Size Generalization for Resource Allocation with Graph Neural Networks

Jiajun Wu, Chengjian Sun and Chenyang Yang
Beihang University, Beijing, China
Email: {jiajunwu,sunchengjian,cyyang}@buaa.edu.cn

Abstract

Size generalization is important for learning wireless policies, which are often with dynamic sizes, say caused by time-varying number of users. Recent works of learning to optimize resource allocation empirically demonstrate that graph neural networks (GNNs) can generalize to different problem scales. However, GNNs are not guaranteed to generalize across input sizes. In this paper, we strive to analyze the size generalization mechanism of GNNs when learning permutation equivariant (PE) policies. We find that the aggregation function and activation functions of a GNN play a key role on its size generalization ability. We take the GNN with mean aggregator, called mean-GNN, as an example to demonstrate a size generalization condition, and interpret why several GNNs in the literature of wireless communications can generalize well to problem scales. To illustrate how to design GNNs with size generalizability according to our finding, we consider power and bandwidth allocation, and suggest to select or pre-train activation function in the output layer of mean-GNN for learning the PE policies. Simulation results show that the proposed GNN can generalize well to the number of users, which validate our analysis for the size generalization condition of GNNs when learning the PE policies.

Index Terms

Size generalization, resource allocation, permutation equivariance, graph neural networks

I. INTRODUCTION

Deep learning has been widely applied for resource allocation with diverse motivations, say improving resource usage efficiency by finding real-time policies from NP-hard problems or from the problems without models. Intelligent resource allocation has been envisioned as one of the most important features of five-generation beyond and six-generation wireless communications.
A core challenge in designing learning-based wireless resource allocation comes from dynamic propagation environments and fluctuating traffic load. For example, wireless channels and number of users may change quickly. When the number of users vary, the scale of a resource allocation problem and hence the input size of a deep neural network (DNN) for optimizing the problem should be changed accordingly. While online learning allows a DNN to make decisions over time-varying channels, it is unable to deal with the dynamic caused by time-varying problem scales. Therefore, designing size generalizable DNNs is critical when applying deep learning for resource allocation, which allows the reuse of DNN without the need of retraining or redesigning. If a DNN is unable to generalize to the size of its input, e.g., fully-connected neural network (FNN), the DNN has to be retrained whenever the problem scale changes unless the DNN is trained using the samples with all possible sizes [8], which incurs non-affordable training complexity for large scale wireless systems.

A promising way to enable generalization is embedding inductive biases into the structure of DNNs. Inductive biases are the constraints imposed on the functions represented by a DNN, which come from the prior knowledge for a class of problems before using data samples [8]. Mismatched inductive biases may lead to performance degradation due to imposing too strong constraints on the learning process, and appropriate inductive biases can accelerate training without performance loss and find the functions that generalize in a desirable way [9]. Hence, finding useful inductive biases is important for size generalization.

As a kind of prior knowledge for multivariate functions, permutation equivariance (PE) property has been shown widely existed in wireless resource allocation policies recently [3,10,12]. A resource allocation policy can usually be represented as a multivariate function, which is a mapping from environment parameters (say channels of multiple users) to the allocated resources to wireless nodes (say powers to the users). For a policy exhibiting the PE property, the mapping is not affected by the order of nodes. Depending on the considered problem, a resource allocation policy may exhibit one-dimensional (1D)-PE, two-dimensional-PE, joint-PE properties or their combinations [3,12]. Graph neural networks (GNNs) [13] and permutation equivariant neural networks (PENN) [14] are two kinds of DNNs that are embedded with permutation equivariant inductive biases and hence can learn the PE policies efficiently. In additional to the PE property, GNNs also harness another kind of prior knowledge: the relation among nodes [13], unless a GNN learns a policy over a complete graph where all nodes are fully connected.

It has been empirically demonstrated in [3,10,12,15,16] that GNNs can generalize to different
in ad hoc or in an interference network was shown able to generalize to 500 or 1000 transceivers when trained using samples with 50 transceivers. In [12], a GNN designed to learn a power allocation policy in cellular network was shown able to generalize to the networks with different numbers of users and cells when trained using samples generated with small networks. In [15,16], a GNN designed to learn a link scheduling policy or joint channel and power allocation policy in device-to-device (D2D) communications was shown with good generalization performance to the number of D2D pairs. However, it has been noticed that GNNs are not guaranteed to generalize across the sizes of graphs. As revealed in [17], the generalization of GNNs to large graph size is hard for some graph topologies. In fact, the size generalization capability of GNNs is not well-understood. As mentioned in [3,10] without explanation, it is the PE property of GNNs that improves their size generalizability. As stated in [12], it is the parameter sharing of GNNs that enables good generalization to unseen sizes since the number of trainable parameters of a GNN is independent from the graph size, which is again owing to the PE property. The reason of size generalization ability of GNNs was not mentioned in [15,16]. Though important, it remains unknown when GNNs can generalize to the problem scales, when cannot, and why.

In this paper, we present a small step in understanding size generalization capability of GNNs. To the best of our knowledge, this is the first attempt to reveal the size generalization mechanism in wireless communications. We notice that learning a resource allocation policy with size generalization ability can be regarded as finding the mapping from environment parameters and problem size to the allocated resources with DNN. A straightforward approach to find the mapping is to train a versatile DNN with the samples of all possible sizes, which does not require size generalization. Since training such a DNN is with prohibitive complexity for large scale systems, we resort to the decomposability of a PE policy in the large size regime. Since the general study for size generalization is extremely difficult, we only analyze case-by-case. To isolate the impact of the PE property from the inductive bias imposed by relation among nodes, we consider GNNs and policies with 1D-PE property since the policies are learned over a complete graph. In the sequel, we call a 1D-PE policy as PE policy for short. In particular, we take the power and bandwidth allocation policies as example to demonstrate how to leverage the decomposability for enabling the GNN to generalize to unseen sizes. While GNNs are very flexible, which can use different aggregators and combiners [13], we consider the commonly used mean, sum, and max aggregators. The major contributions are summarized as follows.
• We find that aggregation and activation functions impose the inductive bias for size generalization of a GNN. We prove that input-output relation of the GNN with mean aggregator (referred to as mean-GNN for short) and size-independent activation functions (SI-AFs) is invariant to the input size in large size regime, but the GNN with sum or max aggregator does not. We provide a size generalization condition, indicating when a GNN can generalize to the size of a PE policy. From the condition, we can interpret why GNNs used for learning resource allocation policies in existing literatures can generalize well to problem scales.

