SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

April 2024, Vol. 67, Iss. 4, 142301:1–142301:16
https://doi.org/10.1007/s11432-023-3880-y

On the size generalizibility of graph neural networks for learning resource allocation

Jiajun WU, Chengjian SUN & Chenyang YANG*

School of Electronics and Information Engineering, Beihang University, Beijing 100191, *China*

Received 21 March 2023/Revised 13 July 2023/Accepted 16 October 2023/Published online 26 March 2024

Abstract Size generalization is important for learning resource allocation policies in wireless systems with time-varying scales. If a neural network for learning a wireless policy is not generalizable to the size of its input, it has to be re-trained whenever the system scale changes, which hinders its practical use due to the unaffordable training costs. Graph neural networks (GNNs) have been shown with size generalization ability empirically when optimizing resource allocation. Yet, are GNNs naturally size generalizable? In this paper, we argue that GNNs are not always size generalizable for resource allocation. We find that the aggregation and activation functions of the GNNs for learning a class of wireless policies play a key role in their size generalization ability. We take the GNN with the mean aggregator, called mean-GNN, as an example to reveal a size generalization condition. To demonstrate how to satisfy the condition, we learn power and bandwidth allocation policies for ultra-reliable low-latency communications and show that selecting or pre-training the activation function in the output layer of mean-GNN can make the GNN size generalizable. Simulation results validate our analysis and evaluate the performance of the learned policies.

Keywords size generalization, graph neural networks, resource allocation, permutation equivariance

1 Introduction

Deep learning has been widely applied for resource allocation [1–3] with diverse motivations, say enabling the real-time implementation of optimized solutions without closed-form expressions [1, 4] or solving problems without closed-form expressions of objective function or constraints [3]. Encouraged by the success of deep learning in a variety of applications, intelligent resource allocation has been envisioned as one of the most important features of the six-generation wireless communications [5].

Generalizability is a key performance metric of deep neural networks (DNNs), especially when learning the resource allocation policies in dynamic environments. For example, wireless channels and user populations may change quickly. When the number of users varies, the scale of a resource allocation problem changes accordingly. If a DNN (e.g., fully-connected neural network (FNN)) is not generalizable to the size of its input, then the DNN for learning a resource allocation policy has to be re-trained whenever the problem scale changes. Online learning allows a DNN to be adaptive to time-varying channels, which however does not enable adaption to time-varying problem scales if the DNN is not size generalizable. In order for an offline-trained DNN to perform well for a policy with different scales, a straightforward approach is to train a versatile DNN with samples of all possible sizes. However, the complexity of training over the dataset generated with numerous system configurations is prohibitive.

A promising way to enable generalization is embedding inductive biases into the structure of DNNs, e.g., imposing constraints on the functions represented by a DNN by harnessing prior knowledge before using data samples [6]. Appropriate inductive biases can reduce training complexity without performance loss [7] and allow the DNNs generalizable in a desirable way [8], while mismatched inductive biases may lead to performance degradation due to imposing too strong constraints on learning. A resource allocation policy can usually be represented as a multivariate function, which is a mapping from environmental parameters (say channels) into the allocated resources to wireless objects (say allocated powers to users). As a kind of prior knowledge of multivariate functions, permutation equivariance (PE) has been shown as a

* Corresponding author (email: cyyang@buaa.edu.cn)

widely existed property of wireless policies [2,9–11]. For a policy exhibiting the PE property, the mapping is not affected by the order of objects. Depending on the considered problem, a resource allocation policy may exhibit one-dimensional (1D)-PE, two-dimensional-PE (e.g., precoding [12]), joint-PE (e.g., power control or link scheduling [2, 9]) properties, their combinations [11]. Graph neural networks (GNNs) [7] and permutation equivariant neural networks [13] are two kinds of DNNs whose structures are embedded with PE inductive biases and hence can learn the PE policies efficiently. In addition to the PE property, GNNs also harness another kind of prior knowledge: the relation among vertices, unless a GNN learns a policy over a complete graph where all vertices are connected.

GNNs have been gaining increasing attention recently in the wireless communication community [2, 9, 11, 14–16]. It has been empirically observed that GNNs can be generalized to unseen graph sizes. In [2, 9], a GNN designed to learn a power control policy in an interference network was shown generalizable to 500 or 1000 transceivers when it is trained using samples with 50 transceivers. In [11], a GNN designed to learn a power allocation policy in the cellular network was shown generalizable to the numbers of users and cells. In [15, 16], the GNNs designed to learn a link scheduling policy and a joint channel and power allocation policy were shown with good generalization performance to the number of device-to-device transceiver pairs. Previous studies believe that GNNs are automatically size generalizable. As mentioned in [2, 9] without explanation, it is the PE inductive bias of GNNs that endows their size generalizability. As stated in [11], GNNs are size generalizable since the number of trainable parameters of a GNN is independent of the graph size, which is again owing to the PE inductive bias.

In [2, 17], size generalizability is also referred to as transferability to large graph sizes, which however differs from a similar notion: scalability, another challenge of intelligent resource allocation. According to the definition for generic networks in [18], an unscalable DNN adds to labour costs (e.g., structure re-designing, high complexity re-training) or harms the system performance. The scalability of a DNN is concerned with whether or not a DNN well-trained and tested in a small scale system can perform well when it is re-trained and tested in large scale systems with affordable training complexity. To evaluate the scalability, the DNN needs to be re-trained when the scale of a problem changes, but when evaluating the size generalizability, the DNN well-trained in certain problem scales is tested in unseen scales without re-training. In [2], it was argued that the PE property of the candidate functions representable by a GNN makes the GNN scalable and transferable (i.e., size generalizable). In [9], both scalability and size generalizability were mentioned, where the definition of scalability agrees with [18].

Despite the empirical success of GNNs in size generalizability, when and why GNNs are size generalizable are still mysterious. Are GNNs naturally size generalizable? Does the size generalizability attribute to the PE inductive bias? In [17, 19–22], few studies endeavour to analyze the size generalization ability of GNNs theoretically. Since the general study is extremely difficult, these studies focused on specific GNNs or particular problems, e.g., GNNs with sum- and mean-pooling for link prediction [19], GNN with sum-pooling for graph classification [20], spectral-GNN [17, 21], and GNNs with max-, min- and sum-pooling for the tasks solved by dynamic programming algorithms [22]. In this paper, we strive to answer the questions by providing counterexamples. To answer the second question, we separate the impact of introducing the PE inductive bias from exploiting the topology information on size generalization. To this end, we consider the policies with the 1D-PE property since they can be learned by GNNs over a complete graph, where the GNNs are with 1D-PE inductive bias [11]. In the sequel, we call a 1D-PE policy as a PE policy for short. To answer the first question, we resort to the decomposability of a PE policy when the problem scale is large, and find a condition for a GNN being generalizable to the size of a PE policy. To demonstrate how to satisfy the condition, we design a size generalizable GNN to learn a PE policy, by taking power and bandwidth allocation policies as an example. Since analyzing the size generalization mechanism of GNNs is challenging, we only consider specific GNNs with mean-, sum- and max-pooling as in [19, 20, 22]. The major contributions are summarized as follows.

• We provide a size generalization condition, which indicates that the size-scaling law of a size generalizable GNN should be aligned with the size-scaling law of a PE policy. We find that the PE inductive bias does not guarantee GNNs to be size generalizable, while appropriate aggregation and activation functions allow the size generalizability of the GNN for learning a PE policy. To the best of our knowledge, this is the first attempt to analyze the size generalization mechanism of GNNs for wireless communications.

• To demonstrate how to use our finding for designing size generalizable GNNs, we provide a method that uses the GNN with a mean aggregator (referred to as mean-GNN for short) and a selected or a pre-trained activation function in the output layer to satisfy the size generalization condition. We take the power and bandwidth allocation in ultra-reliable low-latency communications (URLLC) as an example

to show how to use the method for making the GNNs to learn the two policies with size generalizability.

The rest of the paper is organized as follows. In Section 2, we provide the size generalization condition. In Section 3, we show how to satisfy the condition by designing the activation function in the output layer of mean-GNNs for size-dependent policies. In Section 4, we evaluate the performance of the power and bandwidth allocation policies learned by the GNNs. In Section 5, we provide conclusions.

2 Size generalization condition

In this section, we first introduce the issue of size generalization for learning a resource allocation policy. Then, we show the asymptotic form of a PE policy and the functions learnable by the GNNs with mean-, max-, and sum-pooling with linear processors for aggregation. Finally, we show that mean-GNN with size-independent activation functions (SI-AFs) can be generalized to the size of the PE policy with asymptotic size-invariance property while the GNNs with max- and sum-pooling can not.

2.1 Learning a resource allocation policy with different sizes

A policy for K objects (say users) obtained from a resource allocation problem can usually be represented as a multivariate function, where the allocated resources (say transmit powers) are the output variables and the environmental parameters (say channels) are the input variables. Denote $\boldsymbol{x} \triangleq [x_1, \ldots, x_K]$, $\boldsymbol{y} \triangleq [y_1, \ldots, y_K]$, where x_k and y_k are respectively the input and output variables of the policy for the k-th object, $k = 1, \ldots, K$, and K reflects the scale of the resource allocation problem and is referred to as the size of corresponding policy.

In fact, K can also be regarded as an environmental parameter. Then, the resource allocation policy can be represented as $\boldsymbol{y} = \boldsymbol{F}'(\boldsymbol{x}, K) \triangleq [f'_1(\boldsymbol{x}, K), \ldots, f'_K(\boldsymbol{x}, K)]$[1)]. If we can learn the function $\boldsymbol{F}'(\boldsymbol{x}, K)$ with a DNN, then the DNN can be used for inference in the scenarios of different values of K without the need of re-training. Such a DNN can be obtained simply by training with the samples of $\boldsymbol{x}$ with all possible sizes, which however incurs unaffordable complexity for large scale systems.

2.2 Asymptotic form of a PE policy

If a resource allocation policy $\boldsymbol{F}'(\boldsymbol{x}, K)$ satisfies a PE property $\boldsymbol{\Pi y} = \boldsymbol{F}'(\boldsymbol{\Pi x}, K)$, i.e., it is permutation equivariant to $\boldsymbol{x}$, then the policy is a PE policy [10,11], where $\boldsymbol{\Pi}$ denotes arbitrary permutation matrix.

For a PE policy, the multivariate function degenerates into a scalar function for each object approximately when K is very large, i.e., $f'_k(\boldsymbol{x}, K) \approx f(x_k, K)$, as to be proved in the following. This suggests that $\boldsymbol{F}'(\boldsymbol{x}, K)$ can be learned by learning $f(x_k, K)$ and a size-independent function.

