Learning on Random Balls is Sufficient for Estimating (Some) Graph Parameters

Takanori Maehara
Facebook AI
London, United Kingdom
`tmaehara@fb.com`

Hoang NT
Tokyo Tech & RIKEN AIP
Tokyo, Japan
`hoangnt@net.c.titech.ac.jp`

Abstract

Theoretical analyses for graph learning methods often assume a complete observation of the input graph. Such an assumption might not be useful for handling any-size graphs due to the scalability issues in practice. In this work, we develop a theoretical framework for graph classification problems in the partial observation setting (i.e., subgraph samplings). Equipped with insights from graph limit theory, we propose a new graph classification model that works on a randomly sampled subgraph and a novel topology to characterize the representability of the model. Our theoretical framework contributes a theoretical validation of mini-batch learning on graphs and leads to new learning-theoretic results on generalization bounds as well as size-generalizability without assumptions on the input.

1 Introduction

Going beyond regular structural inputs such as grids (images), sequences (time series, sentences), or general feature vectors is an important research direction of machine learning and computational sciences. Arguably, most interesting objects and problems in nature can be described as graphs [37]. For such reason, graph learning methods, especially Graph Neural Networks (GNN) [60], have recently proven to be a useful solution to many problems in computer vision [10, 12, 24, 63], complex network analyses [22, 30, 73], molecule modeling [17, 32, 41, 45], and physics simulations [3, 31, 54].

The significant value of graph learning models in practice has inspired a large amount of theoretical work dedicated to exploring their representational limits and the possibilities of improving them. Most notably, the representational capability of GNNs has been in the spotlight of recent years. To answer the question "*Can GNNs approximate all functions on graphs?*", researchers discussed universal invariant and equivariant neural networks [27, 40, 43, 51] as *theoretical* upper limits for neural architectures or showed the correspondence between message-passing GNNs (MP-GNNs) to the Weisfeiler-Leman (WL) algorithm [68] as *practical* upper limits [46, 70].

Given an extremely large graph as an input, it is often impractical to keep the whole graph in the working memory. Therefore, practical graph learning methods often utilize neighborhood samplings [22, 73] or random walks [53] to handle this scalability issue. Because existing analyses assumed a complete observation of the input graphs [27, 51, 57], it is unclear what can be learned if we combine graph learning models with random samplings. Thus, the relevant question in this scenario is "*What graph functions are representable by GNNs when we can only observe random neighborhoods?*." This question adds another dimension to the discussion of GNN expressivity; even if we have a powerful GNN (in both theoretical and practical senses), what kind of graph functions can we learn if the input graphs are too large to be computed as a whole?

Contributions This study proposes a theoretical approach to address graph learning problems on large graphs by identifying a novel topology of the graph space. We discuss the graph classification problem in the main part of the paper and extend the discussion to the vertex classification problem in Appendix C. The extension to vertex classification can be realised by viewing it as a rooted-graph classification problem. We first introduce a *random ball sampling GNN (RBS-GNN)*, which is a mathematical model of GNNs implementable in a *random neighborhood* computational model, and prove that the model is universal in the class of estimable functions (Theorem 4). Our main contribution is introducing *randomized Benjamini–Schramm topology* in the space of all graphs and identifying the estimability of the function as the uniform continuity in this topology (Theorem 7). By applying our main theorem, we obtain the following learning-theoretic results.

- We show the equivalent of estimability and continuity (Theorems 4 and 7). This implies the continuity assumption is a sufficient condition for the mini-batch learning on graphs.
- We prove that the functions representable by RBS-GNNs are generalizable by showing an upper bound of the Rademacher complexity of Lipschitz graph functions (Theorem 10).
- We identify size-generalizable functions with estimable functions (Theorem 11). Then, by recognizing the size-generalization as a domain adaptation, we provide a size-generalization error based on the Wasserstein distance (Theorem 13).

Unlike existing studies, which assumed a random graph model [26, 28] or boundedness [11, 27, 51, 58], our framework does not assume anything about the graph class; instead, we assume the continuity of the graph functions. Our results listed above are model-agnostic, i.e., we only discuss the property of the function space, regardless of how GNN models are implemented. The model-agnostic nature of our results gives a systematic view to general graph parameters learning; their generality is especially useful as there are many different GNN architectures in practice [57, 69, 78].

2 Related Work

Large-scale GNNs The success of GNNs, especially vertex classification models like GCN [30] and GraphSAGE [22], has led to various large-scale industrial GNN systems (see [1] and references therein). Aiming to increase computational throughput while maintaining the predictive performance, most of these systems implemented fixed-size neighborhood sampling [22] to enable large-scale batching [23, 73, 76, 77]. GNNs have also been applied to the 3D point clouds classification problem [67], which translates a computer vision problem to the large graph classification problem, and the random sampling was empirically shown to be effective [35]. In this context, our work contributes a theoretical justification for the random sampling procedure.

Graph Parameter Learning Graph function, graph parameter, or graph invariant refer to a (real or integer value) property of graphs, which only depends on the graph structure. In other words, they are functions defined on isomorphism classes of graphs [37]. Determining graph properties from data has long been a topic of interest in theoretical computer science [8, 37] and is an important machine learning task in computational chemistry [9, 17] and biology [6, 16]. Recently, GNNs have been proven successful on a wide range of graph learning benchmark datasets. Current literature analyzed their expressivity to gain a better understanding of the architectures [27, 44, 51]. Several works identify MP-GNNs to the 1-dimensional WL isomorphism test [70] and further improve the GNN architectures to more expressive variants such as k-dimensional WL [46], port-numbered message passing [58], and sparse WL [48]. GNNs are also linked to the representational power of logical expressions [2]. These theoretical results assumed the complete observation of the input graph; therefore, it is difficult to see to what extent these results would hold when the only partial observation is available. By studying the RBS-GNN model, we give an answer to this issue. We use GNNs because they are the most expressive graph learning methods [27, 70]. Nonetheless, our results generalize for other universal (Theorem 4) and partially-universal (Theorem 17) methods.

Generalization Besides expressivity, another challenge in graph learning is to understand the generalization bounds. Scarselli et al. [61] introduced an upper bound for the VC-dimension of functions computable by GNNs, in which the output is defined on a special supervised vertex. Garg et al. [15] derived tighter Rademacher complexity bounds for similar MP-GNNs by considering the local computational tree structures. Liao et al. [36] obtained a generalization gap of MP-GNNs and GCNs [30] using PAC-Bayes techniques. Du et al. [11] obtained a sample complexity using a result in the kernel method for their graph neural tangent kernel model in learning propagation-based functions. Verma and Zhang [66] obtained a generalization gap of single-layer GCNs by analyzing the stability and dependency on the largest eigenvalue of the graph; Lv [39] derived a Rademacher bound for a similar

GCN model with a similar dependency. Keriven et al. [28] assumed an underlying random kernel (similar to graphons [37]) and analyzed the stability of discrete GCN using a continuous counterpart c-GCN. They derived the convergence bounds by looking at stability when diffeomorphisms [42] are applied to the underlying graph kernel, the distribution, and the signals. All these methods placed some assumptions on the graph space; either bounded degree [15, 36, 39], bounded number of vertices [11, 61, 66], or graphs belong to a random model [28]. Therefore, all these results become either inapplicable or unbounded in the general graph space. Our Theorem 10 contributes a complexity bound without assumptions on the graphs.

Property Testing and Constant-Time Local Algorithms Property testing on graphs is a task to identify whether the input graph satisfies a graph property Π or ϵ-far from Π [18]. Often a researcher in this area tries to derive an algorithm whose complexity is constant (i.e., only depends on ϵ) or sublinear in the input size [56]. Several graph properties admit sub-linear (or constant-time) algorithms; the examples include bipartite testing, triangle-free testing, edge connectivity, and matching [49, 74]. Recently, by bridging the constant-time algorithms and the GNN literature, Sato et al. [59] showed that, for each vertex, the neighborhood aggregation procedure of a GNN layer (they called it "node embedding") can be approximated in constant time. However, this does not result in a constant-time learning algorithm for GNNs because we still need to access all the vertices to get the desired outputs. Our results provide the first "fully constant-time" GNNs in the sense that the whole learning and prediction process runs in time independent of the size of the graphs (Section 4).

Statistical Theory for Network Analysis Learning graph property from samples is a traditional topic in statistical network analysis [14, 21, 34]. Klusowski and Wu [33] proved that it is hard to estimate the number of connected components using sublinear-size samples. Bhattacharya et al. [5] showed that the Horvitz–Thompson estimator for the number of subgraphs of constant size is consistent and asymptotic normal if the fourth-moment condition holds. Our result is consistent with these results as the number of connected components is non-continuous and the number of motifs is continuous in the randomized Benjamini–Schramm topology. These studies indicate a future direction of this study; for example, the asymptotic normality and consistency of RBS-GNNs.

3 Preliminaries

3.1 Graphs

A *(directed) graph* G is a tuple (V, E) of the set of vertices V and the set of edges $E \subseteq V \times V$. We use $V(G)$ for V and $E(G)$ for E when the graph is unclear from the context. A graph is *weakly connected* if the underlying undirected graph has a path between any two vertices. A *weakly connected component* is a maximal weakly connected subgraph. Two graphs G and H are *isomorphic* if there is a bijection $\phi : V(G) \to V(H)$ such that $(\phi(u), \phi(v)) \in E(H)$ if and only if $(u, v) \in E(G)$. Let $\mathcal{G}$ be the set of all directed graphs. A *ball of radius r centered at v*, $B_r(v)$ (also simply B), is the set of vertices whose shortest path distance from v is bounded by r. For $U \subseteq V(G)$, $G[U]$ is the subgraph of G induced by U.

A *rooted graph* (G, v) is a graph G augmented with a vertex v in $V(G)$. The isomorphism between (G, v) and (H, u) is defined in the same way as for graphs with the extra requirement that it maps v to u. A *k-rooted graph* $(G, v_1, \ldots, v_k)$ is defined similarly. We often recognize the graph induced by the ball $B_r(v)$ as a rooted graph whose root is v and by the union of k balls as a k-rooted graph.

Modern graph learning problems ask for a function $p : \mathcal{G} \to \mathcal{D}$ from training data, where $\mathcal{D}$ is a "learning-friendly" domain such as the set of real numbers $\mathbb{R}$, a d-dimensional real vector space $Q \subseteq \mathbb{R}^d$, or some finite sets. In most cases, the function p is required to be isomorphism-invariant (or invariant for short). This notion of graph functions coincides with the definition of *graph parameters*. Another term used in the literature is *graph property*, which can be formalized as a graph function whose co-domain is $\{0, 1\}$. Our work focuses on the case in which the co-domain is $\mathbb{R}$.

3.2 Computational Model

Extremely large graphs are usually stored in some complicated storage. Thus, there are some constraints on how we can access the graphs. In the area of property testing, such a situation is modeled by introducing a *computational model*, which is an oracle for accessing the graph.

Importantly, each computational model induces a topology on the graph space. As we will show in later sections, the ability to represent graph functions is related to this topology.

There are three main computational models in the literature: the *adjacency predicate model* [20], the *incidence function model* [19], and the *general graph model* [25, 52]. The adjacency predicate model, also known as the dense graph model, allows randomized algorithms to query whether two vertices are adjacent or not. With the incidence function model, also known as the bounded-degree graph model, algorithms can query a specific neighbor of a vertex. The general graph model lets the algorithms ask for both a specific neighbor and for whether two vertices are adjacent; hence, this is the most realistic model for actual algorithmic applications [18].