• Inspired by our finding, we suggest a method to improve size generalization, which aligns the inductive bias of a GNN to the size-scaling law of a PE policy by using the mean-GNN with a selected or pre-trained activation function of the output layer. We evaluate the size generalization performance of the proposed method by optimizing the power and bandwidth allocation in ultra-reliable low-latency communications (URLLC) in the scenarios with both large and small number of users.

The rest of the paper is organized as follows. In section II, we provide the size generalization condition. In section III, we consider resource allocation policies that are independent from sizes asymptotically, and explain why several GNNs in literature can generalize to the problem scales. In section IV, we consider size-dependent policies, and show how to design the GNNs to enable size generalization. In section V, we evaluate the performance of the proposed GNN for learning the power and bandwidth allocation. In section VI, we provide the conclusion remarks.

II. 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 the PE policy and the size-scaling laws of the GNNs with mean-, max-, and sum-aggregators. Finally, we show that mean-GNN with SI-AFs can generalize to the size of the PE policy with asymptotic size-invariance property, which implies a condition for size generalization.

A. Learning a Policy with Different Sizes

A resource allocation policy for $K$ objects (say users or transceiver pairs) can usually be represented as a multivariate function, where the allocated resources (say transmit powers) $y$ are the output variables and the environmental parameters (say channels) $x$ are the input variables. Denote $x \triangleq [x_1, \ldots, x_K]$, $y \triangleq [y_1, \ldots, y_K]$, where $x_k$ and $y_k$ are respectively the input and output
variables of the policy for the $k$th object, and $K$ reflects the scale of a 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 $y = F'(x, K) \triangleq [f'_1(x, K), \ldots, f'_K(x, K)]$. If we can learn the function $F'(x, K)$ with a DNN, then the DNN can be served as the resource allocation policy with different sizes. To train such a DNN without the need for size generalization, one can use the samples for $x$ with all possible sizes of $K$, which is expensive for large scale systems.

Fortunately, for a PE policy, the multivariate function degenerates into a scalar function of $K$ for each object when $K \to \infty$ as to be shown in the following, i.e., $f'_k(x, K) \to f(x_k, K)$. This inspires us to learn $F'(x, K)$ by learning $f(x_k, K)$ and a size-independent function.

**B. Asymptotic Form of PE Policy**

If the multivariate function $F'(x, K)$ is permutation equivariant to $x$, i.e., the policy does not change with the order of objects, then it is a PE policy [3, 11]. The PE policy satisfies $\Pi y = F'(\Pi x, K)$, where $\Pi$ denotes arbitrary permutation matrix.

In [14], it was proved that permutation invariant functions can be decomposed into continuous outer and inner functions, similar to the Kolmogorov-Arnold representation theorem applicable for arbitrary multivariate continuous function. By extending the proof in [14], it was proved in [18] that the $k$th output variable of any PE policy can be expressed as,

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

(1)

where $\phi(\cdot)$ and $\tilde{f}(\cdot)$ are continuous functions independent from $k$.

The following proposition shows the asymptotic form of (1) for independent and identically distributed (i.i.d.) input variables.

**Proposition 1.** For the PE policy, if $K \to \infty$ and $x_k, k = 1, \cdots, K$ are i.i.d., then

$$y_k = f(x_k, K),$$

(2)

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

**Proof.** See Appendix A

It indicates that a PE policy asymptotically degenerates into a function of $K$, which maps the input variable to the output variable of each object.
C. Impact of Aggregator on the Scaling Law of a GNN with Size

We first consider a GNN with mean aggregator, i.e., mean-GNN, for learning the PE policy with \( K \) objects. Denote the output of the GNN as \( \hat{y} \triangleq [\hat{y}_1, ..., \hat{y}_K] \), and the output of the \( l \)th layer as \( [h^l_1, ..., h^l_K]^T \). The relation between \( h^l_k \) and \( h^{l-1}_k \) of the mean-GNN can be expressed as

\[
h^l_k = \sigma^l(U^l h^{l-1}_k + V^l (\frac{1}{K} \sum_{j \neq k} h^{l-1}_j) + c^l), \quad l = 1, ..., L + 1,
\]

where \( \sigma^l(\cdot) \) is the activation function in the \( l \)th layer, \( U^l \), \( V^l \) and \( c^l \) are the trainable parameters of the \( l \)th layer, and \( L \) is the number of hidden layers. \( h^0_k = x_k \), and \( h^{L+1}_k = \hat{y}_k \).

The term \( \frac{1}{K} \sum_{j \neq k} h^{l-1}_j \) in (3) is an average over \( h^{l-1}_j, j \neq k \). The average is a pooling operation, which is called mean aggregation in literature. In order to impose PE inductive biases on GNNs to learn PE policies efficiently, the aggregation function should satisfy commutative law, hence it can also be other operations such as summation and maximization.