In [13], it was proved that permutation invariant functions can be decomposed into continuous outer and inner functions, similar to the Kolmogorov-Arnold representation theorem that is applicable for arbitrary multivariate continuous functions. By extending the proof in [13], it was proved in [23] that the k-th output variable of any PE policy can be expressed as

$$y_k = f'_k(\boldsymbol{x}, K) \triangleq \tilde{f}\left(x_k, \sum_{j \neq k} \phi(x_j)\right), \quad \forall k, \tag{1}$$

where $\phi(\cdot)$ and $\tilde{f}(\cdot)$ are continuous functions that are the same for every output variable.

The following proposition provides an approximation of (1) for independent and identically distributed (i.i.d.) input variables, which is accurate for a large number of objects.

Proposition 1. For a PE policy, if K is very large and $x_k, k = 1, \ldots, K$ are i.i.d., then

$$y_k \approx f(x_k, K), \tag{2}$$

where the expression of $f(x_k, K)$ is shown in Appendix A.

Proof. See Appendix A.

This indicates that a PE policy (which is a multi-object policy) asymptotically degenerates into a single-object policy that maps the input variable into the output variable of each object.

1) While K is fixed after user scheduling, it is a variable of the function $\boldsymbol{F}'(\boldsymbol{x}, K)$ for reflecting the dependence on K of the resource allocation policy, which may be a misuse of notation.

2.3 Impact of aggregator on the asymptotic form of the functions learned by a GNN

We first consider the mean-GNN for learning a PE policy with K objects, whose update equation is

$$\boldsymbol{h}_k^l = \sigma^l\left(\boldsymbol{U}^l\boldsymbol{h}_k^{l-1} + \left(\frac{1}{K}\sum_{j\neq k}\boldsymbol{V}^l\boldsymbol{h}_j^{l-1}\right) + \boldsymbol{c}^l\right),\ l=1,\ldots,L+1, \tag{3}$$

where $\sigma^l(\cdot)$ is the activation function in the l-th layer, $\boldsymbol{U}^l$, $\boldsymbol{V}^l$ and $\boldsymbol{c}^l$ are the trainable parameters of the l-th layer for ensuring the PE property of each layer, and L is the number of hidden layers. Denote the output of the GNN as $\hat{\boldsymbol{y}} \triangleq [\hat{y}_1,\ldots,\hat{y}_K]$, and the output of the l-th layer as $[\boldsymbol{h}_1^l,\ldots,\boldsymbol{h}_K^l]^{\mathrm{T}}$. The input feature of the GNN is the input variables of the policy, i.e., $\boldsymbol{h}_k^0 = x_k$, and the output of the GNN is the learned output variables, i.e., $\boldsymbol{h}_k^{L+1} = \hat{y}_k$.

The term $\frac{1}{K}\sum_{j\neq k}\boldsymbol{V}^l\boldsymbol{h}_j^{l-1}$ in (3) consists of mean-pooling and linear processing, which is called mean aggregation in previous studies. To impose the PE inductive biases on GNNs, the pooling function should satisfy commutative law, which can also use other operations such as summation and maximization. With such pooling functions and the shared trainable parameters in each layer among objects (i.e., $\boldsymbol{U}^l$, $\boldsymbol{V}^l$ and $\boldsymbol{c}^l$ are same for the update equation of $\boldsymbol{h}_k^l, k=1,\ldots,K$), the GNNs satisfy the PE property [11, 13].

The following proposition indicates that the k-th output of a mean-GNN well-trained for K objects only depends on x_k approximately when the input features $x_k, k=1,\ldots,K$ are i.i.d. and K is large.

Proposition 2. For the mean-GNN, if $x_k, k=1,\ldots,K$ are i.i.d. and K is very large, then

$$\hat{y}_k \approx \hat{q}(x_k, K), \tag{4}$$

where the expression of $\hat{q}(x_k, K)$ is shown in Appendix B and the trainable parameters within it are omitted for notational simplicity.

Proof. See Appendix B.

Denote the probability distribution function of the input features in $\boldsymbol{x}$ of size K as $p^K(x)$. The following proposition provides an asymptotic size-invariance property of the mean-GNN.

Proposition 3. For the mean-GNN, if $p^K(x) = p^{K'}(x)$ for any $K \neq K'$, when K and K' are very large and $\sigma^l(\cdot), l=1,\ldots,L+1$ do not depend on K (i.e., they are SI-AFs), the relation between its input and output of each object is approximately invariant to the input size, i.e.,

$$\hat{y}_k \approx \hat{q}(x_k, K) \approx \hat{q}(x_k, K') \triangleq \hat{q}(x_k). \tag{5}$$

Proof. See Appendix C.

The proposition means that the functions represented by the mean-GNN will not change with K if the input features of the GNN are with identical distribution for different input sizes meanwhile the activation function in each layer is independent from K. The most commonly used activation functions are SI-AFs, e.g., `Softplus`, `Sigmoid`, `Tanh`, `Relu` and its variants. One exception is `Softmax`.

However, if the mean-pooling in (3) is replaced by max- or sum-pooling (the corresponding aggregator is called max- or sum-aggregator in the sequel), then Eq. (4) will still hold but the GNN will no longer be size-invariant, as shown in Proposition 4.

Proposition 4. For the GNN with max- or sum-aggregator, if $x_k, k=1,\ldots,K$ are i.i.d. and K is very large, then $\hat{y}_k \approx \hat{q}(x_k, K)$. However, if $p^K(x) = p^{K'}(x)$ for any $K \neq K'$, K and K' are very large, and $\sigma^l(\cdot), l=1,\ldots,L+1$ are SI-AFs, then its input-output relation will not be invariant to its input size.

Proof. See Appendix D.

Propositions 2–4 indicate that the functions learnable by a GNN with mean-, sum-, or max-aggregators asymptotically degenerates into the functions for learning a single-object policy under i.i.d. input features, while the size-scaling law of a GNN with SI-AFs depend on the pooling function.

2.4 Size generalization condition

According to Propositions 1 and 2 and the universal approximation theorem [23], when the mean-GNN is well-trained for a PE policy with K objects, then $\hat{q}(x_k, K) \approx f(x_k, K)$ if the input features are i.i.d. and K is very large. Further considering Propositions 3 and 4, we can obtain Corollary 1.

Corollary 1. If a PE policy is asymptotically invariant to the size K, i.e.,

$$y_k \approx f(x_k, K) = f(x_k, K') \triangleq f(x_k),\ K \neq K', \tag{6}$$

then the mean-GNN with SI-AF in each layer trained for learning the PE policy with K objects can be generalized to the PE policy with K' objects, while the GNN with max- or sum-aggregator can not, where K and K' are very large.

To help understand the polices with the size-invariance property in (6), we consider a simple power allocation policy. Consider a downlink orthogonal frequency division multiple access (OFDMA) system, where a single-antenna base station (BS) serves K single-antenna users over bandwidth B. The power allocation is optimized to minimize the total transmit power under a rate constraint as follows:

$$\min_{P_k} \ \sum_{k=1}^{K} P_k \ \text{ s.t. } \ B\log\left(1+\frac{P_k g_k}{N_0 B}\right) \geqslant s_0, \quad P_k \geqslant 0, \tag{7}$$

where P_k is the power allocated to the k-th user, g_k is the channel gain of the user, N_0 is the single-side noise spectral density, and s_0 is the minimal data rate required by each user.

The problem is convex and its global optimal solution can be obtained from the Karush-Kuhn-Tucker (KKT) conditions as $P_k^* = \frac{N_0}{g_k} B(2^{\frac{s_0}{B}} - 1)$. The power allocation policy can be expressed as $\boldsymbol{p}^* = \boldsymbol{F}_1(\boldsymbol{g})$, where $\boldsymbol{p}^* = [P_1^*, \ldots, P_K^*]$ and $\boldsymbol{g} = [g_1, \ldots, g_K]$. As shown from the closed-form solution of P_k^*, the policy is independent of K. In other words, the size-scaling law of the policy is that it is invariant to all values of K. This is because the problem can be decoupled into multiple single-user problems, and thus the policy obtained from this problem can always be degenerated into a single-user policy.

Corollary 1 indicates that for a PE policy satisfying the size-invariance property in (6), the mean-GNN with SI-AFs can be generalized to input sizes. This implies a condition for a GNN to learn a PE policy with size generalization ability, which indicates that the PE inductive bias of a GNN does not guarantee the GNN to be size generalizable (recalling that the GNNs in this section satisfy the PE property).

Remark 1. Size generalization condition: a GNN can be generalized to the size of a PE policy if the size-scaling law of the GNN aligns with the size-scaling law of the PE policy.

Remark 2. In practice, the environment parameter x_k may not be i.i.d. among objects, e.g., the channel gains among users are not always i.i.d. in real-world systems. Nonetheless, it has been empirically observed that GNNs are size generalizable even when the i.i.d. assumption does not hold.

Asymptotic size-invariance is the simplest size-scaling law. In practice, however, the scaling law of a PE policy is usually not size-invariant.

3 Size generalization for PE policies without size-invariance property

In this section, we demonstrate how to satisfy the size generalization condition with mean-GNNs for learning the PE policies without the size-invariance property. We first consider a power allocation and bandwidth optimization problem in a single-antenna OFDMA system, where the power allocation policy is with closed-form expression and hence is unnecessary to be learned. We consider such an example for two reasons. One is to help understand Proposition 1 with a multi-user problem that cannot be decoupled into single-user problems. The other is to show that the size generalization condition for the power allocation policy can be satisfied by selecting an activation function for the output layer of mean-GNN with SI-AFs in hidden layers. Then, we consider a practical power and bandwidth allocation problem in URLLC, whose optimal solution cannot be obtained with closed-form under general settings. We proceed to show how to make the mean-GNN size generalizable when learning the policies. To learn the policies from both problems with GNNs, the graphs are complete graphs. Specifically, the users are vertices (whose features are their channel gains), and there are edges between users (that are without any feature) since the resource allocation to the users is coupled due to the total power constraint.

