How Hard Is It for Message-Passing GNNs to Simulate One Step of the Weisfeiler-Lehman Test?

Guanyu Cui
Renmin University of China
cuiguanyu@ruc.edu.cn

Yuhe Guo
Renmin University of China
guoyuhe@ruc.edu.cn

Zhewei Wei*
Renmin University of China
zhewei@ruc.edu.cn

Hsin-Hao Su
Boston College
suhx@bc.edu

Abstract

Message-passing graph neural networks (MPGNNs) are commonly compared with the Weisfeiler-Lehman (WL) test, but this comparison does not quantify the resource parameters a network needs to implement a step of the WL test with bounded-size messages and finite numerical precision. We study the cost of simulating a single step of the WL test on unattributed graphs. We distinguish input-independent, or oblivious, simulation from instance-dependent simulation. In the former, the parameters, or their distributions in randomized models, are fixed before the input instance is known. Our results show that the local update rule of the WL test hides a global relabeling problem. In the oblivious setting, deterministic and zero-error randomized MPGNNs cannot solve this problem in the worst case using only shallow networks with small messages. We complement this lower bound with a nearly matching construction in a stronger rooted, port-aware model. By contrast, when the color set is large, bounded-error randomness can greatly reduce the cost, and a one-layer MPGNN with messages of logarithmic size and a logarithmic number of random bits suffices. We show that this logarithmic number of random bits is essentially necessary for shallow, small-message simulations. When the color set is small, we still obtain a rooted, port-aware simulation, but this construction requires more layers or larger messages. We also prove that this extra cost is partly unavoidable, as small color sets force a nontrivial trade-off between the number of layers and the message size. Finally, instance-dependent simulation can be much shallower, but the required instance-specific parameters are not necessarily easy to find. Together, these results reveal quantitative structure hidden behind the statement that MPGNNs match the WL test.

1 Introduction

Graph neural networks (GNNs) are a central class of models in graph machine learning, with strong empirical performance in applications such as recommendation systems [42, 36, 16, 23, 6, 41], weather forecasting [22], and drug discovery [13, 38, 15, 44]. Alongside these empirical successes, a major theoretical question is to characterize their expressive power. A standard approach is to

*Zhewei Wei is the corresponding author. The work was partially done at Gaoling School of Artificial Intelligence, Beijing Key Laboratory of Research on Large Models and Intelligent Governance, Engineering Research Center of Next-Generation Intelligent Search and Recommendation, MOE, and Pazhou Laboratory (Huangpu), Guangzhou, Guangdong 510555, China.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

compare GNNs with the Weisfeiler-Lehman (WL) test [37][2]. In particular, Xu et al. [39] showed that message-passing GNNs (MPGNNs) are at most as powerful as the WL test and, over countable feature spaces, can attain this upper bound under suitable injectivity assumptions. This perspective has motivated a broad range of more expressive architectures, including higher-order and subgraph-based models [29, 27, 11, 9, 3, 7, 30, 8, 10, 45, 43, 46, 34]. What these results do not quantify, however, is the resource cost of simulating one step of the WL test. In particular, they do not specify the depth, message width, or numerical precision required for an input-independent MPGNN to implement even a single step of the WL test across many inputs.

To study this gap, we focus on a single step of the WL test. This focus is natural because the WL test repeatedly applies the same refinement rule. In one step, each node forms its WL-type, namely its current color together with the multiset of colors of its neighbors. At first glance, simulating this step using MPGNNs appears straightforward, since the relevant information is local and injective multiset encodings are known to exist in the countable-feature setting [39]. The subtle part is the exact, globally consistent relabeling. Nodes with equal WL-types are required to receive the same new color, and distinct WL-types occurring in the current input are required to receive distinct new colors. For a fixed graph and a fixed iteration, this relabeling can be chosen ***after*** observing the WL-types that occur. In contrast, an oblivious MPGNN must use a parameter-selection rule fixed ***before*** the input instance is known. We restrict attention to input colorings that arise along trajectories of the WL test on unattributed graphs, up to relabeling of color names. We therefore ask what resources are required for an MPGNN to simulate[3] one step of the WL test.

To make this question precise, we distinguish two notions of simulation. In *oblivious* or input-independent simulation, all parameter-selection rules are fixed before the input instance is known. Thus, deterministic MPGNNs use a single shared parameter choice, while randomized MPGNNs use an input-independent distribution over parameter choices, with success probability evaluated for each fixed input instance. In *instance-dependent* simulation, by contrast, the parameter choice, or, in the randomized case, the distribution over parameter choices, may depend on the particular input instance. The dependence of parameter selection on the target instance motivates an analogy with inductive and transductive inference in statistical learning theory [35, 33, 28]. Oblivious simulation uses parameters selected independently of the input instance and is analogous to inductive inference, as in GraphSAGE [14]. Instance-dependent representation concerns the existence of suitable parameters for each observed instance and is analogous to transductive inference [19].

All results proved or discussed in this paper are summarized in Table 1. In the oblivious setting, we show that the apparent locality of a single step of the WL test conceals a global relabeling problem. Any deterministic or zero-error randomized oblivious MPGNN must, in the worst case, either use large depth or transmit enough information globally (Theorem 3.1), while a nearly matching upper bound is achievable in the stronger rooted, port-aware model[4] (Theorem 3.2). The bounded-error randomized setting is qualitatively different. When the available color set is sufficiently large, hashing enables a one-layer oblivious MPGNN with logarithmic bandwidth and logarithmic or polylogarithmic randomness to simulate one step of the WL test with high probability (Lemma 3.3 [1], Theorem 3.4). We further show that a logarithmic amount of randomness is essentially necessary for shallow, low-communication simulations (Theorem 3.6). When the color set is only linearly large, however, a nontrivial depth-bandwidth trade-off reappears (Theorem 3.7). Finally, in the instance-dependent setting, we show that shallow representations become possible because the parameters may be chosen after the target instance is known (Theorem 4.1). However, we prove that finding such instance-specific parameters within our decentralized parameter-adaptation framework still requires global communication in the worst case (Theorem 4.2).

2 Preliminaries and Problem Setup

We introduce notation, review MPGNNs and the WL test, and define the simulation problem.

[2]Terminology and initialization conventions vary across the literature. We follow the one-dimensional refinement rule used by Xu et al. [39], which combines a node's current color with the multiset of its neighbors' colors. On unattributed graphs, we start from a constant coloring.

[3]Here and throughout, "simulation" means recovery of the partition after the next step of the WL test, up to a relabeling of color names.

[4]In this model, one node is designated as the root, and each node has locally indexed ports, maintains a separate state for each port, and can send port-specific messages.

Table 1: Summary of results. Here n and m denote the numbers of nodes and edges, respectively; D denotes the diameter; d, w, and p denote the depth, message width, and message precision of an MPGNN, respectively, as defined in Section 2.2; R denotes the number of random bits used; C denotes the size of the color set, as defined in Section 2.4; and p_f denotes the failure probability. The parameter $c_0 \geq 2$ depends on p_f. In our upper-bound constructions, we focus on communication resources (such as d, w and p), and assume sufficiently large local states and sufficiently expressive local functions, including in the rooted, port-aware model.

Setting	Model and Assumptions	Results
Oblivious, Deterministic or Zero-Error Randomized	$\log C = o(m)$	$d = \Omega\left(D + \frac{m}{wp}\right)$ (Thm. 3.1)
	Rooted, Port-Aware MPGNN	$d = O\left(D + \frac{m \log C}{wp}\right)$ (Thm. 3.2)
Oblivious, Bounded-Error Randomized	$p_f = \Omega\left(\frac{1}{\mathrm{poly}(n)}\right)$, $C = \Theta(n^{c_0})$	$d = 1$, $wp = O(\log n)$, $R = O(\log^2 n)$ (Lem. 3.3, [1])
	$C \geq \left\lceil \frac{3n^2}{2p_f} \right\rceil$	$d = 1$, $wp = \Theta(\log C)$, $R = \Theta\left(\log \frac{n \log C}{p_f}\right)$ (Thm. 3.4)
	Rooted, Port-Aware MPGNN	$d = O\left(D + \frac{\log C + n \log(n/p_f)}{wp}\right)$, $R = \Theta\left(\log \frac{n \log C}{p_f}\right)$ (Thm. 3.5)
	$\log C = o(n)$, $dwp = O(\log C)$	$R = \Omega\left(\log \frac{n}{p_f \log C}\right)$ (Thm. 3.6)
	$p_f = \Omega\left(\frac{1}{\mathrm{poly}(n)}\right)$, $C = \Theta(n)$	$d = \Omega\left(D + \min\left\{\log^* n, \frac{n}{wp}\right\}\right)$ (Thm. 3.7)
Instance-Dependent Representation	—	$d = 1$, $wp = \Theta(\log n)$ (Thm. 4.1)
Instance-Dependent Adaptation, Deterministic or Zero-Error Randomized	Graph-Local (Section 4.2) $\log C = o(m)$	$r = \Omega\left(\frac{D}{d} + \frac{m}{dwp}\right)$ (Thm. 4.2)

2.1 Notation

We write $\{\cdot\}$ for sets and $\{\!\{\cdot\}\!\}$ for multisets. The notation $[n]$ stands for $\{1, \ldots, n\}$. Boldface lowercase letters, such as $\boldsymbol{x}$, denote vectors, and boldface uppercase letters, such as $\boldsymbol{X}$, denote matrices. We write $\boldsymbol{x}_i$ for the i-th coordinate of $\boldsymbol{x}$ and $\boldsymbol{X}_{i,:}$ for the i-th row of $\boldsymbol{X}$. We consider simple undirected graphs $G = (V, E)$, where $E \subseteq \{\{u, v\} : u, v \in V, u \neq v\}$. Unless stated otherwise, all graphs in this paper are finite, undirected, and connected. For a node $u \in V$, let $N_G(u) := \{v \in V : \{u, v\} \in E\}$ denote its neighborhood, and write $N(u)$ when G is clear from context. We define $\deg(u) := |N(u)|$, $n := |V|$, and $m := |E|$. The diameter of G is $D := \max_{u,v \in V} \mathrm{dist}_G(u, v)$, where $\mathrm{dist}_G(u, v)$ is the graph distance between u and v.

2.2 Message-Passing Graph Neural Networks

Among the many formulations of GNNs, we use the message-passing GNN (MPGNN) framework from [13]. For the model class studied here, each layer has the form

$$\boldsymbol{X}_{u,:}^{(\ell+1)} = \mathrm{UPD}^{(\ell)}\left(\boldsymbol{X}_{u,:}^{(\ell)}, \left\{\!\!\left\{\mathrm{MSG}^{(\ell)}\left(\boldsymbol{X}_{v,:}^{(\ell)}\right) : v \in N(u)\right\}\!\!\right\}\right), \tag{1}$$

where $\mathrm{MSG}^{(\ell)}(\cdot)$ maps a sender state to a message, and $\mathrm{UPD}^{(\ell)}(\cdot, \cdot)$ combines the current node state with the multiset of incoming messages. The update function is required to be permutation-invariant in its second argument. This formulation makes the communication structure explicit and corresponds to the aggregate-combine viewpoint of Xu et al. [39]. Unless explicitly stated otherwise, this is the anonymous unordered-neighborhood model used throughout the lower-bound statements: nodes have no unique identifiers, no port numbers, and no persistent per-edge states.

In our analysis, the relevant resource measures are communication resources. The **depth** d is the number of message-passing layers. The **width** w is the maximum dimension of a message vector

transmitted along an edge[5], and the **precision** p is the number of bits used to represent each coordinate of a transmitted message. Thus, each edge can transmit at most wp bits per layer, which we call the bandwidth, and at most dwp bits over d layers in one direction. Local memory and computation are unrestricted in our models. For our upper-bound constructions, the local functions are assumed to be sufficiently expressive to perform canonical encoding, hash evaluation, sorting, and ranking exactly.

2.3 Weisfeiler-Lehman Test

The WL test iteratively refines node colors by injectively encoding each node's current color together with the multiset of its neighbors' colors [37, 39]. Let $G = (V, E)$ be a graph equipped with an initial coloring $\chi_G^{(0)} : V \to [C]$. The refinement rule is

$$\chi_G^{(\ell+1)}(u) = \mathrm{HASH}^{(\ell)}\left(\chi_G^{(\ell)}(u), \left\{\!\!\left\{\chi_G^{(\ell)}(v) : v \in N(u)\right\}\!\!\right\}\right), \tag{2}$$

where $\chi_G^{(\ell)}(u)$ is the color of node u after iteration ℓ. The pair $\left(\chi_G^{(\ell)}(u), \left\{\!\!\left\{\chi_G^{(\ell)}(v) : v \in N(u)\right\}\!\!\right\}\right)$ is the *WL-type* of u at iteration ℓ, and $\mathrm{HASH}^{(\ell)}$ is required to be injective on the WL-types that occur at that iteration. For unattributed graphs, we start from the constant coloring $\chi_G^{(0)}(u) = 1$ for all $u \in V$. Since the WL test depends only on the partition induced by a coloring, rather than on the particular names of the colors, we write $\chi \equiv \chi'$ if $\chi(u) = \chi(v) \iff \chi'(u) = \chi'(v)$ for all $u, v \in V$. Accordingly, all simulation guarantees in this paper are stated up to such a relabeling of color names.

2.4 Problem Setup: Simulating One Step of the WL Test

We focus on one step of the WL test for colorings that can arise at arbitrary iterations of the WL test on unattributed graphs started from the constant coloring. Fix a color set $[C]$ large enough to serve as the label space for both input and output colorings. For a graph G, let $\chi_G^{(\ell)}$ denote the coloring after the ℓ-th WL iteration, starting from the constant coloring $\chi_G^{(0)}(u) = 1$ for all $u \in V(G)$. A pair (G, χ) is called a *valid input instance* if $\chi : V(G) \to [C]$ and $\chi \equiv \chi_G^{(\ell)}$ for some iteration $\ell \geq 0$.

We regard an MPGNN, together with its input encoding and output decoding, as a parameterized map from colorings to colorings. Let $\mathcal{P}$ denote the set of all allowable parameter choices of an MPGNN. For $\theta \in \mathcal{P}$ and a graph G, write $\mathcal{N}_\theta(\cdot; G) \colon [C]^{V(G)} \to [C]^{V(G)}$ for the coloring-to-coloring map induced by evaluating the MPGNN with parameters θ on the edge set of G. Here G is not part of the learned parameters; it specifies the graph on which the same parameter choice is evaluated. For a valid input instance (G, χ), define the WL-type of a node u under χ by $\mathrm{type}_{G,\chi}(u) := (\chi(u), \{\!\{\chi(v) : v \in N(u)\}\!\})$. We say that the parameter choice θ *simulates one step of the WL test* on a valid input instance (G, χ) if $\mathcal{N}_\theta(\chi; G)(u) = \mathcal{N}_\theta(\chi; G)(v) \iff \mathrm{type}_{G,\chi}(u) = \mathrm{type}_{G,\chi}(v)$ for all $u, v \in V(G)$.

For a graph family $\mathcal{G}$, an MPGNN **deterministically obliviously** simulates one step of the WL test on $\mathcal{G}$ over $[C]$ if there exists one parameter choice $\theta^\star \in \mathcal{P}$, chosen independently of the input instance, such that $\theta^\star$ simulates one step of the WL test on every valid input instance (G, χ) with $G \in \mathcal{G}$.

Extending deterministic oblivious simulation to the randomized setting involves a subtle choice between two notions of success. Let $\rho \in \{0,1\}^R$ be a random seed drawn from an input-independent distribution. A parameter-generation rule $\Phi : \{0,1\}^R \to \mathcal{P}$, fixed before the input instance is known, maps each seed ρ to the network parameters $\theta_\rho = \Phi(\rho)$. For a failure probability $p_f \in [0, 1)$, consider the following two guarantees over all valid input instances (G, χ) with $G \in \mathcal{G}$:

(i) *Per-instance success.* The simulation succeeds with probability at least $1 - p_f$ on each fixed input, i.e., for all (G, χ), $\Pr_\rho[\theta_\rho \text{ succeeds on } (G, \chi)] \geq 1 - p_f$.

(ii) *Simultaneous success.* With probability at least $1 - p_f$, the sampled parameters yield a simulator that is correct on every input instance, i.e., $\Pr_\rho[\forall (G, \chi),\ \theta_\rho \text{ succeeds on } (G, \chi)] \geq 1 - p_f$.

[5] In this paper, "width" refers to message dimension rather than hidden-state dimension. In many standard models the two coincide, but we keep the distinction explicit because our bounds are stated in terms of transmitted information.

In fact, for $p_f < 1$, setting (ii) is equivalent to the deterministic oblivious setting. Indeed, the simultaneous guarantee implies that some seed $\rho^\star$ succeeds on every instance; fixing this seed gives a deterministic oblivious simulator with no larger depth, width, or precision. Conversely, any deterministic oblivious simulator satisfies (ii) using a point-mass parameter distribution. We therefore adopt (i) as the randomized oblivious extension here. The term "oblivious" means that both the seed distribution and the map Φ from seeds to network parameters are fixed independently of the input instance. For each instance, a seed may be sampled from this same distribution, and the successful seeds may differ across instances. Formally, an MPGNN with R random bits **randomized obliviously** simulates one step of the WL test on $\mathcal{G}$ over $[C]$ with failure probability at most p_f if there exist an input-independent seed distribution and map Φ satisfying (i). The special case $p_f = 0$ of our per-instance guarantee is called **zero-error randomized oblivious** simulation. Because the seed space $\{0,1\}^R$ is finite, any supported seed that failed on any instance would give that instance positive failure probability. Hence every supported seed succeeds on every instance, and fixing any one gives a deterministic oblivious simulator with no larger d, w, p. Together with the converse above, this shows that deterministic and zero-error randomized oblivious simulation are also equivalent in these resources. We therefore treat these two settings together throughout the paper.

In contrast to oblivious simulation, where the parameter choice or its sampling distribution is fixed independently of the input, instance-dependent representation allows the parameters to depend on the input instance. Formally, an MPGNN admits an **instance-dependent representation** of one step of the WL test on $\mathcal{G}$ over $[C]$ if for every valid input instance (G, χ) with $G \in \mathcal{G}$, there exists a parameter choice $\theta_{G,\chi} \in \mathcal{P}$, possibly depending on (G, χ), such that $\theta_{G,\chi}$ simulates one step of the WL test on that instance. This definition concerns only the existence of suitable parameters for each instance, so there is no need to distinguish between deterministic and randomized instance-dependent representation. Whenever random parameter selection succeeds with positive probability on a fixed instance, at least one seed produces a parameter choice that correctly simulates one step of the WL test on that instance. Fixing such a seed separately for each instance gives the required instance-dependent parameters.

Instance-dependent representation concerns only the existence of suitable parameters; we further study **instance-dependent adaptation**, which asks how many adaptation rounds a network needs on a given instance before it can correctly simulate one step of the WL test. The parameter initialization rule and adaptation procedure are fixed independently of the target instance and shared across the graph family. On each valid input instance (G, χ), the procedure performs r adaptation rounds using graph-local communication to adapt a depth-d, width-w, precision-p MPGNN. The adapted network is then evaluated on that same instance using d message-passing layers. Adaptation succeeds on (G, χ) if this evaluation correctly simulates one step of the WL test. As in the randomized oblivious setting, randomized adaptation uses a finite random seed drawn from an input-independent distribution to govern parameter initialization, adaptation, and final evaluation, with the same per-instance success guarantee. By the same finite-seed argument as above, zero-error randomized adaptation is equivalent to deterministic adaptation. The full definition of instance-dependent adaptation is given in Section 4.2.

3 Oblivious Simulation Results

In this section, we analyze the hardness of obliviously simulating one step of the WL test with MPGNNs. Concretely, we study worst-case trade-offs among depth d, width w, and precision p.

3.1 The Deterministic and Randomized Zero-Error Cases

We first consider the deterministic and zero-error randomized settings, which are equivalent in attainable depth, width, and precision as shown in Section 2.4.

One might expect one step of the WL test to be computable by a depth-one MPGNN, because the WL-type of a node depends only on its own color and the multiset of neighboring colors. This intuition overlooks the need for an exact, collision-free relabeling of all WL-types that occur into a bounded color set. In the oblivious setting, this relabeling may require information to be coordinated across distant parts of the graph. The following theorem formalizes this obstruction.