The following proposition shows the scaling law of mean-GNN for i.i.d. input features. It indicates that the \( k \)th output of the mean-GNN well-trained for \( K \) objects depends on \( x_k \) and \( K \) but is independent from \( x_{k'}, k' \neq k \) when \( K \to \infty \).

**Proposition 2.** For the mean-GNN, if \( K \to \infty \) and \( x_k, k = 1 \cdots K \) are i.i.d., then

\[
\hat{y}_k = \hat{q}(x_k, K),
\]

where the expression of \( \hat{q}(x_k, K) \) is shown in the appendix.

**Proof.** See Appendix [B]

Denote the probability distribution function of the features in \( x \) of size \( K \) as \( p^K(x) \). The following proposition shows 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 \( K, K' \to \infty \) and \( \sigma^l(\cdot), l = 1, \cdots, L + 1 \) do not depend on \( K \), then the relation between its input and output of each object is invariant to the input size, i.e.,

\[
\hat{y}_k = \hat{q}(x_k, K) = \hat{q}(x_k, K') \triangleq \hat{q}(x_k).
\]

**Proof.** See Appendix [C]
The proposition means that if the input features are with identical distribution for different input sizes meanwhile the activation function in each layer is independent from $K$, then the functions able to be represented by the mean-GNN do not depend on $K$. Most commonly used activation functions do not depend on the input size, e.g., ReLU and its variants, sigmoid, Softplus and Tanh. One exception is Softmax.

However, the GNN with max-aggregator or sum-aggregator is not size invariant.

**Proposition 4.** For the GNN with maximization or summation as aggregation function, if $K \to \infty$ and $x_{k}, k = 1 \cdots, K$ are i.i.d., then $\hat{y}_{k} = \hat{q}(x_{k}, K)$. However, if $p^{K}(x) = p^{K'}(x)$ for any $K \neq K'$, $K, K' \to \infty$ and $\sigma^{l}(\cdot), l = 1, \cdots, L + 1$ do not depend on $K$, its input-output relation is no longer invariant to the input size.

**Proof.** See Appendix D.

**D. Size Generalization Condition**

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 $K \to \infty$ and features are i.i.d. according to Propositions 1 and 2 and the universal approximation theorem for PE functions [18]. Further considering Proposition 3, we can obtain the following corollary for the PE policy with size-invariance property.

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

$$y_{k} = f(x_{k}, K) = f(x_{k}, K') \triangleq f(x_{k}), \ K \neq K', K, K' \to \infty,$$

then the mean-GNN with SI-AF in each layer trained for learning the PE policy with $K$ objects can generalize to the PE policy with $K'$ objects when $K, K' \to \infty$.

To help understand the PE polices with the size-invariance property, we show $y_{k} = f(x_{k}, K)$ of two toy example policies in Fig.1. For any given input variable $x_{k}$, the output variable $y_{k}$ of “Policy 1” does not depend on $K$, while $y_{k}$ of “Policy 2” increases with $K$. Thus, “Policy 1” satisfies the size-invariance property but “Policy 2” does not.

Asymptotic size-invariance is the simplest size-scaling law. Not all PE policies exhibit the size-invariance property. For a PE policy satisfying (6), the mean-GNN with SI-AFs can generalize to the policy with different sizes, but the GNN with max- or sum-aggregator can not.

This implies a condition for a GNN to learn a PE policy with size generalization ability.
Remark 1. Size generalization condition: a GNN can generalize 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.

III. SIZE GENERALIZATION FOR POLICIES WITH SIZE-INVARIANCE PROPERTY

In this section, we provide three size-invariant PE policies, from which we strive to interpret why existing GNNs used to learn these policies can generalize to the number of objects. We start from a simple policy with closed-form expression to help understand the size-invariance property. We then consider two resource allocation policies without closed-form expressions, to explain why the GNNs in [10, 15] have size generalization ability for learning these policies.

A. Power Allocation Policy with Closed-form

Consider a downlink frequency division multiple access 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 the quality-of-service (QoS) constraint as follows,

$$\begin{align*}
    \min_{P_k} & \quad \sum_{k=1}^{K} P_k \\
    \text{s.t.} & \quad B \log(1 + \frac{P_k g_k}{N_0 B}) \geq s_0, \\
    & \quad P_k \geq 0,
\end{align*}$$

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 \( P_k^* = \frac{N_0}{g_k} B(2^{\frac{s_0}{2}} - 1) \).

The power allocation policy obtained from (7) can be expressed as \( \mathbf{p}^* = \mathbf{F}_1(\mathbf{g}) \), where \( \mathbf{p}^* = [P_1^*, ..., P_K^*] \) and \( \mathbf{g} = [g_1, ..., g_K] \). Since the multivariate function \( \mathbf{F}_1(\mathbf{g}) \) is not affected by the order of users, the policy is a PE Policy.

\( \mathbf{F}_1(\mathbf{g}) \) is independent of \( K \), since the problem can be decoupled into multiple problems each for a single-user. In other words, the power allocation policy is size-invariant for all values of \( K \), and hence can be learned by the mean-GNN with SI-AFs with size generalization ability.

**B. Power Control or Link Scheduling Policy in Interference Networks**

Consider a power control policy for an interference network consisting of \( K \) single-antenna transceiver pairs in [2]. The transmit power is optimized to maximize the sum rate under the power constraint at each transmitter as follows,

\[
\max_{P_k} \sum_{k=1}^{K} \log \left( 1 + \frac{P_k g_{k,k}}{\sum_{j \neq k} P_j g_{j,k} + \sigma_0} \right) \tag{8}
\]

\[ s.t. \ 0 \leq P_k \leq P_{\text{max}}, \]

where \( P_k \) is the transmit power at the \( k \)th transmitter, \( g_{k,j} \) is the channel gain from the \( j \)th transmitter to the \( k \)th receiver, \( \sigma_0 \) is the noise power, and \( P_{\text{max}} \) is the maximal transmit power of each transmitter.