3.1 Power allocation and bandwidth optimization

We still consider the downlink OFDMA system, where a single-antenna BS serves K single-antenna users with maximal transmit power $P_{\max}$. Now we optimize the bandwidth of each user B together with power

allocation to minimize the total bandwidth (equally assigned to users) under both the minimal data rate constraint and total power constraint,

$$\min_{P_k,B} \quad KB \tag{8}$$

$$\text{s.t.} \quad s_k \triangleq B\log(1+P_k g_k/(N_0 B)) \geqslant s_0, \tag{8a}$$

$$\sum_{k=1}^{K} P_k \leqslant P_{\max}, P_k \geqslant 0, B \geqslant 0, \tag{8b}$$

where P_k, g_k, N_0 and s_0 have the same meaning as for the problem in (7). From the KKT conditions of this convex problem, the global optimal solution, denoted as P_k^* and B^*, satisfies $\sum_{k=1}^{K} P_k^* = P_{\max}$, and

$$P_k^* = F(B^*)/g_k, \tag{9}$$

$$F(B^*) \triangleq N_0 B^* \left(2^{\frac{s_0}{B^*}} - 1\right). \tag{10}$$

By substituting (9) into $\sum_{k=1}^{K} P_k^* = P_{\max}$, we have $F(B^*) = \frac{P_{\max}}{\sum_{k=1}^{K} \frac{1}{g_k}}$. By substituting this expression of $F(B^*)$ into (9), we can obtain the optimal power allocated to the k-th user, which can be re-written in the form of `Softmax` function as

$$P_k^* = P_{\max} \frac{\frac{1}{g_k}}{\sum_{k=1}^{K} \frac{1}{g_k}} = P_{\max} \frac{\mathrm{e}^{\ln \frac{1}{g_k}}}{\sum_{k=1}^{K} \mathrm{e}^{\ln \frac{1}{g_k}}}. \tag{11}$$

Denote $F^{-1}(\cdot)$ as the inverse function of $F(B)$ in (10). Then, the optimal bandwidth is re-written as

$$B^* = F^{-1}\left(\frac{P_{\max}}{\sum_{k=1}^{K} \frac{1}{g_k}}\right). \tag{12}$$

Denote the power allocation and bandwidth optimization policies obtained from (8) as $\boldsymbol{p}^* = \boldsymbol{F}_2'(\boldsymbol{g}, K)$ and $B^* = f_3'(\boldsymbol{g}, K)$, respectively, which are PE policies since the functions $\boldsymbol{F}_2'(\boldsymbol{g}, K)$ and $f_3'(\boldsymbol{g}, K)$ remain unchanged when the order of the users changes.

3.1.1 *Asymptotic policies*

Due to the total power constraint, the multi-user problem in (8) cannot be decoupled into single-user problems as the problem in (7). However, when K is very large and $g_k, k = 1, \ldots, K$ are i.i.d., we have $\frac{P_{\max}}{\sum_{k=1}^{K} \frac{1}{g_k}} = \frac{1}{K} \frac{P_{\max}}{(\sum_{k=1}^{K} \frac{1}{g_k})/K} \approx \frac{1}{K} \frac{P_{\max}}{\mathbb{E}(\frac{1}{g_k})}$. Then, the optimal solution can be approximated as

$$P_k^* \approx \frac{1}{K} \frac{P_{\max}}{g_k \mathbb{E}(\frac{1}{g_k})} \triangleq f_2(g_k, K), \quad B^* \approx F^{-1}\left(\frac{1}{K} \frac{P_{\max}}{\mathbb{E}(\frac{1}{g_k})}\right) \triangleq f_3(g_k, K), \tag{13}$$

i.e., the multi-user policy $\boldsymbol{F}_2'(\boldsymbol{g}, K)$ or $f_3'(\boldsymbol{g}, K)$ can be approximately degenerated into a single-user policy when K is large[2)], which agrees with Proposition 1. Besides, both policies are not size-invariant.

3.1.2 *Size generalization by selecting activation function*

In what follows, we show that a mean-GNN with proper activation function at the output layer can be generalized to different values of K when learning the optimal power allocation policy. Denote this GNN as $\mathcal{P}_P(\boldsymbol{g}; \boldsymbol{\theta}_P)$, where $\boldsymbol{\theta}_P$ consists of the trainable parameters. The output of this GNN is $\hat{\boldsymbol{p}} = [\hat{P}_1, \ldots, \hat{P}_K]$.

Since DNNs can be expressed as composite functions, we also express the power allocation policy as a composite of two functions.

From the first equality of (11), the policy can be expressed as a composite function of $f_{e1}(g_k) \triangleq \frac{1}{g_k}$ and a function for normalization (i.e., $\frac{z_k}{\sum_{k=1}^{K} z_k}$).

2) Simulations under Rayleigh channels show that the approximation is accurate for this policy when $K > 30$.

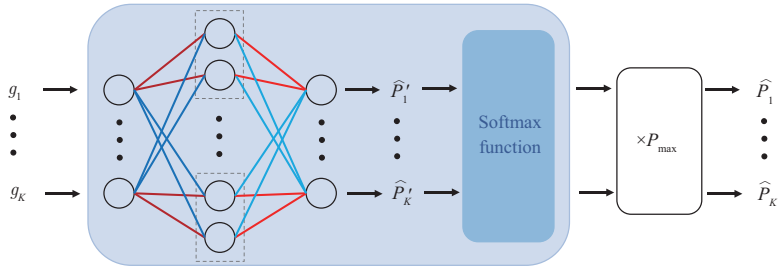

Figure 1 (Color online) Structure of $\mathcal{P}_P(\boldsymbol{g};\boldsymbol{\theta}_P)$. The connections with the same color are with the same weights (i.e., $\boldsymbol{U}^l$ and $\boldsymbol{V}^l$ in (3)).

From the second equality, the policy can also be expressed as a composite function of $f_{e2}(g_k) \triangleq \ln\frac{1}{g_k}$ and the `Softmax` function (i.e., $\sigma_s(z_k,\{z_j\}_{j\neq k}) \triangleq \frac{e^{z_k}}{\sum_{k=1}^K e^{z_k}}$) as

$$\begin{aligned}\boldsymbol{F}'_2(\boldsymbol{g},K) =& P_{\max}\cdot\left[\sigma_s\big(f_{e2}(g_1),\{f_{e2}(g_j)\}_{j\neq 1}\big),\ldots,\sigma_s\big(f_{e2}(g_K),\{f_{e2}(g_j)\}_{j\neq K}\big)\right]\\ \triangleq& P_{\max}\cdot\boldsymbol{\sigma}_s\left(f_{e2}(g_1),\ldots,f_{e2}(g_K)\right).\end{aligned}\tag{14}$$

For this PE policy, the inner function $f_{e1}(g_k)$ or $f_{e2}(g_k)$ is size-invariant, and the outer function $\frac{z_k}{\sum_{k=1}^K z_k}$ or $\sigma_s(\cdot)$ is size-dependent.

From the composite function form of $\boldsymbol{F}'_2(\boldsymbol{g},K)$ in (14), the neural network $\mathcal{P}_P(\boldsymbol{g};\boldsymbol{\theta}_P)$ should consist of hidden layers for learning an inner function (i.e., the mapping from g_k to $\hat{P}'_k$) and an output layer for learning the outer function (i.e., the mapping from $\hat{P}'_k$ to $\hat{P}_k$). When the `Softmax` function is used at the output layer[3)], the output of $\mathcal{P}_P(\boldsymbol{g};\boldsymbol{\theta}_P)$ can be expressed as

$$\begin{aligned}\hat{\boldsymbol{p}} &= P_{\max}\cdot\left[\frac{e^{\hat{P}'_1}}{\sum_{k=1}^K e^{\hat{P}'_k}},\ldots,\frac{e^{\hat{P}'_K}}{\sum_{k=1}^K e^{\hat{P}'_k}}\right]\\ &= P_{\max}\cdot\left[\sigma_s(\hat{P}'_1,\{\hat{P}'_j\}_{j\neq 1}),\ldots,\sigma_s(\hat{P}'_K,\{\hat{P}'_j\}_{j\neq K})\right] = P_{\max}\cdot\boldsymbol{\sigma}_s(\hat{P}'_1,\ldots,\hat{P}'_K),\end{aligned}\tag{15}$$

where $\hat{P}'_k \approx f_{e2}(g_k) = \ln\frac{1}{g_k}$. When K is very large, the k-th output of $\mathcal{P}_P(\boldsymbol{g};\boldsymbol{\theta}_P)$ becomes

$$\hat{P}_k = \frac{P_{\max}e^{\hat{P}'_k}}{\sum_{k=1}^K e^{\hat{P}'_k}} = \frac{1}{K}\frac{P_{\max}e^{\hat{P}'_k}}{\sum_{k=1}^K e^{\hat{P}'_k}/K} \approx \frac{1}{K}\frac{P_{\max}e^{\hat{P}'_k}}{\mathbb{E}(e^{\hat{P}'_k})},$$

which approximately scales with K in the same trend as P_k^* shown in (13).

The structure of $\mathcal{P}_P(\boldsymbol{g};\boldsymbol{\theta}_P)$ is shown in Figure 1, which learns a size-invariant inner function $f_{e2}(g_k)$ before the activation function in the output layer (i.e., $\hat{P}'_k \approx f_{e_2}(g_k)$) and then aligns with the size-scaling law of the power allocation policy (i.e., $1/K$) by `Softmax` function. After the `Softmax` function is multiplied by $P_{\max}$, the total power constraint in (8b) can be satisfied.

Since B^* has to be found from a transcendental equation in (10), $f'_3(\boldsymbol{g},K)$ is without closed-form and cannot be expressed as a composition of outer and inner functions explicitly. Hence, it cannot be learned by mean-GNN with size generalization ability via selecting an activation function or normalization. We show how to deal with this issue in Subsection 3.2.

3.2 Power and bandwidth allocation for URLLC

Consider a downlink multi-antenna OFDMA system supporting URLLC, where a BS equipped with N_t antennas serves K single-antenna users. The power and bandwidth allocation is optimized to minimize

3) From another composite function form of $\boldsymbol{F}'_3(\boldsymbol{g},K)$, $\mathcal{P}_P(\boldsymbol{g};\boldsymbol{\theta}_P)$ can also first learn a size-invariant inner function $f_{e1}(g_k) \triangleq \frac{1}{g_k}$ and then capture the trend of the power allocation policy with K by normalization. We choose `Softmax` function as an example, since it is often used as the activation function of the output layer of DNN.

the total bandwidth required for satisfying the quality of service (QoS) of every user, i.e.,

$$\min_{P_k,B_k} \quad \sum_{k=1}^{K} B_k \tag{16}$$

$$\text{s.t.} \quad C_k^E \geqslant S^E, \tag{16a}$$

$$\sum_{k=1}^{K} P_k \leqslant P_{\max}, \tag{16b}$$

$$P_k \geqslant 0, B_k \geqslant 0, \tag{16c}$$

where P_k and B_k are respectively the power and bandwidth allocated to the k-th user, C_k^E is the effective capacity of the k-th user depending on the queueing delay bound and delay bound violation probability [24], S^E is the effective bandwidth depending on the statistics of packet arrival [25], and $P_{\max}$ is the maximal transmit power. Eq. (16a) is the QoS constraint [26], and Eq. (16b) is the total power constraint.