In this study, we consider the following *random neighborhood model*, which allows us to access the input graph G via the following queries:

- SampleVertex(G): Sample a vertex $u \in V$ uniformly randomly.
- SampleNeighbor(G, u): Sample a vertex v from the neighborhood of u uniformly randomly, where u is an already obtained vertex.
- IsAdjacent(G, u, v): Return whether the vertices u and v are adjacent, where u and v are already obtained vertices.

This model is a randomized version of the general graph model. Czumaj et al. [8] proposed a similar model to analyze edge streaming algorithms for property testing. However, their model does not have the IsAdjacent query, i.e., it is a randomized version of the incidence function model. Note that the computational model naturally specifies the *estimability* of the graph parameters. More formally, we have the following definition of estimability with respect to our random neighborhood model.

Definition 1 (Constant-Time Estimable Graph Parameter). *A graph parameter p is constant-time estimable on the random neighborhood model (estimable for short) if for any $\epsilon > 0$ there exists an integer N and a randomized algorithm $\mathcal{A}$ in the random neighborhood model such that $\mathcal{A}$ performs at most N queries and $|\mathcal{A}(G) - p(G)| < \epsilon$ with probability at least $1 - \epsilon$ for all graphs $G \in \mathcal{G}$.*

Some examples of (non-)estimable graph parameters are:

Example 2. *The number of vertices, min/max degree, and connectivity are not estimable.*

Example 3. *The triangle density and the local clustering coefficient are estimable.*

Additional examples of estimable graph parameters and experimental results are provided in Appendix D. In the next section, we implement a GNN following the proposed random neighborhood computational model. By showing the connection between the GNN and algorithms in the random neighborhood model, we obtain several theoretical results in Section 5.

4 Random Balls Sampling Graph Neural Networks (RBS-GNN)

This section introduces *RBS-GNN*, a theoretical GNN architecture based on the random neighborhood model. RBS stands for “Random Balls Sampling” and also ”Random Benjamini–Schramm” because our random neighborhood model extends the topology of the Benjamini–Schramm convergence [4]. Given an input graph, an RBS-GNN samples k random vertices and proceeds to sample random balls $B_1, \dots, B_k$ rooted at each of these vertices. A random ball of radius r and branching factor b is a subgraph obtained by the procedure RandomBallSample, illustrated in Figure 1 for $r = 1$ and $b = 4$. The exact procedure is presented in Algorithm 1. It is trivial to see that RandomBallSample can be implemented under the random neighborhood model with SampleVertex and SampleNeighbor.

After sampling k random balls, the next step is identifying the induced subgraph $G[B_1 \cup \dots \cup B_k]$ using the IsAdjacent procedure and computing the weakly connected components $C_1, \dots, C_{N_C}$ of the induced subgraph (Step 3 of Figure 1). The classifier part of an RBS-GNN has two trainable components: a multi-layer perceptron g and a GNN f. The output of an RBS-GNN is defined as

$$\text{RBS-GNN}(G) = g\left(\sum_j f(C_j)\right). \tag{1}$$

It should be emphasized that, as mentioned in the end of Section 2, our RBS-GNN can be evaluated in constant time (i.e., only dependent on the hyperparameters) because the subsets returned by RandomBallSample has a constant size regardless of the size of input graph G.

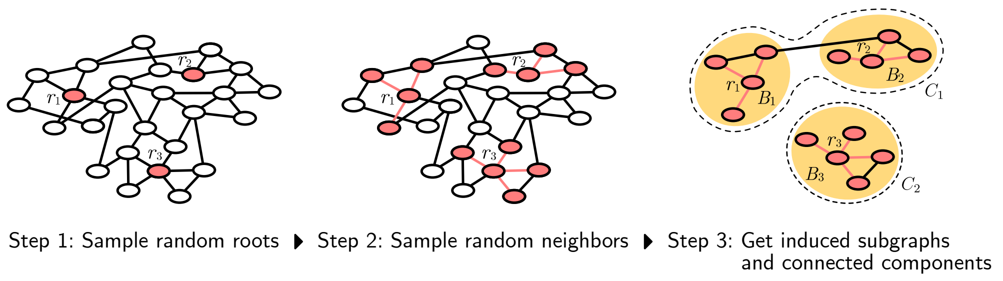

r_1
r_2
r_3
B_1
B_2
B_3
C_1
C_2
Step 1: Sample random roots
Step 2: Sample random neighbors
Step 3: Get induced subgraphs and connected components

Figure 1: Random Balls Sampling Procedure (Algorithm 1). Our computational model is different from the existing general graph model at Step 2, where we sample neighbors randomly instead of taking all neighbors. In Step 3, the randomly sampled edges are shown with color, and the induced edges are black. The weakly connected components C_1 and C_2 are inputs to the GNN.

Algorithm 1 Randomized Benjamini–Schramm GNN

```
procedure RandomBallSample($G$, $b$, $r$)
    layer$[0] \leftarrow []$, ..., layer$[r] \leftarrow []$
    Sample one random vertex from $V(G)$ and insert to layer$[0]$
    for $i = 1, \dots, r$ do
        for $u$ in layer$[i-1]$ do
            Sample $b$ random vertices (with replacement) from $\mathcal{N}(u)$ and insert to layer$[i]$
    return $G[$layer$[0] \cup \cdots \cup$ layer$[r]]$
procedure RBS-GNN($G, f, g, b, r, k$)
    $B_1, \dots, B_k \leftarrow$ RandomBallSample($G, b, r$)            ▷ Runs $k$ times to get $k$ balls.
    $C_1, \dots, C_{N_C} \leftarrow$ WeaklyConnectedComponents($G[B_1 \cup \dots B_k]$)
    return $g(\sum_j f(C_j))$
```

Relation to Existing GNN Models While RBS-GNN is motivated by the random neighborhood model, it has a strong connection with existing message-passing GNNs and optimization techniques in graph learning. When we select f to be a simple message-passing GNN, RBS-GNN is a generalization of the mini-batch version of GraphSAGE (Algorithm 2 in [22]), and the multi-layers perceptron module g acts as the global READOUT as in the GIN [70] architecture. Our analysis technique also applies for other mini-batch approaches of popular graph learning models [30, 64, 65]. This result provides a theoretical validation for the mini-batch approach in practice. Note that this is a positive result for continuous graph parameters, and we make no claim for the non-continuous case. On the other hand, f can also be a more expressive variant such as high-order WL [48], k-treewidth homomorphism density, or a universal approximator [27, 51].

5 Main Result

In this section, we conduct theoretical analyses of RBS-GNN for the graph classification problem. All the proofs are in Appendix A. To simplify the analysis, we assume the hyperparameters k, b, and r have the same value, and by a slight abuse of notation, we denote these values by r. Note that this setting would not alter the notion of estimability. We further simplify the discussion by assuming the graphs have no vertex features. Similar results hold when the vertices have finite-dimensional vertex features; see Appendix B. Additionally, as mentioned in Section 1, we obtained complementary results for the vertex classification problem in Appendix C.

5.1 Universality of RBS-GNN

We first characterize the expressive power of RBS-GNN. The following shows the universality of RBS-GNN, with a universal GNN component f, in the space of the estimable functions.

Theorem 4 (Universality of RBS-GNN)**.** *If a graph parameter $p : \mathcal{G} \to \mathbb{R}$ is estimable (in the random neighborhood model), then it is estimable by an RBS-GNN with a universal GNN f.*

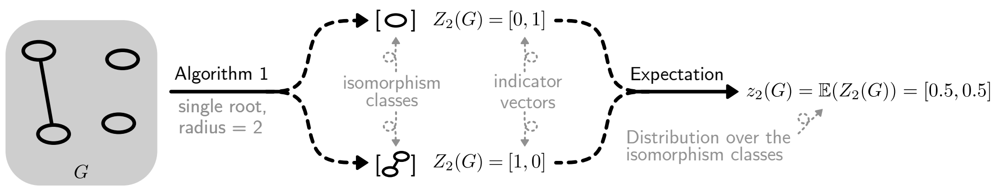


Figure 2: An illustrating example for the r-profile with $k = 1, r = 2$, and a simple graph G.

The proof of this theorem is an adaptation of the proof techniques by Czumaj et al. [8]. We first introduce a *canonical estimator*, which is an algorithm in the random neighborhood model defined by the following procedure. (1) Sample r random balls $B_1, \dots, B_r$ using RandomBallSample(G, r, r); (2) Return a number according to the isomorphism class of the subgraph $G[B_1 \cup \dots \cup B_r]$ induced by the balls. Since the number of random balls, the branching factor, and the radius are constant, we can see that the size of $G[B_1 \cup \dots \cup B_r]$ is bounded by r^{r+2}. Therefore, we can list all isomorphism classes of all graphs having at most r^{r+2} vertices and assign a unique number to each of them. Also, since the induced subgraph is bounded, it is possible to construct a universal approximator GNN [27, 51]. Therefore, we obtained the following.

Lemma 5. *If a graph parameter p is estimable, then it is estimable by a canonical estimator.*

Since a canonical estimator assigns a number according to the isomorphism class of the input, we see that RBS-GNN can approximate the canonical estimator by letting f be a universal approximator for bounded graphs. See the proof in Appendix A.1 for more detail.

Relation to Universality Results Existing universal GNNs assumed that the number of vertices of the input graphs are bounded. Theorem 4 shows that these universal GNNs for bounded graphs can be extended to general graphs by approximating the general graphs using the random balls sampling procedure. As a drawback, the theorem is only applicable to the continuous functions in the randomized Benjamini–Schramm topology introduced below. We emphasize that this drawback shows the limitation of the partial-observation (random neighborhoods) setting.

5.2 Topology of Graph Space: Estimability is Uniform Continuity

The previous section defined the estimability by the existence of an estimation algorithm. Such definition is suitable for algorithmic analysis; however, it is not suitable for further analysis, such as deriving the generalization error bounds. This section rephrases our estimability by the continuity in a new topology induced by a distance between two graphs.

We start with a simple example in Figure 2. The figure shows an input graph G consisting of two isolated vertices and one edge. For simplicity, Algorithm 1 only samples a single ball ($k = 1$) with radius two ($r = 2$). This configuration let us obtain two isomorphism classes with equal probability: a single vertex and a single edge. Hence, the event which each isomorphism class is obtained from Algorithm 1 can be represented by a two-dimensional vector. We denote this vector $Z_2(G)$, which takes value $[0, 1]$ when the single vertex is sampled and value $[1, 0]$ when the single edge is sampled. Clearly, this is a random vector whose expectation defines a distribution over the isomorphism classes with respect to the configuration of Algorithm 1. More generally, we can define the r-profile $Z_r(G)$ and the corresponding isomorphism class distribution $z_r(G)$ for positive integer values of radius r.