The optimal power control policy can be expressed as \( \mathbf{p}^* = \mathbf{F}'_2(\mathbf{G}, K) \), where \( \mathbf{p}^* = [P_1^*, ..., P_K^*] \), and \( \mathbf{G} = [g_{k,j}]_{k,j=1}^{K} \) is the channel matrix whose element in the \( k \)th row and \( j \)th column is \( g_{k,j} \). Since \( \mathbf{F}'_2(\mathbf{G}, K) \) is not affected by the order of the transceiver pairs, the policy is a PE Policy.

The problem in (8) is without closed-form solution. Numerical solution can be obtained via a weighted sum mean-square error minimization (WMMSE) algorithm, where the solution is binary (i.e., \( P_k \in \{0, P_{\text{max}}\} \)) in most cases [2]. In what follows, we numerically show the asymptotic size-invariance property of this complex policy obtained from the WMMSE algorithm.

Considering that \( P_k^* \) also depends on the channel gains of interference links (i.e., \( g_{k,j} \) and \( g_{j,k}, j \neq k \)) when \( K \) is small, we show the probability of \( P_k^* = P_{\text{max}} \), i.e., \( \Pr\{P_k^* = P_{\text{max}}\} \), when \( g_{k,k} \) is given for each value of \( K \). We set \( K = [10, 20, ..., 100] \) and generate 100,000 realizations

\[ ^1 \text{Under Rayleigh fading channel and } P_{\text{max}} = 1, \sigma_0 = 1 \text{ as considered in [2], the ratio of the numerical solution satisfying } P_k \leq 10^{-3} \text{ or } P_k \geq 1 - 10^{-3} \text{ is larger than } 95\%. \]
of $G$ for each value of $K$. Since it is hard to find the same value of $g_{k,k}$ in different realizations, \( \Pr\{P^*_k = P_{\text{max}}\} \) is obtained by counting the frequency when $g_{k,k} \in (g_{k,k} - \delta, g_{k,k} + \delta)$, where $\delta$ is a small value and is set as 0.1. We consider Rayleigh fading channel. The result is shown in Fig. 2(a). We can see that $\Pr\{P^*_k = P_{\text{max}}\} \approx 1$ for $g_{k,k} \geq 2$ and $\Pr\{P^*_k = P_{\text{max}}\} \approx 0$ for $g_{k,k} \leq 1$. Moreover, $\Pr\{P^*_k = P_{\text{max}}\}$ is almost invariant to $K$ when $K \geq 50$. To see this more clearly, we provide the mean value of $\Pr\{P^*_k = P_{\text{max}}\}$ by taking the average over $g_{k,k}$ in Fig. 2(b), which changes slowly with $K$ when $K$ is large. This indicates that the policy obtained from problem (8) is asymptotical size-invariant with high probability.

Fig. 2. Behavior of PE policies with or without the asymptotic size-invariance property versus $K$.

In [15], link scheduling for D2D systems is optimized to maximize the sum rate as follows,

$$\max_{\rho_k} \sum_{k=1}^{K} \log \left( 1 + \frac{\rho_k P_k g_{k,k}}{\sum_{j \neq k} \rho_j P_j g_{k,j} + \sigma_0} \right)$$

(9)

$$s.t. \quad \rho_k \in \{0, 1\},$$

where $\rho_k = 1$ if the $k$th D2D pair is scheduled and $\rho_k = 0$ otherwise. Since the objective function has the same form as that in (8) and the constraint is a special case of that in (8), the optimal policy obtained from (9) is also asymptotical size-invariant with high probability.

In [15], the optimal policy obtained from problem (9) is learned by graph embedding (can be regarded as a GNN), where the aggregation function is summation. In [10], the optimal policy obtained from problem (8) is learned by a GNN, where the aggregation function is maximization. According to Proposition 4, the input-output relation of the GNN with sum- or max-aggregator is not invariant to the input size. Nonetheless, the GNNs have been demonstrated with good size generalization performance in [10,15]. This is because each output of the GNN in [15] is followed by FNN for classification, and the output layer of the GNN in [10] behaves like a
classifier, in order for learning a binary solution. Although the hidden output of the GNN in [10] or [15] is not invariant to the input size, the operation in the output layer for classification makes $I(\hat{q}(x_k, K) > \eta) = I(\hat{q}(x_k, K') > \eta)$ holding with high probability, where $I(\cdot)$ denotes the indicator function and $\eta$ is the threshold for classification. This suggests that when the size-scaling law of a GNN does not match with the size-scaling law of a PE policy, further processing the output may enable the GNN generalizable to the PE policy with different sizes.

IV. SIZE GENERALIZATION FOR POLICIES WITHOUT SIZE-INVARINANCE PROPERTY

In this section, we consider PE policies without size-invariance property, and demonstrate how to satisfy the size generalization condition in Remark 1 by designing mean-GNNs for learning these policies. We first consider power allocation and bandwidth optimization policies where the power allocation policy is with closed-form expression. Despite that it is unnecessary to learn the policy in practice, the goal to provide this policy is for illustrating how to satisfy the size generalization condition by selecting an activation function for the output layer of mean-GNN with SI-AFs in hidden layers. Then, we consider power and bandwidth allocation in URLLC, whose solution is unable to be found via numerical algorithms. We proceed to propose a method to learn the policy with mean-GNN, which can generalize to different values of $K$.