Effective capacity reflects the system service ability for a user with QoS exponent θ. For the k-th user, it can be expressed as $C_k^E = -\frac{1}{\theta}\ln \mathbb{E}_{g_k}\{\mathrm{e}^{-\theta s_k}\}$ packets/frame that depends on α_k [26], and s_k is the achievable rate of the k-th user. To ensure the ultra-low transmission delay, we consider a short frame structure as in [26], where each frame is with duration T_f and consists of a duration τ for data transmission and a duration for signaling. Since the delay bound is typically shorter than coherence time, time diversity is not useful. To guarantee the transmission reliability within the delay bound, we assign each user with different subcarriers in adjacent frames where the frequency interval between adjacent subcarriers exceeds coherence bandwidth [26]. Then, the small scale channels of a user among frames are independent, and the achievable rate of the k-th user can be approximated as $s_k \approx \frac{\tau B_k}{\mu \ln 2}[\ln(1+\frac{\alpha_k P_k g_k}{N_0 B_k}) - \frac{Q_G^{-1}(\epsilon_k^c)}{\sqrt{\tau B_k}}]$ [27], where μ is the packet size, α_k and g_k are respectively the large-scale channel gain and two norm of the channel vector of the k-th user, ϵ_k^c is the decoding error probability, N_0 is the single-sided noise spectral density, and $Q_G^{-1}(\cdot)$ is the inverse of the Gaussian Q-function.

From the KKT conditions, the optimal solution of the problem in (16) should satisfy

$$\sum_{k=1}^{K} P_k^* = P_{\max}, \quad C_k^E = -\frac{1}{\theta}\ln \mathbb{E}_{g_k}\left\{\mathrm{e}^{-\theta s_k(g_k,\alpha_k,P_k^*,B_k^*)}\right\} = S^E, \tag{17}$$

where $s_k(g_k,\alpha_k,P_k^*,B_k^*)$ denotes the rate achieved with P_k^* and B_k^* given channels α_k and g_k.

Since C_k^E is hard to be derived with closed-form expression, the problem in (16) cannot be solved with traditional convex optimization tools. Considering that this problem is quasi-convex [28], the stochastic gradient descent (SGD) method can be used to find a solution of the problem for each given environment parameter (i.e., $\boldsymbol{\alpha} = [\alpha_1,\ldots,\alpha_K]$). However, the SGD method requires a large number of iterations for satisfying the stringent QoS constraint. To reduce the resulting high complexity, we resort to DNNs for learning the optimal power allocation policy $\boldsymbol{p}^* = \boldsymbol{P}'(\boldsymbol{\alpha},K)$ and the bandwidth allocation policy $\boldsymbol{B}^* = \boldsymbol{B}'(\boldsymbol{\alpha},K)$, where $\boldsymbol{p}^* = [P_1^*,\ldots,P_K^*]$, $\boldsymbol{B}^* = [B_1^*,\ldots,B_K^*]$. Since the multivariate functions $\boldsymbol{P}'(\boldsymbol{\alpha},K)$ and $\boldsymbol{B}'(\boldsymbol{\alpha},K)$ are not affected by the order of the users, both are PE policies.

The problem in (16) also cannot be decoupled into single-user problems. However, when K is very large and $\alpha_k, k=1,\ldots,K$ are i.i.d., $\boldsymbol{P}'(\boldsymbol{\alpha},K)$ and $\boldsymbol{B}'(\boldsymbol{\alpha},K)$ can be approximately degenerated into single-user policies. Due to the total power constraint in (16b), P_k^* decreases with K. As a result, B_k^* should increase with K to ensure the QoS in (16a). This indicates that the two policies are not size-invariant.

We use mean-GNNs for learning $\boldsymbol{P}'(\boldsymbol{\alpha},K)$ and $\boldsymbol{B}'(\boldsymbol{\alpha},K)$, which are denoted as $\mathcal{P}_P(\boldsymbol{\alpha};\boldsymbol{\theta}_P)$ and $\mathcal{P}_B(\boldsymbol{\alpha};\boldsymbol{\theta}_B)$, respectively, where $\boldsymbol{\theta}_P$ and $\boldsymbol{\theta}_B$ are the trainable parameters. The input of both GNNs is $\boldsymbol{\alpha}$. The output of $\mathcal{P}_P(\boldsymbol{\alpha};\boldsymbol{\theta}_P)$ is $[\hat{P}_1,\ldots,\hat{P}_K]$, and the output of $\mathcal{P}_B(\boldsymbol{\alpha};\boldsymbol{\theta}_B)$ is $[\hat{B}_1,\ldots,\hat{B}_K]$.

3.2.1 *Size generalization by pre-training "activation function"*

$\boldsymbol{P}'(\boldsymbol{\alpha},K)$ is not with closed-form as in (13) and cannot be expressed as a composite function explicitly as in (14). Nonetheless, it is also not size-invariant due to the same reason as $\boldsymbol{F}_2'(\boldsymbol{g},K)$: the total power constraint. According to the analysis in Subsection 3.1.2, using `Softmax` as the activation function of the

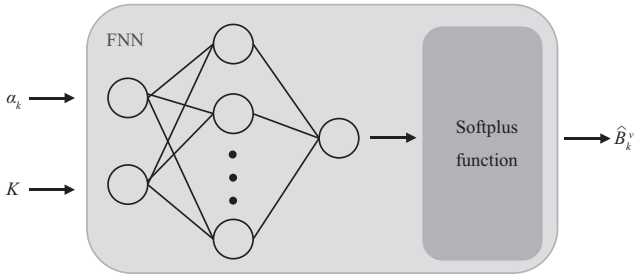

Figure 2 Structure of $\mathcal{B}^v(\alpha_k, K; \boldsymbol{\theta}_v)$ for learning $B^v(\alpha_k, K)$. `Softplus` is used at the output layer to ensure $\hat{B}_k^v > 0$.

output layer of $\mathcal{P}_P(\boldsymbol{\alpha}; \boldsymbol{\theta}_P)$, which is a natural choice to satisfy the power constraint in (16b), can make the mean-GNN generalizable to K.

$\boldsymbol{B}'(\boldsymbol{\alpha}, K)$ is also without closed-form and cannot be expressed as a composite function explicitly. It is not size-invariant due to the QoS constraint and its dependence on P_k^*. Based on a conjecture that $\boldsymbol{B}'(\boldsymbol{\alpha}, K)$ is a composite function of an inner function independent of K and an outer function dependent on K, and inspired by the structure of $\mathcal{P}_P(\boldsymbol{g}; \boldsymbol{\theta}_P)$ in Figure 1, we conceive the following approach to enable a mean-GNN generalizable to K for learning $\boldsymbol{B}'(\boldsymbol{\alpha}, K)$. We first pre-train an "activation function" to learn the implicit outer function that depends on K. Then, we use a mean-GNN with SI-AFs in hidden layers to learn a size-invariant inner function and use the pre-trained "activation function" in the output layer of the mean-GNN to align with the size-scaling law of the bandwidth allocation policy.

Pre-training "activation function". Given a power allocation policy, the PE policy $\boldsymbol{B}'(\boldsymbol{\alpha}, K)$ can be degenerated into a function of α_k and K when K is large as shown in (2), i.e., $B_k^* \approx B^v(\alpha_k, K)$. This function reflects the scaling law of bandwidth allocation policy with K and is identical for all users, which can approximate the implicit outer function of the PE policy in the non-asymptotic regime.

In order to learn the function $B^v(\alpha_k, K)$ defined over all possible values of α_k and K, we introduce a FNN denoted as $\mathcal{B}^v(\alpha_k, K; \boldsymbol{\theta}_v)$, where $\boldsymbol{\theta}_v$ consists of trainable parameters. The structure of $\mathcal{B}^v(\alpha_k, K; \boldsymbol{\theta}_v)$ is shown in Figure 2, where the input is $[\alpha_k, K]$, and the output is $\hat{B}_k^v$.

To reduce the cost of training, we pre-train $\mathcal{B}^v(\alpha_k, K; \boldsymbol{\theta}_v)$ and assume equal power allocation during the pre-training. The samples for α_k are generated from random realizations of large scale channel and the samples for K are all possible numbers of users in the considered system. The FNN can be pre-trained in a supervised manner, where the trainable parameters are found from

$$\boldsymbol{\theta}_v = \arg\min_{\boldsymbol{\theta}_v} \mathbb{E}_{\alpha_k, K} \left\{ \left(\hat{B}^v(\alpha_k, K; \boldsymbol{\theta}_v) - B^{v*} \right)^2 \right\} \tag{18}$$

by estimating the expectation as empirical mean from samples, $\hat{B}^v(\alpha_k, K; \boldsymbol{\theta}_v)$ is the output of the FNN, and B^{v*} is the label obtained by numerically solving the equation in (17) with $P_k^* = \frac{P_{\max}}{K}$. The FNN can also be pre-trained in an unsupervised manner, where the trainable parameters are found from

$$\boldsymbol{\theta}_v = \arg\min_{\boldsymbol{\theta}_v} \mathbb{E}_{\alpha_k, K} \left\{ \left(S^E + \frac{1}{\theta} \ln \mathbb{E}_{g_k} \left\{ \mathrm{e}^{-\theta s_k(g_k, \alpha_k, \frac{P_{\max}}{K}, \hat{B}^v(\alpha_k, K; \boldsymbol{\theta}_v))} \right\} \right)^2 \right\}, \tag{19}$$

where the loss function is obtained from the second KKT condition in (17), since the optimally allocated bandwidth should satisfy the condition.

Learning bandwidth allocation policy. The structure of $\mathcal{P}_B(\boldsymbol{\alpha}; \boldsymbol{\theta}_B)$ for learning $\boldsymbol{B}'(\boldsymbol{\alpha}, K)$ with size generalization ability is shown in Figure 3. It consists of a mean-GNN with SI-AFs in hidden layers for learning the size-invariant inner function (i.e., the mapping from $\alpha_k, k = 1, \ldots, K$ to $\hat{B}'_k, k = 1, \ldots, K$) and the pre-trained FNN in the output layer for learning the outer function (i.e., the mapping from $\hat{B}'_k$ to $\hat{B}_k$), where the learned bandwidth allocated to the k-th user is $\hat{B}_k = \hat{B}'_k \cdot \hat{B}_k^v$. To ensure $\hat{B}_k \geqslant 0$, `Softplus` is used at the last hidden layer of the mean-GNN.