Theorem 3.1. *For every sufficiently large integer n, every integer $m \in [n, n^2]$, and every integer $D \in [8, n]$, there exists a family $\mathcal{G}_{n,m,D}$ of connected graphs, each with $\Theta(n)$ nodes, $\Theta(m)$ edges,*

and diameter D. Then, for any color set $[C]$ with $C \geq \max_{G \in \mathcal{G}_{n,m,D}} |V(G)|$[6] and $\log C = o(m)$, any deterministic or zero-error randomized oblivious MPGNN that simulates one step of the WL test on $\mathcal{G}_{n,m,D}$ over $[C]$ must satisfy $d = \Omega\left(D + \frac{m}{wp}\right)$.

The proof is deferred to Appendix A. The proof idea is to construct a family of graphs in which two distant regions encode an instance $(\boldsymbol{a}, \boldsymbol{b})$ of the Equality problem from communication complexity [40, 21]. If an oblivious MPGNN correctly performs one step of the WL test on these graphs, then it can be used to construct a zero-error communication protocol for the Equality problem. Since each edge can transmit at most wp bits per layer, d layers can transmit only $O(dwp)$ bits across the communication bottleneck in each graph in the family. The zero-error communication lower bound for Equality yields the $\Omega\left(\frac{m}{wp}\right)$ term, while the separation between the relevant graph regions gives the additional $\Omega(D)$ locality term.

The lower bound gives a necessary amount of communication, measured through the product of depth, width, and precision, for oblivious exact simulation. We next ask to what extent this lower bound can be matched by an explicit construction. A natural source of upper-bound algorithms is the CONGEST model [24, 25, 31, 12] from distributed computing (see Appendix B for the definition), which was used by Loukas [26] as a computational abstraction for MPGNNs. However, standard CONGEST algorithms often assume unique node identifiers or port-level control. Unique identifiers would break the anonymity and permutation equivariance of the standard MPGNN model, and may not generalize well to graphs with unseen identifiers, while port-level control is not available in the unordered-neighborhood update of Equation (1).

For the upper bound only, we therefore consider a rooted, port-aware relaxation of the anonymous MPGNN model. One node receives a special one-bit marker designating it as the root. In addition, each node can distinguish its incident edge ports, maintain a hidden state for each port, send messages depending on the corresponding port state, and update each port state using the node state, the current port state, and the message received through that port (see Appendix B for a detailed definition). This relaxation is strictly stronger than the anonymous unordered-neighborhood MPGNN model used in the lower-bound theorem. The following theorem gives an upper bound for simulating one step of the WL test under this model.

Theorem 3.2. *Given any family of graphs with at most n nodes, at most m edges, and diameter at most D, where each graph is supplied with an arbitrary designated root marker, and a color set $[C]$ large enough to represent all input and output colors under consideration (e.g., $C \geq n$), there exists an oblivious rooted, port-aware MPGNN with depth $d = O\left(D + \frac{m \log C}{wp}\right)$, sufficiently large node and port state dimensions, and sufficiently expressive message, node-update, and port-update functions, that simulates one step of the WL test in both the deterministic and zero-error randomized settings.*

The proof is deferred to Appendix B. The proof idea is to use the root marker and port states to construct a rooted spanning tree. The WL-types that occur are then aggregated to the root, where duplicate types are removed and the distinct types are sorted to assign ranks, which serve as the new color names. The resulting type-to-color map is broadcast back to all nodes along the tree, after which each node computes its new color by applying this map to its own WL-type. This result shows that, in the stronger rooted, port-aware model, the deterministic lower bound is matched up to the $\log C$ factor in the communication term. Whether a comparable deterministic upper bound can be achieved in the fully anonymous model remains open.

3.2 The Bounded-Error Randomized Case

We then turn to bounded-error randomized simulation, where the failure probability may be positive but must be at most p_f, leading to a picture quite different from the deterministic and zero-error randomized cases.

Aamand et al. [1] explored this question using universal hashing and pseudorandomness as tools. We now restate their more efficient construction as Lemma 3.3.

[6] Kiefer et al. [20] constructed a family of graphs for which the WL test requires $|V(G)| - 1$ rounds to stabilize, and hence requires $|V(G)|$ colors. Therefore, in the oblivious setting, the color set size C should be at least $|V(G)|$.

Lemma 3.3 (cf. Section 3 of [1])**.** *Given any family of graphs with at most n nodes and a failure probability $p_f \in (0,1)$ with $p_f = \Omega\left(\frac{1}{\mathrm{poly}(n)}\right)$, there exists an MPGNN with depth $d = 1$ and bandwidth $wp = O(\log n)$ that simulates one step of the WL test with probability at least $1 - p_f$, using $R = O(\log^2 n)$ random bits and a color set $[C]$ of size $C = \Theta(n^{c_0})$ for some constant c_0 depending on p_f*[7]*.*

We describe the construction behind Lemma 3.3 in Appendix C. The advantage of their construction is that its local computation can be implemented by ReLU networks, and the authors also analyze lower bounds for it. A limitation of that construction is that the required color-set size depends on the failure probability p_f, and the constant c_0 is difficult to determine precisely. Motivated by this, we present two alternative constructions in Theorem 3.4 and Theorem 3.5. The first construction gives an upper bound for large color sets in the fully anonymous MPGNN model.

Theorem 3.4. *Given any family of graphs with at most n nodes, a failure probability $p_f \in (0,1)$, and a color set $[C]$ with $C \geq \left\lceil \frac{3n^2}{2p_f} \right\rceil$, there exists an MPGNN with depth $d = 1$, $wp \geq \lceil \log_2 C \rceil$, sufficiently large node state dimension, and sufficiently expressive message and node-update functions, that simulates one step of the WL test with probability at least $1 - p_f$, using $R = \Theta\left(\log \frac{n \log C}{p_f}\right)$ random bits.*

The second construction gives an upper bound for arbitrary color sets in the rooted, port-aware model.

Theorem 3.5. *Given any family of graphs with at most n nodes and diameter at most D, where each graph is supplied with an arbitrary designated root marker, a failure probability $p_f \in (0,1)$, a color set $[C]$ with $C \geq n$, there exists a rooted, port-aware MPGNN with depth $d = O\left(D + \frac{\log C + n \log(n/p_f)}{wp}\right)$, sufficiently large node and port state dimensions, and sufficiently expressive message, node-update, and port-update functions, that simulates one step of the WL test with probability at least $1 - p_f$, using $R = \Theta\left(\log \frac{n \log C}{p_f}\right)$ random bits.*

The proofs of Theorem 3.4 and Theorem 3.5 are deferred to Appendix D. The proofs mainly rely on almost universal hashing [5, 2]. Once each node has formed its WL-type, it can hash this type to a range $[M]$ of size $M = \Theta\left(\frac{n^2}{p_f}\right)$. By a union bound over all pairs among the at most n WL-types that occur, the probability that any two distinct WL-types that occur collide is at most p_f. If the size of the color set is at least $\Theta\left(\frac{n^2}{p_f}\right)$, then the hash value itself can be treated as the new color, and no communication beyond forming the WL-type is needed. When C is below the $\Theta\left(\frac{n^2}{p_f}\right)$ threshold, we additionally apply a construction similar to Theorem 3.2 to sort the hash values and use their ranks as the new colors. In the first construction (Theorem 3.4), consider the practical setting where $p_f = \Omega\left(\frac{1}{\mathrm{poly}(n)}\right)$ and $\left\lceil \frac{3n^2}{2p_f} \right\rceil \leq C \leq \mathrm{poly}(n)$. If $wp = k \log n$ for a sufficiently large constant $k > 0$, then $d = 1$. In this regime, the MPGNN depth and per-layer bandwidth wp are comparable to those in Lemma 3.3, but we reduce the number of random bits from $O(\log^2 n)$ to $\Theta\left(\log \frac{n \log C}{p_f}\right) = \Theta(\log n)$, and we can explicitly characterize the required color-set size. In the second construction (Theorem 3.5), the depth and total communication requirements are larger, but the color-set size can be any $C \geq n$ and no longer depends on p_f.

Both the construction of Aamand et al. [1] and our large-color construction achieve low total communication dwp. However, it remains unclear whether the number of random bits required by such constructions is close to optimal. In Theorem 3.6 below, we prove a lower bound on the required number of random bits.

Theorem 3.6. *For every sufficiently large integer n, there exists a family $\mathcal{G}_n$ of connected graphs, each with $\Theta(n)$ nodes. Then, for any failure probability $p_f \in (0, 1/9)$ and any color set $[C]$ with $C \geq \max_{G \in \mathcal{G}_n} |V(G)|$ and $\log C = o(n)$, any randomized oblivious MPGNN with $dwp = O(\log C)$*

[7] For their hashing-based construction, ensuring that, with probability at least $1 - p_f$, no two distinct WL-types that occur collide requires a range of size on the order of $\frac{n^2}{p_f}$ by a union-bound argument. In particular, this implies $c_0 \geq 2$.

that simulates one step of the WL test on $\mathcal{G}_n$ over $[C]$ with probability at least $1 - p_f$ must use $R = \Omega\left(\log \frac{n}{p_f \log C}\right)$ random bits.

The proof of Theorem 3.6 is deferred to Appendix E. The proof again reduces the Equality problem to simulating a single step of the WL test, and then applies the lower bound from [18] on the communication complexity of the Equality problem in terms of the error probability p_f and the number of random bits R. When instantiated with $wp = O(\log C)$, the construction in Theorem 3.4 has $dwp = O(\log C)$. For polynomial-size color sets satisfying $C \geq \left\lceil \frac{3n^2}{2p_f} \right\rceil$ and $p_f \in (0, 1/9)$, the upper and lower bounds on the number of random bits match up to constant factors.

When the allowed color-set size is below the $\Theta\left(\frac{n^2}{p_f}\right)$ threshold required by Theorem 3.4, Theorem 3.5 provides an alternative upper bound on the trade-off among depth, width, and precision. A natural question is whether this upper bound can be improved under such a restricted color space. In Theorem 3.7 below, we show that when $C = \Theta(n)$, any randomized oblivious MPGNN that simulates one step of the WL test on the hard family must have depth $d = \Omega\left(D + \min\left\{\log^* n, \frac{n}{wp}\right\}\right)$.

Theorem 3.7. *For every sufficiently large integer n and every integer $D \in [8, n]$, there exists a family $\mathcal{G}_{n,D}$ of connected graphs, each with $\Theta(n)$ nodes and diameter $\Theta(D)$. Then, for any failure probability $p_f \in (0, 1/6)$ satisfying $p_f = \Omega\left(\frac{1}{\mathrm{poly}(n)}\right)$, and any color set $[C]$ with $C \geq \max_{G \in \mathcal{G}_{n,D}} |V(G)|$ and $C = \Theta(n)$, any randomized oblivious MPGNN that simulates one step of the WL test on $\mathcal{G}_{n,D}$ over $[C]$ with probability at least $1 - p_f$ must satisfy $d = \Omega\left(D + \min\left\{\log^* n, \frac{n}{wp}\right\}\right)$.*

The proof of Theorem 3.7 is deferred to Appendix F. The proof reduces the $\mathrm{ExistsEqual}$ problem (see, e.g., [17]) to simulating one step of the WL test with success probability at least $1 - p_f$.

To summarize, bounded-error randomness leads to a markedly different picture from the deterministic and zero-error regimes. When a sufficiently large polynomial-size color set is available, for example when $C = \mathrm{poly}(n)$, $C = \Omega\left(\frac{n^2}{p_f}\right)$, and $p_f = \Omega\left(\frac{1}{\mathrm{poly}(n)}\right)$, one step of the WL test can be simulated by an oblivious MPGNN with $d = 1$, $wp = O(\log n)$, and $R = \Theta(\log n)$ random bits. At the minimum color set size $C = \Theta(n)$, however, for $p_f = \Omega\left(\frac{1}{\mathrm{poly}(n)}\right)$, Theorem 3.7 rules out constant-depth simulations with logarithmic per-layer bandwidth $wp = \Theta(\log n)$ in the anonymous MPGNN model. This highlights that bounded-error randomness can dramatically reduce the cost of simulating one step of the WL test in the large-color regime, while small color sets still impose a nontrivial communication-depth trade-off.

We also note that the lower-bound proofs in this section do not rely on anonymity or unordered neighborhood aggregation in the basic MPGNN model. They use only two constraints that remain valid in the rooted, port-aware strengthening used for our upper bounds: each edge transmits at most $O(wp)$ bits per layer, and a depth-d message-passing computation has locality radius at most d. Consequently, the lower bounds in this section also hold for the rooted, port-aware model.

4 Instance-Dependent Simulation Results

In the previous section, we studied the hardness of simulating one step of the WL test with MPGNNs in the oblivious setting. In this section, we remove this input-independence requirement and study instance-dependent simulation.

We separate the issue into two questions. First, if the parameters may be chosen by an oracle after seeing the target instance, can a shallow MPGNN represent one step of the WL test on that instance? Second, if such parameters exist, can they be found efficiently by a local target-instance adaptation procedure? The answer to the first question is positive: shallow instance-specific parameters always exist. The answer to the second question is negative in the worst case: under a decentralized parameter-adaptation model, finding such parameters still requires global communication.

4.1 Instance-Dependent Representation

As discussed in Section 2.4, there is no need to distinguish between deterministic and randomized instance-dependent representation. We first show that, once input-independence is removed, the representational difficulty of one step of the WL test essentially disappears.

Theorem 4.1. *Given any valid input instance (G, χ), where G is a graph with n nodes, any color set $[C]$ with $C \geq n$, and any width w and precision p satisfying $wp \geq \lceil \log_2 n \rceil$, there exists an instance-dependent parameter choice $\theta_{G,\chi}$ for a deterministic depth-one MPGNN with width w, precision p, sufficiently large hidden-state dimension, and instance-dependent embedding, message, update, and decoding functions, such that the resulting MPGNN simulates one step of the WL test on the fixed instance (G, χ), that is, for all $u, v \in V(G)$, $\mathcal{N}_{\theta_{G,\chi}}(\chi; G)(u) = \mathcal{N}_{\theta_{G,\chi}}(\chi; G)(v) \iff \mathrm{type}_{G,\chi}(u) = \mathrm{type}_{G,\chi}(v)$.*

The proof of Theorem 4.1 is deferred to Appendix G. The proof first compresses each input color injectively into $\lceil \log_2 n \rceil$-bit codes, so that each node can transmit its compressed color to all neighbors in one layer. Each node then forms a canonical vectorization of its compressed WL-type. Finally, because at most n compressed WL-types occur and $C \geq n$, an instance-dependent decoder can injectively map the occurring WL-types to colors in $[C]$.

4.2 Instance-Dependent Adaptation

As discussed in Section 2.4, the existence of suitable parameters for each instance does not imply that they can be found efficiently. We therefore study instance-dependent adaptation, for which we give the full definition below. A motivating setting is target-specific adaptation on a graph too large for a single machine, where machines storing different graph partitions must communicate to coordinate the adaptation. Our framework captures this coordination when the adaptation procedure obeys the communication constraints defined below.

The parameter initialization rule and adaptation algorithm are fixed independently of the target instance and shared across the graph family. The initial parameters θ_0 may, for example, be fixed, hand-designed, or pretrained on data independent of the target family. Given a target instance (G, χ), each node initially knows only its own input color, its incident edges or ports, and the target-independent initialization. Any additional target-dependent information, including any loss, gradient, collision statistic, or other statistic used by the adaptation rule, must be obtained through local computation and communication along the edges of G. The desired refined color labels are not provided during adaptation.

One adaptation round allows arbitrary local computation and communication consistent with a forward-backward computation of a depth-d, width-w, precision-p MPGNN. In each round, at most $O(dwp)$ bits can cross each edge, and information can propagate at most $O(d)$ hops. After r rounds, the algorithm produces an adapted configuration $\Gamma_{r;G,\chi} = \textsc{Adapt}^{(r)}(\theta_0, G, \chi)$ containing parameters and node-local states. Each node may use only the part of this configuration stored locally or received through subsequent message passing. After adaptation, a depth-d, width-w, precision-p message-passing computation initialized with $\Gamma_{r;G,\chi}$ produces $\chi' = \mathcal{N}_{\Gamma_{r;G,\chi}}(\chi; G)$. Each edge transmits at most wp bits per layer in each direction. The procedure succeeds on (G, χ) if $\chi'(u) = \chi'(v) \iff \mathrm{type}_{G,\chi}(u) = \mathrm{type}_{G,\chi}(v)$ for all $u, v \in V(G)$.

As discussed in Section 2.4, deterministic and zero-error randomized adaptation are equivalent. We next establish a worst-case lower bound on the number of adaptation rounds in these settings.

Theorem 4.2. *For every sufficiently large integer n, every integer $m \in [n, n^2]$, and every integer $D \in [8, n]$, there exists a family $\mathcal{G}_{n,m,D}$ of connected graphs, each with $\Theta(n)$ nodes, $\Theta(m)$ edges, and diameter D. For any color set $[C]$ with $C \geq \max_{G \in \mathcal{G}_{n,m,D}} |V(G)|$ and $\log C = o(m)$, any deterministic or zero-error randomized decentralized parameter-adaptation framework that uses $r \geq 1$ adaptation rounds to adapt the parameters of a depth-d, width-w, precision-p MPGNN, so that for every valid input instance (G, χ) with $G \in \mathcal{G}_{n,m,D}$ the resulting MPGNN simulates one step of the WL test on (G, χ), must satisfy $r = \Omega\left(\frac{D}{d} + \frac{m}{dwp}\right)$.*

The proof is deferred to Appendix H. The proof reduces Equality to repeated simulation of adaptation and final inference on the hard graphs of Theorem 3.1. When $r = 0$, the initial network must

already be an oblivious simulator. For $r \geq 1$, the reduction also accounts for communication during adaptation and yields the stated lower bound on the number of adaptation rounds.

5 Discussion and Limitations

Our results show that the qualitative comparison between MPGNNs and the WL test conceals a quantitative relabeling problem. This motivates a finer-grained view of GNN expressivity under explicit resource constraints, with three implications for GNN design. First, allowing randomization and a small failure probability can reduce the depth and communication needed to simulate one step of the WL test with a sufficiently large color space. Second, rooted aggregation and port-aware routing yield explicit upper bounds, motivating further study of this model and its possible resource advantages over fully anonymous MPGNNs. Finally, deterministic or zero-error randomized graph-local adaptation can require global communication even when a depth-one representation exists, so representational capacity and adaptation cost should be assessed separately.

Meanwhile, our results have several limitations. First, some of our upper bounds in the oblivious setting are proved in a rooted, port-aware relaxation, and whether comparable bounds can be obtained in the fully anonymous unordered-neighborhood MPGNN model remains open. Second, in the randomized setting, the intermediate regime between $C = \Theta(n)$ and $C = \Theta\left(\frac{n^2}{p_f}\right)$ is not fully characterized. Third, our upper bounds allow large local memory and expressive local computations; incorporating explicit constraints on local state size or on implementability by standard neural architectures is an important direction for future work. Fourth, our upper bounds apply to every valid instance under their stated assumptions, and hence under any distribution supported on such instances, whereas our lower bounds are witnessed by specially constructed hard families. Establishing comparable average-case lower bounds for natural or real-world graph distributions remains open. Finally, the lower bound for instance-dependent adaptation relies on a local adaptation model, and it does not preclude centralized algorithms with global access to the whole target graph.

Extending our bounds to higher-order WL tests, such as k-WL, requires a corresponding communication model beyond standard anonymous MPGNNs on the original graph. A natural direction is to study relabeling on message-passing structures over tuples of vertices [29], with new reductions tailored to their communication pattern.

Acknowledgments and Disclosure of Funding

This research was supported in part by National Natural Science Foundation of China (No. 92470128, No. U2241212). We also wish to acknowledge the support provided by the fund for building world-class universities (disciplines) of Renmin University of China, by Engineering Research Center of Next-Generation Intelligent Search and Recommendation, Ministry of Education, by Intelligent Social Governance Interdisciplinary Platform, Major Innovation & Planning Interdisciplinary Platform for the "Double-First Class" Initiative, Public Policy and Decision-making Research Lab, and Public Computing Cloud, Renmin University of China.

References

[1] Anders Aamand, Justin Y. Chen, Piotr Indyk, Shyam Narayanan, Ronitt Rubinfeld, Nicholas Schiefer, Sandeep Silwal, and Tal Wagner. Exponentially improving the complexity of simulating the weisfeiler-lehman test with graph neural networks. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

[2] Noga Alon, Oded Goldreich, Johan Håstad, and René Peralta. Simple constructions of almost k-wise independent random variables. *Random Structures & Algorithms*, 3(3):289–304, 1992.