For an integer r, an r-*profile* $Z_r(G)$ of a graph G is a random variable of the (k-rooted[1]) isomorphism class of $G[B_1 \cup \dots \cup B_r]$, where each B_j is obtained from RandomBallSample(G, r, r). As RandomBallSample(G, r, r) produces a graph of size at most r^r, we can identify $Z_r(G)$ as a random finite-dimensional vector. Let $z_r(G) = \mathbb{E}[Z_r(G)]$ be the probability distribution over the isomorphism classes in terms of the k-rooted graph isomorphism, where the expectation is taken over SampleVertex and SampleNeighbor. The *sampling distance* of two graphs is defined by

$$d(G, H) = \sum_{r=1}^{\infty} 2^{-r} d_{TV}(z_r(G), z_r(H)), \qquad (2)$$

[1]For simplicity, we let $k = r$.

where d_{TV} is the total variation distance of two probability distributions given by $d_{TV}(p, q) = (1/2)\|p - q\|_1$. It should be emphasized that the sampling distance allows us to compare any two graphs even though they have a different number of vertices. We call the topology on the set of all graphs $\mathcal{G}$ induced by this sampling distance *randomized Benjamini–Schramm topology*. A graph parameter is defined to be *uniformly continuous in the randomized Benjamini–Schramm topology* by the followings.

Definition 6 (Uniform Continuity in the RBS Topology)**.** *A graph parameter $p : \mathcal{G} \to \mathbb{R}$ (resp. a randomized algorithm $\mathcal{A}$) is uniformly continuous if for any $\epsilon > 0$ there exists $\delta > 0$ such that for any G and H, $d(G, H) \leq \delta$ implies $|p(G) - p(H)| < \epsilon$ (resp. $|\mathcal{A}(G) - \mathcal{A}(H)| \leq \epsilon$ with probability at least $1 - \epsilon$).*

This topology connects the estimability in terms of the continuity as follows.

Theorem 7. *A graph parameter p is estimable in the random neighborhood model if and only if it is uniformly continuous in the randomized Benjamini–Schramm topology.*

The "if" direction of this theorem is given by the triangle inequality and the (optimal) coupling theorem [7]; the "only-if" is proved using the fact that the graph space is totally bounded as follows.

Lemma 8 (Totally Boundedness of Graph Space)**.** *For any $\epsilon > 0$, there exists a set of graphs $\{H_1, \ldots, H_C\}$ with $C \leq 2^{2^{(\log 1/\epsilon)^{O(\log 1/\epsilon)}}}$ such that $\min_{j \in \{1,\ldots,C\}} d(G, H_j) \leq \epsilon$ for all G.*

Theorem 7 allows us to apply existing "functional analysis techniques" to analyze the estimable functions. We present such applications in Section 6.

Intuition of the Sampling Distance and Relation to Benjamini–Schramm Topology The intuition behind our definition of the sampling distance reflects the idea that two graphs are similar when random samples from these graphs look similar, where the similarities on small samples are more important to the similarities on large samples. This definition generalizes the Benjamini–Schramm topology for the space $\mathcal{G}_D$ of all graphs of degree bounded by D, which uses a different definition of the "r-profile:" Let us define the r-profile by the union of r balls of radius r whose centers are sampled randomly. Then, the sampling distance defined using this r-profile induces a topology called the *Benjamini–Schramm topology*. This topology was first studied by Benjamini and Schramm [4] to analyze the planar packing problem, and now it is widely used to analyze the limit of bounded degree graphs, where the limit object is identified as the graphing; see [38]. A practical issue of the Benjamini–Schramm topology is that it is only applicable to bounded degree graphs, where many real-world extremely large graphs are complex networks having power-law degree distributions (i.e., unbounded degree). We addressed this issue by introducing the randomized Benjamini–Schramm topology, which is applicable to all graphs.

6 Theoretical Applications

6.1 Robustness Against Perturbation

The continuity immediately implies the robustness against the structural perturbation, i.e., for any $\epsilon > 0$ there exists $\delta > 0$ such that the output of RBS-GNN does not change more than ϵ if the graph is perturbed at most δ in the sampling distance. As the perturbation in sampling distance may not be intuitive in practice, we here provide a bound regarding the additive perturbation edges.

Proposition 9. *Let G be a graph and let G' be the graph obtained from G by adding $\delta|V(G)|$ edges completely randomly where $0 < \delta < 1$. Then $d(G, G') = O(1/\log(1/\delta))$.*

This result indicates that to change the output of RBS-GNN, one needs to add linearly many random edges; it is impractical in extremely large graphs. Note that the "adversarial" perturbation can change the distance more easily, especially if there is a "hub" in the graph; see Appendix A.3 for details.

6.2 Rademacher Complexity

Thus far, we only discussed the expressibility of the functions regardless of the learnability. Here, we derive the Rademacher complexity for the class of Lipschitz functions in the random Benjamini–Schramm topology. This gives an algorithm-independent bound of the learnability of the functions.

Theorem 10. *Let n be the number of training instances. The Rademacher complexity R_n of the set of 1-Lipschitz functions that maps to $[0,1]$ is $(\log\log n)^{-O(1/\log\log\log\log n)}$. It is $o(1/\log\log\log n)$.*

This result implies that, by minimizing the empirical error of n instances, we can achieve the generalization gap of $o(1/\log\log\log n)$ with high probability. To the extent of our knowledge, this is the first Rademacher bound for the general graph space, which guarantees the asymptotic convergence on any graph learning problem without assuming any graph structure.

Comparison with Existing Results The significant difference between existing studies [11, 15, 36, 39, 61, 66] and our bound (Theorem 10) is that ours is independent of any structural property, such as the maximum number of vertices, the maximum degree, and the spectrum of the graphs. Thus, ours can be applied to any graph distribution. Simply put, this is a consequence of the totally boundedness of the graph space (Lemma 8): For any $\epsilon > 0$, the space of all graphs is approximated by finitely many graphs; hence any graph parameter is bounded by the values among them, which is a constant depending on ϵ. The drawback of this generality is its poor dependency on the number of instances n, which leaves significant room for quantitative improvement. One possible way to improve the bound is by assuming some properties of the graph distribution because the above derivation is distribution-agnostic; a concrete strategy for improvement is left for future works.

6.3 Size-Generalizability

One interesting topic of GNNs is *size-generalization*, which is a property that a model trained on small graphs should perform well on larger graphs. Size-generalization is observed in several tasks [29]; however, it has also been proved that some classes of GNNs do not naturally generalize [72]. Hence, we want to know about the conditions for GNNs to generalize.

We need to distinguish the "approximation-theoretic" size-generalizability and the "learning-theoretic" size-generalizability. The former is the possibility of size-generalization, which is proved by showing the existence of size-generalizing models. This, however, does not mean that a size-generalizable model is obtained by training; thus, we need to introduce the latter. The latter is the degree of size-generalizability when we train a model using a dataset (or a distribution); it is proved by bounding the generalization error.

6.3.1 Approximation-Theoretic Size-Generalizability

We say that a function p is *size-generalizable in approximation-theoretic sense* if for any $\epsilon > 0$, there exists $N > 0$ such that we can construct an algorithm $\mathcal{A}$ using dataset $\{(p(G_{\leq N}), G_{\leq N}) : |V(G_{\leq N})| \leq N\}$ such that $|p(G) - \mathcal{A}(G)| \leq \epsilon$ with probability at least $1-\epsilon$ for all $G \in \mathcal{G}$. This gives one mathematical formulation of the size-generalizability as it requires to fit algorithm $\mathcal{A}$ to all graphs using the dataset of bounded graphs. In this definition, we have the following theorem.

Theorem 11. *Estimable functions are size-generalizable in the approximation-theoretic sense.*

This theorem is proved by constructing a size-generalizable algorithm. We first pick the continuity constant δ for ϵ using Theorem 7. Then, we construct a δ-net using Lemma 8. By storing all the values $p(G_i)$ for the graphs in the δ-net, we obtain a size-generalizable algorithm, where N is the maximum number of the vertices in the δ-net.

6.3.2 Learning-Theoretic Size-Generalizability

From the learning theoretic viewpoint, size-generalization is a domain adaptation from the distribution of smaller graphs to the distribution of larger graphs [72]. Thus, it is natural to utilize the domain adaptation theory [55]. Especially since we have introduced the sampling distance defined on all pairs of graphs irrelevant to their sizes, we here employ the Wasserstein distance-based approach [62].

We start from a general situation. Let $\mathcal{D}_1$ and $\mathcal{D}_2$ be joint distributions of graphs and their labels, and $\mathcal{G}_1$ and $\mathcal{G}_2$ be the corresponding marginal distributions of graphs. We abbreviate $\mathbb{E}_1$ and $\mathbb{E}_2$ for the expectations on $\mathcal{D}_1$ and $\mathcal{D}_2$, respectively. The Wasserstein distance between $\mathcal{G}_1$ and $\mathcal{G}_2$ is given by

$$W(\mathcal{G}_1, \mathcal{G}_2) = \inf_{\pi} \mathbb{E}_{(G_1,G_2)\sim\pi} d(G_1, G_2), \tag{3}$$

where d is the sampling distance of the graphs, and π runs over the couplings between these distributions. Let $\lambda = \inf_h \{\mathbb{E}_1|y - h(G)| + \mathbb{E}_2|y - h(G)|\}$ be the optimal combined error, where $\inf_h$ runs over all 1-Lipschitz functions. We have the following lemma.

Lemma 12. *For any* 1*-Lipschitz functions* h *and* h'*, we have the following.*

$$\mathbb{E}_1|y - f(G)| \leq \mathbb{E}_2|y - f(G)| + 2W(\mathcal{G}_1, \mathcal{G}_2) + \lambda, \tag{4}$$

Combining this result with the Rademacher complexity (Theorem 10), we obtain the following generalization bound.

Theorem 13. *Let* $\epsilon > 0$*. Let* $(y_{21}, G_{21}), \ldots, (y_{2n}, G_{2n})$ *be independently drawn from* $\mathcal{D}_2$*. If* $\lambda = O(\epsilon)$ *and* $n \geq 2^{2^{2^{\tilde{\Omega}(1/\epsilon)}}}$*, then, for any* 1*-Lipschitz function* h*, we have*

$$\mathbb{E}_{(G_1,y_1)\sim\mathcal{D}_1}[|y_1 - h_1(G)|] \leq \frac{1}{n}\sum_{i=1}^{n}|y_{2i} - h(G_{2i})| + 2W(\mathcal{D}_1, \mathcal{D}_2) + O(\epsilon) \tag{5}$$

with probability at least $1 - \epsilon$*.*

The condition $\lambda = O(\epsilon)$ requires the existence of a "consistent rule" among both $\mathcal{D}_1$ and $\mathcal{D}_2$. For example, this condition holds when the labels are generated by $y = f(G) + \epsilon\mathcal{N}(0,1)$ for some 1-Lipschitz function f, where $\mathcal{N}(0,1)$ is the standard normal distribution. We can obtain the size-generalization bound by applying the above theorem for the distribution of large graphs $\mathcal{G}_1$ and of small graphs $\mathcal{G}_2$. Thus, we only need to evaluate their Wasserstein distance. The Wasserstein distance can be large in the worst-case; thus, we here consider concrete examples of graph distributions.

First, we consider the case that undirected graphs are drawn from the *configuration model of* d*-regular graphs*. In this model, a graph is constructed by the following procedure: (1) It creates N vertices with d half-edges; (2) Then, it pairs the half-edges and connects them to obtain edges. We see that a learning problem on this distribution is size-generalizable.