3.2.2 *Unsupervised learning for the policies*

Since the stringent QoS constraints in URLLC is hard to satisfy by supervised learning, we train the two GNNs for learning the power and bandwidth allocation policies in an unsupervised manner, where a GNN for learning the Lagrangian multiplier function for controlling the QoS constraint (denoted as

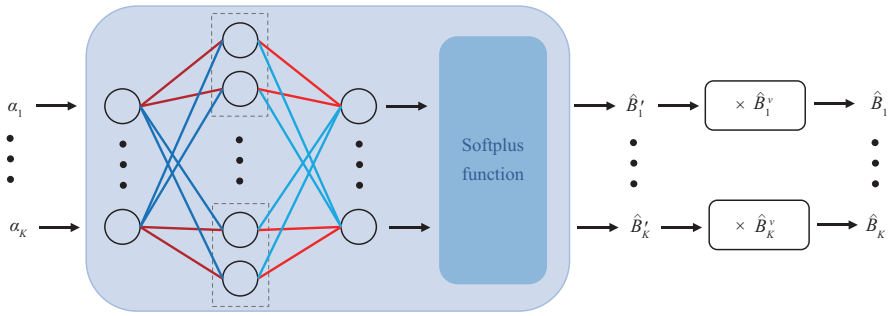

Figure 3 (Color online) Structure of $\mathcal{P}_B(\boldsymbol{\alpha};\boldsymbol{\theta}_B)$ for learning $\boldsymbol{B}'(\boldsymbol{\alpha},K)$. The connections with the same color are with the same weights.

Algorithm 1 Training procedure

1: Initialize $\mathcal{P}_P(\boldsymbol{\alpha};\boldsymbol{\theta}_P)$, $\mathcal{P}_B(\boldsymbol{\alpha};\boldsymbol{\theta}_B)$ and $\mathcal{P}_\lambda(\boldsymbol{\alpha};\boldsymbol{\theta}_\lambda)$ with random parameters $\boldsymbol{\theta}_P$, $\boldsymbol{\theta}_B$ and $\boldsymbol{\theta}_\lambda$;
2: **for** $t=1,2,\ldots$ **do**
3: Sample a batch of $(\boldsymbol{g},\boldsymbol{\alpha})$ as $\mathcal{S}$;
4: Input $\boldsymbol{\alpha}$ into $\mathcal{P}_P(\boldsymbol{\alpha};\boldsymbol{\theta}_P)$, $\mathcal{P}_B(\boldsymbol{\alpha};\boldsymbol{\theta}_B)$ and $\mathcal{P}_\lambda(\boldsymbol{\alpha};\boldsymbol{\theta}_\lambda)$ to obtain $\hat{P}_k$, $\hat{B}'_k$ and $\hat{\lambda}_k$, respectively;
5: **for** $k=1,\ldots,K$ **do**
6: Input $[\alpha_k,K]$ into the pre-trained $\mathcal{B}^v(\alpha_k,K;\boldsymbol{\theta}_v)$ to obtain $\hat{B}^v_k$;
7: Scale $\hat{B}'_k$ with $\hat{B}^v_k$ to obtain $\hat{B}_k$, i.e., $\hat{B}_k=\hat{B}'_k\cdot\hat{B}^v_k$;
8: **end for**
9: Update the parameters $\boldsymbol{\theta}_P$, $\boldsymbol{\theta}_B$ and $\boldsymbol{\theta}_\lambda$ as follows:

$$\boldsymbol{\theta}_P^{t+1}\leftarrow\boldsymbol{\theta}_P^t-\frac{\delta_t}{|\mathcal{S}|}\sum_{(\boldsymbol{g},\boldsymbol{\alpha})\in\mathcal{S}}\nabla_{\boldsymbol{\theta}_P}\mathcal{P}_P(\boldsymbol{\alpha};\boldsymbol{\theta}_P)\frac{\partial L^t(\boldsymbol{g},\boldsymbol{\alpha},\hat{P}_k,\hat{B}_k,\hat{\lambda}_k)}{\partial\hat{P}_k},$$

$$\boldsymbol{\theta}_B^{t+1}\leftarrow\boldsymbol{\theta}_B^t-\frac{\delta_t}{|\mathcal{S}|}\sum_{(\boldsymbol{g},\boldsymbol{\alpha})\in\mathcal{S}}\nabla_{\boldsymbol{\theta}_B}\mathcal{P}_B(\boldsymbol{\alpha};\boldsymbol{\theta}_B)\frac{\partial L^t(\boldsymbol{g},\boldsymbol{\alpha},\hat{P}_k,\hat{B}_k,\hat{\lambda}_k)}{\partial\hat{B}_k}\hat{B}^v_k,$$

$$\boldsymbol{\theta}_\lambda^{t+1}\leftarrow\boldsymbol{\theta}_\lambda^t+\frac{\delta_t}{|\mathcal{S}|}\sum_{(\boldsymbol{g},\boldsymbol{\alpha})\in\mathcal{S}}\nabla_{\boldsymbol{\theta}_\lambda}\mathcal{P}_\lambda(\boldsymbol{\alpha};\boldsymbol{\theta}_\lambda)\frac{\partial L^t(\boldsymbol{g},\boldsymbol{\alpha},\hat{P}_k,\hat{B}_k,\hat{\lambda}_k)}{\partial\hat{\lambda}_k};$$

10: **end for**

$\mathcal{P}_\lambda(\boldsymbol{\alpha};\boldsymbol{\theta}_\lambda)$) is also trained [26].

The joint training procedure for $\mathcal{P}_P(\boldsymbol{\alpha};\boldsymbol{\theta}_P)$, $\mathcal{P}_B(\boldsymbol{\alpha};\boldsymbol{\theta}_B)$ and $\mathcal{P}_\lambda(\boldsymbol{\alpha};\boldsymbol{\theta}_\lambda)$ is provided in Algorithm 1, where t is the number of iterations, $\boldsymbol{g}\triangleq[g_1,\ldots,g_K]$, δ_t is the learning rate, $|\cdot|$ denotes cardinality, $\nabla_{\boldsymbol{x}}\boldsymbol{y}=[\nabla_{\boldsymbol{x}}y_1,\ldots,\nabla_{\boldsymbol{x}}y_K]$, $\nabla_{\boldsymbol{x}}y_i=[\frac{\partial y_i}{\partial x_1},\ldots,\frac{\partial y_i}{\partial x_K}]^{\mathrm{T}}$, and $L^t(\boldsymbol{g},\boldsymbol{\alpha},\hat{P}_k,\hat{B}_k,\hat{\lambda}_k)\triangleq\sum_{k=1}^K[\hat{B}_k+\hat{\lambda}_k(\mathrm{e}^{-\theta s_k}-\mathrm{e}^{-\theta S^E})]$.

4 Simulation results

In this section, we validate our analyses by evaluating the size generalization performance of the DNNs for learning the power and bandwidth allocation policies in URLLC. To this end, the DNNs are trained using the samples generated in the scenario with a given number of users, but tested using the samples in the scenarios with other numbers of users without re-training.

4.1 Simulation setup

The cell radius is 250 m. All users are randomly located in a road with a minimal distance of 50 m from the BS. The path loss model is $35.3+37.6\log(d_k)$, where d_k is the distance between the k-th user and the BS. $P_{\max}=43$ dBm, $N_t=8$, and $N_0=-173$ dbm/Hz. The required overall packet loss probability and the end-to-end delay bound are respectively $\epsilon_{\max}=10^{-5}$ and $D_{\max}=1$ ms. After subtracting the downlink transmission delay and the decoding delay from the delay bound, the queueing delay is bounded by 0.8 ms [26]. As in [29], the decoding error probability ϵ_k^c and the queueing delay violation probability ϵ_k^q are set as $\epsilon_k^c=\epsilon_k^q=\epsilon_{\max}/2$. The time for data transmission and the frame duration are $\tau=0.05$ ms and $T_f=0.1$ ms, respectively. The packet size is $\mu=20$ bytes [30]. The packets arrive at the BS according to the Poisson process with an average arrival rate of $a=0.2$ packets/frame. Then, $S^E=\frac{a}{\theta}(\mathrm{e}^\theta-1)$ packets/frame and $\theta=\ln[1-\frac{\ln(\epsilon_{\max}/2)}{aD_{\max}/T_f}]$ [29].

Table 1 Generating samples in training, validation and test sets

	Number of users K in samples	Number of samples generated for each K	Total number of samples
Training set	[10]	2000	2000
Validation set	[10]	200	200
Test set	[1,2,5,10,50,100,200]	100	700

Table 2 Hyper-parameters

Hyper-parameter	GNN			FNN $\mathcal{B}^v$
	$\mathcal{P}_P$	$\mathcal{P}_B$	$\mathcal{P}_\lambda$	
Number of neurons in hidden layers		$[4K]$		$[200, 100, 100, 50]$
Activation function of the hidden layers		`Leaky ReLU` (i.e., $y = x, x \geqslant 0$, $y = 0.01x$, $x < 0$)		
Activation function of the output layer	`Softmax`	`Softplus`	`Softplus`	`Softplus`
Learning rate		$\frac{0.01}{1+0.01t}$		$\frac{0.01}{1+0.001t}$
Batch size		10		100
Epochs		5000		2500

4.2 Sample generation and fine-tuned hyper-parameters

For notational simplicity, denote the three GNNs as $\mathcal{P}_P(\boldsymbol{\alpha};\boldsymbol{\theta}_P) \triangleq \mathcal{P}_P$, $\mathcal{P}_B(\boldsymbol{\alpha};\boldsymbol{\theta}_B) \triangleq \mathcal{P}_B$ $\mathcal{P}_\lambda(\boldsymbol{\alpha};\boldsymbol{\theta}_\lambda) \triangleq \mathcal{P}_\lambda$ and the FNN as $\mathcal{B}^v(\alpha_k, K;\boldsymbol{\theta}_v) \triangleq \mathcal{B}^v$ in the sequel.

Since the loss function in (18) is simpler than (19) and the label for pre-training $\mathcal{B}^v$ can be easily obtained by solving (17) with the bi-section method, we use supervised learning to pre-train this FNN. Denote one sample for pre-training $\mathcal{B}^v$ as $[(\alpha_k, K), B^{v*}]$, where α_k is obtained from the path loss model with randomly generated locations, and B^{v*} is the label. We generate 4000 samples to pre-train the FNN, where 20 samples are generated for every K from 1 to 200.

Denote one sample for training $\mathcal{P}_P$, $\mathcal{P}_B$, and $\mathcal{P}_\lambda$ as $[\boldsymbol{g}, \boldsymbol{\alpha}]$, which is a realization of the small and large-scale channel gains of all users. In each realization, $\boldsymbol{g}$ is randomly generated from Rayleigh distribution, and $\boldsymbol{\alpha}$ is obtained from the path loss model with randomly generated locations of all users. As shown in Table 1, 2000 samples are generated for $K = 10$ (i.e., all these samples for the vectors $\boldsymbol{g}$ and $\boldsymbol{\alpha}$ are of size 10) when generating training set, 200 samples are generated for $K = 10$ when generating validation set, and 100 samples are generated for every K in $[1, 2, 5, 10, 50, 100, 200]$ (i.e., 100 samples for $\boldsymbol{g}$ and $\boldsymbol{\alpha}$ are scalars, 100 samples for $\boldsymbol{g}$ and $\boldsymbol{\alpha}$ are of size two, 100 samples for the vectors are of size five, etc.) when generating test set. Therefore, the total number of samples in the training, validation, and test sets are 2000, 200, and 700, respectively.