[3] Emily Alsentzer, Samuel G. Finlayson, Michelle M. Li, and Marinka Zitnik. Subgraph neural networks. 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.

[4] Joshua Brody, Amit Chakrabarti, Ranganath Kondapally, David P Woodruff, and Grigory Yaroslavtsev. Certifying equality with limited interaction. *Algorithmica*, 76(3):796–845, 2016.

[5] J Lawrence Carter and Mark N Wegman. Universal classes of hash functions. In *Proceedings of the ninth annual ACM symposium on Theory of computing*, pages 106–112, 1977.

[6] Jiajia Chen, Jiancan Wu, Jiawei Chen, Xin Xin, Yong Li, and Xiangnan He. How graph convolutions amplify popularity bias for recommendation? *Frontiers of Computer Science*, 18(5):185603, 2024.

[7] Leonardo Cotta, Christopher Morris, and Bruno Ribeiro. Reconstruction for powerful graph representations. 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 1713–1726, 2021.

[8] Jiarui Feng, Yixin Chen, Fuhai Li, Anindya Sarkar, and Muhan Zhang. How powerful are k-hop message passing graph neural networks. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

[9] Jiarui Feng, Lecheng Kong, Hao Liu, Dacheng Tao, Fuhai Li, Muhan Zhang, and Yixin Chen. Extending the design space of graph neural networks by rethinking folklore weisfeiler-lehman. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

[10] Fabrizio Frasca, Beatrice Bevilacqua, Michael M. Bronstein, and Haggai Maron. Understanding and extending subgraph gnns by rethinking their symmetries. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

[11] 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.

[12] Mohsen Ghaffari. Distributed graph algorithms. Course Notes for Distributed Algorithms, 2022. Accessed: September, 2024.

[13] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 1263–1272. PMLR, 2017.

[14] William L. Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 1024–1034, 2017.

[15] Jiaqi Han, Jiacheng Cen, Liming Wu, Zongzhao Li, Xiangzhe Kong, Rui Jiao, Ziyang Yu, Tingyang Xu, Fandi Wu, Zihe Wang, Hongteng Xu, Zhewei Wei, Deli Zhao, Yang Liu, Yu Rong, and Wenbing Huang. A survey of geometric graph neural networks: data structures, models and applications. *Frontiers of Computer Science*, 19(11):1911375, 2025.

[16] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yong-Dong Zhang, and Meng Wang. Lightgcn: Simplifying and powering graph convolution network for recommendation. In Jimmy X. Huang, Yi Chang, Xueqi Cheng, Jaap Kamps, Vanessa Murdock, Ji-Rong Wen, and Yiqun Liu, editors, *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020*, pages 639–648. ACM, 2020.

[17] Dawei Huang, Seth Pettie, Yixiang Zhang, and Zhijun Zhang. The communication complexity of set intersection and multiple equality testing. In *Proceedings of the Thirty-First Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '20, page 1715–1732, USA, 2020. Society for Industrial and Applied Mathematics.

[18] Dale Jacobs, John Jeang, Vladimir Podolskii, Morgan Prior, and Ilya Volkovich. Communication complexity of equality and error-correcting codes. In C. Aiswarya, Ruta Mehta, and Subhajit Roy, editors, *45th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2025)*, volume 360 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 37:1–37:19, 2025.

[19] Thorsten Joachims. Transductive inference for text classification using support vector machines. In Ivan Bratko and Saso Dzeroski, editors, *Proceedings of the Sixteenth International Conference on Machine Learning (ICML 1999), Bled, Slovenia, June 27 - 30, 1999*, pages 200–209. Morgan Kaufmann, 1999.

[20] Sandra Kiefer and Brendan D. McKay. The iteration number of colour refinement. In Artur Czumaj, Anuj Dawar, and Emanuela Merelli, editors, *47th International Colloquium on Automata, Languages, and Programming, ICALP 2020, Saarbrücken, Germany (Virtual Conference), July 8-11, 2020*, LIPIcs, pages 73:1–73:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[21] Eyal Kushilevitz and Noam Nisan. *Communication complexity*. Cambridge University Press, 1997.

[22] Remi Lam, Alvaro Sanchez-Gonzalez, Matthew Willson, Peter Wirnsberger, Meire Fortunato, Ferran Alet, Suman Ravuri, Timo Ewalds, Zach Eaton-Rosen, Weihua Hu, et al. Learning skillful medium-range global weather forecasting. *Science*, 382(6677):1416–1421, 2023.

[23] Yongquan Liang, Qiuyu Song, Zhongying Zhao, Hui Zhou, and Maoguo Gong. BA-GNN: Behavior-aware graph neural network for session-based recommendation. *Frontiers of Computer Science*, 17(6):176613, 2023.

[24] Nathan Linial. Distributive graph algorithms-global solutions from local data. In *28th Annual Symposium on Foundations of Computer Science, Los Angeles, California, USA, 27-29 October 1987*, pages 331–335. IEEE Computer Society, 1987.

[25] Nathan Linial. Locality in distributed graph algorithms. *SIAM J. Comput.*, 21(1):193–201, 1992.

[26] 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.

[27] Haggai Maron, Heli Ben-Hamu, Hadar Serviansky, and Yaron Lipman. Provably powerful graph networks. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 2153–2164, 2019.

[28] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of machine learning*. MIT press, 2018.

[29] Christopher Morris, Martin Ritzert, Matthias Fey, William L. Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The*

Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019, pages 4602–4609. AAAI Press, 2019.

[30] Pál András Papp, Karolis Martinkus, Lukas Faber, and Roger Wattenhofer. Dropgnn: Random dropouts increase the expressiveness of graph neural 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 21997–22009, 2021.

[31] David Peleg. *Distributed computing: a locality-sensitive approach*. Society for Industrial and Applied Mathematics, USA, 2000.

[32] Mert Saglam and Gábor Tardos. On the communication complexity of sparse set disjointness and exists-equal problems. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, pages 678–687. IEEE, 2013.

[33] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning - From Theory to Algorithms*. Cambridge University Press, 2014.

[34] Qian Tao, Xiyuan Wang, Muhan Zhang, Shuxian Hu, Wenyuan Yu, and Jingren Zhou. Exact acceleration of subgraph graph neural networks by eliminating computation redundancy. *Frontiers of Computer Science*, 20(12):2012626, 2026.

[35] Vladimir Naumovich Vapnik. *The Nature of Statistical Learning Theory, Second Edition*. Statistics for Engineering and Information Science. Springer, 2000.

[36] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. Neural graph collaborative filtering. In Benjamin Piwowarski, Max Chevalier, Éric Gaussier, Yoelle Maarek, Jian-Yun Nie, and Falk Scholer, editors, *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2019, Paris, France, July 21-25, 2019*, pages 165–174. ACM, 2019.

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

[38] Zhaoping Xiong, Dingyan Wang, Xiaohong Liu, Feisheng Zhong, Xiaozhe Wan, Xutong Li, Zhaojun Li, Xiaomin Luo, Kaixian Chen, Hualiang Jiang, et al. Pushing the boundaries of molecular representation for drug discovery with the graph attention mechanism. *Journal of medicinal chemistry*, 63(16):8749–8760, 2019.

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

[40] Andrew Chi-Chih Yao. Some complexity questions related to distributive computing (preliminary report). In *Proceedings of the Eleventh Annual ACM Symposium on Theory of Computing*, STOC '79, pages 209–213, New York, NY, USA, 1979. Association for Computing Machinery.

[41] Zhihang Yi, Hairong Wang, Fangping Chen, Zhaojing Xu, and Jianling Yang. Graph contrastive learning view construction methods in recommender systems: a survey. *Frontiers of Computer Science*, 20(7):2007338, 2026.

[42] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In Yike Guo and Faisal Farooq, editors, *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*, pages 974–983. ACM, 2018.

[43] Bohang Zhang, Shengjie Luo, Liwei Wang, and Di He. Rethinking the expressive power of gnns via graph biconnectivity. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

[44] Xuan Zhang, Cheng Chen, Xiaoting Wang, Haitao Jiang, Wei Zhao, and Xuefeng Cui. MetaGIN: a lightweight framework for molecular property prediction. *Frontiers of Computer Science*, 19(5):195912, 2025.

[45] Cai Zhou, Xiyuan Wang, and Muhan Zhang. From relational pooling to subgraph gnns: A universal framework for more expressive graph neural networks. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 42742–42768. PMLR, 2023.

[46] Junru Zhou, Jiarui Feng, Xiyuan Wang, and Muhan Zhang. Distance-restricted folklore weisfeiler-leman gnns with provable cycle counting power. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

A Communication Complexity Background and Proof of Theorem 3.1

A.1 Introduction to Communication Complexity

The proof of Theorem 3.1 relies on tools from communication complexity, so we briefly review the necessary background to make the required concepts self-contained. Communication complexity was first introduced by Yao [40]. In two-party communication complexity, two parties, Alice and Bob, wish to compute a function $f : X \times Y \to Z$, where X and Y are their input domains, respectively. They agree on a strategy beforehand but are separated before receiving their inputs $(x, y) \in X \times Y$. They then exchange messages to compute $f(x, y)$, with the goal of minimizing the total number of bits exchanged.

In deterministic communication, the strategy is fixed, and the minimum number of bits required to compute f in this setting is known as the deterministic communication complexity, denoted by $D(f)$. Similarly, in randomized communication, where Alice and Bob can use random bits and a two-sided error of ϵ is allowed, the minimum number of bits required is the randomized communication complexity. If the randomness is private, it is denoted by $R_\epsilon^{\mathsf{prv}}(f)$, and if it is public, it is denoted by $R_\epsilon^{\mathsf{pub}}(f)$.

The Equality (EQ) problem between two n-bit strings, denoted by $\mathrm{EQ}_n : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$, is defined as

$$\mathrm{EQ}_n(\boldsymbol{x}, \boldsymbol{y}) = \begin{cases} 1, & \boldsymbol{x} = \boldsymbol{y}, \\ 0, & \text{otherwise.} \end{cases}$$

It is one of the most well-studied problems in two-party communication complexity. We summarize its communication complexity under different settings in Table 2 below.

Table 2: Communication complexity of the EQ_n function under different settings.

Function	Deterministic	Randomized			
		Private Coin		Public Coin	
	$D(\cdot)$	$R_0^{\mathsf{prv}}(\cdot)$	$R_{1/3}^{\mathsf{prv}}(\cdot)$	$R_0^{\mathsf{pub}}(\cdot)$	$R_{1/3}^{\mathsf{pub}}(\cdot)$
EQ_n	$\Theta(n)^\dagger$	$\Theta(n)^\dagger$	$\Theta(\log n)^\dagger$	$\Theta(n)^*$	$\Theta(1)^\dagger$

† The proofs can be found in [21].

* This follows from the bound $R_0^{\mathsf{prv}}(f) = O\left(R_0^{\mathsf{pub}}(f) + \log n\right)$ (see Exercise 3.15 in [21]).

A.2 Proof of Theorem 3.1

We now prove Theorem 3.1.

Theorem 3.1. *For every sufficiently large integer n, every integer $m \in [n, n^2]$, and every integer $D \in [8, n]$, there exists a family $\mathcal{G}_{n,m,D}$ of connected graphs, each with $\Theta(n)$ nodes, $\Theta(m)$ edges, and diameter D. Then, for any color set $[C]$ with $C \geq \max_{G \in \mathcal{G}_{n,m,D}} |V(G)|$ and $\log C = o(m)$, any deterministic or zero-error randomized oblivious MPGNN that simulates one step of the WL test on $\mathcal{G}_{n,m,D}$ over $[C]$ must satisfy $d = \Omega\left(D + \frac{m}{wp}\right)$.*

Proof. At a high level, we prove the result by reducing Equality to the problem of simulating one step of the WL test. To be concrete, for every sufficiently large positive integer n, every $m \in [n, n^2]$, every $D \in [8, n]$, and for every EQ_m instance $(\boldsymbol{a}, \boldsymbol{b}) \in \{0,1\}^m \times \{0,1\}^m$, we construct a connected uncolored graph G with $\Theta(n)$ nodes, $\Theta(m)$ edges, and diameter exactly D, together with two distinguished nodes $x^{(A)}$ and $x^{(B)}$ satisfying $\mathrm{dist}_G\left(x^{(A)}, x^{(B)}\right) = \Theta(D)$, such that $\chi_G^{(3)}\left(x^{(A)}\right) = \chi_G^{(3)}\left(x^{(B)}\right) \iff \boldsymbol{a} = \boldsymbol{b}$. Thus, by chaining together three oblivious MPGNNs that simulate one step of the WL test, and then comparing the colors of two nodes via communication, we obtain a communication protocol for solving the Equality problem. The communication-complexity lower bound for Equality then yields the desired lower bound on the relevant parameters of the MPGNN.

Construction. The entire construction is divided into four steps. In the first step, we construct a fixed basic graph which is partitioned between Alice (A) and Bob (B). In the second step, given an input instance $(\boldsymbol{a}, \boldsymbol{b})$ of EQ_m, we add edges to the basic graph and assign virtual colors to certain nodes, thereby obtaining a virtually colored hard instance. In the third step, we further augment the above virtually colored hard instance by adding nodes and edges to decolorize it, thus obtaining an uncolored hard instance. In the last step, we add filler nodes so that the resulting graph satisfies the desired size requirement.

Step One: Fixed Basic Graph. Let $r \coloneqq \lceil \sqrt{m} \rceil$, and $q \coloneqq \left\lceil \frac{m}{r} \right\rceil$ be two positive integers. For each side $\sigma \in \{A, B\}$, create nodes $x^{(\sigma)}$, $w_1^{(\sigma)}, \cdots, w_q^{(\sigma)}$, $u_1^{(\sigma)}, \cdots, u_r^{(\sigma)}$, and $v_1^{(\sigma)}, \cdots, v_r^{(\sigma)}$; and add the fixed edges $\left\{x^{(\sigma)}, w_i^{(\sigma)}\right\}$ for $i \in [q]$, $\left\{x^{(\sigma)}, u_j^{(\sigma)}\right\}$ and $\left\{x^{(\sigma)}, v_j^{(\sigma)}\right\}$ for all $j \in [r]$. Then, connect the two sides by a path, for example by adding nodes $c_1, \cdots, c_{D-7}$ and edges $\left\{x^{(A)}, c_1\right\}, \{c_1, c_2\}, \cdots, \left\{c_{D-7}, x^{(B)}\right\}$[8].

Next, Alice holds the induced subgraph on the vertices with superscript (A), while Bob holds the induced subgraph on the vertices with superscript (B). We partition the connecting path at an arbitrary edge, assigning all vertices on one side of this edge to Alice and all vertices on the other side to Bob. Thus the Alice-Bob cut contains exactly one graph edge. Figure 1 below gives an illustration of the fixed basic graph. The edges $\left\{x^{(\sigma)}, u_j^{(\sigma)}\right\}$ and $\left\{x^{(\sigma)}, v_j^{(\sigma)}\right\}$ are omitted, and the connecting path is simplified.

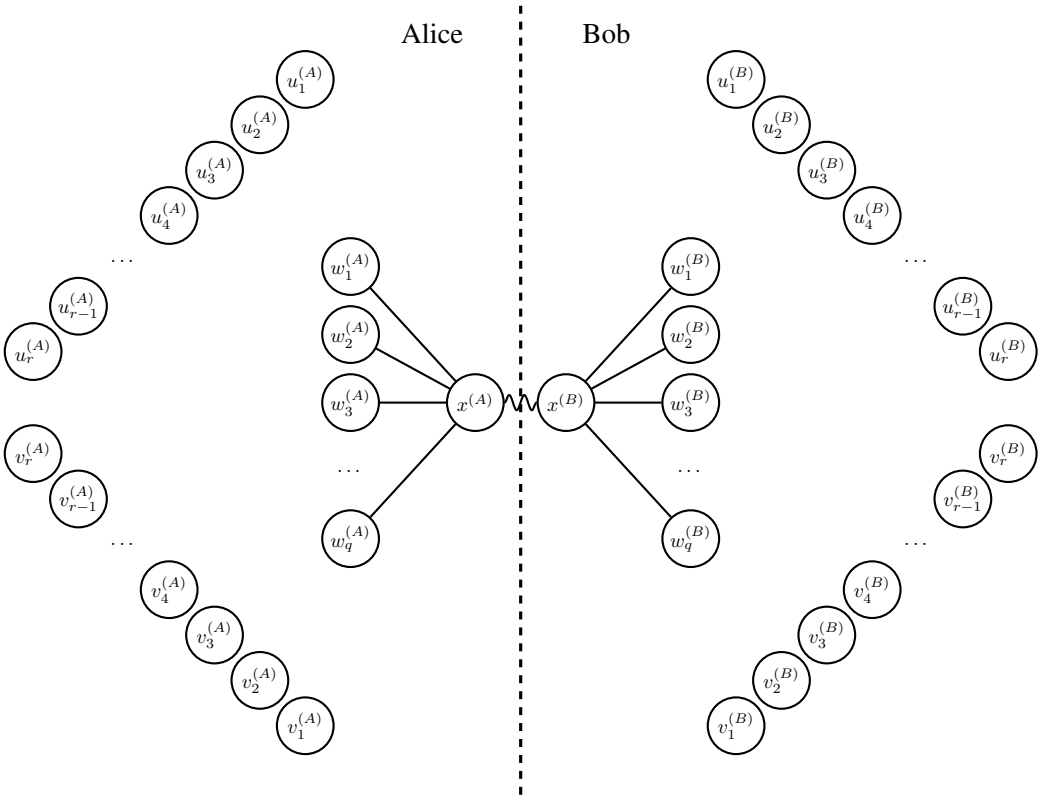

Figure 1: The constructed basic graph. The edges $\left\{x^{(\sigma)}, u_j^{(\sigma)}\right\}$ and $\left\{x^{(\sigma)}, v_j^{(\sigma)}\right\}$ are omitted, and the connecting path is simplified.

Step Two: Virtually Colored Hard Instance. We now encode an instance of EQ_m and assign virtual colors to selected nodes. Fix an injection $\tau : [m] \to [q] \times [r]$ by $\tau(k) = \left(\left\lceil \frac{k}{r} \right\rceil, (k-1) \bmod r + 1\right)$. Given inputs $\boldsymbol{a}, \boldsymbol{b} \in \{0,1\}^m$, define bits $a_{i,j}$ and $b_{i,j}$ as follows. If $(i, j) = \tau(k)$ for some $k \in [m]$,

[8]The distinguished nodes are used only by the reduction to specify which two output colors Alice and Bob compare. They are not given as additional node features to the MPGNN or to the WL test.

set $a_{i,j} \coloneqq \boldsymbol{a}_k$ and $b_{i,j} \coloneqq \boldsymbol{b}_k$. Otherwise set $a_{i,j} = b_{i,j} \coloneqq 0$. For every $(i,j) \in [q] \times [r]$, Alice adds the edge $\left\{w_i^{(A)}, u_j^{(A)}\right\}$ if $a_{i,j} = 0$, and the edge $\left\{w_i^{(A)}, v_j^{(A)}\right\}$ if $a_{i,j} = 1$. Bob adds the analogous edge $\left\{w_i^{(B)}, u_j^{(B)}\right\}$ if $b_{i,j} = 0$, and the edge $\left\{w_i^{(B)}, v_j^{(B)}\right\}$ if $b_{i,j} = 1$.

We next assign some virtual colors to certain nodes. Define virtual colors by $\kappa\left(w_i^{(\sigma)}\right) = i$, $\kappa\left(u_j^{(\sigma)}\right) = q + j$, and $\kappa\left(v_j^{(\sigma)}\right) = q + r + j$. The nodes $x^{(A)}$, $x^{(B)}$ and connecting nodes c_i do not need virtual colors. Figure 2 below gives an illustration of the virtually colored hard instance.