Proposition 14. *Let* $\mathcal{G}$ *be a distribution of random* d*-regular graphs generated by the configuration model, and* $\mathcal{G}_{\leq N}$ *be the distribution conditioned on only graphs of size bounded by* N*. If* $N \geq (\log 1/\epsilon)^{\Omega(\log 1/\epsilon)}$ *then* $W(\mathcal{G}, \mathcal{G}_{\leq D}) = O(\epsilon)$*.*

This result can be generalized to a general distribution of graphs with large girth. Next, we consider the case where undirected graphs are drawn from a graphon. A *graphon* $\mathcal{W}$ is a function $\mathcal{W} : [0,1] \times [0,1] \to [0,1]$. A graph G_N is drawn from $\mathcal{W}$ if we first draw N random numbers $x_1, \ldots, x_N \in [0,1]$ uniformly randomly. Then, for each pairs (x_i, x_j), we put an edge with probability $\mathcal{W}(x_i, x_j)$. This model extends Erdos–Renyi random graph and stochastic block model; see [37] for more detail.

Proposition 15. *Let* $\mathcal{W}$ *be a graphon. Let* N_1 *and* N_2 *be integers with* $N_1 < N_2$*. Let* $\mathcal{G}_{N_i}$ *be a distribution of graphs of* N_i *vertices drawn from* $\mathcal{W}$*. If* $N_1 \geq 2^{O(1/\epsilon^2)}$ *then* $W(\mathcal{G}_{N_1}, \mathcal{G}_{N_2}) \leq \epsilon$*.*

Finally, we consider the case that $\mathcal{D}_N$ is obtained from $\mathcal{D}$ by the metric projection. Let Π_N be the projection onto the space of graphs of size at most N, i.e., $\Pi(G) = \mathrm{argmin}_{G_N : |V(G_N)| \leq N} d(G, G_N)$.

Proposition 16. *Let* $\mathcal{G}$ *be any graph distribution and let* $\mathcal{G}_{\leq N} = \Pi(\mathcal{G})$ *be the projected distribution of graphs of size at most* N*. For any* $\epsilon > 0$*, there exists* N *such that* $W(\mathcal{G}, \mathcal{G}_{\leq N}) \leq \epsilon$*.*

A drawback of this result is that an explicit bound of N is not known, even for its deterministic variant in the bounded degree graphs (See Proposition 19.10 in [37]). The only known bound is for the bounded degree graphs with large girth [13].

Comparison with Existing Results Size-generalization of GNNs is reported on several tasks, but its theoretical analysis is limited. Yehudai et al. [72] studied the size-generalizability using the concept of d-pattern, which is information obtained from d-ball; it is similar to our r-profile. Their results are approximation-theoretic as they showed the (non-)existence of size-generalizable models but did not show how such models can be obtained by training on data. Xu et al. [71] proved the size-generalization of the max-degree function under several conditions on the training data and GNNs. Their result is essentially an approximation-theoretic as it assumes the dataset lies in and spans a certain space that is sufficient to identify the max-degree function.

6.4 Partially-Universal RBS-GNNs

Thus far, we assumed the universal GNNs are plugged into the RBS-GNNs for theoretical analysis. This assumption achieves the maximum expressive power in this framework; however, in practice, we often use expressive but more efficient GNNs such as GCN [30], GIN [70], or GAT [64]. Here, we discuss what will be changed if we made this modification.

Let $\equiv$ be an equivalence relation on graphs. We assume that $\equiv$ is *consistent with the weakly connected component decomposition*, i.e., if $G_1 \equiv H_1$ and $G_2 \equiv H_2$ then $G_1 + G_2 \equiv H_1 + H_2$, where the "$+$" symbol denotes the disjoint union of two graphs. We say that a function h (resp. a randomized algorithm $\mathcal{A}$) is $\equiv$*-indistinguishable* if $h(G) = h(G')$ (resp. $\mathcal{A}(G) = \mathcal{A}(G')$ given the random sample) for all $G \equiv G'$. A GNN is $\equiv$*-universal* if it can learn any $\equiv$-indistinguishable functions. For example, it is known that GIN is universal with respect to the WL indistinguishable functions [70]. Let RBS-GNN[$\equiv$] be a class of RBS-GNNs that uses an $\equiv$-universal GNN f in Equation (1). The following shows the partial universality of this architecture.

Theorem 17. *If f is estimable and $\equiv$-indistinguishable, then it is estimable by an RBS-GNN[$\equiv$].*

One application of this theorem is extending the expressivity of GraphSAGE to the partial observation setting. GraphSAGE can represent the local clustering coefficient if we have the complete observation of the graph [22, Theorem 1]. We can prove the local clustering coefficient is estimable (Proposition 24 in Appendix D). By applying Theorem 17 to the equivalence relation $G_1 \equiv G_2$ defined by $f(G_1) = f(G_2)$ for all function f representable by GraphSAGE, we obtain the following.

Proposition 18. *The mini-batch version of the GraphSAGE (Algorithm 2 in [22]) can estimate the local clustering coefficient.*

Comparison with Existing Studies Equivalence relations associated with GNNs are mainly studied in the context of the "limitation" of GNNs: If a GNN is $\equiv$-indistinguishable, then it cannot learn any function h that is non $\equiv$-indistinguishable. Morris et al. [46] proved a message-passing type GNN cannot distinguish two graphs having the same d-patterns. Garg et al. [15] identified indistinguishable graphs of several GNNs, including GCN [30], GIN [70], GPNGNN [58], and DimeNet [32].

On the other hand, we should use non-universal GNNs in practice because more expressive GNNs have higher computational costs (e.g., universal GNNs [27] is more costly than the graph isomorphism test). We here considered RBS-GNN[$\equiv$] because $\equiv$-universal GNNs are theoretically tractable classes of non-universal GNNs. With similar motivation, [51] proposed GNNs parameterized by information aggregation pattern and proved the $\equiv$-universality, where $\equiv$ is induced by the aggregation patterns.

7 Conclusion

We answered the question "*What graph functions are representable by GNNs when we can only observe random neighborhoods?*" by proving the functions representable by RBS-GNNs coincides with the estimable functions in the random neighborhood model, which is equivalent to the uniformly continuous functions in the randomized Benjamini–Schramm topology. The word "(Some)" in our title was meant to emphasize the restriction to continuous graph functions. The result holds without any assumption on the input graphs, such as the boundedness. This result gives us a "functional analysis view" of graph learning problems and leads to several new learning-theoretic results. The weakness of our result is the poor dependency on the number of training instances, which is the trade-off for generality. We believe addressing this issue will be an interesting future direction. Another future direction links to the asymptotic behaviors of RBS-GNNs. Motivating example includes the motif counting problem, in which Bhattacharya et al. [5] proved the Horvitz–Thompson estimator is consistent and asymptotically normal. A corresponding result for the RBS-GNN will allow us to obtain a confidence interval of an estimation and to perform statistical testing.

Potential Impact Our work contributes an understanding of general graph learning models whose inputs are random samples of arbitrarily large graphs. Due to the theoretical nature of our results, we believe there will not be a direct nor indirect negative societal impact.

Acknowledgement We would like to thank the anonymous reviewers and the area chairs for their thoughtful comments which help us improve our manuscript. HN is partially supported by the Japanese Government MEXT SGU Scholarship No. 205144.

References

[1] Sergi Abadal, Akshay Jain, Robert Guirado, Jorge López-Alonso, and Eduard Alarcón. Computing Graph Neural Networks: A Survey from Algorithms to Accelerators. *ArXiv preprint*, abs/2010.00130, 2020. URL https://arxiv.org/abs/2010.00130.

[2] Pablo Barceló, Egor V. Kostylev, Mikaël Monet, Jorge Pérez, Juan L. Reutter, and Juan Pablo Silva. The Logical Expressiveness of Graph Neural Networks. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[3] Peter W. Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, and Koray Kavukcuoglu. Interaction Networks for Learning about Objects, Relations and Physics. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 4502–4510, 2016.

[4] Itai Benjamini and Oded Schramm. Recurrence of Distributional Limits of Finite Planar Graphs. In *Selected Works of Oded Schramm*, pages 533–545. Springer, 2011.

[5] Bhaswar B Bhattacharya, Sayan Das, and Sumit Mukherjee. Motif Estimation via Subgraph Sampling: The Fourth Moment Phenomenon. *The Annals of Statistics*, To Appear, 2021.

[6] Karsten M Borgwardt, Cheng Soon Ong, Stefan Schönauer, SVN Vishwanathan, Alex J Smola, and Hans-Peter Kriegel. Protein Function Prediction via Graph Kernels. *Bioinformatics*, 21: 47–56, 2005.

[7] Juan A Cuestaalbertos, L Ruschendorf, and Araceli Tuerodiaz. Optimal Coupling of Multivariate Distributions and Stochastic Processes. *Journal of Multivariate Analysis*, 46(2):335–361, 1993.

[8] Artur Czumaj, Hendrik Fichtenberger, Pan Peng, and Christian Sohler. Testable Properties in General Graphs and Random Order Streaming. *ArXiv preprint*, abs/1905.01644, 2019. URL https://arxiv.org/abs/1905.01644.

[9] Asim Kumar Debnath, Rosa L Lopez de Compadre, Gargi Debnath, Alan J Shusterman, and Corwin Hansch. Structure-Activity Relationship of Mutagenic Aromatic and Heteroaromatic Nitro Compounds: Correlation with Molecular Orbital Energies and Hydrophobicity. *Journal of Medicinal Chemistry*, 34(2):786–797, 1991.

[10] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, NeurIPS 2016, December 5-10, 2016, Barcelona, Spain*, pages 3837–3845, 2016.

[11] Simon S. Du, Kangcheng Hou, Ruslan Salakhutdinov, Barnabás Póczos, Ruosong Wang, and Keyulu Xu. Graph Neural Tangent Kernel: Fusing Graph Neural Networks with Graph Kernels. In *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 5724–5734, 2019.

[12] Matthias Fey, Jan Eric Lenssen, Frank Weichert, and Heinrich Müller. SplineCNN: Fast Geometric Deep Learning With Continuous B-Spline Kernels. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 869–877. IEEE Computer Society, 2018.

[13] Hendrik Fichtenberger, Pan Peng, and Christian Sohler. On Constant-Size Graphs That Preserve the Local Structure of High-Girth Graphs. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2015, August 24-26, 2015, Princeton, NJ, USA*, volume 40 of *LIPIcs*, pages 786–799. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2015.

[14] Ove Frank. Estimation of Graph Totals. *Scandinavian Journal of Statistics*, pages 81–89, 1977.

[15] Vikas K. Garg, Stefanie Jegelka, and Tommi S. Jaakkola. Generalization and Representational Limits of Graph Neural Networks. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 3419–3430. PMLR, 2020.

[16] Thomas Gärtner. A Survey of Kernels for Structured Data. *ACM SIGKDD Explorations Newsletter*, 5(1):49–58, 2003.

[17] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural Message Passing for Quantum Chemistry. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, August 6-11, 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 1263–1272. PMLR, 2017.

[18] Oded Goldreich. Introduction to Testing Graph Properties. In *Property Testing*, pages 105–141. Springer, 2010.

[19] Oded Goldreich and Dana Ron. Property Testing in Bounded Degree Graphs. In *Proceedings of the Twenty-ninth Annual ACM Symposium on Theory of Computing*, pages 406–415, 1997.

[20] Oded Goldreich, Shari Goldwasser, and Dana Ron. Property Testing and Its Connection to Learning and Approximation. *Journal of ACM*, 45(4):653–750, 1998.

[21] Leo A Goodman. On the Estimation of the Number of Classes in a Population. *The Annals of Mathematical Statistics*, 20(4):572–579, 1949.

[22] William L. Hamilton, Zhitao Ying, and Jure Leskovec. Inductive Representation Learning on Large Graphs. In *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.