A. Power Allocation and Bandwidth Optimization Policy

We still consider the downlink frequency division multiple access system, where a single-antenna BS serves $K$ single-antenna users with maximal transmit power $P_{\text{max}}$. Now we optimize the bandwidth (equally allocated to each user) together with power allocation to minimize the total bandwidth under both the QoS constraint and the power constraint,

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

s.t. $s_k \triangleq B \log(1 + \frac{P_k g_k}{N_0 B}) \geq s_0$, \hspace{1cm} (10a)

$$\sum_{k=1}^{K} P_k \leq P_{\text{max}},$$ \hspace{1cm} (10b)

$$P_k \geq 0, B \geq 0.$$ \hspace{1cm} (10c)
It is not hard to prove that this problem is convex. From the KKT conditions, we find that the global optimal solution, denoted as \( P_k^* \) and \( B^* \), satisfies the equality of (10b), and

\[
P_k^* = \frac{1}{g_k} F(B^*),
\]

(11)

\[
F(B^*) \triangleq N_0 B^*(2^{\frac{g}{B^*}} - 1).
\]

(12)

By substituting (11) into \( \sum_{k=1}^{K} P_k^* = P_{\text{max}} \), we have \( F(B^*) = \frac{P_{\text{max}}}{\sum_{k=1}^{K} \frac{1}{g_k}} \). By substituting this expression of \( F(B^*) \) into (11), we can obtain the optimal power allocated to the \( k \)th user and re-write it in the form of softmax function as

\[
P_k^* = P_{\text{max}} \frac{\frac{1}{g_k}}{\sum_{k=1}^{K} \frac{1}{g_k}} = P_{\text{max}} \frac{e^{\ln \frac{1}{g_k}}}{\sum_{k=1}^{K} e^{\ln \frac{1}{g_k}}},
\]

(13)

Denote \( F^{-1}(\cdot) \) as the inverse function of \( F(B) \) in (12). The optimal bandwidth allocated to the \( k \)th user can be re-written as

\[
B^* = F^{-1}\left( P_{\text{max}} \frac{\sum_{k=1}^{K} \frac{1}{g_k}}{\sum_{k=1}^{K} \frac{1}{g_k}} \right).
\]

(14)

Denote the power allocation and bandwidth optimization policies obtained from (10) as \( p^* = F'_3(g, K) \) and \( B^* = f'_4(g, K) \), respectively, which are PE policies since the functions \( F'_3(g, K) \) and \( f'_4(g, K) \) remain unchanged when the order of the users changes.

From the first equality of (13), the power allocation 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., \( \frac{z_k}{\sum_{k=1}^{K} z_k} \)). 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 as

\[
F'_3(g, K) = P_{\text{max}} \cdot \left[ \sigma_s\left(f_{e2}(g_1), \{f_{e2}(g_j)\}_{j \neq 1}\right), \cdots, \sigma_s\left(f_{e2}(g_K), \{f_{e2}(g_j)\}_{j \neq K}\right) \right]
\]

\[
\triangleq P_{\text{max}} \cdot \sigma_s\left(f_{e2}(g_1), \cdots, f_{e2}(g_K)\right),
\]

(15)

where \( \sigma_s(z_k, \{z_j\}_{j \neq k}) \triangleq \frac{e^{z_k}}{\sum_{k=1}^{K} e^{z_k}} \). 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.

1) Asymptotic Policies: To show that both policies are not size-invariant, we provide their asymptotic form.

When \( K \to \infty \), we have \( \frac{P_{\text{max}}}{\sum_{k=1}^{K} \frac{1}{g_k}} = \frac{1}{K} \frac{P_{\text{max}}}{\sum_{k=1}^{K} \frac{1}{g_k}} / K \to \frac{1}{K} \frac{P_{\text{max}}}{\sum_{k=1}^{K} \frac{1}{g_k}} \). Then, the optimal solution in large \( K \) regime can be expressed as
\[ P_k^* \rightarrow \frac{1}{K} \frac{P_{\text{max}}}{g_k \mathbb{E}(\frac{1}{g_k})} \triangleq f_3(g_k, K), \quad B^* \rightarrow F^{-1}(\frac{1}{K} \frac{P_{\text{max}}}{\mathbb{E}(\frac{1}{g_k})}) \triangleq f_4(g_k, K). \]  \hspace{1cm} (16)

In order to show how large the value of \( K \) is sufficient for \( P_k^* \) to be solely dependent on \( g_k \) and \( K \) as shown in (16), we provide the values of \( P_k^* \) versus \( K \) for a given value of \( g_k \) in Fig. 2(c). Since \( P_k^* \) also depends on the channel gains of other users (i.e., \( g_j, j \neq k \)) when \( K \) is small as shown in (13), we show the mean value and standard deviation of \( P_k^* \) versus \( K \).

We consider Rayleigh fading channel, set \( K = [2, 3, ..., 100] \), and generate 100,000 realizations of the channel vector \( g \) for each value of \( K \). Since it is hard to find the same value of \( g_k \) in different realizations of \( g \), we calculate the statistics of \( P_k^* \) when \( g_k \in (g_k' - \delta, g_k' + \delta) \), where \( \delta = 0.1 \). Since the results are similar for different values of \( g_k' \), set \( g_k' = 1 \). We can see that the standard deviation, which is caused by \( g_j, j \neq k \), approaches zero when \( K \geq 30 \).

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 generalize to different values of \( K \) when learning the optimal power allocation policy. Denote this GNN as \( \mathcal{P}_P(g; \theta_P) \), where \( \theta_P \) is the model parameters. The output of this GNN is \( \hat{p} = [\hat{P}_1, ..., \hat{P}_K] \).