The three datasets in Table 1 are used as follows unless otherwise specified.

The fine-tuned hyper-parameters are shown in Table 2, where $[*_1, *_2, \ldots, *_L]$ denotes that a DNN is with $*_l$ neurons in the l-th hidden layer. To accelerate the training procedure, we scale the large-scale channel gains as $\ln(\boldsymbol{\alpha})+30$ since they are usually small, and scale K as $K/200$ to make the input variables in the same order of magnitude when pre-training $\mathcal{B}^v$.

4.3 Size generalization performance

The system performance is measured by the availability and total bandwidth achieved by $[\hat{P}_1, \ldots, \hat{P}_K]$ and $[\hat{B}_1, \ldots, \hat{B}_K]$. Availability is the percentage of the users with satisfied QoS among all the users in the system, which is one of the key performance metrics for URLLC [31], and total bandwidth is the sum of the bandwidth of all users achieved by a learned policy. The two metrics are respectively defined as $A_K \triangleq \frac{1}{K|\mathcal{N}_K|}\sum_{(\boldsymbol{g},\boldsymbol{\alpha})\in\mathcal{N}_K}\sum_{k=1}^{K} I(\epsilon_k < \epsilon_{\max})$, $W_K^r \triangleq \frac{1}{|\mathcal{N}_K|}\sum_{(\boldsymbol{g},\boldsymbol{\alpha})\in\mathcal{N}_K}\sum_{k=1}^{K}\hat{B}_k$ MHz, where $\mathcal{N}_K$ is a subset in the test set containing the samples for the channel vectors with the same size, and ϵ_k is the packet loss probability of the k-th user achieved by a learned policy.

Considering the randomness of small training sets, we train the GNNs 10 times. In each time, the training set is randomly generated and the GNNs are with randomly initialized trainable parameters while the test set remains unchanged. The values of the two metrics are obtained by selecting the second worst test result, hence are with confidence level of 90%.

4.3.1 *Reliability controlling*

Despite that the learned solution is unable to satisfy the QoS with probability one, the packet loss probability can be ensured by setting a more stringent requirement for reliability than the required reliability $\epsilon_{\max}$ during training (i.e., $\epsilon_D < \epsilon_{\max}$), as suggested in [26]. With such a conservative design,

Table 3 Impact of ϵ_D on system performance[a)]

ϵ_D	$K=1$		$K=2$		$K=5$		$K=10$		$K=50$		$K=100$		$K=200$	
	A_K	W_K^r	A_K	W_K^r	A_K	W_K^r	A_K	W_K^r	A_K	W_K^r	A_K	W_K^r	A_K	W_K^r
10^{-5}	0.57	0.13	0.26	0.27	0.32	0.77	0.44	1.66	0.42	10.29	0.41	22.95	0.45	51.81
8×10^{-6}	1.00	0.14	0.81	0.28	0.84	0.78	0.91	1.68	0.77	10.38	0.73	23.15	0.63	52.27
6×10^{-6}	**1.00**	**0.14**	**1.00**	**0.28**	**1.00**	**0.79**	**1.00**	**1.70**	**1.00**	**10.55**	**1.00**	**23.53**	**1.00**	**53.14**
5×10^{-6}	1.00	0.14	1.00	0.28	1.00	0.80	1.00	1.72	1.00	10.64	1.00	23.74	1.00	53.60

a) The bold numbers denote the conservative design that ensures availability of 1.00 in all test scenarios with the minimum increasing total bandwidth, and the corresponding system performance under this design.

Table 4 Training complexity for achieving the availability of 1.00, $\epsilon_D = 6\times10^{-6}$

Size K of training samples	Number of training samples	Training time (s)
[2]	5000	1290.32
[5]	4000	1289.91
[8]	3000	1185.11
[10]	2000	875.43

even if the overall packet loss probability achieved by the learned solution exceeds ϵ_D, the probability of violating the original requirement $\epsilon_{\max}$ is low (e.g., the availability is 1.00 for $\epsilon_D = 6\times10^{-6}$ as shown in Table 3). As shown by the simulation results in Table 3, the availability can be remarkably improved with a slight increase of the total bandwidth by reducing ϵ_D. For $\epsilon_D = 6\times10^{-6}$, the availability achieves 1.00 with a bandwidth loss $\frac{53.14-51.81}{51.81} \approx 2.57\%$ from $\epsilon_D = 10^{-5}$ when $K = 200$ in the test set. The results in the table indicate that the required reliability $\epsilon_{\max}$ can be ensured when the well-trained GNNs with $K = 10$ in the training set (see Table 1) are tested in the scenarios with different number of users (i.e., $K = 1, 2, 5, 50, 100$ or 200) from the training set.

4.3.2 *Impact of small K*

In previous analytical analysis, K is assumed very large. Here we show that the designed GNNs can still be applied to a small number of users. In Table 4, we show the minimal number of samples and the time required to train the mean-GNNs with pre-trained activation function in the output layer when the training samples are with different sizes for achieving the availability of 1.00 in all test scenarios, where the test samples are generated with the size of $1, 2, 5, 10, 50, 100$ or 200 in each scenario. It is shown that the policies learned by the mean-GNNs can achieve good generalization performance for large size (e.g., $K = 200$) even when it is trained with samples of small size (e.g., $K = 2$). However, the training complexity increases when K is small, because the output of the pre-trained FNN, i.e., $\mathcal{B}_v$, is not an accurate approximation of B_k^*, which needs more training samples to make $\mathcal{P}_B$ to well learn the policy.

4.3.3 *Performance comparison*

To validate our analyses, we compare three GNNs with the PE inductive bias and an FNN that is not with PE inductive bias, where all DNNs are with fine-tuned hyper-parameters.

• "P-GNN". This legend denotes the mean-GNNs satisfying the size generalization condition, where $\mathcal{P}_P$ with `Softmax` in the output layer is used to learn power allocation and $\mathcal{P}_B$ with the pre-trained "activation function" in the output layer is used to learn bandwidth allocation, i.e., $\hat{B}_k = \hat{B}'_k\hat{B}^v_k$.

• "M-GNN". This legend denotes the existing mean-GNNs, where $\mathcal{P}_P$ with `Softmax` in the output layer is used to learn the power allocation policy (as in "P-GNN"), and a mean-GNN with SI-AFs in all layers (which is $\mathcal{P}_B$ without pre-trained "activation function" at the output layer and does not satisfy the size generalization condition) is used to learn the bandwidth allocation policy, i.e., $\hat{B}_k = \hat{B}'_k$.

• "S-GNN". This legend denotes the GNNs with sum-aggregator that does not satisfy the size generalization condition, where $\mathcal{P}_P$ with `Softmax` in output layer is used to learn power allocation and $\mathcal{P}_B$ with the pre-trained "activation function" in the output layer is used to learn bandwidth allocation.

• "FNN". The power and bandwidth allocation policies are learned by two FNNs. Each FNN is with one hidden layer consisting of 300 neurons. Since the input of an FNN can only be with a fixed size, both the input size and output size of each of the two FNNs are $K_{\max}$. When the training or test samples are generated with $K < K_{\max}$, the input vector is padded with zeros. The learning rate is $\frac{0.001}{1+0.001t}$, the batch size is 100 and the other hyper-parameters are the same as those of the GNNs in Table 2.

For "FNN", we use the samples with $K_{\max} = 200$ for training such that it can also be used for the test samples with $K < 200$ by padding with zeros. For "M-GNN", we respectively use the training samples of sizes $K = 10$ and $K = 200$, and use "M-GNN^K" to denote the M-GNN trained with the samples

Table 5 Availability and total bandwidth of the learned policies in the test set[a)]

Method	$K=1$		$K=2$		$K=5$		$K=10$		$K=50$		$K=100$		$K=200$	
	A_K	W_K^r	A_K	W_K^r	A_K	W_K^r	A_K	W_K^r	A_K	W_K^r	A_K	W_K^r	A_K	W_K^r
P-GNN	1.00	0.14	1.00	0.28	1.00	0.79	1.00	1.70	1.00	10.55	1.00	23.53	1.00	53.14
M-GNN^{10}	1.00	0.17	1.00	0.33	1.00	0.86	1.00	**1.72**	0.00	8.65	0.00	17.29	0.00	34.50
M-GNN^{200}	1.00	0.27	1.00	0.52	1.00	1.35	1.00	2.70	1.00	13.61	1.00	27.20	1.00	**54.22**
S-GNN	0.83	0.12	0.65	0.22	0.77	0.72	1.00	**1.71**	1.00	15.06	1.00	72.24	1.00	557.34
FNN	1.00	0.42	1.00	0.78	1.00	2.02	0.98	3.78	0.91	17.16	0.88	31.04	1.00	**55.19**

a) These bold numbers respectively denote the total bandwidth achieved by the learned policy from M-GNN, S-GNN and FNN in scenarios where their system performance are close to that of P-GNN.

generated in the scenario of K users. Since $\epsilon_D = 6 \times 10^{-6}$ is unable to control the reliability achieved by the policies learned by "M-GNN" and "FNN", the values of ϵ_D chosen for "M-GNN^{10}", "M-GNN^{200}", and "FNN" are 5×10^{-6}, 3×10^{-6}, and 2×10^{-6}, respectively.

The system performance of the learned policies tested with the samples respectively generated in the scenarios with $K = 1$, 2, 5, 10, 50, 100, and 200 is shown in Table 5. It is shown that when $K = 10$ and $K = 200$ in the test set where "M-GNN^{10}" and "M-GNN^{200}" do not need to be generalized, their availability is 1.00 and the required total bandwidth (highlighted in boldface) is close to "P-GNN". The availability of "M-GNN^{10}" is 0.00 in the test scenarios where $K > 10$, i.e., the QoS of all users are not satisfied. Although the availability of "M-GNN^{10}" is 1.00 in the test scenarios where $K < 10$, the bandwidth it required is much larger than that of "P-GNN". For the test scenarios where $K < 200$, although the availability of "M-GNN^{200}" is 1.00, the bandwidth it required is much larger than that of "P-GNN". This is because "M-GNN" cannot learn the scaling law of B_k^* with K. When the well-trained "M-GNN^{10}" is used for inference in the test scenarios where $K > 10$, the QoS cannot be satisfied since $\hat{B}_k$ is lower than required. When the well-trained "M-GNN^{10}" is used for inference in the test scenarios where $K < 10$ or the well-trained "M-GNN^{200}" is used for inference when $K < 200$, much larger total bandwidth is required than "P-GNN". The well-trained "S-GNN" only performs close to "P-GNN" when $K = 10$ in the test set, but is either with lower availability or much larger total bandwidth than "P-GNN" for other values of K. This indicates that "S-GNN" is not size generalizable, even when `Softmax` or the pre-trained "activation functions" is used at the output layer, which validates Proposition 4. Similarly, the well-trained "FNN" only performs close to "P-GNN" when $K = 200$ in the test set. These results show that "P-GNN" is size generalizable, which validates the conjecture that the bandwidth allocation policy can indeed (at least approximately) be decomposed into a size-invariant inner function and a size-dependent outer function. As two counterexamples, the results for "M-GNN" and "S-GNN" indicate that the PE inductive bias of a GNN does not ensure the GNN to be size generalizable.