[23] Zhihao Jia, Sina Lin, Mingyu Gao, Matei Zaharia, and Alex Aiken. Improving the Accuracy, Scalability, and Performance of Graph Neural Networks with ROC. In *Proceedings of Machine Learning and Systems 2020, MLSys 2020, Austin, TX, USA, March 2-4, 2020*. MLSys.org, 2020.

[24] Michael Kampffmeyer, Yinbo Chen, Xiaodan Liang, Hao Wang, Yujia Zhang, and Eric P. Xing. Rethinking Knowledge Graph Propagation for Zero-Shot Learning. In *2019 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 11487–11496. IEEE Computer Society, 2019.

[25] Tali Kaufman, Michael Krivelevich, and Dana Ron. Tight Bounds for Testing Bipartiteness in General Graphs. *SIAM Journal on Computing*, 33(6):1441–1483, 2004.

[26] Tatsuro Kawamoto, Masashi Tsubaki, and Tomoyuki Obuchi. Mean-Field Theory of Graph Neural Networks in Graph Partitioning. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 4366–4376, 2018.

[27] Nicolas Keriven and Gabriel Peyré. Universal Invariant and Equivariant Graph Neural Networks. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 7090–7099, 2019.

[28] Nicolas Keriven, Alberto Bietti, and Samuel Vaiter. Convergence and Stability of Graph Convolutional Networks on Large Random Graphs. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, Virtual*, 2020.

[29] Elias B. Khalil, Hanjun Dai, Yuyu Zhang, Bistra Dilkina, and Le Song. Learning Combinatorial Optimization Algorithms over Graphs. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, NeurIPS 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 6348–6358, 2017.

[30] Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017*. OpenReview.net, 2017.

[31] Thomas N. Kipf, Ethan Fetaya, Kuan-Chieh Wang, Max Welling, and Richard S. Zemel. Neural Relational Inference for Interacting Systems. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 2693–2702. PMLR, 2018.

[32] Johannes Klicpera, Janek Groß, and Stephan Günnemann. Directional Message Passing for Molecular Graphs. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[33] Jason M Klusowski and Yihong Wu. Estimating the Number of Connected Components in a Graph via Subgraph Sampling. *Bernoulli*, 26(3):1635–1664, 2020.

[34] Eric D Kolaczyk and Gábor Csárdi. *Statistical Analysis of Network Data with R*, volume 65. Springer, 2014.

[35] Itai Lang, Asaf Manor, and Shai Avidan. SampleNet: Differentiable Point Cloud Sampling. In *2020 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 7575–7585. IEEE Computer Society, 2020.

[36] Renjie Liao, Raquel Urtasun, and Richard Zemel. A PAC-Bayesian Approach to Generalization Bounds for Graph Neural Networks. In *9th International Conference on Learning Representations, ICLR 2021, Virtual, April 25-29, 2021*. OpenReview.net, 2021.

[37] László Lovász. *Large Networks and Graph Limits*. American Mathematical Society, 2012.

[38] László Lovász and Balázs Szegedy. Limits of Dense Graph Sequences. *Journal of Combinatorial Theory, Series B*, 96(6):933–957, 2006.

[39] Shaogao Lv. Generalization Bounds for Graph Convolutional Neural Networks via Rademacher Complexity. *ArXiv preprint*, abs/2102.10234, 2021. URL https://arxiv.org/abs/2102.10234.

[40] Takanori Maehara and Hoang NT. A Simple Proof of the Universality of Invariant/Equivariant Graph Neural Networks. *ArXiv preprint*, abs/1910.03802, 2019. URL https://arxiv.org/abs/1910.03802.

[41] Pierre Mahé and Jean-Philippe Vert. Graph Kernels Based on Tree Patterns for Molecules. *Machine Learning*, 75(1):3–35, 2009.

[42] Stéphane Mallat. Group Invariant Scattering. *Communications on Pure and Applied Mathematics*, 65(10):1331–1398, 2012.

[43] Haggai Maron, Heli Ben-Hamu, Hadar Serviansky, and Yaron Lipman. Provably Powerful Graph Networks. In *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.

[44] Haggai Maron, Heli Ben-Hamu, Nadav Shamir, and Yaron Lipman. Invariant and Equivariant Graph Networks. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[45] Andreas Mayr, Günter Klambauer, Thomas Unterthiner, and Sepp Hochreiter. DeepTox: Toxicity Prediction using Deep Learning. *Frontiers in Environmental Science*, 3:80, 2016.

[46] 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, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 4602–4609. AAAI Press, 2019.

[47] Christopher Morris, Nils M. Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. TUDataset: A Collection of Benchmark Datasets for Learning With Graphs. In *ICML 2020 Workshop on Graph Representation Learning and Beyond (GRL+ 2020)*, 2020. URL www.graphlearning.io.

[48] Christopher Morris, Gaurav Rattan, and Petra Mutzel. Weisfeiler and Leman Go Sparse: Towards Scalable Higher-order Graph embeddings. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, Virtual*, 2020.

[49] Huy N. Nguyen and Krzysztof Onak. Constant-Time Approximation Algorithms via Local Improvements. In *49th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2008, October 25-28, 2008, Philadelphia, PA, USA*, pages 327–336. IEEE Computer Society, 2008.

[50] Hoang NT and Takanori Maehara. Revisiting Graph Neural Networks: All We Have is Low-pass Filters. *ArXiv preprint*, abs/1905.09550, 2019. URL https://arxiv.org/abs/1905.09550.

[51] Hoang NT and Takanori Maehara. Graph Homomorphism Convolution. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 7306–7316. PMLR, 2020.

[52] Michal Parnas and Dana Ron. Testing the Diameter of Graphs. *Random Structures & Algorithms*, 20(2):165–183, 2002.

[53] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. DeepWalk: Online Learning of Social Representations. In *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2014, New York, NY, USA, August 24-27, 2014*, pages 701–710. ACM, 2014.

[54] Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter W Battaglia. Learning Mesh-Based Simulation with Graph Networks. *ArXiv preprint*, abs/2010.03409, 2020. URL https://arxiv.org/abs/2010.03409.

[55] Ievgen Redko, Emilie Morvant, Amaury Habrard, Marc Sebban, and Younes Bennani. *Advances in Domain Adaptation Theory*. Elsevier, 2019.

[56] Ronitt Rubinfeld and Asaf Shapira. Sublinear Time Algorithms. *SIAM Journal on Discrete Mathematics*, 25(4):1562–1588, 2011.

[57] Ryoma Sato. A Survey on the Expressive Power of Graph Neural Networks. *ArXiv preprint*, abs/2003.04078, 2020. URL https://arxiv.org/abs/2003.04078.

[58] Ryoma Sato, Makoto Yamada, and Hisashi Kashima. Approximation Ratios of Graph Neural Networks for Combinatorial Problems. In *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 4083–4092, 2019.

[59] Ryoma Sato, Makoto Yamada, and Hisashi Kashima. Constant Time Graph Neural Networks. *ArXiv preprint*, abs/1901.07868, 2019. URL https://arxiv.org/abs/1901.07868.

[60] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The Graph Neural Network Model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2008.

[61] Franco Scarselli, Ah Chung Tsoi, and Markus Hagenbuchner. The Vapnik–Chervonenkis Dimension of Graph and Recursive Neural Networks. *Neural Networks*, 108:248–259, 2018.

[62] Jian Shen, Yanru Qu, Weinan Zhang, and Yong Yu. Wasserstein Distance Guided Representation Learning for Domain Adaptation. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, AAAI 2018, New Orleans, LA, USA, February 2-7, 2018*, pages 4058–4065. AAAI Press, 2018.

[63] Diego Valsesia, Giulia Fracastoro, and Enrico Magli. Learning Localized Generative Models for 3D Point Clouds via Graph Convolution. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[64] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018*. OpenReview.net, 2018.

[65] Petar Velickovic, William Fedus, William L. Hamilton, Pietro Liò, Yoshua Bengio, and R. Devon Hjelm. Deep Graph Infomax. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[66] Saurabh Verma and Zhi-Li Zhang. Stability and Generalization of Graph Convolutional Neural Networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019*, pages 1539–1548. ACM, 2019.

[67] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic Graph CNN for Learning on Point Clouds. *ACM Transaction on Graphics*, 38(5), 2019.

[68] 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.

[69] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A Comprehensive Survey on Graph Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[70] 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.

[71] Keyulu Xu, Mozhi Zhang, Jingling Li, Simon S Du, Ken-ichi Kawarabayashi, and Stefanie Jegelka. How Neural Networks Extrapolate: From Feedforward to Graph Neural Networks. In *9th International Conference on Learning Representations, ICLR 2021, Virtual, April 25-29, 2021*. OpenReview.net, 2021.

[72] Gilad Yehudai, Ethan Fetaya, Eli Meirom, Gal Chechik, and Haggai Maron. From Local Structures to Size Generalization in Graph Neural Networks. *ArXiv preprint*, abs/2010.08853, 2020. URL https://arxiv.org/abs/2010.08853.

[73] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, and Jure Leskovec. Graph Convolutional Neural Networks for Web-Scale Recommender Systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2018, London, UK, August 19-23, 2018*, pages 974–983. ACM, 2018.

[74] Yuichi Yoshida, Masaki Yamamoto, and Hiro Ito. An Improved Constant-Time Approximation Algorithm for Maximum Matchings. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing*, STOC 2009, page 225–234, New York, NY, USA, 2009. ACM.

[75] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabás Póczos, Ruslan Salakhutdinov, and Alexander J. Smola. Deep Sets. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, NeurIPS 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 3391–3401, 2017.

[76] Dalong Zhang, Xin Huang, Ziqi Liu, Zhiyang Hu, Xianzheng Song, Zhibang Ge, Zhiqiang Zhang, Lin Wang, Jun Zhou, Yang Shuang, et al. AGL: A Scalable System for Industrial-purpose Graph Machine Learning. *ArXiv preprint*, abs/2003.02454, 2020. URL https://arxiv.org/abs/2003.02454.

[77] Da Zheng, Chao Ma, Minjie Wang, Jinjing Zhou, Qidong Su, Xiang Song, Quan Gan, Zheng Zhang, and George Karypis. DistDGL: Distributed Graph Neural Network Training for Billion-Scale Graphs. *ArXiv preprint*, abs/2010.05337, 2020. URL https://arxiv.org/abs/2010.05337.

[78] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph Neural Networks: A Review of Methods and Applications. *AI Open*, 1:57–81, 2020.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See the discussion after each result.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See the conclusion.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See the beginning of Section 5 and 6.
   (b) Did you include complete proofs of all theoretical results? [Yes] Proof ideas are described in the main content, and complete proofs are provided in Appendix A.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Appendix D.2.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See result tables and figures.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Appendix D.2, paragraph "Computational Recourses".
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] See Appendix D and citation [47].
   (b) Did you mention the license of the assets? [N/A] There is no explicit license provided by the original creator in both their website and their manuscript [47].
   (c) Did you include any new assets either in the supplemental material or as a URL? [No] We did not curate any new asset.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Complete proofs

A.1 Theorem 4

Theorem 4 states the universality of the proposed RBS-GNN. This theorem was proposed to address the question "Even if we have a powerful GNN, what kind of graph functions can we learn if the input graphs are too large to be computed as a whole?" posed in the Introduction. We think of this theorem from two perspectives. In one view, this theorem extends the universality of "complete-observation" GNNs in to "partial-observation". In another, the theorem reduced the universality of GNNs to only universal on estimable functions. This section presents proofs leading up to Theorem 4.