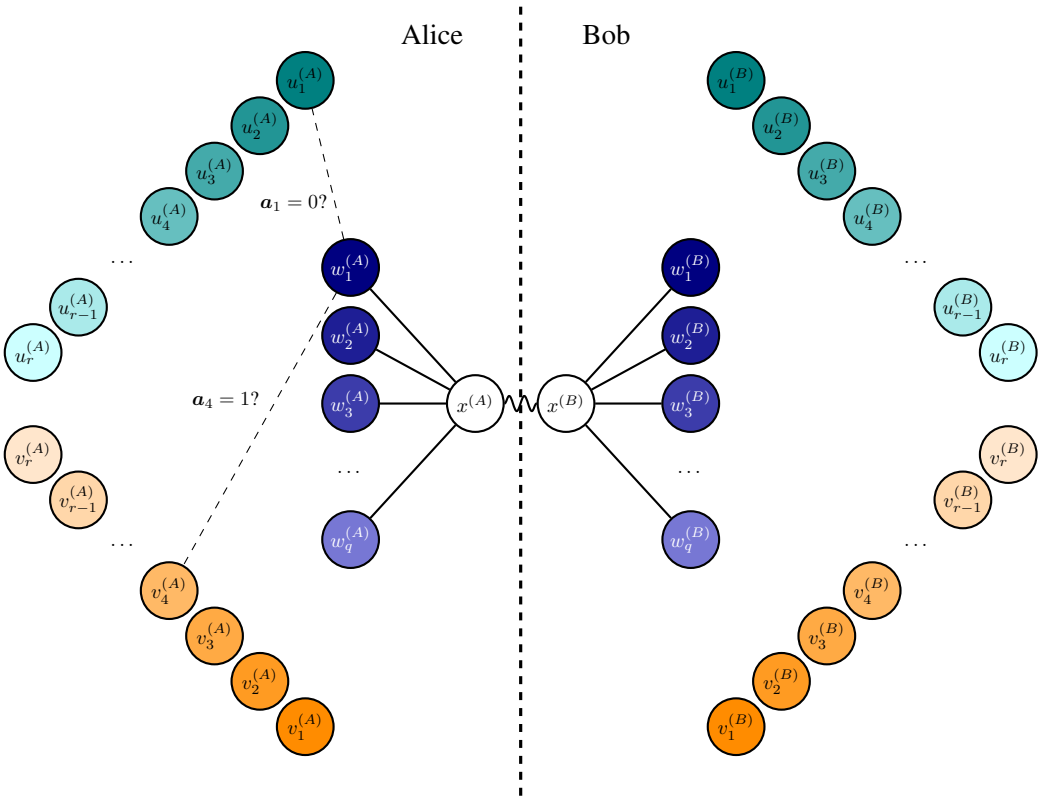

Figure 2: The constructed virtually colored hard instance graph. The edges $\left\{x^{(\sigma)}, u_j^{(\sigma)}\right\}$ and $\left\{x^{(\sigma)}, v_j^{(\sigma)}\right\}$ are omitted, and the connecting path is simplified.

Step Three: Decolorization. We perform the decolorization process separately on Alice's side and Bob's side, so no new edge crosses the Alice-Bob cut. The idea is to add auxiliary nodes and connect certain nodes to them in order to adjust their degrees, thereby encoding the desired colors.

Let $Z^{(\sigma)} \coloneqq \left\{w_i^{(\sigma)} : i \in [q]\right\} \cup \left\{u_j^{(\sigma)}, v_j^{(\sigma)} : j \in [r]\right\}$ be the colored nodes on each side, and $K \coloneqq q + 2r$ be the number of used virtual colors. Let Δ_0 be an upper bound on the degree of every node in $Z^{(\sigma)}$ before decolorization. Indeed, before decolorization, each $w_i^{(\sigma)}$ has degree $r+1$, while each $u_j^{(\sigma)}$ or $v_j^{(\sigma)}$ has degree at most $q+1$. Hence $\Delta_0 = q + r + 1$ is a valid upper bound. We will assign a distinct target degree $T_k \coloneqq \Delta_0 + k$ for each virtual color $k \in [K]$, and also define the padding target degree $T_\perp \coloneqq \Delta_0 + K + 1$. Then all T_k are pairwise distinct, all are larger than Δ_0, and none equals $T_\perp$.

Now fix one side $\sigma \in \{A, B\}$. For each node $z \in Z^{(\sigma)}$, let $d_0(z)$ be its current degree in the virtually colored hard instance, and define $p_z \coloneqq T_{\kappa(z)} - d_0(z)$. Since $T_{\kappa(z)} > \Delta_0 \geq d_0(z)$, we have $p_z > 0$. Let $M^{(\sigma)} \coloneqq \max_{z \in Z^{(\sigma)}} p_z > 0$. Create padding nodes $P_1^{(\sigma)}, \cdots, P_{M^{(\sigma)}}^{(\sigma)}$. Since $M^{(\sigma)} > 0$, there

exists at least one padding node on each side. For every $z \in Z^{(\sigma)}$, connect z to the first p_z padding nodes, namely $P_1^{(\sigma)}, \cdots, P_{p_z}^{(\sigma)}$. Hence the resulting degree of z becomes $d_0(z) + p_z = T_{\kappa(z)}$. Also, every padding node $P_i^{(\sigma)}$ is adjacent to some node in $Z^{(\sigma)}$ by the definition of $M^{(\sigma)}$.

Finally, we further modify the degrees of the padding nodes to ensure that they have the same first-round WL color. Let $s_i^{(\sigma)} := \left|\left\{z \in Z^{(\sigma)} : p_z \geq i\right\}\right|$ be the number of neighbors currently adjacent to $P_i^{(\sigma)}$. Since $s_i^{(\sigma)} \leq \left|Z^{(\sigma)}\right| = K < T_\perp$, we can regularize the degrees of the padding nodes to $T_\perp$. Let $R^{(\sigma)} := \max_{i \in [M^{(\sigma)}]}(T_\perp - s_i^{(\sigma)})$. Create regularizer nodes $Q_1^{(\sigma)}, \cdots, Q_{R^{(\sigma)}}^{(\sigma)}$. Since $R^{(\sigma)} > 0$, there exists at least one regularizer node on each side. For each $i \in [M^{(\sigma)}]$, connect $P_i^{(\sigma)}$ to $Q_1^{(\sigma)}, \cdots, Q_{T_\perp - s_i^{(\sigma)}}^{(\sigma)}$. Then every padding node has degree exactly $s_i^{(\sigma)} + (T_\perp - s_i^{(\sigma)}) = T_\perp$. Moreover, every regularizer node is adjacent to some padding node by the definition of $R^{(\sigma)}$, so this step does not create isolated nodes. Figure 3 below gives an illustration of the decolorization process.

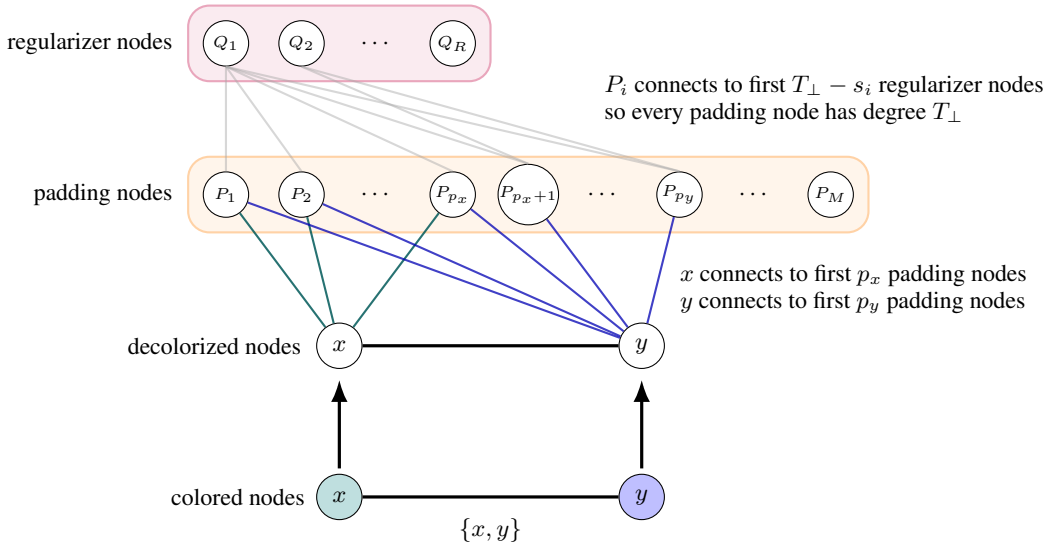

Figure 3: An illustration of the decolorization process.

Step Four: Filler Nodes. Finally, to make the number of nodes $\Theta(n)$, we add $3n$ new leaves adjacent to $x^{(A)}$ and $3n$ new leaves adjacent to $x^{(B)}$. We denote the constructed graph by $G_{(n,m,D;\boldsymbol{a},\boldsymbol{b})}$, and for notational convenience, we sometimes simply write it as G.

Analysis of Graph Parameters. Next, we verify that the constructed graph has the desired number of nodes, number of edges, and diameter.

- ***Number of Nodes.*** For each side $\sigma \in \{A, B\}$, there is one $x^{(\sigma)}$-node, $q = \Theta(\sqrt{m})$ $w^{(\sigma)}$-nodes, $r = \Theta(\sqrt{m})$ $u^{(\sigma)}$-nodes, $r = \Theta(\sqrt{m})$ $v^{(\sigma)}$-nodes, and $\Theta(n)$ filler nodes attached to $x^{(\sigma)}$. In addition, there are at most $M^{(\sigma)} \leq \max_{z \in Z^{(\sigma)}} T_{\kappa(z)} \leq \Delta_0 + K + 1 = \Theta(\sqrt{m})$ padding nodes, and at most $R^{(\sigma)} \leq T_\perp = \Delta_0 + K + 1 = \Theta(\sqrt{m})$ regularizer nodes. Together with the $\Theta(D)$ connecting-path nodes, the whole graph has $\Theta(\sqrt{m} + n + D) = \Theta(n)$ nodes.
- ***Number of Edges.*** For each side $\sigma \in \{A, B\}$, there are $q = \Theta(\sqrt{m})$ edges between the $x^{(\sigma)}$-node and the $w^{(\sigma)}$-nodes, $2r = \Theta(\sqrt{m})$ edges between the $x^{(\sigma)}$-node and the $u^{(\sigma)}$- and $v^{(\sigma)}$-nodes, and $\Theta(n)$ edges between the $x^{(\sigma)}$-node and the filler nodes. For each $w^{(\sigma)}$-node, there are $r = \Theta(\sqrt{m})$ edges connecting it to the $u^{(\sigma)}$-nodes or the $v^{(\sigma)}$-nodes. For every node $z \in Z^{(\sigma)}$, there are $p_z \leq T_{\kappa(z)} \leq \Delta_0 + K + 1 = O(\sqrt{m})$ edges connecting it to padding nodes. Hence, this part contributes $O((q + 2r)\sqrt{m}) = O(m)$ edges. For every padding node, there are at most $T_\perp$ edges connecting it to regularizer nodes. Hence, this part contributes $O(M^{(\sigma)}\sqrt{m}) = O(m)$

edges. Together with the $\Theta(D)$ edges on the path connecting the two sides, the whole graph has $\Theta(m+n+D) = \Theta(m)$ edges.

- ***Diameter.*** Let $L = D - 6$ be the length of the connecting path between $x^{(A)}$ and $x^{(B)}$. Every non-path vertex attached to the A-side is within distance at most 3 from $x^{(A)}$, and every non-path vertex attached to the B-side is within distance at most 3 from $x^{(B)}$. Moreover, every path vertex lies on the $x^{(A)}$-$x^{(B)}$ path of length L. Thus any two vertices attached to opposite side gadgets have distance at most $3 + L + 3 = D$. Two non-path vertices attached to the same side gadget have distance at most $6 \le D$. Distances involving path vertices are also at most D, since every path vertex is within distance at most L from one of $x^{(A)}$ and $x^{(B)}$, and every non-path vertex is within distance at most 3 from the corresponding endpoint. Hence the diameter is at most D.

  On each side there exists at least one regularizer node at distance exactly 3 from the corresponding distinguished node. Since the only connection between the two side gadgets is the connecting path, a regularizer node on the A-side and a regularizer node on the B-side are at distance exactly $3 + L + 3 = D$. Therefore, the diameter of G is exactly D.

Analysis of WL Colors. Let $\chi^{(t)}$ denote the coloring after t rounds of the WL test starting from the uniform coloring.

- ***Round*** 1**.** At round 1, WL colors are determined by the degrees of the nodes.
  - (1) For any $z \in Z \coloneqq Z^{(A)} \cup Z^{(B)}$, we have $\deg_G(z) = T_{\kappa(z)}$. Thus, for any $z, z' \in Z$, $\chi^{(1)}(z) = \chi^{(1)}(z')$ if and only if $\kappa(z) = \kappa(z')$, since the target degrees $\Delta_0 + 1 = T_1 < T_2 < \cdots < T_K = \Delta_0 + K < T_\perp = \Delta_0 + K + 1$ are distinct. In other words, $\chi^{(1)}\left(w_i^{(A)}\right) = \chi^{(1)}\left(w_i^{(B)}\right)$ for each $i \in [q]$, and $\chi^{(1)}\left(u_j^{(A)}\right) = \chi^{(1)}\left(u_j^{(B)}\right)$ and $\chi^{(1)}\left(v_j^{(A)}\right) = \chi^{(1)}\left(v_j^{(B)}\right)$ for each $j \in [r]$; moreover, these round-1 WL colors are pairwise distinct.
  - (2) In addition, all padding nodes have the same round-1 WL color, which is different from that of the nodes in $Z^{(A)} \cup Z^{(B)}$, since they all have the distinct target degree $T_\perp$.
  - (3) Moreover, since $\deg_G\left(x^{(A)}\right) = \deg_G\left(x^{(B)}\right) = q + 2r + 3n + 1$, we have $\chi^{(1)}\left(x^{(A)}\right) = \chi^{(1)}\left(x^{(B)}\right)$. Note that $m \le n^2$, so $\sqrt{m} \le n$, and hence $r = \lceil \sqrt{m} \rceil \le n$. Also, since $r \ge \sqrt{m}$, we have $\frac{m}{r} \le \sqrt{m} \le r$, and therefore $q = \left\lceil \frac{m}{r} \right\rceil \le r$. Consequently, $q + r \le 2n$. Meanwhile, $\deg_G\left(x^{(\sigma)}\right) - T_\perp = q + 2r + 3n + 1 - (2q + 3r + 2) = 3n - (q + r) - 1$. Using $q + r \le 2n$, we obtain $3n - (q + r) - 1 \ge 3n - 2n - 1 = n - 1 > 0$ for all $n \ge 2$. Therefore, the degrees of the nodes $x^{(A)}$ and $x^{(B)}$ are greater than $T_\perp$, so their round-1 color is also different from all the colors mentioned above.
  - (4) Finally, all filler nodes adjacent to the x-nodes have the same round-1 WL color, and all connecting-path nodes have the same round-1 WL color. The filler-node color and the connecting-path-node color are different from the round-1 colors of the nodes in $Z^{(A)} \cup Z^{(B)}$, the padding nodes, and the distinguished nodes $x^{(A)}, x^{(B)}$. Indeed, filler nodes have degree 1, connecting-path nodes have degree 2, nodes in $Z^{(A)} \cup Z^{(B)}$ have degrees in $\{T_1, \cdots, T_K\}$, padding nodes have degree $T_\perp$, and $x^{(A)}, x^{(B)}$ have degree larger than $T_\perp$. The colors of the regularizer nodes are irrelevant for the subsequent argument.
- ***Round*** 2**.** The round-2 WL color of a node is determined by its round-1 WL color together with the multiset of round-1 WL colors of its neighbors.
  - (1) Fix a side $\sigma \in \{A, B\}$ and consider the nodes $u_j^{(\sigma)}$ and $v_j^{(\sigma)}$. For each $j \in [r]$, define $S_{0,j}^{(A)} \coloneqq \{i \in [q] : a_{i,j} = 0\}$, $S_{1,j}^{(A)} \coloneqq \{i \in [q] : a_{i,j} = 1\}$, $S_{0,j}^{(B)} \coloneqq \{i \in [q] : b_{i,j} = 0\}$, and $S_{1,j}^{(B)} \coloneqq \{i \in [q] : b_{i,j} = 1\}$. The neighbors of $u_j^{(\sigma)}$ consist of the node $x^{(\sigma)}$, the nodes $w_i^{(\sigma)}$ with $i \in S_{0,j}^{(\sigma)}$, and $T_{q+j} - \left(1 + \left|S_{0,j}^{(\sigma)}\right|\right)$ padding nodes. Similarly, the neighbors of $v_j^{(\sigma)}$ consist of the node $x^{(\sigma)}$, the nodes $w_i^{(\sigma)}$ with $i \in S_{1,j}^{(\sigma)}$, and $T_{q+r+j} - \left(1 + \left|S_{1,j}^{(\sigma)}\right|\right)$ padding nodes. By the round-1 analysis, we have $\chi^{(1)}\left(u_j^{(A)}\right) = \chi^{(1)}\left(u_j^{(B)}\right)$, $\chi^{(1)}\left(v_j^{(A)}\right) = \chi^{(1)}\left(v_j^{(B)}\right)$.

Moreover, the round-1 colors of the nodes $w_1, \cdots, w_q$ are pairwise distinct and agree across the two sides for the same index, while the padding color is common and is distinct from all colors of the w-nodes. Therefore the multiset of round-1 colors seen by $u_j^{(A)}$ equals the one seen by $u_j^{(B)}$ if and only if $S_{0,j}^{(A)} = S_{0,j}^{(B)}$. Since the entries are binary, this is equivalent to $(a_{1,j}, \cdots, a_{q,j}) = (b_{1,j}, \cdots, b_{q,j})$. Hence $\chi^{(2)}\left(u_j^{(A)}\right) = \chi^{(2)}\left(u_j^{(B)}\right) \iff (a_{1,j}, \cdots, a_{q,j}) = (b_{1,j}, \cdots, b_{q,j})$. The same argument applied to the sets $S_{1,j}^{(\sigma)}$ gives $\chi^{(2)}\left(v_j^{(A)}\right) = \chi^{(2)}\left(v_j^{(B)}\right) \iff (a_{1,j}, \cdots, a_{q,j}) = (b_{1,j}, \cdots, b_{q,j})$.

- (2) Next, consider a node $w_i^{(\sigma)}$. Its neighbors consist of one node $x^{(\sigma)}$, one of $u_j^{(\sigma)}$ or $v_j^{(\sigma)}$ for each $j \in [r]$, and $T_i - r - 1$ padding nodes. By our analysis of the round-1 colors, we have $\chi^{(1)}\left(w_i^{(A)}\right) = \chi^{(1)}\left(w_i^{(B)}\right)$, and $\chi^{(1)}\left(x^{(A)}\right) = \chi^{(1)}\left(x^{(B)}\right)$, and the padding neighbors also contribute the same color with the same multiplicity. Thus, we have $\chi^{(2)}\left(w_i^{(A)}\right) = \chi^{(2)}\left(w_i^{(B)}\right)$ if and only if $(a_{i,1}, \cdots, a_{i,r}) = (b_{i,1}, \cdots, b_{i,r})$.
- (3) Then, consider the node $x^{(\sigma)}$. It has q neighbors $w_i^{(\sigma)}$, r neighbors $u_i^{(\sigma)}$, r neighbors $v_i^{(\sigma)}$, $3n$ filler-node neighbors, and one connecting-path node neighbor. By our analysis of the round-1 colors, we have $\chi^{(1)}\left(x^{(A)}\right) = \chi^{(1)}\left(x^{(B)}\right)$, $\chi^{(1)}\left(w_i^{(A)}\right) = \chi^{(1)}\left(w_i^{(B)}\right)$ for each $i \in [q]$, and $\chi^{(1)}\left(u_j^{(A)}\right) = \chi^{(1)}\left(u_j^{(B)}\right)$ and $\chi^{(1)}\left(v_j^{(A)}\right) = \chi^{(1)}\left(v_j^{(B)}\right)$ for each $j \in [r]$. Moreover, all filler nodes adjacent to the x-nodes have the same round-1 WL color, and all connecting-path nodes have the same round-1 WL color. Therefore, $\chi^{(2)}\left(x^{(A)}\right) = \chi^{(2)}\left(x^{(B)}\right)$.
- (4) Moreover, all filler nodes adjacent to the x-nodes have the same round-2 WL color, since they all have the same round-1 color and the round-1 color of their neighbor $x^{(\sigma)}$ is identical; and connecting-path nodes c_1 and c_{D-7} have the same round-2 WL color for a similar reason. We do not need to consider the round-2 WL colors of the regularizer nodes, the padding nodes, or the other connecting-path nodes.

- ***Round*** 3**.** The round-3 WL color of a node is determined by its round-2 WL color together with the multiset of round-2 WL colors of its neighbors. For node $x^{(\sigma)}$, we have already shown that $\chi^{(2)}\left(x^{(A)}\right) = \chi^{(2)}\left(x^{(B)}\right)$. The filler-node contribution and the connecting-path contribution are also identical on the two sides. It remains to compare the round-2 colors contributed by the nodes in $Z^{(\sigma)}$.