Remark 3. We have also evaluated the scalability of the four DNNs, according to the definition of scalability in [15,18]. To this end, we consider four scenarios each with $K = 10, 50, 100, 200$, respectively, where the DNNs are first re-trained (with re-tuned hyper-parameters) and then tested in each scenario. Our simulation results show that "P-GNN", "M-GNN" and "S-GNN" achieve good system performance with slightly higher training complexity while "FNN" is with high complexity for training to achieve comparable performance, when K is large (not provided here due to the lack of space). In other words, "P-GNN", "M-GNN" and "S-GNN" are scalable while "FNN" is not scalable in terms of training complexity. These counterexamples show that a scalable GNN may not be size generalizable, e.g., "M-GNN" and "S-GNN" are scalable but not size generalizable.

Remark 4. We also evaluated the size generalizability of "P-GNN" in the scenarios where $\alpha_k, k = 1, \ldots, K$ are not i.i.d. by setting the users randomly located in two roads. The simulation results are the same as in Table 5 (not provided due to the space limitation). This indicates that even when the i.i.d. assumption in the propositions does not hold, the proposed method is still applicable.

5 Conclusion

In this paper, we showed that GNNs are not naturally size generalizable and solely embedded with PE inductive bias does not ensure a GNN size generalizable for PE policies. We found a size generalization condition and demonstrated that aggregators and activation functions impose an inductive bias on the size generalizability of GNNs for learning PE policies. To showcase how to make GNNs generalizable to the sizes by satisfying the condition, we took power and bandwidth allocation as examples. For the power allocation policy, we showed that it can be learned by the mean-GNN with size generalization ability

when the activation function at the output layer is selected for satisfying the total power constraint, which happens to satisfy the size generalization condition. For the bandwidth allocation policy, we provided a method to make the GNN generalizable to size by scaling its output with a pre-trained "activation function". Simulation results validated our analyses and showed that the mean-GNNs with selected and pre-trained "activation function" can learn the power and bandwidth allocation policies with good generalization ability to the number of users. However, our analyses and method are only applicable to the GNNs with mean-, sum-, or max-aggregators when learning 1D-PE policies. Why the GNNs are generalizable to the number of users when learning power control/allocation and link scheduling policies in various interference networks and how to design GNNs for optimizing multi-user precoding with size generalizability in various system configurations are still open.

Acknowledgements This work was supported by National Natural Science Foundation of China (Grant Nos. 61731002, 62271024).

References

1 Sun H R, Chen X Y, Shi Q J, et al. Learning to optimize: training deep neural networks for interference management. IEEE Trans Signal Process, 2018, 66: 5438–5453
2 Eisen M, Ribeiro A. Optimal wireless resource allocation with random edge graph neural networks. IEEE Trans Signal Process, 2020, 68: 2977–2991
3 Liu D, Sun C J, Yang C Y, et al. Optimizing wireless systems using unsupervised and reinforced-unsupervised deep learning. IEEE Network, 2020, 34: 270–277
4 Zhang C Z, Guo J, Yang C Y. When the gain of predictive resource allocation for content delivery is large? Sci China Inf Sci, 2023, 66: 222302
5 Kato N, Mao B M, Tang F X, et al. Ten challenges in advancing machine learning technologies toward 6G. IEEE Wireless Commun, 2020, 27: 96–103
6 Mitchell T M. The Need for Biases in Learning Generalizations. Rutgers Computer Science Technical Report CBM-TR-117. 1980
7 Keriven N, Peyré G. Universal invariant and equivariant graph neural networks. In: Proceedings of the Advances in Neural Information Processing Systems, 2019
8 Geman S, Bienenstock E, Doursat R. Neural networks and the bias/variance dilemma. Neural Comput, 1992, 4: 1–58
9 Shen Y F, Shi Y M, Zhang J, et al. Graph neural networks for scalable radio resource management: architecture design and theoretical analysis. IEEE J Sel Areas Commun, 2021, 39: 101–115
10 Sun C J, Wu J J, Yang C Y. Improving learning efficiency for wireless resource allocation with symmetric prior. IEEE Wireless Commun, 2022, 29: 162–168
11 Guo J, Yang C Y. Learning power allocation for multi-cell-multi-user systems with heterogeneous graph neural networks. IEEE Trans Wireless Commun, 2022, 21: 884–897
12 Zhao B, Guo J, Yang C Y. Learning precoding policy: CNN or GNN? In: Proceedings of the IEEE Wireless Communications and Networking Conference, 2022
13 Manzil Z, Satwik Z, Siamak R, et al. Deep sets. In: Proceedings of the Advances in Neural Information Processing Systems, 2017
14 He S W, Xiong S W, Ou Y Y, et al. An overview on the application of graph neural networks in wireless networks. IEEE Open J Commun Soc, 2021, 2: 2547–2565
15 Lee M, Yu G, Li G Y. Graph embedding-based wireless link scheduling with few training samples. IEEE Trans Wireless Commun, 2021, 20: 2282–2294
16 Chen T R, Zhang X R, You M L, et al. A GNN-based supervised learning framework for resource allocation in wireless IoT networks. IEEE Int Things J, 2022, 9: 1712–1724
17 Ruiz L, Chamon L, Ribeiro A. Graphon neural networks and the transferability of graph neural networks. In: Proceedings of the Advances in Neural Information Processing Systems, 2020
18 Bondi A B. Characteristics of scalability and their impact on performance. In: Proceedings of the International Workshop on Software and Performance, 2000
19 Zhou Y Z, Kutyniok G, Ribeiro B. OOD link prediction generalization capabilities of message-passing GNNs in larger test graphs. In: Proceedings of the Advances in Neural Information Processing Systems, 2022
20 Yehudai G, Fetaya E, Meirom E, et al. From local structures to size generalization in graph neural networks. In: Proceedings of the International Conference on Machine Learning, 2021
21 Maskey S, Levie R, Kutyniok G. Transferability of graph neural networks: an extended graphon approach. Appl Comput Harmonic Anal, 2023, 63: 48–83
22 Xu K, Zhang M Z, Li J L, et al. How neural networks extrapolate: from feedforward to graph neural networks. In: Proceedings of the International Conference on Learning Representations, 2021
23 Sannai A, Takai Y, Cordonnier, M. Universal approximations of permutation invariant/equivariant functions by deep neural networks. 2019. ArXiv:1903.01939
24 Wu D P, Negi R. Effective capacity: a wireless link model for support of quality of service. IEEE Trans Wireless Commun, 2003, 24: 630–643
25 Kelly F. Notes on effective bandwidths. In: Stochastic Networks: Theory and Applications. Oxford: Oxford University Press, 1996
26 Sun C J, Yang C Y. Learning to optimize with unsupervised learning: training deep neural networks for URLLC. In: Proceedings of the IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, 2019
27 Yang W, Durisi G, Koch T, et al. Quasi-static multiple-antenna fading channels at finite blocklength. IEEE Trans Inform Theor, 2014, 60: 4232–4265
28 Sun C J, She C Y, Yang C Y, et al. Optimizing resource allocation in the short blocklength regime for ultra-reliable and low-latency communications. IEEE Trans Wireless Commun, 2018, 18: 402–415
29 She C Y, Yang C Y, Quek T Q S. Joint uplink and downlink resource configuration for ultra-reliable and low-latency communications. IEEE Trans Commun, 2018, 66: 2266–2280

30 3GPP. Study on Scenarios and Requirements for Next Generation Access Technologies. Technical Specification Group Radio Access Network, Technical Report 38.913, Release 14. 2016
31 Popovski P, Mange G, Roos A. Deliverable d6.3 intermediate system evaluation results. 2014. https://metis2020.com/wp-content/uploads/deliverables/METIS_D6.3_v1.pdf

Appendix A Proof of Proposition 1

Denote $[x_j]_{j\neq k}$ as a vector obtained by removing x_k from $\boldsymbol{x} \triangleq [x_1, \ldots, x_K]$. After ranking the elements in this vector in descending (or ascending) order, the resulting vector is denoted as $[x_j^d]_{j\neq k}$. Since the summation $\sum_{j\neq k} \phi(x_j)$ is independent of the order of each term, we have $\sum_{j\neq k} \phi(x_j) = \sum_{j\neq k} \phi(x_j^d)$.

When K is very large and the elements in $\boldsymbol{x}$ are i.i.d., each element in $[x_j^d]_{j\neq k}$ approaches to a deterministic function of K[4)]. Therefore, $\sum_{j\neq k} \phi(x_j^d)$ can be asymptotically expressed as a function of K, i.e., $\sum_{j\neq k} \phi(x_j^d) \approx \bar{f}(K)$. Then, from (1) we obtain $y_k = \tilde{f}\big(x_k, \sum_{j\neq k} \phi(x_j^d)\big) \approx \tilde{f}(x_k, \bar{f}(K)) \triangleq f(x_k, K)$.

Appendix B Proof of Proposition 2

For notational simplicity but without loss of generality, the activation function in each layer is the same in this appendix, i.e., $\sigma^l(\cdot) = \sigma(\cdot)$.