From the composite function form of \( \mathcal{F}_3'(g, K) \) in (15), the neural network \( \mathcal{P}_P(g; \theta_P) \) should consist of hidden layers to learn an inner function (i.e., the mapping from \( g_k \) to \( \hat{P}_k' \)) and an output layer to learn 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\(^2\) the output of \( \mathcal{P}_P(g; \theta_P) \) can be expressed as

\[
\hat{p} = P_{\text{max}} \cdot \left[ \frac{e^{\hat{P}_1'}}{\sum_{k=1}^{K} e^{\hat{P}_k'}}, ..., \frac{e^{\hat{P}_K'}}{\sum_{k=1}^{K} e^{\hat{P}_k'}} \right] \\
= P_{\text{max}} \cdot \left[ \sigma_s(\hat{P}_1', \{\hat{P}_j'\}_{j \neq 1}), ..., \sigma_s(\hat{P}_K', \{\hat{P}_j'\}_{j \neq K}) \right] = P_{\text{max}} \cdot \sigma_s(\hat{P}_1', ..., \hat{P}_K'),
\]

where \( \hat{P}_k' = f_{e2}(g_k) = \ln \frac{1}{g_k} \). When \( K \rightarrow \infty \), the \( k \)th output of \( \mathcal{P}_P(g; \theta_P) \) becomes

\[
\hat{P}_k = \frac{P_{\text{max}} e^{\hat{P}_k'}}{\sum_{k=1}^{K} e^{\hat{P}_k'}} = \frac{1}{K} \sum_{k=1}^{K} e^{\hat{P}_k'} \rightarrow \frac{1}{K} \frac{P_{\text{max}} e^{\hat{P}_k'}}{\mathbb{E}(e^{\hat{P}_k'})},
\]

which scales with \( K \) in the same trend as \( P_k^* \) shown in (16).

\(^2\)From another composite function form of \( \mathcal{F}_3'(g, K) \), \( \mathcal{P}_P(g; \theta_P) \) can also first learn a size-invariant inner function \( f_{e1}(g_k) \) and then capture the trend of the power allocation policy with \( K \) by choosing other function for normalization. We choose softmax function as an example, since it is often used as the activation function of the output layer of DNN.
The structure of $\mathcal{P}_P(g; \theta_P)$ is shown in Fig. 3 which learns a size-invariant inner function $f_{e2}(g_k)$ before the activation function in the output layer (i.e., $\hat{P}_k' \approx f_{e2}(g_k)$) and then captures the asymptotic trend of the power allocation policy with $K$ (i.e., $1/K$) by softmax function.

**Remark 2.** In order to make $\hat{P}_k \approx P^*_k$, $\hat{P}_k'$ only needs to satisfy $\hat{P}_k' \approx \ln \frac{1}{g_k} + \zeta(K)$, where $\zeta(K)$ denotes any function of $K$. This is because when $\hat{P}_k' \approx \ln \frac{1}{g_k} + \zeta(K)$, we have

$$\hat{P}_k \approx \frac{P_{\text{max}} e^{\ln \frac{1}{g_k} + \zeta(K)}}{\sum_{k=1}^{K} e^{\ln \frac{1}{g_k} + \zeta(K)}} = \frac{P_{\text{max}} e^{\ln \frac{1}{g_k} + \zeta(K)}}{e^{\zeta(K)} \sum_{k=1}^{K} e^{\ln \frac{1}{g_k}}} = \frac{P_{\text{max}} e^{\ln \frac{1}{g_k}}}{\sum_{k=1}^{K} e^{\ln \frac{1}{g_k}}} = P^*_k.$$  

It implies that the GNN used to learn the inner function does not necessarily be invariant to $K$. This may explain why the power allocation policy can be learned with size generalization ability by a GNN using sum-aggregator and softmax in the output layer in [12].

Since $B^*$ has to be found from a transcendental equation in (12), $f'_4(g, K)$ has no closed-form and can not 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 will show how to deal with this issue in the next subsection.

**B. Power and Bandwidth Allocation Policy for URLLC**

Consider a downlink orthogonal frequency division multiple access system supporting URLLC, where a BS equipped with $N_t$ antennas serves $K$ single-antenna users. To improve the resource usage efficiency, the power and bandwidth allocation is optimized to minimize the total bandwidth required to satisfy the QoS of every user with maximal transmit power $P_{\text{max}}$ at the BS as,

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

s.t. 

$$C^E_k \geq S^E,$$  

$$\sum_{k=1}^{K} P_k \leq P_{\text{max}},$$

where $C^E_k$ is the total available bandwidth and $S^E$ is the required bandwidth for a user.
\begin{equation}
P_k \geq 0, B_k \geq 0,
\end{equation}

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 reflecting the system service ability for a user with QoS exponent $\theta$ depending on the queueing delay bound and delay bound violation probability \cite{19, 20}, and $S^E$ is the effective bandwidth depending on the statistics of packet arrival \cite{21, 22}. \eqref{18b} is the QoS requirement of each user \cite{23}, and \eqref{18b} is the total transmit power constraint.

To ensure the ultra-low transmission delay, we consider a short frame structure as in \cite{23}, where each frame is with duration $T_f$ and consists of a duration for data transmission and a duration for signalling. The duration for data transmission in each frame is $\tau$. Since the delay bound is typically shorter than the channel coherence time, time diversity cannot be exploited. To guarantee the transmission reliability within the delay bound, we assign each user with different subcarriers in adjacent frames \cite{23}. When the frequency interval between adjacent subcarriers is larger than the coherence bandwidth, the small scale channel gains of a user among frames are independent. Effective capacity can be expressed as $C_k^E = -\frac{1}{\theta} \ln \mathbb{E}_{g_k} \{e^{-\theta s_k}\}$ packets/frame \cite{23}, and $s_k$ is the achievable rate of the $k$th user that can be approximated as \cite{24}

\begin{equation}
s_k \approx \frac{\tau B_k}{\mu \ln 2} \left[ \ln \left( 1 + \frac{\alpha_k P_k g_k}{N_0 B_k} \right) - \frac{Q^{-1}_G(\epsilon_k)}{\sqrt{\tau B_k}} \right],
\end{equation}

where $\mu$ is the packet size, $\alpha_k$ and $g_k$ are respectively the large-scale channel gain and two norm of the channel vector of the $k$th user, $\epsilon_k$ is the decoding error probability, and $Q^{-1}_G(\cdot)$ is the inverse of the Gaussian Q-function.