  - If $\boldsymbol{a} = \boldsymbol{b}$, then every row and every column of the matrices $(a_{i,j})$ and $(b_{i,j})$ agree. By the round-2 analysis above, we then have $\chi^{(2)}\left(w_i^{(A)}\right) = \chi^{(2)}\left(w_i^{(B)}\right)$ for all $i \in [q]$, and $\chi^{(2)}\left(u_j^{(A)}\right) = \chi^{(2)}\left(u_j^{(B)}\right)$, $\chi^{(2)}\left(v_j^{(A)}\right) = \chi^{(2)}\left(v_j^{(B)}\right)$ for all $j \in [r]$. Hence the entire multiset of round-2 colors seen by $x^{(A)}$ is the same as the one seen by $x^{(B)}$, and therefore $\chi^{(3)}\left(x^{(A)}\right) = \chi^{(3)}\left(x^{(B)}\right)$.
  - Conversely, suppose $\boldsymbol{a} \neq \boldsymbol{b}$. Since τ is injective and all unused entries are set to 0 on both sides, there exists some $(i, j) \in [q] \times [r]$ such that $a_{i,j} \neq b_{i,j}$. Then the i-th rows differ, so by the round-2 analysis of the w-nodes, $\chi^{(2)}\left(w_i^{(A)}\right) \neq \chi^{(2)}\left(w_i^{(B)}\right)$. Because WL colors always refine previous-round colors, any node whose round-2 color equals $\chi^{(2)}\left(w_i^{(A)}\right)$ must have the same round-1 color as $w_i^{(A)}$. Among the neighbors of $x^{(B)}$, the only node with this round-1 color is $w_i^{(B)}$. However, $\chi^{(2)}\left(w_i^{(B)}\right) \neq \chi^{(2)}\left(w_i^{(A)}\right)$. Therefore the color $\chi^{(2)}\left(w_i^{(A)}\right)$ appears with multiplicity one in the round-2 neighbor-color multiset of $x^{(A)}$, but with multiplicity zero in the corresponding multiset of $x^{(B)}$. Hence the two round-2 neighbor-color multisets are different, and therefore $\chi^{(3)}\left(x^{(A)}\right) \neq \chi^{(3)}\left(x^{(B)}\right)$.

Thus, from the above analysis, we obtain $\chi^{(3)}\left(x^{(A)}\right) = \chi^{(3)}\left(x^{(B)}\right) \iff \boldsymbol{a} = \boldsymbol{b}$.

Lower Bound. Now suppose that we have an oblivious MPGNN with depth d, width w, and precision p that can simulate one step of the WL test with zero error. We now derive the lower bound from two perspectives:

- ***Communication Complexity.*** We can use this MPGNN to obtain a communication protocol for EQ_m. By composing three copies of the MPGNN, we obtain an MPGNN of depth $3d$, width w, and precision p that simulates three steps of the WL test. We then apply it to the hard-instance graph constructed above and compare the colors of $x^{(A)}$ and $x^{(B)}$.

  Alice and Bob simulate the MPGNN on their respective sides. The Alice-Bob cut contains exactly one graph edge. Therefore, in each message-passing layer, only $O(wp)$ bits need to cross the cut. Over $3d$ layers, the total communication is $O(dwp)$ bits. Finally, the simulator outputs labels in the color set $[C]$, so the colors of $x^{(A)}$ and $x^{(B)}$ can be compared using $O(\log C)$ bits of communication. Therefore, we obtain an $O(dwp + \log C)$-bit communication protocol for EQ_m.

  Since the deterministic and randomized zero-error communication complexities of EQ_m satisfy $D(\mathrm{EQ}_m) = R_0(\mathrm{EQ}_m) = \Omega(m)$ bits, we get $O\left(dwp + \log C\right) = \Omega(m)$. When $\log C = o(m)$, this implies $dwp = \Omega(m)$, and hence $d = \Omega\left(\frac{m}{wp}\right)$.

- ***Locality.*** Let $\ell \coloneqq \mathrm{dist}_G\left(x^{(A)}, x^{(B)}\right) = \Theta(D)$. We claim that the composed MPGNN of depth $3d$ must satisfy $3d \geq \frac{\ell}{2} = \Theta(D)$.

  Suppose for contradiction that $3d < \frac{\ell}{2}$. Then, by locality of message passing, the output color of $x^{(A)}$ after the composed MPGNN is independent of Bob's input $\boldsymbol{b}$, and the output color of $x^{(B)}$ is independent of Alice's input $\boldsymbol{a}$. Therefore, there exist functions $F, H : \{0,1\}^m \to [C]$ such that the output color of $x^{(A)}$ is $F(\boldsymbol{a})$ and the output color of $x^{(B)}$ is $H(\boldsymbol{b})$[9].

  Since the composed MPGNN simulates three steps of the WL test with zero error, and the hard instance satisfies $\chi_G^{(3)}\left(x^{(A)}\right) = \chi_G^{(3)}\left(x^{(B)}\right) \iff \boldsymbol{a} = \boldsymbol{b}$, we must have $F(\boldsymbol{a}) = H(\boldsymbol{b}) \iff \boldsymbol{a} = \boldsymbol{b}$, for all $\boldsymbol{a}, \boldsymbol{b} \in \{0,1\}^m$. In particular, $F(\boldsymbol{a}) = H(\boldsymbol{a})$ for every $\boldsymbol{a} \in \{0,1\}^m$. Moreover, F must be injective: if $F(\boldsymbol{a}) = F(\boldsymbol{a}')$, then $F(\boldsymbol{a}') = F(\boldsymbol{a}) = H(\boldsymbol{a})$, which implies $\boldsymbol{a}' = \boldsymbol{a}$ by the equivalence above. Thus $|\mathrm{Im}(F)| = 2^m$, and hence $C \geq 2^m$. This contradicts the assumption $\log C = o(m)$ for sufficiently large m. Therefore $3d \geq \frac{\ell}{2} = \Theta(D)$, and hence $d = \Omega(D)$.

Combining the two bounds gives $d = \Omega\left(\max\left\{D, \frac{m}{wp}\right\}\right) = \Omega\left(D + \frac{m}{wp}\right)$. □

B CONGEST, Our Rooted Port-Aware MPGNN, and Proof of Theorem 3.2

B.1 CONGEST Model and Our Rooted Port-Aware MPGNN Model

Before proving Theorem 3.2, we first introduce the CONGEST model, the rooted, port-aware MPGNN model used in our construction, and several CONGEST algorithms that can be transferred to this model.

The CONGEST model was introduced to study the number of communication rounds required to solve distributed computing problems in bandwidth-limited networks [24, 25, 31, 12]. Due to the similarity between the update process in the CONGEST model (Equation (3)) and the message-passing mechanism of MPGNNs (Equation (1)), prior work [26] has proposed using the CONGEST model to characterize MPGNNs and derived lower bounds for tasks such as cycle detection. The definition of the CONGEST model is:

Definition B.1 (CONGEST Model)**.** The CONGEST model is defined on an n-node graph $G = (V = [n], E)$. Initially, each node knows the total number of nodes n, its own initial features, and its unique identifier, which lies in $[\mathrm{poly}(n)]$ or $[n]$. In each communication round, a node receives an $O(w)$-word message[10] from each neighbor, performs local computation based on its current

[9] For zero-error randomized simulation, fix any seed in the support as justified in Section 2.4 and apply the deterministic argument below.

[10] In the standard setting, message sizes are restricted to $O(\log n)$ bits, that is, $O(1)$ words when $p = \Theta(\log n)$. In [26], this constraint is relaxed to $O(w)$ words, that is, $O(w \log n)$ bits when $p = \Theta(\log n)$, to match the width of MPGNNs.

knowledge, and then sends an $O(w)$-word message to each neighbor, where the messages sent to different neighbors may be different. This process can be described by the following update rule:

$$s_u^{(\ell+1)} = \mathrm{UPD}^{(\ell)}\left(s_u^{(\ell)}, \left\{\!\!\left\{\left(v, \mathrm{MSG}^{(\ell)}\left(s_v^{(\ell)}, u, v\right)\right) : v \in N(u)\right\}\!\!\right\}\right), \tag{3}$$

where $s_u^{(\ell)}$ denotes the internal state (which may not necessarily be a vector) of node u.

In [26], the authors view the CONGEST model as a computational model for MPGNNs, so that MPGNN constructions can be obtained by designing CONGEST algorithms.

However, the use of unique identifiers in CONGEST is a rather strong assumption. While it greatly enhances the computational power of the model, it also breaks anonymity. Moreover, when the update and message functions depend directly on node IDs, it may destroy the permutation invariance or permutation equivariance property commonly required by MPGNNs, and may also make it difficult to generalize to graphs with previously unseen IDs. For this reason, we do not adopt CONGEST directly as the model for studying upper bounds. Instead, in order to obtain concrete constructions while keeping nodes from being uniquely identified and preserving the permutation invariance or permutation equivariance of the model as much as possible, we work with the following weaker rooted, port-aware model.

Definition B.2 (Rooted, Port-Aware MPGNN Model)**.** Our rooted, port-aware MPGNN model is defined on a graph $G = (V, E)$ with one designated root node r. Initially, each node u receives its input feature vector $\boldsymbol{x}_u$ and a root bit $\boldsymbol{b}_u \in \{0, 1\}$, where $\boldsymbol{b}_u = 1$ if and only if $u = r$. Nodes do not have other unique identifiers. Each node u has an arbitrary but fixed local ordering of its incident ports, denoted by $P(u) = [\deg(u)]$. These port numbers are purely local and are not globally consistent across endpoints of an edge.

At layer ℓ, node u maintains an arbitrary finite node state $\boldsymbol{s}_u^{(\ell)}$ and arbitrary finite port states $\left(\boldsymbol{s}_{u,p}^{(\ell)}\right)_{p\in P(u)}$. Through each port p, it receives a message $\boldsymbol{m}_{u,p}^{(\ell)}$ of at most wp bits. It then applies a shared local transition rule

$$\left(\boldsymbol{s}_u^{(\ell+1)}, \left(\boldsymbol{s}_{u,p}^{(\ell+1)}\right)_{p\in P(u)}, \left(\boldsymbol{m}_{u,p}^{(\ell+1)}\right)_{p\in P(u)}\right) = \mathrm{UPD}^{(\ell)}\left(\boldsymbol{s}_u^{(\ell)}, \boldsymbol{b}_u, \left(p, \boldsymbol{s}_{u,p}^{(\ell)}, \boldsymbol{m}_{u,p}^{(\ell)}\right)_{p\in P(u)}\right), \tag{4}$$

and sends $\boldsymbol{m}_{u,p}^{(\ell+1)}$ through port p. The same transition rule $\mathrm{UPD}^{(\ell)}$ is used at all nodes. Thus the model has root information and local port awareness, but no unique node identifiers.

B.2 Three CONGEST Algorithms

We then present several classical CONGEST algorithms and point out that they can still be implemented in the weaker rooted, port-aware MPGNN model.

First, a spanning tree rooted at a node u can be constructed using the FLOOD algorithm in the CONGEST model.

Lemma B.3 (cf. Chapter 3 of [31], FLOOD Algorithm)**.** *There exists a CONGEST algorithm in which a designated node $u \in V$ can construct a spanning tree T rooted at u with depth $\mathrm{depth}(T) = \max_v \{d_G(u, v)\}$ in $\max_v \{d_G(u, v)\} = O(D)$ rounds, where D is the diameter of the graph.*

The idea behind the FLOOD algorithm is straightforward. Initially, the source node u sends a special token to all its neighbors. Each node, upon receiving the token for the first time, stores it and forwards it to its neighbors. If a node receives the token again, it discards it and does nothing. Note that the round complexity of FLOOD is $O(D)$ and it does not depend on the CONGEST parameters w (the number of words per edge per round) or p (the number of bits per word), since the token can be encoded using a constant number of bits.

Additionally, the following lemmas describe the ability of the CONGEST model to broadcast and collect messages to and from a designated node.

Lemma B.4 (cf. Chapter 4 of [31], DOWNCAST Algorithm)**.** *There exists a CONGEST algorithm where given M messages (of $\Theta(\log n)$ bits) stored at a designated node $u \in V$, and a spanning tree T rooted at u, the messages can be broadcast to other nodes in $O(\mathrm{depth}(T) + M)$ rounds.*

Lemma B.5 (cf. Chapter 4 of [31], UPCAST Algorithm)**.** *There exists a CONGEST algorithm where given M messages (of $\Theta(\log n)$ bits) stored at different nodes and a spanning tree T rooted at u, the messages can be collected at node u in $O(\mathrm{depth}(T) + M)$ rounds.*

The key idea behind these two algorithms is to use a queue as a buffer to pipeline messages along the spanning tree. Take DOWNCAST as an example. Each node maintains a queue as a buffer. In each round, the root sends one message to each of its children. Any other node enqueues the message received from its parent on the spanning tree, then dequeues the message at the head of the queue and forwards it to all of its children. Under this strategy, the root releases one message per round, so it terminates after M rounds. Moreover, a message that leaves the root in round r reaches its destination by round at most $r + \mathrm{depth}(T)$. The UPCAST algorithm proceeds in the opposite direction and can be analyzed similarly.

It is important to note that the conclusions for the DOWNCAST and UPCAST algorithms above are derived under the standard CONGEST model, where each edge can transmit only $O(1)$ messages of size $\Theta(\log n)$ bits per communication round. If the total encoded information to be broadcast or collected is B bits, and we relax the per-edge restriction to allow the transmission of w messages of p bits each per round, then the round complexities of both algorithms reduce to $O\left(\mathrm{depth}(T) + \frac{B}{wp}\right)$ by streaming the encoded information as detailed below.

B.3 Implementations in the Rooted, Port-Aware MPGNN Model

In fact, none of the above three algorithms uses the unique identifiers assumed in the CONGEST model. They only require that each node knows whether it is the root, and that each edge can decide whether to continue transmitting information according to whether it is a tree edge. Therefore, it is intuitive that all three CONGEST algorithms can also be implemented in our rooted, port-aware MPGNN model. The following lemma formalizes this claim.

Lemma B.6 (Rooted, Port-Aware MPGNN Implementations of FLOOD, DOWNCAST, and UPCAST)**.** *Let $G = (V, E)$ be an n-node graph with a designated root r and diameter D. Let $\mathrm{depth}(T)$ denote the depth of a rooted tree T, namely the maximum tree distance from the root to any node. Suppose that in each MPGNN layer, each direction on an edge can transmit at most w words, and each word consists of p bits, so each edge can transmit at most wp bits per layer in each direction. Then the following CONGEST algorithms admit rooted, port-aware MPGNN implementations.*

1. **FLOOD.** *There exists a rooted, port-aware MPGNN of network depth $d_{\text{FLOOD}} = O(D)$ that constructs a spanning tree T rooted at r. The constructed tree satisfies $\mathrm{depth}(T) = \max_{v \in V} d_G(r, v) = O(D)$.*
2. **DOWNCAST.** *Suppose a spanning tree T rooted at r is already encoded in the port states, and suppose the root stores a payload of total length B bits. There exists a rooted, port-aware MPGNN of network depth $d_{\text{DOWNCAST}} = O\left(\mathrm{depth}(T) + \left\lceil \frac{B}{wp} \right\rceil\right)$ that broadcasts the payload from r to all nodes.*
3. **UPCAST.** *Suppose a spanning tree T rooted at r is already encoded in the port states, and suppose the nodes jointly store payloads of total encoded length B bits, including the stream delimiters described below. There exists a rooted, port-aware MPGNN of network depth $d_{\text{UPCAST}} = O\left(\mathrm{depth}(T) + \left\lceil \frac{B}{wp} \right\rceil\right)$ that collects all payloads at the root r.*

Proof. We describe how to simulate the three algorithms in the rooted, port-aware MPGNN model.

For every node u and every port $p \in P(u)$, the port state stores a role $\rho_{u,p} \in \{\mathsf{unknown}, \mathsf{parent}, \mathsf{child}, \mathsf{offtree}\}$[11]. By the definition of the rooted, port-aware model, node u receives the port-indexed collection of incoming messages and port states. Hence it can distinguish its incident ports, compare their local indices, and update the corresponding port states jointly. We write $\iota_{u,p} \in [\deg(u)]$ for the local index of port p. This index is purely local and is not a unique node identifier.

[11] In practice, this role information is encoded using two bits; we write it symbolically here only for ease of exposition.

We use a fixed alphabet containing the data symbols $0, 1$ and the control symbols needed below. Each symbol has a constant-length binary encoding, so a logical communication round transmitting up to wp symbols per edge in each direction can be implemented by a constant number of MPGNN layers. Unused slots contain an idle symbol, which receivers discard. All rounds in the following implementations are logical communication rounds.

- For FLOOD, initially only the root r is active. The root sends a FLOOD token through all ports. When a non-root node u receives a FLOOD token for the first time, it chooses one receiving port as its parent port. If several tokens arrive in the same round, u chooses the port with the smallest local index $\iota_{u,p}$. It marks this port as parent, sends a ACK token back through this parent port, and forwards the FLOOD token through all other ports. Whenever a node receives a ACK token through a port, it marks that port as child. After the flooding and acknowledgement phase has run for $O(D)$ rounds, any port that has not been marked as parent or child is marked as offtree.

  A node at graph distance h from the root first receives the FLOOD token after h rounds, and therefore chooses as parent a neighbor at distance $h-1$ from the root. Hence the chosen parent pointers form a spanning tree rooted at r, and the tree depth is $\mathrm{depth}(T) = \max_{v \in V} d_G(r, v)$. Since every node is reached within at most D rounds, and the ACK confirmations only add a constant number of rounds, the required network depth is $d_{\text{FLOOD}} = O(D)$.

- For DOWNCAST, assume that a rooted spanning tree T has already been encoded in the port states. The root splits its B-bit payload into $L = \left\lceil \frac{B}{wp} \right\rceil$ packets, each containing at most wp data symbols. Each node maintains a FIFO queue. Initially, the root queue contains all L packets, while other queues are empty. In each round, the root sends the head packet of its queue to all child ports. Every non-root node accepts data packets only from its parent port, enqueues the received packet, and if its queue is nonempty, forwards the head packet to all child ports. This behavior is implemented by the port-indexed local transition rule: the node ignores data packets received on non-parent ports and produces data packets only on child ports[12].

  The resulting procedure is the standard pipelined broadcast on a rooted tree. The root releases one packet per round, and a packet needs at most $\mathrm{depth}(T)$ additional rounds to reach every node. Therefore all L packets reach all nodes after $O\left(\mathrm{depth}(T) + L\right) = O\left(\mathrm{depth}(T) + \left\lceil \frac{B}{wp} \right\rceil\right)$ rounds. Hence the network depth is $d_{\text{DOWNCAST}} = O\left(\mathrm{depth}(T) + \left\lceil \frac{B}{wp} \right\rceil\right)$.

- For UPCAST, again assume that the rooted spanning tree T has already been encoded in the port states. The idea is to assemble each subtree's payloads into an ordered stream and pipeline it toward the root. Delimiters let the root recover the individual payloads.

  Each node u forms a subtree stream encoding the payloads of all nodes in the subtree of T rooted at u. It maintains a separate buffer for each child port. The stream begins with the payload at u and a separator, followed by the subtree streams of its children in local port order and a final end marker. Separators and end markers use binary encodings distinct from those of data and idle symbols. The subtree streams received from children are forwarded verbatim, so this encoding has only constant-factor expansion of the payload bits and $O(n)$ additional framing bits. Each node forwards the longest available prefix of its stream, up to wp symbols per round. The subtree stream from the current child is forwarded as its symbols arrive, until its matching outer end marker is reached, while streams from later child ports are buffered. When this subtree stream ends, the node fills any remaining slots in the same round from the next part of its stream. The root uses the same rule to store the combined stream and then parses it to recover all payloads.

  Let L_u be the number of symbols in the subtree stream of u, and let h_u be its subtree height. We show by induction on t that after t rounds, each node u has forwarded (or stored at the root) at least $\min\{L_u, wp \max\{0, t - h_u\}\}$ symbols. The claim is immediate for $t \le h_u$. For $t > h_u$, the induction hypothesis ensures that each child v has supplied at least $\min\{L_v, wp(t-h_u)\}$ symbols by round $t-1$. Thus either the subtree streams of all children are complete, or the first incomplete stream already supplies $wp(t-h_u)$ symbols, so a prefix of length at least $\min\{L_u, wp(t-h_u)\}$ is available. Together with the previous round's bound and the capacity to forward wp symbols,

[12] For example, one can add a gating vector to each port and use the Hadamard product to control message passing.

this proves the claim. Since $L_r = \Theta(B)$, the number of logical rounds, and hence the MPGNN depth, is $O\left(\mathrm{depth}(T) + \lceil B/(wp) \rceil\right)$.