When K is very large and $x_k, k = 1, \ldots, K$ are i.i.d., we first prove a statement that $\boldsymbol{h}_k^l, k = 1, \ldots, K$ are i.i.d. is true for $l = 0, 1, \ldots, L$ using mathematical induction. When $l = 0$, we have $\boldsymbol{h}_k^0 = x_k$, Since $x_k, k = 1, \ldots, K$ are i.i.d., the statement is true. Assume that the statement is true for $l = n$. When $l = n + 1$, from (3) we have

$$\boldsymbol{h}_k^{n+1} = \sigma\left(\boldsymbol{U}^{n+1}\boldsymbol{h}_k^n + \frac{\boldsymbol{V}^{n+1}}{K}\left(\sum_{j\neq k}\boldsymbol{h}_j^n\right) + \boldsymbol{c}^{n+1}\right). \tag{B1}$$

Since $\boldsymbol{h}_k^n, k = 1, \ldots, K$ are i.i.d., $\frac{1}{K}(\sum_{j\neq k}\boldsymbol{h}_j^n) \approx \mathbb{E}^K(\boldsymbol{h}_k^n) = \mathbb{E}^K(\boldsymbol{h}^n)$ when K is large[5)], where $\mathbb{E}^K(\cdot)$ denotes expectation. Then, by substituting $\mathbb{E}^K(\boldsymbol{h}^n)$ into (B1), we have

$$\boldsymbol{h}_k^{n+1} \approx \sigma\Big(\boldsymbol{U}^{n+1}\boldsymbol{h}_k^n + \boldsymbol{V}^{n+1}\mathbb{E}^K(\boldsymbol{h}^n) + \boldsymbol{c}^{n+1}\Big), \tag{B2}$$

which is a function of $\boldsymbol{h}_k^n$ and hence is also i.i.d. among k. Consequently, $\boldsymbol{h}_k^l, k = 1, \ldots, K$ are i.i.d. for $l = 0, 1, \ldots, L$. Thereby, $\frac{1}{K}\sum_{j\neq k}\boldsymbol{h}_j^{l-1} \approx \mathbb{E}^K(\boldsymbol{h}_k^{l-1}) = \mathbb{E}^K(\boldsymbol{h}^{l-1})$ when K is very large. By substituting $\mathbb{E}^K(\boldsymbol{h}^{l-1})$ into (3), we can obtain

$$\boldsymbol{h}_k^l \approx \sigma\Big(\boldsymbol{U}^l\boldsymbol{h}_k^{l-1} + \boldsymbol{V}^l\mathbb{E}^K\left(\boldsymbol{h}^{l-1}\right) + \boldsymbol{c}^l\Big). \tag{B3}$$

Then, the k-th output of the mean-GNN can be approximated as a function of K and x_k, i.e.,

$$\hat{y}_k \approx \sigma\Big(\boldsymbol{U}^{L+1}\sigma\big(\cdots\boldsymbol{U}^2\sigma(\boldsymbol{U}^1x_k + \boldsymbol{V}^1\mathbb{E}^K(x) + \boldsymbol{c}^1) + \boldsymbol{V}^2\mathbb{E}^K(\boldsymbol{h}^1) + \boldsymbol{c}^2\cdots\big) + \boldsymbol{V}^{L+1}\mathbb{E}^K(\boldsymbol{h}^L) + \boldsymbol{c}^{L+1}\Big) \triangleq \hat{q}(x_k, K). \tag{B4}$$

Appendix C Proof of Proposition 3

In order to prove $\hat{q}(x_k, K) \approx \hat{q}(x_k, K')$, we only need to prove $\mathbb{E}^K(\boldsymbol{h}^l) \approx \mathbb{E}^{K'}(\boldsymbol{h}^l)$, $l = 0, 1, \ldots, L$ according to (4). Since $p^K(x) = p^{K'}(x)$, we have $\mathbb{E}^K(x) = \mathbb{E}^{K'}(x)$. In the following, we prove that

$$p^K(\boldsymbol{h}^l) \approx p^{K'}(\boldsymbol{h}^l) \tag{C1}$$

is true for $l = 0, 1, \ldots, L$ using mathematical induction. When $l = 0$, we have $\boldsymbol{h}^0 = x$, hence (C1) is true. Assume that (C1) is true for $l = n$. In order to prove $p^K(\boldsymbol{h}^{n+1}) \approx p^{K'}(\boldsymbol{h}^{n+1})$, we need to prove that $\Pr^K\{\boldsymbol{h}^{n+1} \leqslant \boldsymbol{h}\} \approx \Pr^{K'}\{\boldsymbol{h}^{n+1} \leqslant \boldsymbol{h}\}$ is true for any $\boldsymbol{h}$, where $\Pr^K\{\cdot\}$ denotes probability. In other words, we need to prove $\Pr^K\{\boldsymbol{h}_k^{n+1} \leqslant \boldsymbol{h}\} \approx \Pr^{K'}\{\boldsymbol{h}_k^{n+1} \leqslant \boldsymbol{h}\}$. When K and K' are very large, from (B2) we can derive that

$$\Pr^K\Big\{\boldsymbol{h}_k^{n+1} \leqslant \boldsymbol{h}\Big\} \approx \Pr^K\Big\{\boldsymbol{h}_k^n \leqslant A^K(\boldsymbol{h})\Big\}, \tag{C2}$$

$$\Pr^{K'}\Big\{\boldsymbol{h}_k^{n+1} \leqslant \boldsymbol{h}\Big\} \approx \Pr^{K'}\Big\{\boldsymbol{h}_k^n \leqslant A^{K'}(\boldsymbol{h})\Big\}, \tag{C3}$$

where $A^K(\boldsymbol{h}) \triangleq \frac{1}{\boldsymbol{U}^{n+1}}(\sigma^{-1}(\boldsymbol{h}) - \boldsymbol{V}^{n+1}\mathbb{E}^K(\boldsymbol{h}^n) - \boldsymbol{c}^{n+1})$, and $\sigma^{-1}(\cdot)$ denotes the inverse function of $\sigma(\cdot)$. Again, we omit the superscript in the activation function for notational simplicity.

Since $p^K(\boldsymbol{h}^n) \approx p^{K'}(\boldsymbol{h}^n)$, we have $\mathbb{E}^K(\boldsymbol{h}^n) \approx \mathbb{E}^{K'}(\boldsymbol{h}^n)$. When $\sigma(\cdot)$ does not depend on K, we have $A^K(\boldsymbol{h}) \approx A^{K'}(\boldsymbol{h})$. Hence, we can obtain $\Pr^K\{\boldsymbol{h}_k^n \leqslant A^K(\boldsymbol{h})\} \approx \Pr^{K'}\{\boldsymbol{h}_k^n \leqslant A^{K'}(\boldsymbol{h})\}$, which indicates that $\Pr^K\{\boldsymbol{h}_k^{n+1} \leqslant \boldsymbol{h}\} \approx \Pr^{K'}\{\boldsymbol{h}_k^{n+1} \leqslant \boldsymbol{h}\}$ (i.e., $p^K(\boldsymbol{h}^{n+1}) \approx p^{K'}(\boldsymbol{h}^{n+1})$) according to (C2) and (C3). Hence, (C1) is true for $l = 0, \ldots, L$. Thus, $\mathbb{E}^K(\boldsymbol{h}^l) \approx \mathbb{E}^{K'}(\boldsymbol{h}^l)$.

4) David H A, Nagaraja H N. Order Statistics. Hoboken: John Wiley and Sons, 2003.
5) Rice J A. Mathematical Statistics and Data Analysis. Boston: Cengage Learning, 2006.

Appendix D Proof of Proposition 4

Again, for notational simplicity, the activation function in each layer is the same in this appendix. We first show that the input-output relation of the GNN with the max-aggregator is not invariant to the size of the input vector. For this GNN, the relation between $\boldsymbol{h}_k^l$ and $\boldsymbol{h}_k^{l-1}$ can be expressed as

$$\boldsymbol{h}_k^l = \sigma\Big(\boldsymbol{U}^l \boldsymbol{h}_k^{l-1} + \boldsymbol{V}^l \max_{j\neq k}^{K} \boldsymbol{h}_j^{l-1} + \boldsymbol{c}^l\Big),$$

where $\max_{j\neq k}^{K}(\cdot)$ denotes the maximization operation.

Analogous to Appendix B, we can derive that $\boldsymbol{h}_k^l, k = 1,\ldots,K$ are i.i.d. for $l = 0,\ldots,L$ when K is very large since $x_k, k = 1,\ldots,K$ are i.i.d.. Then, we have $\max_{j\neq k}^{K} \boldsymbol{h}_j^{l-1} \approx D_{\boldsymbol{h}^{l-1}}^{-1}(\frac{K}{K+1})$ according to[4)], where $D_{\boldsymbol{h}^{l-1}}^{-1}(\cdot)$ denotes the inverse of the probability distribution function of $\boldsymbol{h}^{l-1}$. Then, the relation between the output and the input of the GNN can be approximated as

$$\begin{aligned}\hat{y}_k &\approx \sigma\left(\boldsymbol{U}^{L+1}\sigma\left(\cdots \boldsymbol{U}^2\sigma\left(\boldsymbol{U}^1 x_k + \boldsymbol{V}^1 D_x^{-1}\left(\frac{K}{K+1}\right) + \boldsymbol{c}^1\right) + \boldsymbol{V}^2 D_{\boldsymbol{h}^1}^{-1}\left(\frac{K}{K+1}\right) + \boldsymbol{c}^2 \cdots\right) + \boldsymbol{V}^{L+1} D_{\boldsymbol{h}^L}^{-1}\left(\frac{K}{K+1}\right) + \boldsymbol{c}^{L+1}\right)\\ &= \hat{q}(x_k, K).\end{aligned}$$

Since the probability distribution function of the continuous random variable is a strictly monotonically increasing function, $D_{\boldsymbol{h}^l}^{-1}(\frac{K}{K+1})$ also strictly increases monotonically. Then, we have $D_{\boldsymbol{h}^l}^{-1}(\frac{K}{K+1}) \neq D_{\boldsymbol{h}^l}^{-1}(\frac{K'}{K'+1})$ for any $K \neq K'$. Thus, $\hat{q}(x_k, K) \neq \hat{q}(x_k, K')$.

We then show that the input-output relation of the GNN with the sum-aggregator is not invariant to the input size. For this GNN, the relation between $\boldsymbol{h}_k^l$ and $\boldsymbol{h}_k^{l-1}$ can be expressed as

$$\boldsymbol{h}_k^l = \sigma\left(\boldsymbol{U}^l \boldsymbol{h}_k^{l-1} + \boldsymbol{V}^l \sum_{j\neq k}^{K} \boldsymbol{h}_j^{l-1} + \boldsymbol{c}^l\right).$$

When K is very large, again considering that x_k is i.i.d. among k, we can derive that $\boldsymbol{h}_k^l$ is also i.i.d. across k. According to the central-limit theorem, $\sum_{j\neq k}^{K} \boldsymbol{h}_j^{l-1}$ follows a normal distribution, whose mean and variance increase with K. This suggests that $\sum_{j\neq k}^{K} \boldsymbol{h}_j^{l-1}$ depends on K, which can be approximated as $\tilde{\boldsymbol{h}}^{l-1}(K)$. Then, the relation between the output and the input of the GNN can be approximated as follows that is no longer equal to $\hat{q}(x_k, K')$,

$$\hat{y}_k \approx \sigma\Big(\boldsymbol{U}^{L+1}\sigma\Big(\cdots \boldsymbol{U}^2\sigma\Big(\boldsymbol{U}^1 x_k + \boldsymbol{V}^1 \tilde{\boldsymbol{h}}^0(K) + \boldsymbol{c}^1\Big) + \boldsymbol{V}^2 \tilde{\boldsymbol{h}}^1(K) + \boldsymbol{c}^2 \cdots\Big) + \boldsymbol{V}^{L+1} \tilde{\boldsymbol{h}}^L(K) + \boldsymbol{c}^{L+1}\Big) = \hat{q}(x_k, K).$$