From the KKT conditions, the optimal solution of the problem in \eqref{18} should satisfy

\begin{align}
\sum_{k=1}^{K} P_k^* &= P_{\text{max}}, \quad \text{(20a)} \\
C_k^E &= -\frac{1}{\theta} \ln \mathbb{E}_{g_k} \{e^{-\theta s_k(g_k, \alpha_k, P_k^*, B_k^*)}\} = S^E, \quad \text{(20b)}
\end{align}

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

Since the effective capacity is hard to derive, we can not obtain closed-form solution of problem \eqref{18}. When channel distribution is unknown, the solution cannot be obtained even with numerical algorithms. Therefore, we resort to DNNs to learn the mapping from environment parameters to $P_k^*$ and $B_k^*$. We can see from \eqref{20b} that the effective capacity of the $k$th user depends on $\alpha_k$, and hence the environment parameters are the large scale channel gains.
Denote the optimal power allocation policy as \( p^* = P'(\alpha, K) \) and the bandwidth allocation policy as \( B^* = B'(\alpha, K) \), where \( p^* = [P^*_1, ..., P^*_K] \), \( B^* = [B^*_1, ..., B^*_K] \), and \( \alpha = [\alpha_1, ..., \alpha_K] \). Since the multivariate functions \( P'(\alpha, K) \) and \( B'(\alpha, K) \) are not affected by the order of the users, both are PE policies and hence can be respectively learned by GNNs.

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

Due to the total power constraint shown in (20a), \( P^*_k \) decreases with \( K \). As a result, \( B^*_k \) should increase with \( K \) to ensure the QoS in (20b). This indicates that the two policies do not satisfy the size-invariance property.

1) Size Generalization by Pre-training “Activation Function”: \( P'(\alpha, K) \) cannot be expressed as a composite function in closed-form as in (15), but it is not size-invariant due to the same reason as \( F_3^*(g, K) \): the summation constraint, which is also a kind of prior knowledge. Inspired by the analysis in section IV-A2, we can choose softmax as the activation function of the output layer of \( \mathcal{P}_P(\alpha; \theta_P) \) in order for the mean-GNN generalizable to \( K \).

\( B'(\alpha, K) \) also cannot be expressed as a composite function in closed-form, which is not size-invariant due to the QoS constraint and its dependence with \( P^*_k \). Inspired by the structure of \( \mathcal{P}_P(g; \theta_P) \) in Fig. 3, we conceive the following approach to enable mean-GNN generalizable to \( K \) for learning \( B'(\alpha, K) \). We first pre-train an “activation function” to find the scaling law of \( B'(\alpha, K) \) with \( K \). 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. Such an approach is based on a conjecture that \( B'(\alpha, K) \) is a composite function of an inner function independent of \( K \) and an outer function dependent on \( K \).

Given a power allocation policy, \( B'(\alpha, K) \) can be degenerated into a function of \( \alpha_k \) and \( K \) when \( K \to \infty \) as shown in (2) i.e., \( B^*_k = B'^*(\alpha_k, K) \). 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 find the function \( B'^*(\alpha_k, K) \) for all possible values of \( \alpha_k \) and \( K \), we introduce a FNN denoted as \( B'^*(\alpha_k, K; \theta_v) \), where \( \theta_v \) is the model parameters. The structure of \( B'^*(\alpha_k, K; \theta_v) \) is shown in Fig. 4, where the input is \( [\alpha_k, K] \), and the output is \( \hat{B}'^*_k \).

To reduce the cost for training, we pre-train \( B'^*(\alpha_k, K; \theta_v) \) and assume equal power allocation during the pre-training. The samples for \( \alpha_k \) 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 supervised manner, where the model parameters are found from

$$
\theta_v = \arg \min_{\theta_v} \mathbb{E}_{\alpha_k, K} \left\{ (\hat{B}^v(\alpha_k, K; \theta_v) - B^{\text{true}})^2 \right\},
$$

where $\hat{B}^v(\alpha_k, K; \theta_v)$ denotes the function learned by $B^v(\alpha_k, K; \theta_v)$, and $B^{\text{true}}$ is the label obtained by numerically solving the equation in (20b) with bi-section method by setting $P_k^* = \frac{P_{\text{max}}}{K}$ and estimating the expectation as empirical mean from samples. The FNN can also be pre-trained in unsupervised manner, where the model parameters are found from

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

where the loss function is from the KKT condition in (20b).

The structure of $\mathcal{B}^v(\alpha_k, K; \theta_v)$ for learning $B^v(\alpha_k, K)$ with size generalization ability is shown in Fig. 5. It composes a mean-GNN with SI-AFs in hidden layers for learning the size-invariant inner function (i.e., the mapping from $\alpha_k$ to $\hat{B}_k'$) 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 $\hat{B}_k \triangleq \hat{B}_k' \cdot \hat{B}_k^v$. To ensure $\hat{B}_k \geq 0$, Softplus is used at the output layer of $B^v$, before the pre-trained activation function of the mean-GNN.

It is worthy to note that the prior knowledge for the bandwidth allocation problem (i.e., the optimal solution should satisfy the KKT condition) has been embedded in the pre-trained FNN, which serves as the inductive bias for aligning the size-scaling law of $\mathcal{P}_B(\alpha; \theta_B)$ to the size-scaling law of the policy.