Therefore the three CONGEST algorithms have rooted, port-aware MPGNN implementations with the claimed network depths. □

B.4 Proof of Theorem 3.2

With these tools in hand, we are now ready to prove the upper bound in the theorem.

Theorem 3.2. *Given any family of graphs with at most n nodes, at most m edges, and diameter at most D, where each graph is supplied with an arbitrary designated root marker, and a color set $[C]$ large enough to represent all input and output colors under consideration (e.g., $C \geq n$), there exists an oblivious rooted, port-aware MPGNN with depth $d = O\left(D + \frac{m \log C}{wp}\right)$, sufficiently large node and port state dimensions, and sufficiently expressive message, node-update, and port-update functions, that simulates one step of the WL test in both the deterministic and zero-error randomized settings.*

Proof. By the equivalence established in Section 2.4, it suffices to give a deterministic construction; this also establishes the zero-error randomized guarantee and the simultaneous guarantee with $p_f < 1$. We design a deterministic rooted, port-aware MPGNN with depth $d = O\left(D + \frac{m \log C}{wp}\right)$ and width w words per edge per round and word precision p bits. The main idea is to encode each WL-type canonically and sort the distinct encoded types at the root, and use the resulting rank as the new color, which automatically satisfies the hash function condition. We present the framework of our construction and analyze the depth for each step:

1. Each node u sends a message $\boldsymbol{x}_u$ to its neighbors and receives messages from them to form its WL-type as $M_u = (\boldsymbol{x}_u, \{\!\{\boldsymbol{x}_v : v \in N(u)\}\!\})$. This process can be done by a rooted, port-aware MPGNN with depth $O\left(\max\left\{1, \frac{\log C}{wp}\right\}\right) = O\left(1 + \frac{\log C}{wp}\right)$.
2. The root node r initiates the FLOOD algorithm to construct a BFS spanning tree rooted at r. This process can be done by an MPGNN with depth $O(D)$.
3. The UPCAST algorithm is used to collect all sets M_u at the root node r along the spanning tree. This process can be done by an MPGNN with depth $O\left(D + \frac{m \log C}{wp}\right)$, since there are $\sum_{u \in V} O(\deg(u)) = O(m)$ colors to gather, each of size $O(\log C)$ bits[13].
4. The root node merges all M_u and deduplicates them to form the set $K = \{M_u : u \in V\}$.
5. The root node sorts K according to an arbitrarily fixed total order on all WL-types, for example lexicographic order, and forms a mapping $\mathrm{rk} : K \to [|K|]$ that assigns to each element of K its rank in $[|K|]$, represented by the set of pairs $R = \{(k, \mathrm{rk}(k)) : k \in K\}$.
6. The DOWNCAST algorithm is used to send the mapping R back to every node along the spanning tree. This process can be done by an MPGNN with depth $O\left(D + \frac{m \log C}{wp}\right)$, since the size of R is $\sum_{u \in V} O(\deg(u) \log C) + O(n \log n) = O(m \log C)$ bits.
7. Finally, each node determines its new color from its own WL-type and the returned mapping rk.

Thus, one step of the WL test can be computed by an MPGNN with depth

$$d = O\left(1 + \frac{\log C}{wp} + D + D + \frac{m \log C}{wp} + D + \frac{m \log C}{wp}\right) = O\left(D + \frac{m \log C}{wp}\right).$$

□

[13] We use a self-delimiting encoding of each WL-type, including its degree or separators. This adds $O(n \log n)$ bits, which is absorbed by $O(m \log C)$ for connected graphs and $C \geq n$.

C Introduction to the Bounded-Error Randomized Construction by Aamand et al.

In this section, we introduce the MPGNN construction of Aamand et al. [1] for bounded-error randomized simulation of one step of the WL test.

Their Definition 1.1 states a pairwise success guarantee for each fixed graph G and input coloring χ. Specifically, for any fixed pair of nodes $u, v \in V$, they require that if $\mathrm{type}_{G,\chi}(u) = \mathrm{type}_{G,\chi}(v)$, then $\chi'(u) = \chi'(v)$ with probability 1, and if $\mathrm{type}_{G,\chi}(u) \neq \mathrm{type}_{G,\chi}(v)$, then $\chi'(u) \neq \chi'(v)$ with probability at least $1 - q$, where χ' is the computed new coloring. In contrast, in our formulation, simulating one step of the WL test with success probability at least $1 - p_f$ means producing the correct new colors of all nodes with probability at least $1 - p_f$. By a union bound over all unordered node pairs of this fixed instance, their pairwise guarantee implies our all-pairs correctness guarantee when setting $q = O\left(\frac{p_f}{n^2}\right)$. This gives a per-instance guarantee as defined in Section 2.4.

In their construction, each node carries a color label $\boldsymbol{x}_u \in \{0,1\}^{c \log n}$ for a constant c, which is equivalent to a color in $[n^{c_0}]$ for some constant c_0 and can be represented by $\Theta(\log n)$ bits. In our single-step setting, we resample the shared randomness for each fixed instance and encode its input colors into these vectors in the message function. Each node exchanges its encoded color with its neighbors and performs local computation to obtain its new color, so the total message exchanged on each edge is $wp = \Theta(\log n)$ bits. The construction requires access to public randomness: all nodes can read a shared random vector $\boldsymbol{b} \in [F]^{c \log n}$, where each coordinate is uniform in $[F]$. In addition, the construction samples $t = c \log n$ ϵ-biased vectors $\boldsymbol{a}_1, \cdots, \boldsymbol{a}_t \in \{0,1\}^F$. By Corollary 3.3 of [1], each $\boldsymbol{a}_i$ can be generated using $O(\log F + \log(1/\epsilon))$ truly random bits. To bound the number of random bits, note that their analysis bounds the failure probability by $\frac{1}{F}$, so to ensure $\frac{1}{F} \leq q$ it suffices to take $F = \Theta\left(\frac{n^2}{p_f}\right)$, which implies $\log F = \Theta(\log n)$ when $p_f = \Omega\left(\frac{1}{\mathrm{poly}(n)}\right)$. Finally, the parameters c and ϵ are chosen appropriately to satisfy $\left(\frac{1+\epsilon}{2}\right)^{c \log n} \leq q = O\left(\frac{p_f}{n^2}\right)$, which yields the result.

D Hash Function Background and Proofs of Theorem 3.4 and Theorem 3.5

In this section, we present the proofs of Theorem 3.4 and Theorem 3.5.

D.1 Introduction to Hash Functions

We begin by introducing hash functions as a key proof tool. Hash functions are widely used in computer science, for example in cryptography, pseudorandomness, and data structures. Given two finite sets U and $[M]$ (usually $|U| \gg M$), a hash family $\mathcal{H}$ is a collection of functions from U to $[M]$ (i.e., $\mathcal{H} \subseteq [M]^U$). Carter and Wegman [5] defined universal hash functions as follows: given a hash family $\mathcal{H}$ from U to $[M]$, we say that $\mathcal{H}$ is universal if for any two distinct inputs $x \neq y$ in U, the collision probability satisfies $\Pr[h(x) = h(y)] \leq \frac{1}{M}$, where h is sampled uniformly at random from $\mathcal{H}$. In some applications, it suffices to relax the bound $\frac{1}{M}$ and consider ϵ-almost universal hashing: for $\frac{1}{M} \leq \epsilon < 1$, we say that $\mathcal{H}$ is ϵ-almost universal if for any two distinct inputs $x \neq y$ in U, $\Pr[h(x) = h(y)] \leq \epsilon$.

Suppose we want to design a universal hash family from $U = \{0,1\}^u$ to $[M] = \{0,1\}^t$. A simple choice is the family of linear hash functions $\mathcal{H} = \left\{h_{\boldsymbol{A}}(\boldsymbol{x}) \coloneqq \boldsymbol{A}\boldsymbol{x} : \boldsymbol{A} \in \{0,1\}^{t \times u}\right\}$, where the multiplication between a binary matrix and a binary vector is over $\mathrm{GF}(2)$ (i.e., modulo 2). Sampling a hash function from $\mathcal{H}$ requires tu random bits to choose a uniformly random matrix $\boldsymbol{A}$. If we allow $O\left(\frac{1}{M}\right)$-almost universal hashing, then a hash function can be sampled using only $\Theta(\log u + t)$ random bits. This is formalized in Lemma D.1:

Lemma D.1. *For any constant $c > 1$, there exists a $\frac{c}{2^t}$-almost universal hash family $\mathcal{H}$ from $\{0,1\}^u$ to $\{0,1\}^t$, and sampling $h \in \mathcal{H}$ can be done using $\Theta(\log u + t)$ random bits.*

Proof. We define a family $\mathcal{H} = \{h_{x,y} : x, y \in \mathrm{GF}(2^b)\}$ where each $h_{x,y}$ is a linear map of the form $h_{x,y}(\boldsymbol{z}) = \boldsymbol{A}(x,y)\boldsymbol{z}$ for some binary matrix $\boldsymbol{A}(x,y) \in \{0,1\}^{t\times u}$ generated from the seed (x,y). We will choose b later such that the family is $\frac{c}{2^t}$-almost universal.

We first fix b and a binary bijective representation map $\mathrm{bin} : \mathrm{GF}(2^b) \to \{0,1\}^b$ such that $\mathrm{bin}(0) = (0,0,\cdots,0)$ and $\mathrm{bin}(u+v) = \mathrm{bin}(u) \oplus \mathrm{bin}(v)$ for all $u, v \in \mathrm{GF}(2^b)$. For example, viewing an element $\alpha \in \mathrm{GF}(2^b)$ as a binary polynomial $a_1 + a_2 x + \cdots + a_b x^{b-1}$ of degree at most $b-1$ with $a_i \in \{0,1\}$, one can set $\mathrm{bin}(\alpha) = (a_1, a_2, \cdots, a_b) \in \{0,1\}^b$.

Let $N = tu \geq 2$. We generate a matrix $\boldsymbol{A} = \boldsymbol{A}(x,y)$ using limited randomness as follows:

1. Sample two elements x, y independently and uniformly at random from $\mathrm{GF}(2^b)$.
2. For $i = 1, \cdots, N$, let $r_i \coloneqq \langle \mathrm{bin}(x^{i-1}), \mathrm{bin}(y)\rangle$, where $\langle\cdot,\cdot\rangle$ denotes the inner product over $\mathrm{GF}(2)$. Concretely, for $\boldsymbol{a} = (\boldsymbol{a}_1, \cdots, \boldsymbol{a}_b)$ and $\boldsymbol{b} = (\boldsymbol{b}_1, \cdots, \boldsymbol{b}_b)$ in $\{0,1\}^b$, we define $\langle \boldsymbol{a}, \boldsymbol{b}\rangle \coloneqq \sum_{j=1}^{b} \boldsymbol{a}_j \boldsymbol{b}_j \bmod 2$.
3. Fill the bits $r_1, \ldots, r_N$ into the matrix $\boldsymbol{A}$ row by row, i.e., for each $j \in [t]$ and $k \in [u]$ we set $\boldsymbol{A}_{j,k} \coloneqq r_{(j-1)u+k}$.

Let $\boldsymbol{r} = (r_1, \ldots, r_N) \in \{0,1\}^N$ denote this sampled string. We now analyze the collision probability. Fix any two distinct inputs $\boldsymbol{x}, \boldsymbol{x}' \in \{0,1\}^u$. A collision under $h_{x,y}$ occurs when $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{A}\boldsymbol{x}' \iff \boldsymbol{A}(\boldsymbol{x} \oplus \boldsymbol{x}') = \boldsymbol{0} \in \{0,1\}^t$. Since $\boldsymbol{x} \neq \boldsymbol{x}'$, letting $\boldsymbol{d} \coloneqq \boldsymbol{x} \oplus \boldsymbol{x}' \neq \boldsymbol{0}$, the collision probability reduces to the probability that the random matrix $\boldsymbol{A}$ maps a fixed nonzero vector $\boldsymbol{d} \in \{0,1\}^u$ to the zero vector.

Define a random variable $\boldsymbol{z} \coloneqq \boldsymbol{A}\boldsymbol{d} \in \{0,1\}^t$. For each $\boldsymbol{s} \in \{0,1\}^t$, define $\chi_{\boldsymbol{s}}(\boldsymbol{z}) = (-1)^{\langle \boldsymbol{s}, \boldsymbol{z}\rangle}$, where $\langle \boldsymbol{s}, \boldsymbol{z}\rangle$ denotes the mod-2 inner product. It is straightforward to verify the identity $\mathbb{I}[\boldsymbol{z} = \boldsymbol{0}] = 2^{-t} \sum_{\boldsymbol{s}\in\{0,1\}^t} \chi_{\boldsymbol{s}}(\boldsymbol{z})$. Then

$$
\begin{aligned}
\Pr[h_{x,y}(\boldsymbol{x}) = h_{x,y}(\boldsymbol{x}')] = \Pr[\boldsymbol{z} = \boldsymbol{0}] &= \mathbb{E}[\mathbb{I}[\boldsymbol{z} = \boldsymbol{0}]] \\
&= 2^{-t} \sum_{\boldsymbol{s}\in\{0,1\}^t} \mathbb{E}[\chi_{\boldsymbol{s}}(\boldsymbol{z})] \\
&= 2^{-t} \left(\mathbb{E}[\chi_{\boldsymbol{0}}(\boldsymbol{z})] + \sum_{\boldsymbol{0}\neq\boldsymbol{s}\in\{0,1\}^t} \mathbb{E}[\chi_{\boldsymbol{s}}(\boldsymbol{z})] \right) \\
&\leq 2^{-t} \left(1 + (2^t - 1) \max_{\boldsymbol{0}\neq\boldsymbol{s}\in\{0,1\}^t} \left| \mathbb{E}[\chi_{\boldsymbol{s}}(\boldsymbol{z})] \right| \right) \\
&\leq 2^{-t} + \max_{\boldsymbol{0}\neq\boldsymbol{s}\in\{0,1\}^t} \left| \mathbb{E}[\chi_{\boldsymbol{s}}(\boldsymbol{z})] \right|.
\end{aligned} \tag{5}
$$

It remains to bound $\max_{\boldsymbol{0}\neq\boldsymbol{s}\in\{0,1\}^t} \left|\mathbb{E}[\chi_{\boldsymbol{s}}(\boldsymbol{z})]\right|$. We first rewrite $\langle \boldsymbol{s}, \boldsymbol{z}\rangle$ as a linear test on $\boldsymbol{r}$. For any fixed $\boldsymbol{s} \in \{0,1\}^t$ and $\boldsymbol{d} \in \{0,1\}^u$,

$$
\langle \boldsymbol{s}, \boldsymbol{z}\rangle = \sum_{j=1}^{t} \boldsymbol{s}_j \boldsymbol{z}_j \bmod 2 = \sum_{j=1}^{t} \boldsymbol{s}_j \left(\sum_{k=1}^{u} \boldsymbol{A}_{j,k} \boldsymbol{d}_k \right) \bmod 2 = \sum_{j=1}^{t} \sum_{k=1}^{u} (\boldsymbol{s}_j \boldsymbol{d}_k) \boldsymbol{A}_{j,k} \bmod 2.
$$

Define $\boldsymbol{w} = \boldsymbol{w}(\boldsymbol{s}, \boldsymbol{d}) \in \{0,1\}^N$ by $\boldsymbol{w}_{(j-1)u+k} \coloneqq \boldsymbol{s}_j \boldsymbol{d}_k$ for all $j \in [t]$ and $k \in [u]$. Since $r_{(j-1)u+k} = \boldsymbol{A}_{j,k}$ by construction, we have $\langle \boldsymbol{s}, \boldsymbol{z}\rangle = \langle \boldsymbol{w}, \boldsymbol{r}\rangle$, and therefore $\mathbb{E}[\chi_{\boldsymbol{s}}(\boldsymbol{z})] = \mathbb{E}\left[(-1)^{\langle \boldsymbol{s}, \boldsymbol{z}\rangle}\right] = \mathbb{E}\left[(-1)^{\langle \boldsymbol{w}, \boldsymbol{r}\rangle}\right]$. Moreover, because $\boldsymbol{d} \neq \boldsymbol{0}$, we have $\boldsymbol{w} \neq \boldsymbol{0}$ whenever $\boldsymbol{s} \neq \boldsymbol{0}$. Hence it suffices to upper bound $\left|\mathbb{E}\left[(-1)^{\langle \boldsymbol{w}, \boldsymbol{r}\rangle}\right]\right|$ for all $\boldsymbol{0} \neq \boldsymbol{w} \in \{0,1\}^N$.

Proposition 3 of [2] implies $\max_{\mathbf{0}\neq \boldsymbol{w}\in\{0,1\}^N} \left|\mathbb{E}\left[(-1)^{\langle \boldsymbol{w},\boldsymbol{r}\rangle}\right]\right| \leq \frac{N-1}{2^b}$. For completeness, we prove this bound below.

Fix some $\boldsymbol{w} \in \{0,1\}^N \setminus \{\mathbf{0}\}$, and consider the linear test $\langle \boldsymbol{w}, \boldsymbol{r}\rangle = \sum_{i=1}^N \boldsymbol{w}_i \boldsymbol{r}_i \bmod 2$. In the sampling procedure above, $\boldsymbol{r} \in \{0,1\}^N$ is a function of the field elements $x, y \in \mathrm{GF}(2^b)$. Thus, we can rewrite

$$\begin{aligned}\langle \boldsymbol{w}, \boldsymbol{r}(x,y)\rangle &= \sum_{i=1}^N \boldsymbol{w}_i \left\langle \mathrm{bin}(x^{i-1}), \mathrm{bin}(y)\right\rangle \\ &= \left\langle \mathrm{bin}\left(\sum_{i=1}^N \boldsymbol{w}_i x^{i-1}\right), \mathrm{bin}(y)\right\rangle,\end{aligned}$$

where the second equality uses linearity of the inner product in its first argument and the property of bin. Let $p_{\boldsymbol{w}}(X) \coloneqq \sum_{i=1}^N \boldsymbol{w}_i X^{i-1}$ be a polynomial over $\mathrm{GF}(2)$, and then $\langle \boldsymbol{w}, \boldsymbol{r}(x,y)\rangle = \langle \mathrm{bin}(p_{\boldsymbol{w}}(x)), \mathrm{bin}(y)\rangle$. We consider two cases:

- If $p_{\boldsymbol{w}}(x) = 0$, then we have $\langle \boldsymbol{w}, \boldsymbol{r}(x,y)\rangle = \langle \mathrm{bin}(0), \mathrm{bin}(y)\rangle = 0$. Hence the probability $\Pr\left[\langle \boldsymbol{w}, \boldsymbol{r}(x,y)\rangle = 1 \;\middle|\; p_{\boldsymbol{w}}(x) = 0\right] = 0$.

- If $p_{\boldsymbol{w}}(x) \neq 0$, then for any fixed nonzero vector $\mathrm{bin}(p_{\boldsymbol{w}}(x)) \in \{0,1\}^b$, exactly half of the $\mathrm{bin}(y) \in \{0,1\}^b$ make the inner product $\langle \mathrm{bin}(p_{\boldsymbol{w}}(x)), \mathrm{bin}(y)\rangle$ equal to 0. Since y is uniform over $\mathrm{GF}(2^b)$ and bin is a bijection, $\mathrm{bin}(y)$ is uniform over $\{0,1\}^b$. Therefore, $\Pr\left[\langle \boldsymbol{w}, \boldsymbol{r}(x,y)\rangle = 0 \;\middle|\; p_{\boldsymbol{w}}(x) \neq 0\right] = \Pr\left[\langle \boldsymbol{w}, \boldsymbol{r}(x,y)\rangle = 1 \;\middle|\; p_{\boldsymbol{w}}(x) \neq 0\right] = \frac{1}{2}$.