Proof of Lemma 5. We construct a canonical estimator from the original estimator. Let $\mathcal{E}$ be the original estimator and let N be the total number of queries of to achieve accuracy $\epsilon^2/2$. We first construct an estimator $\mathcal{E}_1$. $\mathcal{E}_1$ samples an N random balls using $\mathsf{RandomBallSample}(G, N, N)$ and simulates $\mathcal{E}$ on $\mathcal{E}_1$ using a permutation π over the vertices of $B_1 \cup \cdots \cup B_r$. Because of the simulation, we obtain

$$\mathrm{Prob}_{S,\pi}\left[|f(G) - \mathcal{E}_1(G \mid S, \pi)| > \epsilon^2/2\right] < \epsilon^2/2. \tag{6}$$

Then, we construct the final estimator $\mathcal{E}_2$. $\mathcal{E}_2$ returns the expected value of the output of $\mathcal{E}_1$ over all simulations. Here,

$$\begin{aligned}\mathbb{E}_S[|f(G) - \mathcal{E}_2(G \mid S)|] &= \mathbb{E}_S[|f(G) - \mathbb{E}_\pi[\mathcal{E}_1(G \mid S, \pi)]|] && (7)\\ &\le \mathbb{E}_{S,\pi}[|f(G) - \mathcal{E}_1(G \mid S, \pi)|] && (8)\\ &\le \epsilon^2. && (9)\end{aligned}$$

Thus, by the Markov inequality,

$$\mathrm{Prob}_S[|f(G) - \mathcal{E}_2(G \mid S)]| > \epsilon] \le \epsilon. \tag{10}$$

□

Using Lemma 5, we obtain the proof for Theorem 4.

Proof of Theorem 4. The output of the canonical estimator is determined by the isomorphism class of the subgraph induced by the balls. Hence, it is determined by the isomorphism classes of the weakly connected components of the induced subgraph. This means that we can write the canonical estimator as a function from the set of weakly-connected graphs: $h(\{C_1, \ldots, C_l\})$. Here, we can injectively map each C_j as a finite-dimensional vector z_j using a universal neural network f since it has a size bounded by a constant (depending on N). Also, the number of connected components is bounded by a constant (depending on N). Thus, we can identify the function h as a permutation-invariant function with a constant number of arguments whose inputs are finite-dimensional vectors. Therefore, we can apply Theorem 9 in [75], which shows that there exists continuous functions g and ϕ such that $h(z_1, \ldots, z_m) = g(\sum_i \rho(z_m))$. for all $z_1, \ldots, z_m$. Because ϕ is approximated by a neural network, we can combine it with the GNN f; therefore, we obtain the proof. □

A.2 Theorem 7

Theorem 7 is perhaps the most important contribution of this work. This theorem continues to analyze the concept of estimable graph functions by providing a topology in which estimable functions are continuous and vice versa. We find that the result is quite useful when we want to apply functional analysis techniques to analyze graph learning problems. This section presents the proofs of Lemma 8 and Theorem 7.

Proof of Lemma 8. This is a variant of [37, Proposition 19.10], which is for a different topology (different computational model). We can prove our lemma by the same strategy, but here we provide a proof for completeness.

Let $r = \lceil \log 2/\epsilon \rceil$. We choose a maximal set of graphs $H_1, \ldots, H_N$ such that for all $i \ne j$, $d_{TV}(z_s(H_i), z_s(H_j)) > \epsilon/4$ holds on some $s \le r$. We can see that such a set exists (see below). By the maximality, for any graph G, there exists j such that $d_{TV}(z_s(G), z_s(H_j)) \leq \epsilon/4$ for all $s \leq r$, which implies $d(G, H) \leq (1/2)^r + \epsilon/2 \leq \epsilon$.

We show the upper bound of C. In the proof, we represent G by an r-tuple $(z_1(G), \dots, z_r(G))$ of probability distributions, where each $z_s(G)$ lies on $2^{s^s \times s^s} = 2^{(\log 1/\epsilon)^{O(\log 1/\epsilon)}}$-dimensional simplex for $s \leq r$. Because the packing number of d-dimensional simplex in the total variation distance (equivalently in the l_1 metric) is $(1/\epsilon)^{O(d)}$, we cannot choose more than $2^{2^{(\log 1/\epsilon)^{O(\log 1/\epsilon)}}}$ points whose pairwise distance is at least $\epsilon/4$. □

Proof of Theorem 7. Suppose f is estimable. For any $\epsilon > 0$, we choose a canonical estimator $\mathcal{A}$ of accuracy ϵ. Then we have $|f(G) - f(H)| \leq |f(G) - \mathcal{A}(G)| + |\mathcal{A}(G) - \mathcal{A}(H)| + |\mathcal{A}(H) - f(H)|$. Here, the first and last terms are at most ϵ by the definition of $\mathcal{A}$ with probability at least $1 - \epsilon$, respectively. We take $\delta = 2^{-r}\epsilon$ for the second term. Then, for any G, H with $d(G, H) \leq \delta$, we have $d_{TV}(z_r(G), z_r(H)) \leq \epsilon$; hence, by the optimal coupling theorem,[2] there exists a coupling between $Z_r(G)$ and $Z_r(H)$ such that $P(Z_r(G) \neq Z_r(H)) = d_{TV}(z_r(G), z_r(H)) \leq \epsilon$. Thus, the output of the algorithm $\mathcal{A}$ coincides on G and H with a probability at least $1 - \epsilon$. Therefore we have $|f(G) - f(H)| \leq 3\epsilon$ with probability at least $1 - 3\epsilon$. By taking the expectation, we obtain the result.

Suppose f is uniformly continuous. For any $\epsilon > 0$, let $\delta > 0$ be the corresponding constant in the continuity definition. Take a $\delta/2$-net $\{H_1, \dots, H_C\}$ and let $r = \lceil \log 4/\delta \rceil$. The algorithm $\mathcal{A}$ performs random sapling to estimate the distribution $(z_1(G), \dots, z_r(G))$ with accuracy $\delta/2$ with probability at least $1 - \epsilon$. Then, it outputs $f(H_j)$, where H_j is the nearest neighborhood of G. By the construction, the algorithm finds H_j with $d(G, H_j) \leq \delta$ with probability at least $1 - \epsilon$. Therefore, we have $|f(G) - \mathcal{A}(G)| = |f(G) - f(H_j)| \leq \epsilon$ with probability as least $1 - \epsilon$. □

It should be emphasized that the space of all graphs equipped with the randomized Benjamini–Schramm topology is *not* compact, because there is a continuous but not uniformly continuous function; the average degree function is such an example.

A.3 Proofs for Applications

This section provides the proofs for the theoretical applications section in the main part (Section 6). Most notably, the proofs for Theorem 10, 11, and 13 are provided here.

Proof of Proposition 9. Let M be the endpoints of the random edges. Then, M induces a uniform distribution on the vertices of G. Any run with $Z_r(G) \cap M = \emptyset$ can be coupled with $Z_r(G')$; so the coupling probability is

$$\begin{aligned} P(M \cap Z_r(G) = \emptyset) &= \sum_{x \in M} P(x \notin Z_r(G)) && (11) \\ &\leq |M| r^r / n && (12) \\ &= 2 r^r \delta. && (13) \end{aligned}$$

By the optimal coupling theorem, we have $d_{TV}(z_r(G), z_r(G')) = 2r^r\delta$. Therefore,

$$\begin{aligned} d(G, G') &= \sum_{r=1}^{\infty} 2^{-r} d_{TV}(G_r, G'_r) && (14) \\ &\leq \sum_{r=1}^{\infty} 2^{-r} \min\{1, 2r^r\delta\} && (15) \\ &\leq s^s \delta + 2^{-s} && (16) \end{aligned}$$

for any s. By putting $s = \log\log(1/\delta)$, we obtain the result. □

If M is chosen adversarially, we cannot obtain the inequality (12). In particular, if M contains a vertex x with a large PageRank, as the probability of $x \in Z_r(G)$ is large, we cannot bound the distance.

[2]See [7], or `pages.uoregon.edu/dlevin/AMS_shortcourse/ams_coupling.pdf` (May, 2021).

Proof of Theorem 10. We use the following inequality that bounds the Rademacher complexity by the covering number $C_{\mathcal{F}}(\epsilon)$ of the function space $\mathcal{F}$:

$$R_n(\mathcal{F}) \le \inf_{\epsilon > 0} \left\{ \epsilon + O\left(\sqrt{\frac{\log C_{\mathcal{F}}(\epsilon)}{n}} \right) \right\}. \tag{17}$$

We choose an $\epsilon/2$-net of the graphs of size $C(\epsilon/2)$ by Lemma 8. Then, we define an (external) ϵ-cover of the space of 1-Lipschitz functions by the piecewise constant functions whose values are discretized by $\epsilon/2$, where the pieces are the Voronoi regions of the ϵ-net; it is easy to verify this is an ϵ-cover of the space of 1-Lipschitz functions. This shows $C_{\mathcal{F}}(\epsilon) \le (2/\epsilon)^{C(\epsilon/2)} = 2^{2^{2^{2^{O(\log(1/\epsilon)\log\log(1/\epsilon))}}}}$. By substituting ϵ satisfying $O(\log(1/\epsilon)\log\log(1/\epsilon)) = \log\log\log(n/\log n))$, we obtain the result.

$\log C_{\mathcal{F}}(\epsilon) = 2^{2^{2^{O(\log 1/\epsilon \log\log 1/\epsilon})}}$. We set ϵ to be $O(\log 1/\epsilon \log\log 1/\epsilon) = \log\log\log(n/\log n)^2$. Then, by definition, $\sqrt{\log C_{\mathcal{F}}(\epsilon)/n} = 1/\log n$.

We try to evaluate ϵ. We see ϵ satisfies $\log 1/\epsilon \log\log 1/\epsilon = \Omega(\log\log\log n)$.

Recall that $x \log x = y$ iff $y = e^{W(x)}$ where $W(x)$ is the Lambert W function. Since $W(x) = \log(x/\log x) + \Theta(\log\log x/\log x)$, we have $y \ge x/\log x$. By using this formula, we have $\log 1/\epsilon = \Omega(\log\log\log n/\log\log\log\log n)$. □

Proof of Theorem 11. By Theorem 7, an estimable function f is uniformly continuous. Let δ be the constant for ϵ for the continuity. By Lemma 8, there is an δ-net $\{H_1, \dots, H_C\}$ and let $N(\delta)$ be the maximum number of vertices in the graphs in the δ-net. Our algorithm $\mathcal{A}$ outputs $\mathcal{A}(G) = f(H_j)$ where H_j is the nearest neighbor of G. This algorithm achieves the accuracy of ϵ because $d(G, H_j) \le \delta$. Also, the algorithm can be constructed only accessing graphs of size at most $N = \max_j |V(H_j)|$. Hence, f is size-generalizable. □

Proof of Lemma 12. This is an adaptation of [62] to our metric space. Their proof only uses the "easy" direction of the Kantorovich–Rubinstein duality, which holds on any metric space. Hence, we obtain this lemma.

