 2021b.

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

Christopher Morris, Floris Geerts, Jan Tönshoff, and Martin Grohe. Wl meet vc, 2023.

Ryan L. Murphy, Balasubramaniam Srinivasan, Vinayak A. Rao, and Bruno Ribeiro. Relational pooling for graph representations. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 4663–4673. PMLR, 2019.

Hoang NT and Takanori Maehara. Graph homomorphism convolution. *CoRR*, abs/2005.01214, 2020.

Chendi Qian, Gaurav Rattan, Floris Geerts, Christopher Morris, and Mathias Niepert. Ordered subgraph aggregation networks. *CoRR*, abs/2206.11168, 2022.

Patrick Reiser, Marlen Neubert, André Eberhard, Luca Torresi, Chen Zhou, Chen Shao, Houssam Metni, Clint van Hoesel, Henrik Schopmans, Timo Sommer, and Pascal Friederich. Graph neural networks for materials science and chemistry. *CoRR*, abs/2208.09481, 2022.

Jan G. Rittig, Martin Ritzert, Artur M. Schweidtmann, Stefanie Winkler, Jana M. Weber, Philipp Morsch, K. Alexander Heufer, Martin Grohe, Alexander Mitsos, and Manuel Dahmen. Graph machine learning for design of high-octane fuels. *CoRR*, abs/2206.00619, 2022.

Ryoma Sato, Makoto Yamada, and Hisashi Kashima. Random features strengthen graph neural networks. In Carlotta Demeniconi and Ian Davidson, editors, *Proceedings of the 2021 SIAM International Conference on Data Mining, SDM 2021, Virtual Event, April 29 - May 1, 2021*, pages 333–341. SIAM, 2021.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Trans. Neural Networks*, 20(1):61–80, 2009.

Franco Scarselli, Ah Chung Tsoi, and Markus Hagenbuchner. The Vapnik-Chervonenkis dimension of graph and recursive neural networks. *Neural Networks*, 108:248–259, 2018.

Artur M. Schweidtmann, Jan G. Rittig, Jana M. Weber, Martin Grohe, Manuel Dahmen, Kai Leonhard, and Alexander Mitsos. Physical pooling functions in graph neural networks for molecular property prediction. *Comput. Chem. Eng.*, 172:108202, 2023.

Luc Segoufin. M. grohe, descriptive complexity, canonisation, and definable graph structure theory, cambridge university press, cambridge, 2017, x + 544 pp. *The Bulletin of Symbolic Logic*, 2017.

Martin Simonovsky and Nikos Komodakis. Simonovsky, komodakis - 2017 - dynamic edge-conditioned filters in convolutional neural networks on graphs.pdf. *Cvpr*, 2017.

Jonathan M. Stokes, Kevin Yang, Kyle Swanson, Wengong Jin, Andres Cubillos-Ruiz, Nina M. Donghia, Craig R. MacNair, Shawn French, Lindsey A. Carfrae, Zohar Bloom-Ackerman, Victoria M. Tran, Anush Chiappino-Pepe, Ahmed H. Badran, Ian W. Andrews, Emma J. Chory, George M. Church, Eric D. Brown, Tommi S. Jaakkola, Regina Barzilay, and James J. Collins. A deep learning approach to antibiotic discovery. *Cell*, 2020.

Rajat Talak, Siyi Hu, Lisa Peng, and Luca Carlone. Neural trees for learning on graphs. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 26395–26408, 2021.

Jan Tönshoff, Martin Ritzert, Hinrikus Wolf, and Martin Grohe. Graph learning with 1d convolutions on random walks. *CoRR*, abs/2102.08786, 2021.

Clément Vignac, Andreas Loukas, and Pascal Frossard. Building powerful and equivariant graph neural networks with structural message-passing. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Oriol Vinyals, Samy Bengio, and Manjunath Kudlur. Order matters: Sequence to sequence for sets. 2016.

Yu. B. Weisfeiler and A. Leman. The reduction of a graph to canonical form and the algebra which appears therein. *Nauchno-Technicheskaya Informatsia (NTI Series)*, 1968.

Lingfei Wu, Peng Cui, Jian Pei, Liang Zhao, and Le Song. *Graph Neural Networks*, pages 27–37. Springer Nature Singapore, Singapore, 2022.

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

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are gnn. *Int. Conf. on Learning Representations*, 2019b.

Pinar Yanardag and S. V. N. Vishwanathan. Deep graph kernels. In Longbing Cao, Chengqi Zhang, Thorsten Joachims, Geoffrey I. Webb, Dragos D. Margineantu, and Graham Williams, editors, *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, NSW, Australia, August 10-13, 2015*, pages 1365–1374. ACM, 2015.