Since $\deg(p_{\boldsymbol{w}}) \leq N-1$ and $p_{\boldsymbol{w}}$ is nonzero, it has at most $N-1$ roots in $\mathrm{GF}(2^b)$, and thus $\Pr[p_{\boldsymbol{w}}(x) = 0] \leq \frac{N-1}{2^b}$. By the law of total probability,

$$\begin{aligned}\Pr[\langle \boldsymbol{w}, \boldsymbol{r}(x,y)\rangle = 1] &= \Pr\left[\langle \boldsymbol{w}, \boldsymbol{r}(x,y)\rangle = 1 \;\middle|\; p_{\boldsymbol{w}}(x) = 0\right]\Pr[p_{\boldsymbol{w}}(x) = 0] \\ &\quad + \Pr\left[\langle \boldsymbol{w}, \boldsymbol{r}(x,y)\rangle = 1 \;\middle|\; p_{\boldsymbol{w}}(x) \neq 0\right]\Pr[p_{\boldsymbol{w}}(x) \neq 0] \\ &= 0 + \frac{1}{2}\Big(1 - \Pr[p_{\boldsymbol{w}}(x) = 0]\Big) = \frac{1}{2} - \frac{1}{2}\Pr[p_{\boldsymbol{w}}(x) = 0],\end{aligned}$$

and similarly,

$$\Pr[\langle \boldsymbol{w}, \boldsymbol{r}(x,y)\rangle = 0] = \Pr[p_{\boldsymbol{w}}(x) = 0] + \frac{1}{2}\Big(1 - \Pr[p_{\boldsymbol{w}}(x) = 0]\Big) = \frac{1}{2} + \frac{1}{2}\Pr[p_{\boldsymbol{w}}(x) = 0].$$

Therefore,

$$\begin{aligned}\left|\mathbb{E}\left[(-1)^{\langle \boldsymbol{w}, \boldsymbol{r}(x,y)\rangle}\right]\right| &= \Big|1 \cdot \Pr[\langle \boldsymbol{w}, \boldsymbol{r}\rangle = 0] - 1 \cdot \Pr[\langle \boldsymbol{w}, \boldsymbol{r}\rangle = 1]\Big| \\ &= \Pr[p_{\boldsymbol{w}}(x) = 0] \leq \frac{N-1}{2^b}.\end{aligned}$$

This proves $\max_{\mathbf{0}\neq \boldsymbol{w}\in\{0,1\}^N} \left|\mathbb{E}\left[(-1)^{\langle \boldsymbol{w},\boldsymbol{r}\rangle}\right]\right| \leq \frac{N-1}{2^b}$, and hence $\max_{\mathbf{0}\neq \boldsymbol{s}\in\{0,1\}^t} \left|\mathbb{E}\left[\chi_{\boldsymbol{s}}(\boldsymbol{z})\right]\right| \leq \frac{N-1}{2^b}$. Substituting into Equation (5), we obtain $\Pr[h_{x,y}(\boldsymbol{x}) = h_{x,y}(\boldsymbol{x}')] \leq 2^{-t} + \frac{N-1}{2^b}$. To make this probability at most $\frac{c}{2^t}$, it suffices to ensure $\frac{N-1}{2^b} \leq \frac{c-1}{2^t}$, namely $2^b \geq \frac{N-1}{c-1}2^t$, equivalently $b \geq t + \log_2(N-1) - \log_2(c-1)$. Thus, setting $b \coloneqq \lceil t + \log_2(tu-1) - \log_2(c-1)\rceil$ satisfies the desired collision-probability bound, and $\mathcal{H}$ is $\frac{c}{2^t}$-almost universal.

Finally, sampling $h_{x,y}$ requires sampling (x,y) uniformly from $\mathrm{GF}(2^b)^2$, which takes $R = 2b = \Theta(t + \log(tu)) = \Theta(\log u + t)$ random bits. □

D.2 Proof of Theorem 3.4 and Theorem 3.5

Now we can prove Theorem 3.4 and Theorem 3.5 using Lemma D.1 as our main tool.

Theorem 3.4. *Given any family of graphs with at most n nodes, a failure probability $p_f \in (0,1)$, and a color set $[C]$ with $C \geq \left\lceil \frac{3n^2}{2p_f} \right\rceil$, there exists an MPGNN with depth $d = 1$, $wp \geq \lceil \log_2 C \rceil$, sufficiently large node state dimension, and sufficiently expressive message and node-update functions, that simulates one step of the WL test with probability at least $1 - p_f$, using $R = \Theta\left(\log \frac{n \log C}{p_f}\right)$ random bits.*

Proof. We construct an anonymous unordered-neighborhood MPGNN with depth $d = 1$, width w words per edge per round, and word precision p bits. We outline the construction and analyze the depth of each step:

1. Each node u sends its current color $\boldsymbol{x}_u$ to its neighbors and receives their colors to form its WL-type $(\boldsymbol{x}_u, \{\!\{\boldsymbol{x}_v : v \in N(u)\}\!\})$. This step can be done by a model with depth 1 since $wp \geq \lceil \log_2 C \rceil$.
2. Each node encodes its WL-type into a binary vector in a canonical way. For example, it can first encode the WL-type as a vector in $\{0, 1, \cdots, C\}^n$ by setting the first entry to $\boldsymbol{x}_u$, the next $\deg(u) \leq n-1$ entries to the sorted multiset $\{\!\{\boldsymbol{x}_v : v \in N(u)\}\!\}$, and padding the remaining entries with a special symbol 0 not used as a color. It then converts each entry into $\lceil \log_2(C+1) \rceil$ bits, yielding a binary vector $\boldsymbol{c}_u \in \{0,1\}^{n\lceil \log_2(C+1) \rceil}$.
3. Let $F \coloneqq \left\lceil \frac{3n^2}{2p_f} \right\rceil \leq C$. We apply Lemma D.1 to construct a $\frac{2}{2^{\lceil \log_2 F \rceil}}$-almost universal hash family $\mathcal{H}_1$ from $\{0,1\}^{n\lceil \log_2(C+1) \rceil}$ to $\{0,1\}^{\lceil \log_2 F \rceil}$, and use $\Theta\left(\log\left(n \log C\right) + \log F\right) = \Theta\left(\log \frac{n \log C}{p_f}\right)$ random bits to sample a hash function $h_1 \in \mathcal{H}_1$.
4. We then construct another universal hash family $\mathcal{H}_2$ from $\{0,1\}^{\lceil \log_2 F \rceil}$ to $[F]$ via a standard construction (e.g., $\{h_{a,b}(x) = 1 + ((ax+b) \bmod P) \bmod F : a \in [P-1], b \in \{0, 1, \cdots, P-1\}\}$ for some prime $P \geq 2^{\lceil \log_2 F \rceil}$), and use $\Theta(\log F) = \Theta\left(\log \frac{n}{p_f}\right)$ random bits to sample a function $h_2 \in \mathcal{H}_2$.
5. Finally, we set the new color of node u to be $\boldsymbol{y}_u \coloneqq h_2(h_1(\boldsymbol{c}_u))$.

Next, we bound the collision probability. Fix any $u, v \in V$ such that $\boldsymbol{c}_u \neq \boldsymbol{c}_v$. We have

$$\begin{aligned}
\Pr\left[\boldsymbol{y}_u = \boldsymbol{y}_v\right] &= \Pr\left[h_2(h_1(\boldsymbol{c}_u)) = h_2(h_1(\boldsymbol{c}_v)) \;\middle|\; h_1(\boldsymbol{c}_u) = h_1(\boldsymbol{c}_v)\right] \Pr\left[h_1(\boldsymbol{c}_u) = h_1(\boldsymbol{c}_v)\right] \\
&\quad + \Pr\left[h_2(h_1(\boldsymbol{c}_u)) = h_2(h_1(\boldsymbol{c}_v)) \;\middle|\; h_1(\boldsymbol{c}_u) \neq h_1(\boldsymbol{c}_v)\right] \Pr\left[h_1(\boldsymbol{c}_u) \neq h_1(\boldsymbol{c}_v)\right] \\
&\leq \Pr\left[h_1(\boldsymbol{c}_u) = h_1(\boldsymbol{c}_v)\right] + \frac{1}{F}\left(1 - \Pr\left[h_1(\boldsymbol{c}_u) = h_1(\boldsymbol{c}_v)\right]\right) \\
&\leq \frac{1}{F} + \left(1 - \frac{1}{F}\right) \Pr\left[h_1(\boldsymbol{c}_u) = h_1(\boldsymbol{c}_v)\right] \\
&\leq \frac{1}{F} + \left(1 - \frac{1}{F}\right) \frac{2}{2^{\lceil \log_2 F \rceil}} \leq \frac{3}{F}.
\end{aligned}$$

By a union bound, the overall failure probability is

$$\Pr\left[\exists u \neq v \in V \text{ such that } \boldsymbol{c}_u \neq \boldsymbol{c}_v \text{ and } \boldsymbol{y}_u = \boldsymbol{y}_v\right] \leq \sum_{\substack{\{u,v\} \subseteq V \\ \boldsymbol{c}_u \neq \boldsymbol{c}_v}} \Pr\left[\boldsymbol{y}_u = \boldsymbol{y}_v\right] \leq \frac{3n^2}{2F} \leq p_f.$$

□

We now prove Theorem 3.5.

Theorem 3.5. *Given any family of graphs with at most n nodes and diameter at most D, where each graph is supplied with an arbitrary designated root marker, a failure probability $p_f \in (0,1)$, a color set $[C]$ with $C \geq n$, there exists a rooted, port-aware MPGNN with depth $d = O\left(D + \frac{\log C + n\log(n/p_f)}{wp}\right)$, sufficiently large node and port state dimensions, and sufficiently expressive message, node-update, and port-update functions, that simulates one step of the WL test with probability at least $1 - p_f$, using $R = \Theta\left(\log \frac{n \log C}{p_f}\right)$ random bits.*

Proof. We prove this theorem by constructing a rooted, port-aware MPGNN with bandwidth w words per edge per round and word precision p bits. The construction proceeds as follows:

1. First, we can use a rooted, port-aware MPGNN of depth $O\left(1 + \frac{\log C}{wp}\right)$ with $\Theta\left(\log \frac{n \log C}{p_f}\right)$ random bits, where each node first forms its WL-type and then maps the canonical encoding of this WL-type to a temporary color $\boldsymbol{y}_u \in \left[\left\lceil \frac{3n^2}{2p_f} \right\rceil\right]$.
2. The root node r initiates FLOOD to construct a BFS spanning tree rooted at r. This can be done by a model with depth $O(D)$.
3. We use UPCAST to collect all node colors $\boldsymbol{y}_u$ at the root node r along the spanning tree. This can be done by a model with depth $O\left(D + \frac{n\log(n/p_f)}{wp}\right)$ rounds, since there are $O(n)$ colors to gather, each of size $O\left(\log \left\lceil \frac{3n^2}{2p_f} \right\rceil\right) = O\left(\log \frac{n}{p_f}\right)$ bits, and each edge can transmit wp bits per round.
4. The root node merges and deduplicates the colors $\boldsymbol{y}_u$ to form the set $K = \{\boldsymbol{y}_u : u \in V\}$, then sorts K and assigns new colors according to rank. Concretely, it produces a set $R = \{(\boldsymbol{y}_u, \boldsymbol{z}_u) : u \in V\}$, where $\boldsymbol{z}_u \coloneqq |\{\boldsymbol{y}_v : v \in V, \boldsymbol{y}_v < \boldsymbol{y}_u\}| + 1$ is the rank of $\boldsymbol{y}_u$ among the distinct values in $\{\boldsymbol{y}_v : v \in V\}$.
5. The DOWNCAST algorithm is used to send the mapping R back to every node along the spanning tree. This process can be carried out by an MPGNN with depth $O\left(D + \frac{n\log(n/p_f)}{wp}\right)$, since there are $O(n)$ messages to transmit, each of size $O(\log(n/p_f) + \log n) = O(\log(n/p_f))$ bits.
6. Finally, each node determines its new color from its own color $\boldsymbol{y}_u$ and the returned mapping R.

The algorithm succeeds whenever the hashing subroutine is collision-free on the WL-types that occur. Therefore the failure probability is at most p_f. □

E Proof of Theorem 3.6

In this section, we prove Theorem 3.6.

Theorem 3.6. *For every sufficiently large integer n, there exists a family $\mathcal{G}_n$ of connected graphs, each with $\Theta(n)$ nodes. Then, for any failure probability $p_f \in (0, 1/9)$ and any color set $[C]$ with $C \geq \max_{G \in \mathcal{G}_n} |V(G)|$ and $\log C = o(n)$, any randomized oblivious MPGNN with $dwp = O(\log C)$ that simulates one step of the WL test on $\mathcal{G}_n$ over $[C]$ with probability at least $1 - p_f$ must use $R = \Omega\left(\log \frac{n}{p_f \log C}\right)$ random bits.*

Proof. We reduce EQ_n to simulating one step of the WL test. We use the hard-instance construction from the proof of Theorem 3.1 with parameters $m = n$ and $D = 8$. Therefore, we have a graph family $\mathcal{G}_n = \{G_{\boldsymbol{a},\boldsymbol{b}} : \boldsymbol{a}, \boldsymbol{b} \in \{0,1\}^n\}$ consisting of connected uncolored graphs with $\Theta(n)$ nodes and $\Theta(n)$ edges. Moreover, the construction has an Alice-Bob partition whose cut contains exactly one graph edge, and it contains two distinguished nodes $x^{(A)}$ and $x^{(B)}$ such that, if $\chi_G^{(t)}$ denotes the coloring after t rounds of the WL test starting from the uniform coloring, then $\chi_G^{(3)}(x^{(A)}) = \chi_G^{(3)}(x^{(B)})$ if and only if $\boldsymbol{a} = \boldsymbol{b}$.

Now suppose that there exists a randomized oblivious MPGNN $\mathcal{M}$ with depth d, width w, precision p, and R random bits that simulates one step of the WL test with failure probability at most p_f, and suppose further that $dwp = O(\log C)$. Alice and Bob use $\mathcal{M}$ three times in sequence to simulate three steps of the WL test on $G_{\boldsymbol{a},\boldsymbol{b}}$. The composed simulator has depth $3d$ and uses $3R$ random bits. Using three independent copies of $\mathcal{M}$ and conditioning on the correctness of the previous simulated rounds, all three rounds are correct with probability at least $(1-p_f)^3 > 1 - 3p_f > 2/3$. During this simulation, the Alice-Bob cut contains exactly one graph edge. Hence each message-passing layer requires only $O(wp)$ bits of communication across the cut. Over $3d$ layers, the total communication used to simulate the MPGNN is $O(dwp)$. After the simulation, Alice and Bob compare the output colors of $x^{(A)}$ and $x^{(B)}$. Since the output set is $[C]$, this costs another $O(\log C)$ bits. Therefore, we obtain a randomized public-coin protocol for EQ_n with communication cost $k = O(dwp + \log C) = O(\log C)$, randomness $3R$, and failure probability at most $3p_f < 1/3$.

By the limited-randomness lower bound for Equality [18], any randomized public-coin protocol for EQ_n using ρ random bits and failure probability ε must have communication cost $k \geq \Omega\left(\min\left\{n, \frac{n}{\varepsilon 2^\rho}\right\}\right)$. Applying this bound with $\rho = 3R$ and $\varepsilon = 3p_f$ gives $k \geq \Omega\left(\min\left\{n, \frac{n}{p_f 2^{3R}}\right\}\right)$. Since $k = O(\log C)$ and $\log C = o(n)$, the first alternative $k = \Omega(n)$ is impossible for sufficiently large n. Hence we must have $O(\log C) \geq \Omega\left(\frac{n}{p_f 2^{3R}}\right)$. Rearranging gives $2^{3R} \geq \Omega\left(\frac{n}{p_f \log C}\right)$. Taking logarithms yields $R = \Omega\left(\log \frac{n}{p_f \log C}\right)$, as claimed. □

F ExistsEqual Problem and Proof of Theorem 3.7

In this section, we prove Theorem 3.7. Before that, we introduce a communication complexity problem called $\mathrm{ExistsEqual}_{U,k}$ (or $\exists\mathrm{EQ}_{U,k}$ for short).

Definition F.1 ($\mathrm{ExistsEqual}_{U,k}$, cf. [17])**.** Given a set U and a positive integer k, Alice and Bob hold vectors $\boldsymbol{x} \in U^k$ and $\boldsymbol{y} \in U^k$, respectively. They must determine whether there exists an index $i \in [k]$ such that $\boldsymbol{x}_i = \boldsymbol{y}_i$.

The communication complexity of $\mathrm{ExistsEqual}_{U,k}$ has been studied extensively. For error probability $p_f = \Theta(1)$ and for $p_f = \Omega\left(\frac{1}{\mathrm{poly}(k)}\right)$, Saglam and Tardos [32] and Brody et al. [4] showed that, when $|U| = \Omega(k)$, the communication complexity of $\mathrm{ExistsEqual}_{U,k}$ is $\Omega\left(k \log^{(r)} k\right)$, where $\log^{(i)}(x)$ denotes the iterated logarithm, defined by $\log^{(1)}(x) \coloneqq \log x$ and $\log^{(i)}(x) \coloneqq \log\left(\log^{(i-1)}(x)\right)$ for $i \geq 2$, and r denotes the number of communication rounds.

We are now ready to present the proof of Theorem 3.7.

Theorem 3.7. *For every sufficiently large integer n and every integer $D \in [8, n]$, there exists a family $\mathcal{G}_{n,D}$ of connected graphs, each with $\Theta(n)$ nodes and diameter $\Theta(D)$. Then, for any failure probability $p_f \in (0, 1/6)$ satisfying $p_f = \Omega\left(\frac{1}{\mathrm{poly}(n)}\right)$, and any color set $[C]$ with $C \geq \max_{G \in \mathcal{G}_{n,D}} |V(G)|$ and $C = \Theta(n)$, any randomized oblivious MPGNN that simulates one step of the WL test on $\mathcal{G}_{n,D}$ over $[C]$ with probability at least $1 - p_f$ must satisfy $d = \Omega\left(D + \min\left\{\log^* n, \frac{n}{wp}\right\}\right)$.*

Proof. We reduce the $\exists\mathrm{EQ}$ problem to simulating one step of the WL test. The proof follows the same high-level idea as the proof of Theorem 3.1. We first construct a virtually colored hard instance, and then encode the virtual colors by node degrees using shared padding and regularizer nodes.

Construction. We reduce from $\exists\mathrm{EQ}_{U,n}$, where U is the set of all multisets of size 2 over $[n]$. Thus $|U| = \binom{n+1}{2} = \Theta(n^2)$.

Step One: Fixed Basic Graph. For each side $\sigma \in \{A, B\}$, create nodes $x^{(\sigma)}$, nodes $w_1^{(\sigma)}, \cdots, w_n^{(\sigma)}$, and nodes $u_{i,1}^{(\sigma)}, u_{i,2}^{(\sigma)}$ for every $i \in [n]$. Add edges $\left\{x^{(\sigma)}, w_i^{(\sigma)}\right\}$, $\left\{w_i^{(\sigma)}, u_{i,1}^{(\sigma)}\right\}$, and $\left\{w_i^{(\sigma)}, u_{i,2}^{(\sigma)}\right\}$ for every $i \in [n]$. Then, connect the two sides by a path of length $\Theta(D)$, for example by adding nodes

$c_1, \cdots, c_D$ and edges $\left\{x^{(A)}, c_1\right\}, \{c_1, c_2\}, \cdots, \left\{c_D, x^{(B)}\right\}$. Figure 4 below gives an illustration of the fixed basic graph, where the connecting path is simplified.

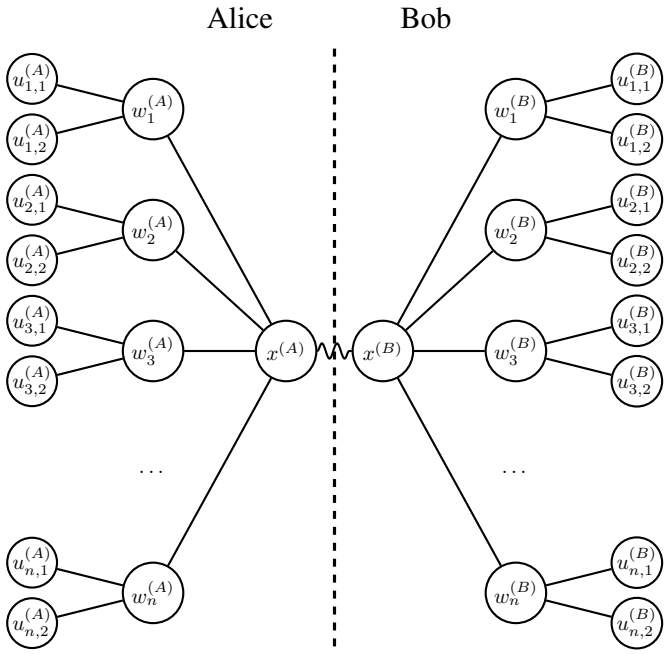

Figure 4: The constructed basic graph. The connecting path is simplified.