To be self-contained, we will give a proof. For any 1-Lipschitz function f and any coupling π between $\mathcal{D}_1$ and $\mathcal{D}_2$, we have the following "easy" direction of the Kantorovich–Rubinstein duality:

$$\mathbb{E}_1[f(G_1)] - \mathbb{E}_2[f(G_2)] = \mathbb{E}_{(G_1,G_1)\sim\pi}\mathbb{E}[f(G_1) - f(G_2)] \tag{18}$$
$$\le \mathbb{E}_{(G_1,G_2)\sim\pi}\mathbb{E}[d(G_1, G_2)] \tag{19}$$
$$\le W(\mathcal{G}_1, \mathcal{G}_2). \tag{20}$$

By putting $f = (h - h')/2$, we obtain

$$\mathbb{E}_1|h(G) - h'(G)| - \mathbb{E}_2|h(G) - h'(G)| \le 2W(\mathcal{G}_1, \mathcal{G}_2). \tag{21}$$

Hence,

$$\mathbb{E}_1|y - h(G)| \le \mathbb{E}_1|y - h'(G)| + \mathbb{E}_1|h(G) - h'(G)| \tag{22}$$
$$= \mathbb{E}_1|y - h'(G)| + \mathbb{E}_1[h(G) - h'(G)| + \mathbb{E}_2|h(G) - h'(G)| - \mathbb{E}_2|h(G) - h'(G)| \tag{23}$$
$$\le \mathbb{E}_1|y - h'(G)| + \mathbb{E}_2|h(G) - h'(G)| + 2W(\mathcal{G}_1, \mathcal{G}_2) \tag{24}$$
$$\le \mathbb{E}_2|y - h(G)| + \mathbb{E}_1|y - h'(G)| + \mathbb{E}_2|h(G) - h'(G)| + 2W(\mathcal{G}_1, \mathcal{G}_2). \tag{25}$$

By taking the infimum over h', we obtain the theorem. □

Proof of Theorem 13. We obtain the result by combining Theorem 10 and Lemma 12. □

Proof of Proposition 14. Let $\mathcal{G}_N$ be the distribution of random d-regulra graphs of size N. Then, we can see that $Z_r(G_N) \mid G_N \sim \mathcal{G}_N$ has no cycle with probability at least $1 - r^{O(r)}/N$. This implies that we can couple $Z_r(G) \mid G \sim \mathcal{G}$ and $Z_r(G_{\le N}) \mid G_{\le N} \sim \mathcal{G}_{\le N}$ with probability at least $1 - r^{O(r)}/N$. By putting $r = \log 1/\epsilon$, we obtain the result. □

Proof of Proposition 15. This follows from the proof of Lemma 10.31 and Exercise 10.31 in [37]. □

Proof of Proposition 16. We can choose N by the maximum number of vertices in the ϵ-net. Then, we have $d(G, \Pi(G)) \leq \epsilon$. Thus the Wasserstein distance is bounded by ϵ. □

Proof of Theorem 17. We introduce a helper concept, $\equiv$*-indistinguishably estimable*, which is a class of functions that is estimable by $\equiv$-indistinguishable computation on r-profile. By the same argument as Theorem 4, we can show that RBS-GNN[$\equiv$] can represent $\equiv$-indistinguishably estimable function.

Now we prove that if a function f is estimable and $\equiv$-indistinguishable, then it is $\equiv$-indistinguishably estimable. Because f is estimable, there exists $\delta > 0$ such that $|f(G) - f(H)| \leq \epsilon/2$ if $d(G, H) < \delta$. We fix a δ-net $\{H_1, \dots, H_C\}$ of the graph space. We consider a quotient space of the graphs by $\equiv$, select a representative $[G]$ to each quotient, and assign H_i to $[G]$ which is the nearest neighbor of the representative G.

Our estimator $\mathcal{A}$ is the following. First, we obtain a subgraph S by sampling sufficiently many vertices to be $d(S, G) \leq \delta$ with probability at least $1 - \epsilon$. Second, we take the representative $[S]$ of the equivalent class containing S. Finally, we output the value $f(H)$, where H is the nearest neighbor of $[S]$. By construction, $\mathcal{A}$ is an $\equiv$-indistinguishable computation after the sampling. Here,

$$|f(G) - \mathcal{A}(G)| \leq |f(G) - f(S)| + |f(S) - f([S])| + |f([S]) - f(H)| \leq \epsilon \tag{26}$$

with probability at least $1 - \epsilon$, where the first term in the right-hand side is at most $\epsilon/2$ with probability at least $1 - \epsilon$ due to the sampling and continuity, the second term is zero due to the $\equiv$-indistinguishability, and the last term is at most $\epsilon/2$ due to the δ-net and uniform continuity. □

B Extended Results: Finite-Dimensional Vertex Features

We can extend our framework to the vertex-featured case. We assume the vertex features are in $[0, 1]^d$. This assumption is well-aligned with the pre-processing step in practice where vertex features are normalized [30, 50, 51].

The estimability is defined similarly, where we additionally assume that the estimation is uniformly continuous with respect to the vertex features on the sampled subgraph (in the standard topology of $\mathbb{R}^d$). Then, we can prove the RBS-GNN can estimate arbitrary estimable vertex-featured graph parameters.

The difficulty is how to define the topology on the vertex-featured graphs. As in the non-featured case, We want to define $Z_r(G)$ by the "frequency" of the graphs. However, since there are uncountably many vertex-featured graphs, we need a technique. To address this issue, we fix an ϵ-net on $[0, 1]^d$; it has the cardinality of $(1/\epsilon)^d$. Then, we approximate the vertex features of the sampled graph by the elements of the ϵ-net by the uniform continuity. Then, the number of "vertex-featured graphs" of N vertices is bounded by $((1/\epsilon)^d)^{O(N \times N)}$; hence we can define the randomized Benjamini–Schramm topology. The space is totally bounded since the ϵ-net is constructed by combining the ϵ-net of the graph and the ϵ-net of $[0, 1]^d$. Note that this makes no significant difference on the size of the ϵ-net since the difference is absorbed in the nested power.

C Extended Results: Vertex Classification Problems

In this section, we extend our framework for the graph classification problem to the vertex classification problem.

The vertex classification problem is usually defined as follows. We are given a set of graphs $G_1, \dots, G_N$ with the "supervised vertices" $S_1 \subseteq V(G_1), \dots, S_N \subseteq V(G_N)$ and the labels y_u on the supervised vertices. The task is to find an equivariant function $h\colon G \mapsto (y_1, \dots, y_n) \in \mathcal{Y}^{V(G)}$. This formulation, however, is not suitable for large graphs because it needs to output values to all the vertices. Here, we recognize a vertex classification problem as a rooted graph classification problem as in [50]: The input of the problem is a set of pairs $(y_{v_i}, (G_i, v_i))$ of rooted graphs (G_i, v_i) and the label y_{v_i}. The goal is to find a function h such that $h((G, v)) \approx y_v$. It should be noted that a graph G

with a supervised nodes $S \subseteq V(G)$ in the original formulation is transformed to $|S|$ rooted graphs $\{(G, v) : v \in S\}$.

Our framework for the graph classification problem is easily extended to the rooted graph classification problem. We first modify our computational model by assuming the root of the graph is available at the beginning of the computation. Then, the estimability of the function is defined in the same way using this computational model. Then, we modify the RBS-GNN to have one additional ball centered at the root vertex, i.e.,

$$\text{RBS-GNN}((G, v)) = g\left(f(C_0), \sum_{j=1}^{N_C} f(C_j)\right) \tag{27}$$

where $B_0, \dots, B_k$ are the random balls obtained by RandomBallSample where the root of B_0 is conditioned by v, and $C_0, \dots, C_{CS}$ are the weakly connected components of $G[B_0 \cup \dots \cup B_k]$ where C_0 contains v. We can prove that any estimable vertex parameter in the random neighborhood model for the rooted graph is estimable using the RBS-GNN. Also, by extending the randomized Benjamini–Schramm topology to the rooted graphs, we see the estimability coincides with the uniform continuity in the randomized Benjamini–Schramm topology of rooted graphs.

Using this topology, we can obtain the vertex classification version of the results in Section 6. As the number of rooted graphs of N vertices is N times larger than the number of non-rooted graphs of N vertices, the covering number of the rooted graph space is larger than that of the non-rooted graph space. But this makes no significant difference because this gap is absorbed in the nested logarithm.

This formulation gives several consequences on the vertex classification problem.

- We can evaluate the required number of supervised vertices to obtain the desired accuracy. In this formulation, each supervised vertex v corresponds to a rooted graph (G, v). Thus, if the supervised vertices are chosen randomly, the required number of supervised vertices is evaluated by the Rademacher complexity of the model. In particular, we can obtain a model with an accuracy ϵ from the constantly many supervised vertices.
- We can characterize the difficulty of a vertex classification problem with different supervision using transfer learning. Imagine a situation that the supervised vertices have large degrees, but we want to predict the vertex property on low-degree vertices. This situation can be recognized that the training and test rooted graph distributions, $\mathcal{G}_{\text{train}}$ and $\mathcal{G}_{\text{test}}$, are different. Therefore, we can apply Lemma 12 to obtain an estimation of the test error, which involves the Wasserstein distance of these distributions.

D Extended Results: Practical Applications

Many real-world graph parameters are estimable in our framework. This section provides a review of graph parameters and their estimability in our random neighborhood model. Most notably, this section demonstrates the usage of Theorem 7 for practical graph parameters and provide the proof for Proposition 18. In addition, we provide experimental results on real-world datasets to verify our theoretical claims.

D.1 Graph Parameters

For convenience, we re-state the random neighborhood model and the *estimable* definitions here. The original definitions were provided in Section 3.2.

Definition 19 (Random Neighborhood Model)**.** *The random neighborhood computational model allows the following three queries given an input graph G:*

- SampleVertex(G): Sample a vertex $u \in V$ uniformly randomly.
- SampleNeighbor(G, u): Sample a vertex v from the neighborhood of u uniformly randomly, where u is an already obtained vertex.
- IsAdjacent(G, u, v): Return whether the vertices u and v are adjacent, where u and v are already obtained vertices.

This computational model induces an estimability definition and a topology, named random Benjamini-Schramm, on the graph space $\mathcal{G}$.

Definition 20 (Constant-Time Estimable Graph Parameter)**.** *A graph parameter p is constant-time estimable on the random neighborhood model (estimable for short) if for any $\epsilon > 0$ there exists an integer N and a randomized algorithm $\mathcal{A}$ in the random neighborhood model such that $\mathcal{A}$ performs at most N queries and $|\mathcal{A}(G) - p(G)| < \epsilon$ with probability at least $1 - \epsilon$ for all graphs $G \in \mathcal{G}$.*

D.1.1 Non-estimable Graph Parameters

We first see that an estimable parameter is bounded as follows.

Proposition 21. *An estimable parameter p is bounded.*

Proof. Recall that an estimable parameter is uniformly continuous (Theorem 7). Let δ be the constant for $\epsilon = 1$ for the continuity. Let us fix an δ-net $\{G_1, G_2, \dots\}$ of the graph space using the totally boundedness of the space (Lemma 8). Then, p is bounded by $C = 1 + \max_i p(G_i)$. In fact, for any graph $G \in \mathcal{G}$, we have

$$p(G) \leq |p(G) - p(G_i)| + |p(G_i)| \leq C \tag{28}$$

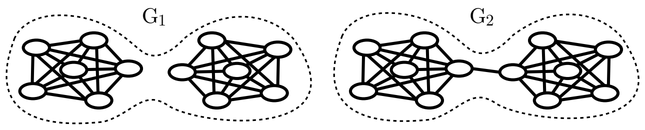
where G_i is the nearest neighbor of G in the ϵ-net. □

Example 2 in Section 3.2 states that the number of vertices and the min/max degrees are not estimable; these immediately follow form Proposition 21 since they are unbounded parameters. Similarly, the average degree is unbounded so it is not estimable. The connectivity function is an example of bounded but non-estimable graph parameter.