Step Two: Virtually Colored Hard Instance. Given an input instance $(\boldsymbol{a}, \boldsymbol{b}) \in U^n \times U^n$, Alice and Bob assign virtual colors as follows. For every $i \in [n]$, set $\kappa\left(w_i^{(A)}\right) = \kappa\left(w_i^{(B)}\right) = i$. If $\boldsymbol{a}_i = \{\!\{\alpha, \beta\}\!\}$ with $\alpha \leq \beta$, then Alice sets $\kappa\left(u_{i,1}^{(A)}\right) = \alpha$ and $\kappa\left(u_{i,2}^{(A)}\right) = \beta$. Bob does the analogous assignment using $\boldsymbol{b}_i$. The nodes $x^{(A)}$, $x^{(B)}$ and connecting nodes c_i do not need virtual colors.

Step Three: Decolorization. We now decolorize this virtually colored graph separately on Alice's side and Bob's side, so no new edge crosses the Alice-Bob cut. Let $Z^{(\sigma)} \coloneqq \left\{w_i^{(\sigma)} : i \in [n]\right\} \cup \left\{u_{i,1}^{(\sigma)}, u_{i,2}^{(\sigma)} : i \in [n]\right\}$ be the set of virtually colored nodes on side σ, and let $K \coloneqq n$ be the number of virtual colors used. Let $\Delta_0 = 3$ be an upper bound on the degrees of nodes in $Z^{(\sigma)}$ before decolorization. For every virtual color $k \in [K]$, define a distinct target degree $T_k = \Delta_0 + k = 3 + k$, and define the padding target degree $T_\perp = 3n + 1$ larger than $|Z^{(\sigma)}|$. Then all T_k are pairwise distinct, all are larger than Δ_0, and none equals $T_\perp$.

For each $z \in Z^{(\sigma)}$, let $d_0(z)$ be its current degree before decolorization and set $p_z = T_{\kappa(z)} - d_0(z)$. Since $T_{\kappa(z)} > \Delta_0 \geq d_0(z)$, we have $p_z > 0$. Create shared padding nodes $P_1^{(\sigma)}, \ldots, P_{M^{(\sigma)}}^{(\sigma)}$, where $M^{(\sigma)} = \max_{z \in Z^{(\sigma)}} p_z$, and connect each $z \in Z^{(\sigma)}$ to $P_1^{(\sigma)}, \ldots, P_{p_z}^{(\sigma)}$. Thus every virtually colored node z obtains degree exactly $T_{\kappa(z)}$.

We further modify the degrees of the padding nodes to ensure that they have the same first-round WL color. Let $s_i^{(\sigma)} \coloneqq \left|\left\{z \in Z^{(\sigma)} : p_z \geq i\right\}\right|$ be the number of neighbors currently adjacent to $P_i^{(\sigma)}$. Since $s_i^{(\sigma)} \leq \left|Z^{(\sigma)}\right| = 3n < T_\perp$, we can regularize the degrees of the padding nodes to $T_\perp$. Let $R^{(\sigma)} \coloneqq \max_{i \in [M^{(\sigma)}]}(T_\perp - s_i^{(\sigma)})$. Create regularizer nodes $Q_1^{(\sigma)}, \cdots, Q_{R^{(\sigma)}}^{(\sigma)}$. Since $R^{(\sigma)} > 0$, there exists at least one regularizer node on each side. For each $i \in [M^{(\sigma)}]$, connect $P_i^{(\sigma)}$ to $Q_1^{(\sigma)}, \cdots, Q_{T_\perp - s_i^{(\sigma)}}^{(\sigma)}$. Then every padding node has degree exactly $s_i^{(\sigma)} + (T_\perp - s_i^{(\sigma)}) = T_\perp$.

Moreover, every regularizer node is adjacent to some padding node by the definition of $R^{(\sigma)}$, so this step does not create isolated nodes.

Finally, add equal numbers of filler leaves adjacent to $x^{(A)}$ and $x^{(B)}$, if necessary, so that $\deg(x^{(A)}) = \deg(x^{(B)}) = 3n + 2$ and this common degree is larger than $T_\perp$. We denote the resulting graph by $G_{\boldsymbol{a},\boldsymbol{b}}$, and let $\mathcal{G}_{n,D}$ be the family of all such graphs.

Analysis of Graph Parameters. Similar to the proof of Theorem 3.1, the decolorized graph is connected and has $\Theta(n)$ nodes and $\Theta(D)$ diameter.

Analysis of WL Colors. Let $\chi_G^{(t)}$ denote the coloring after t steps of the WL test starting from the uniform coloring on the decolorized graph $G = G_{\boldsymbol{a},\boldsymbol{b}}$.

- ***Round*** 1**.** In round 1, WL colors are determined by degrees. By construction, every virtually colored node $z \in Z^{(A)} \cup Z^{(B)}$ has degree $T_{\kappa(z)}$, and the target degrees $T_1, \cdots, T_K$ are pairwise distinct. Hence, for any $z, z' \in Z^{(A)} \cup Z^{(B)}$, we have $\chi_G^{(1)}(z) = \chi_G^{(1)}(z')$ if and only if $\kappa(z) = \kappa(z')$. Thus the first step of the WL test recovers the intended virtual colors on all w-nodes and u-nodes. Also, $\chi_G^{(1)}(x^{(A)}) = \chi_G^{(1)}(x^{(B)})$, because the two distinguished nodes have the same degree. The padding nodes all have the same round-1 color, corresponding to degree $T_\perp$.
- ***Round*** 2**.** Fix $i, j \in [n]$. If $i \neq j$, then we have $\chi_G^{(2)}\left(w_i^{(A)}\right) \neq \chi_G^{(2)}\left(w_j^{(B)}\right)$, because $\chi_G^{(1)}\left(w_i^{(A)}\right) \neq \chi_G^{(1)}\left(w_j^{(B)}\right)$. Thus only nodes with the same index can possibly match. Now fix $i \in [n]$. The round-2 color of $w_i^{(A)}$ is determined by its round-1 color together with the multiset of round-1 colors of its neighbors. These neighbors consist of $x^{(A)}$, the two nodes $u_{i,1}^{(A)}, u_{i,2}^{(A)}$, and a number of padding nodes determined only by the virtual color of $w_i^{(A)}$. The same holds for $w_i^{(B)}$. The contribution from $x^{(A)}$ and $x^{(B)}$ is identical, and the padding contribution is also identical on the two sides. Therefore, $\chi_G^{(2)}\left(w_i^{(A)}\right) = \chi_G^{(2)}\left(w_i^{(B)}\right)$ if and only if the two multisets of virtual colors assigned to $u_{i,1}^{(A)}, u_{i,2}^{(A)}$ and $u_{i,1}^{(B)}, u_{i,2}^{(B)}$ are equal, which holds if and only if $\boldsymbol{a}_i = \boldsymbol{b}_i$. Consequently, the two sets of round-2 colors $\left\{\chi_G^{(2)}\left(w_i^{(A)}\right) : i \in [n]\right\}$ and $\left\{\chi_G^{(2)}\left(w_i^{(B)}\right) : i \in [n]\right\}$ intersect if and only if there exists $i \in [n]$ such that $\boldsymbol{a}_i = \boldsymbol{b}_i$. In other words, two steps of the WL test on the decolorized graph solve the predicate $\exists\mathrm{EQ}_{U,n}$.

Lower Bound. Suppose that there exists a randomized oblivious MPGNN $\mathcal{M}$ with depth d, width w, and precision p that simulates one step of the WL test on $\mathcal{G}_{n,D}$ with failure probability at most p_f. By composing two independent copies of $\mathcal{M}$, Alice and Bob can simulate two steps of the WL test on $G_{\boldsymbol{a},\boldsymbol{b}}$. The composed simulator has depth $2d$, and it succeeds with probability at least $(1 - p_f)^2 > 1 - 2p_f > 2/3$.

- ***The Communication Term.*** First consider the communication lower bound. Alice and Bob simulate the composed MPGNN across the Alice-Bob cut. The cut contains exactly one graph edge, and hence each message-passing layer requires only $O(wp)$ bits of communication across the cut. Over $2d$ layers, the simulation uses $O(dwp)$ bits. After the simulation, Alice sends Bob a C-bit mask indicating which colors appear in the set $\left\{\chi_G^{(2)}\left(w_i^{(A)}\right) : i \in [n]\right\}$. Since $C = \Theta(n)$, this costs $O(n)$ bits. Bob accepts if and only if one of the colors in $\left\{\chi_G^{(2)}\left(w_i^{(B)}\right) : i \in [n]\right\}$ appears in Alice's mask. By the WL analysis above, this protocol solves $\exists\mathrm{EQ}_{U,n}$ with failure probability at most $2p_f < 1/3$, communication cost $O(dwp + n)$, and round complexity $O(d)$. The round-sensitive lower bound for $\exists\mathrm{EQ}_{U,n}$ with $|U| = \Theta(n^2)$ [32, 4] gives communication $\Omega(n \log^{(r)} n)$ for r-round public-coin protocols. If $r = o(\log^* n)$, then $\log^{(r)} n \to \infty$, so the required communication is $\omega(n)$. Thus an $O(n)$-communication protocol must have $r = \Omega(\log^* n)$. If $dwp = o(n)$, the constructed protocol has $O(n)$ communication and $O(d)$ rounds, so $d = \Omega(\log^* n)$. Hence either $d = \Omega(\log^* n)$ or $dwp = \Omega(n)$. Equivalently, $d = \Omega\left(\min\left\{\log^* n, \frac{n}{wp}\right\}\right)$.

- ***The Diameter Term.*** It remains to prove the diameter term. The two side gadgets are connected by a path of length $\Theta(D)$. Suppose, for contradiction, that $d = o(D)$. Then the depth-$2d$ composed MPGNN cannot transmit information from Alice's side to Bob's side through the connecting path. Thus the colors output on Alice's w-nodes are functions only of Alice's input and the public randomness, and the colors output on Bob's w-nodes are functions only of Bob's input and the public randomness. Alice can therefore locally compute her color set, send the same $O(n)$-bit color mask to Bob, and Bob can decide whether the two color sets intersect. This gives a one-way ($r = 1$) randomized protocol for $\exists\mathrm{EQ}_{U,n}$ with $O(n)$ communication and bounded error. However, the one-way randomized communication complexity of $\exists\mathrm{EQ}_{U,n}$ with $|U| = \Theta(n^2)$ is $\Omega(n \log n)$ for bounded error [32, 4]. This contradicts the $O(n)$-bit one-way protocol above. Therefore $d = \Omega(D)$.

Combining $d = \Omega(D)$ and $d = \Omega\left(\min\left\{\log^* n, \frac{n}{wp}\right\}\right)$ gives the desired lower bound $d = \Omega\left(D + \min\left\{\log^* n, \frac{n}{wp}\right\}\right)$. □

G Proof of Theorem 4.1

In this section we prove Theorem 4.1.

Theorem 4.1. *Given any valid input instance (G, χ), where G is a graph with n nodes, any color set $[C]$ with $C \geq n$, and any width w and precision p satisfying $wp \geq \lceil \log_2 n \rceil$, there exists an instance-dependent parameter choice $\theta_{G,\chi}$ for a deterministic depth-one MPGNN with width w, precision p, sufficiently large hidden-state dimension, and instance-dependent embedding, message, update, and decoding functions, such that the resulting MPGNN simulates one step of the WL test on the fixed instance (G, χ), that is, for all $u, v \in V(G)$, $\mathcal{N}_{\theta_{G,\chi}}(\chi; G)(u) = \mathcal{N}_{\theta_{G,\chi}}(\chi; G)(v) \iff \mathrm{type}_{G,\chi}(u) = \mathrm{type}_{G,\chi}(v)$.*

Proof. Let $K = \{\chi(u) : u \in V(G)\}$ be the set of input colors that occur in the instance. Since $|K| \leq |V(G)| = n$, we can choose an injective compression function $c : K \to \{0,1\}^{\lceil \log_2 n \rceil}$. The instance-dependent construction first uses an input embedding function that maps each color $k \in K$ to $c(k)$ and maps the remaining colors in $[C] \setminus K$ arbitrarily. Since $wp \geq \lceil \log_2 n \rceil$, each node can send the compressed color to all neighbors in one message-passing layer.

After this layer, node u has access to the compressed WL-type $(c(\chi(u)), \{\!\{c(\chi(v)) : v \in N(u)\}\!\})$, represented by a canonical vectorization, for example by the method used in Theorem 3.4. We denote this vector by $\boldsymbol{t}_u$. Because c is injective on the input colors that occur, this compressed WL-type is the same for two nodes $u, v \in V(G)$ if and only if $\mathrm{type}_{G,\chi}(u) = \mathrm{type}_{G,\chi}(v)$.

There are at most n compressed WL-types that occur in the input instance. Since $C \geq n$, we can choose an injective mapping from these vectorized compressed WL-types $\boldsymbol{t}_u$ to $[C]$. The instance-dependent decoder implements this lookup table. Therefore, two nodes receive the same output color if and only if they have the same WL-type. □

H Proof of Theorem 4.2

In this section we prove Theorem 4.2.

Theorem 4.2. *For every sufficiently large integer n, every integer $m \in [n, n^2]$, and every integer $D \in [8, n]$, there exists a family $\mathcal{G}_{n,m,D}$ of connected graphs, each with $\Theta(n)$ nodes, $\Theta(m)$ edges, and diameter D. For any color set $[C]$ with $C \geq \max_{G \in \mathcal{G}_{n,m,D}} |V(G)|$ and $\log C = o(m)$, any deterministic or zero-error randomized decentralized parameter-adaptation framework that uses $r \geq 1$ adaptation rounds to adapt the parameters of a depth-d, width-w, precision-p MPGNN, so that for every valid input instance (G, χ) with $G \in \mathcal{G}_{n,m,D}$ the resulting MPGNN simulates one step of the WL test on (G, χ), must satisfy $r = \Omega\left(\frac{D}{d} + \frac{m}{dwp}\right)$.*

Proof. By the seed-fixing argument in Section 2.4, it suffices to prove the deterministic case. We reduce the Equality problem to the decentralized parameter-adaptation process. Given an instance

$(\boldsymbol{a}, \boldsymbol{b}) \in \{0,1\}^m \times \{0,1\}^m$, Alice and Bob construct their two sides of the hard graph $G_{\boldsymbol{a},\boldsymbol{b}}$ from the proof of Theorem 3.1. The initial parameter θ_0, the adaptation algorithm, and all target-independent information are fixed before the inputs $(\boldsymbol{a}, \boldsymbol{b})$ are known.

Starting from the uniform coloring $\widehat{\chi}^{(0)}$, Alice and Bob simulate three consecutive applications of the decentralized parameter-adaptation framework. Concretely, for $t = 0, 1, 2$, they run $\theta_r^{(t)} = \textsc{Adapt}^{(r)}(\theta_0, G_{\boldsymbol{a},\boldsymbol{b}}, \widehat{\chi}^{(t)})$ and then compute $\widehat{\chi}^{(t+1)} = \mathcal{N}_{\theta_r^{(t)}}\left(\widehat{\chi}^{(t)}; G_{\boldsymbol{a},\boldsymbol{b}}\right)$. By induction on t, each $\widehat{\chi}^{(t)}$ is equivalent to the coloring $\chi^{(t)}_{G_{\boldsymbol{a},\boldsymbol{b}}}$ after t steps of the WL test. Indeed, $\widehat{\chi}^{(0)}$ is the uniform coloring, and the success guarantee of the framework implies that one application maps any valid input coloring to the coloring obtained after one further step of the WL test, up to relabeling.

We first prove the communication lower bound. During one application of the decentralized parameter-adaptation framework, the r adaptation rounds transmit at most $O(rdwp)$ bits across the Alice-Bob cut, while the final depth-d inference pass transmits at most $O(dwp)$ bits across the cut. Repeating the framework three times only changes the communication by a constant factor. After the three applications, Alice holds the final color of $x^{(A)}$, and Bob holds the final color of $x^{(B)}$. They compare these two colors using $O(\log C)$ additional bits.

By the hard-instance property proved in Theorem 3.1, $\chi^{(3)}_{G_{\boldsymbol{a},\boldsymbol{b}}}\left(x^{(A)}\right) = \chi^{(3)}_{G_{\boldsymbol{a},\boldsymbol{b}}}\left(x^{(B)}\right) \iff \boldsymbol{a} = \boldsymbol{b}$. Since $\widehat{\chi}^{(3)} \equiv \chi^{(3)}_{G_{\boldsymbol{a},\boldsymbol{b}}}$, comparing the two final output colors solves EQ_m. Therefore the decentralized parameter-adaptation framework yields a deterministic communication protocol for EQ_m with communication $O(rdwp + dwp + \log C)$. The deterministic communication complexity of EQ_m is $\Omega(m)$. Since $\log C = o(m)$, we obtain $(r+1)dwp = \Omega(m)$.

We next prove the locality lower bound. Let $\ell = \mathrm{dist}_{G_{\boldsymbol{a},\boldsymbol{b}}}\left(x^{(A)}, x^{(B)}\right) = \Theta(D)$. To simulate one WL step, the procedure runs r adaptation rounds followed by a depth-d inference pass. Information can travel $O(d)$ hops per adaptation round and at most d hops during inference. Across all three simulated WL steps, the propagation radius is therefore $O(rd)$ for $r \geq 1$ and $O(d)$ for $r = 0$. Suppose, for contradiction, that this radius is less than $\ell/2$ on every graph in the hard family. Then the final output color of $x^{(A)}$ is independent of Bob's input $\boldsymbol{b}$, and the final output color of $x^{(B)}$ is independent of Alice's input $\boldsymbol{a}$. Hence there exist functions $F, H : \{0,1\}^m \to [C]$ such that the final output color of $x^{(A)}$ is $F(\boldsymbol{a})$, while the final output color of $x^{(B)}$ is $H(\boldsymbol{b})$. Correctness of the adaptation framework and the hard-instance property imply $F(\boldsymbol{a}) = H(\boldsymbol{b}) \iff \boldsymbol{a} = \boldsymbol{b}$ for all $\boldsymbol{a}, \boldsymbol{b} \in \{0,1\}^m$. In particular, $F(\boldsymbol{a}) = H(\boldsymbol{a})$ for every $\boldsymbol{a}$. Moreover, F must be injective. If $F(\boldsymbol{a}) = F(\boldsymbol{a}')$, then $F(\boldsymbol{a}') = F(\boldsymbol{a}) = H(\boldsymbol{a})$, which implies $\boldsymbol{a}' = \boldsymbol{a}$. Therefore $|\mathrm{Im}(F)| = 2^m$, and hence $C \geq 2^m$. This contradicts $\log C = o(m)$. Thus the propagation radius is at least $\ell/2 = \Omega(D)$ on some graph in the family, so $(r+1)d = \Omega(D)$.

For $r = 0$, these two bounds give the necessary condition $d = \Omega\left(D + \frac{m}{wp}\right)$ for simulation without adaptation. Combining the two bounds for $r \geq 1$, we obtain $r = \Omega\left(\frac{D}{d} + \frac{m}{dwp}\right)$. □

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction summarize the simulation settings in Section 2.4 and the resource bounds in Sections 3 and 4, also collected in Table 1.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Section 5 discusses the limitations of our models and theoretical results.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: Sections 2–4 specify the models, assumptions, and theorem statements. Full proofs appear in Appendices A–H.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [N/A]

   Justification: Our paper is purely theoretical and does not include experiments.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [N/A]

Justification: Our paper is purely theoretical and does not include experiments.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [N/A]

   Justification: Our paper is purely theoretical and does not include experiments.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [N/A]

   Justification: Our paper is purely theoretical and does not include experiments.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [N/A]

   Justification: Our paper is purely theoretical and does not include experiments.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We confirm that our paper conforms, in every respect, with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: Our paper only theoretically studies upper and lower bounds on the resources required for MPGNNs to implement one step of the WL test under different settings. Therefore, we view our work as purely theoretical, with no concrete or direct societal impacts.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: We do not use models that have a high risk for misuse.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [N/A]

    Justification: Our paper does not use existing assets.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: Our paper does not release new assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: Our paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: Our paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: We confirm that the core method development in this research does not involve LLMs as any important, original, or non-standard components.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.