Proposition 22. *$p(G) = 1[G$ is connected$]$ is not estimable.*

Proof. We prove this proposition by giving a counter example showing p violates the definition for continuity. Since continuity is equivalent to estimability, such counter example would also disprove the estimability of p.

We first fix $\epsilon = 1/2$. Then, we choose two graphs G_1 and G_2 such that G_1 is the disjoint union of two cliques of size N and G_2 is obtained from G_1 by adding one edge between them. The figure below demonstrates for the case $N = 6$.



We see that $d(G_1, G_2) \leq \delta$ for any chosen $\delta > 0$ if N is sufficiently large. This is because the distribution $z_r(G_1)$ and $z_r(G_2)$ of isomorphism classes only different at the event that one of the two connected vertices of G_2 is sampled. The probability for such event becomes increasingly insignificant when N is sufficiently large. Hence, the distance $d(G_1, G_2)$ can be arbitrarily small as stated above. However, by the definition of the connectivity function, for any N we always have $|p(G_1) - p(G_2)| = 1 > 1/2$. Hence, p is not continuous, and by Theorem 7, it is also not estimable. □

D.1.2 Estimable Graph Parameters

The following propositions proves statements in Example 3.

Proposition 23. *The triangle density is a uniformly continuous parameter and estimable.*

Using Theorem 7, it is clear that we only need to prove estimability or continuity. We show both proofs for this case as a demonstration. For simplicity, we assume the input graph G is undirected.

Proof for Estimability. We show that the triangle density is estimable by constructing a random algorithm and prove that this algorithm estimates the triangle density to an arbitrary precision dependent only on the number of random samples (Definition 1). The randomized algorithm can be implemented under the random neighborhood computational model (Definition 19).

Algorithm 2 Triangle Density Estimation in the Random Neighborhood Model

```
1: procedure IsTriangle($G$, $u$, $v$, $q$)
2:     $uv \leftarrow$ IsAdjacent($G$, $u$, $v$);
3:     $uq \leftarrow$ IsAdjacent($G$, $u$, $q$);
4:     $qv \leftarrow$ IsAdjacent($G$, $q$, $v$);
5:     return $uv \wedge uq \wedge qv$;                          ▷ $\wedge$ is the logical "and".
6: procedure TriangleDensity($G$, $T$)
7:     triangles $\leftarrow 0$;
8:     for $i$ in $1, \dots, T$ do
9:         $u, v, q \leftarrow$ SampleVertex($G$);               ▷ Runs 3 times to get 3 samples.
10:        triangles $\leftarrow$ triangles $+$ IsTriangle($G, u, v, q$);
11:    return $\frac{1}{T}$triangles;
```

The procedure IsTriangle in Algorithm 2 return 1 if the three input vertices induce a triangle and 0 otherwise. Let X be the output of IsTriangle given three random vertices $u, v,$ and q from graph G, and $\bar{X}$ be the output of TriangleDensity. By definition, the expectation $\mathbb{E}(X)$ is the true triangle density $p_\triangle$. Since the $p_\triangle$ and Var(X) are clearly finite, we can apply the Chebyshev's concentration bound to the sample average $\bar{X}$ with T samples to obtain the following. For any $\epsilon > 0$,

$$\mathbb{P}(|\bar{X} - p_\triangle)| \geq \epsilon) \leq \frac{\text{Var}(X)}{\epsilon^2 T}. \tag{29}$$

This bound shows that if we take $T = O(\epsilon^{-3})$ samples, then with probability at least $1 - \epsilon$ we obtain an estimation less than ϵ from the true value. Note that T is only dependent on the precision ϵ and not the size of G; this shows the intuition behind the constant-time nature of our RBS-GNN.

□

Proof for Continuity. We now prove that the triangle density $p_\triangle$ is uniformly continuous in the randomized Benjamini-Schramm topology. For a given $\epsilon > 0$, we choose $\delta = 2^3\epsilon$. Let G_1 and G_2 be two graphs satisfying $d(G_1, G_2) \leq \delta$. We denote two random variable X_1 and X_2 to represent the event that random sampling from G_1 and G_2 obtained a triangle. X_1 and X_2 follows $z_\triangle(G_1)$ and $z_\triangle(G_2)$ distributions, respectively. By the optimal coupling theorem, the random sampling on G_1 and G_2 can be coupled with probability at least $1 - \epsilon$.

$$d_{TV}(z_\triangle(G_1), z_\triangle(G_2)) = \min_{(X_1, X_2)-\text{couplings}} \mathbb{P}(X_1 \neq X_2) \tag{30}$$

Hence, by the definition of the triangle density, these differs at most ϵ. □

Using a similar technique, we can prove the estimability or equivalently uniformly continuity of the local clustering coefficient.

Proposition 24. *The local clustering coefficient is uniformly continuous and estimable.*

D.2 Graph Classification in Random Neighborhood Model

We show the results for RBS-GNN on social networks datasets in the TUDatasets repository [47]: COLLAB, REDDIT-BINARY, and REDDIT-MULTI5K. We preprocess these datasets in the same way as proposed by Xu et al. [70]. Because of this pre-processing, each vertex has a feature vector representing its position in the degree distribution. The reason for such setting is because in 1-WL, the degree determines the initial coloring [48]. A summary of the datasets is given in Table 1.

To simulate the random ball sampling procedure of RBS-GNN, we pre-sample the original datasets and use these random samples in both training and testing. Table 1 shows the sampling setting we used to report the results in Table 2. We prepared multiple other settings for r, b, and k, see the provided source code for more detail (`supp/notebooks/Preprocessing.ipynb`). Let $N_G(\cdot)$ be the neighborhood function of the sampled input graph, we construct a K-layers f as follows.

$$h_G^{(\ell)}(u) = \text{MLP}^{(\ell)}\left(\sum_{v \in N_G(u)} h^{(\ell-1)}(v)\right), \ell = 1, \dots, K \tag{31}$$

Table 1: Overview of the graph classification datasets. This is a small part of the TUDataset [47]. $|\mathcal{G}|$ denotes the total number of graphs in the dataset, $\overline{\nu}(G)$ denotes the average number of nodes per graph, $|c|$ denotes the number of classes, d denotes the dimensionality of vertex features (created by [70]). r denotes the radius of random balls and also the branching factor. k denotes the number of random balls. $\%$ denotes the average coverage of random balls in terms of the number of edges. $\%$ Memory denotes the relative data storage size.

DATASETS	$\lvert\mathcal{G}\rvert$	$\overline{\nu}(G)$	$\lvert c\rvert$	d	r	b	k	$\%\lvert E(G)\rvert$	% Memory
COLLAB	5000	74.5	3	367	2	5	3	58.3 ± 22.9	55.7
RDT-BINARY	2000	429.6	2	566	3	5	3	37.3 ± 28.6	14.9
RDT-MULTI5K	5000	508.5	5	734	3	5	3	23.8 ± 17.7	14.2

$$f(G) = \sum_{\ell=1,\ldots,K} \sum_{u \in G} h_G^{(\ell)}(u), \tag{32}$$

where MLP$^\ell$ is a single layer MLP with no activation. Let g be a 2-layers MLP with ReLU activation functions, the final output of RBS-GNN[$\equiv_2$] is given similar to Equation (1):

$$\text{RBS-GNN}[\equiv_2](G) = g\left(\sum_j f(G)\right). \tag{33}$$

The implemented RBS-GNN is denoted by RBS-GNN[$\equiv_2$] because its GNN component f resembles a message-passing GNN such as GIN [70]. In all our experiments, the graph neural network f has 4 propagation layers and 5 MLP layers, each of these layers have 32 ReLU hidden units (see `supp/src/rbsgnn/models/mpgnn_batched.py` and `supp/notebooks/Cross-Validation Scores.ipynb`). Regularization methods are weight decay (10^{-3}), learning rate decay (initialized at 0.01, step size 50, $\gamma = 0.5$), and dropout (0.5).

Computational Resources We run all our experiments on a single computer having a single Intel CPU (i7-8700K3.70GHz), 64GB DDR4 memory, and a NVIDIA GeForce GTX 1080Ti GPU with CUDA 11.3 (driver version 465.31). The system runs Linux Kernel 5.12.6. Our model's prototype is implemented using Python 3.9 and PyTorch 1.8.1+cu111 (see `supp/src/requirements.txt` for the detail of the Python environment).

Cross-Validation Scores Reporting the 10-folds (also 3-folds and 5-folds) cross-validation scores for graph learning model is a common task in the literature [11, 69, 70]. We compare our practical implementation of RBS-GNN to existing benchmarks in terms of 10-folds cross validation scores. We show the results for other baselines reported by Xu et al. [70]. These baselines includes WL-subtree, PatchySan, and AWL (see Section 7 in [70] for more detail). Note that RBS-GNN only has access to partial inputs for both training and testing procedures. The fractions of observed edges and storage memory are shown in Table 1.

Table 2: Best test accuracy (in percentage) for the graph classification task. Note that RBS-GNN only has access to 20$\sim$60 percent of edges and 15$\sim$50 percent of node features (Table 1).

MODELS	COLLAB	RDT-BINARY	RDT-MULTI5K
GIN-0	80.2 ± 1.9	92.4 ± 2.5	57.5 ± 1.5
WL subtree	78.9 ± 1.9	81.0 ± 3.1	52.5 ± 2.1
PatchySan	72.6 ± 2.2	86.3 ± 1.6	49.1 ± 0.7
AWL	73.9 ± 1.9	87.9 ± 2.5	54.7 ± 2.9
RBS-GNN[$\equiv_2$]	80.3 ± 1.5	79.0 ± 1.9	44.0 ± 1.4

The result in Table 2 shows that at best we can achieve similar result for COLLAB while observing only 55.7% of the data. Note that this experiment is different from any random pooling or drop-out techniques because we use random balls for *both* train and test. The results for REDDIT datasets are also quite similar to the results of complete-observation models. As shown in Table 1, we only observe about 14% of the REDDIT original datasets. We selected such extreme example to show that although by a small observation, in some cases GNNs can still predict well. This observation implies that the true labeling function of these dataset is smooth in the random Benjamini-Schramm topology.

Size Generalization Our Theorem 11 identified size-generalizability with estimability. We verify this theoretical result experimentally as follows. For each dataset, we split into two sets of train and test data (0.5/0.5 split). Furthermore, the train set consists of smaller graphs while the test set has larger ones. We then report the test accuracy for targets being the global clustering coefficient (estimable) and the max-degree (non-estimable). In machine learning, it is often more beneficial to work with classification rather than regression; therefore, we categorized the clustering coefficients and the max-degree values into 5 classes. Each class represents a 20-percentile proportion of the data (see `supp/notebooks/preprocessing.ipynb`). The test accuracy is reported in Table 3.

Table 3: Test accuracy (in percentage) of RBS-GNN[$\equiv_2$] for the size-generalization task.

	Global Clustering			*Max-Degree*	
COLLAB	RDT-BINARY	RDT-MULTI5K	COLLAB	RDT-BINARY	RDT-MULTI5K
23.7 ± 3.7	33.7 ± 1.1	40.0 ± 2.0	16.1 ± 2.1	43.4 ± 1.2	39.8 ± 3.1