2) Unsupervised Learning for the Policies: Since the closed-form expression of the effective capacity is unable to be derived in general cases, the labels for supervised learning can not be obtained by solving problem (18). Moreover, the constraints can not be ensured to be satisfied by supervised learning. To control the stringent constraint in URLLC, we train the GNNs in
unsupervised manner [23]. To this end, we first transform (18) into its primal-dual problem as

\[
\max_{\lambda_k} \min_{P_k, B_k} L(\alpha, P_k, B_k, \lambda_k) \triangleq \sum_{k=1}^{K} \left[ B_k + \lambda_k (E_{\theta_k} \{e^{-\theta_u k}\} - e^{-\theta_S E}) \right]
\]

\[\text{(23)}\]

s.t. \[\sum_{k=1}^{K} P_k \leq P_{\text{max}}, \quad P_k \geq 0, B_k \geq 0, \lambda_k \geq 0, \]

\[\text{(23a), (23b)}\]

where \(L(\alpha, P_k, B_k, \lambda_k)\) is the Lagrangian function of problem (18), and \(\lambda_k\) is the Lagrangian multiplier for the QoS constraint.

According to the proof in [23], the variable optimization problem in (23) can be equivalently transformed into the following functional optimization problem,

\[
\max_{\lambda_k(\alpha)} \min_{P'_k(\alpha, K), B'_k(\alpha, K)} \mathbb{E}_\alpha \{L(\alpha, P'_k(\alpha, K), B'_k(\alpha, K), \lambda_k(\alpha))\}
\]

\[\text{(24)}\]

s.t. (23a), (23b).

Since the optimal Lagrangian multiplier function is not affected by the order of users, we can use a GNN \(P_\lambda(\alpha; \theta_\lambda)\) to approximate \(\lambda^*_k(\alpha)\), where \(\alpha\) is the input and \(\theta_\lambda\) is the model parameters. Denote the output as \([\hat{\lambda}_1, ..., \hat{\lambda}_K]\). By respectively replacing \(P'_k(\alpha, K), B'_k(\alpha, K)\) and \(\lambda_k(\alpha)\) in (24) by the functions \(\hat{P}'_k(\alpha, K; \theta_P), \hat{B}'_k(\alpha, K; \theta_B)\) and \(\hat{\lambda}_k(\alpha; \theta_\lambda)\) learned by three GNNs, the optimal model parameters of the GNNs can be found from the following problem,

\[
\max_{\theta_\lambda} \min_{\theta_P, \theta_B} \mathbb{E}_\alpha \left\{ L(\alpha, \hat{P}_k(\alpha; \theta_P), \hat{B}_k(\alpha; \theta_B), \hat{\lambda}_k(\alpha; \theta_\lambda)) \right\}
\]

\[\text{(25)}\]

s.t. (23a), (23b).

The constraint in (23a) can be satisfied by selecting \texttt{softmax} as the activation function in the output layer of \(P_\lambda(\alpha; \theta_P)\), and the constraint in (23b) can be satisfied by selecting \texttt{softplus}
as the activation function in the output layer of \( P_B(\alpha; \theta_B), B^u(\alpha_k, K; \theta_v) \) and \( P_\lambda(\alpha; \theta_\lambda) \).

The joint training procedure for \( P_P(\alpha; \theta_P), P_B(\alpha; \theta_B) \) and \( P_\lambda(\alpha; \theta_\lambda) \) is summarized in Algorithm 1, where \( t \) denotes the number of iterations, \( g \triangleq [g_1, ..., g_K] \), \( \delta_t \) is the learning rate, \(|\cdot|\) denotes cardinality, \( \nabla_{xy} = [\nabla_x y_1, ..., \nabla_x y_K], \nabla_x y_i = [\frac{\partial y_i}{\partial x_1}, ..., \frac{\partial y_i}{\partial x_K}]^T \) and \( L^t(\boldsymbol{g}, \alpha, \hat{P}_k, \hat{B}_k, \hat{\lambda}_k) \triangleq \sum_{k=1}^{K} \left[ \hat{B}_k + \hat{\lambda}_k(e^{-\theta_{sk}} - e^{-\theta_{S^E}}) \right] \).

**Algorithm 1 Training Procedure.**

1: Initialize \( P_P(\alpha; \theta_P), P_B(\alpha; \theta_B) \) and \( P_\lambda(\alpha; \theta_\lambda) \) with random parameters \( \theta_P, \theta_B \) and \( \theta_\lambda \).
2: for \( t = 1, 2, ... \) do
3: Sample a batch of \( (\boldsymbol{g}, \alpha) \) as \( S \).
4: Input \( \alpha \) into \( P_P(\alpha; \theta_P), P_B(\alpha; \theta_B) \) and \( P_\lambda(\alpha; \theta_\lambda) \) to obtain \( \hat{P}_k, \hat{B}_k' \) and \( \hat{\lambda}_k \), respectively.
5: for \( k = 1, ..., K \) do
6: Input \([\alpha_k, K]\) into the pre-trained \( B^u(\alpha_k, K; \theta_v) \) to obtain \( \hat{B}_k^u \).
7: Scale \( \hat{B}_k' \) with \( \hat{B}_k^u \) to obtain \( \hat{B}_k \), i.e., \( \hat{B}_k = \hat{B}_k' \cdot \hat{B}_k^u \).
8: end for
9: Update the parameters \( \theta_P, \theta_B \) and \( \theta_\lambda \) via stochastic gradient descend method as follows,

\[
\theta_P^{t+1} \leftarrow \theta_P^t - \frac{\delta_t}{|S|} \sum_{(\boldsymbol{g}, \alpha) \in S} \nabla_{\theta_P}P_P(\alpha; \theta_P) \frac{\partial L^t(\boldsymbol{g}, \alpha, \hat{P}_k, \hat{B}_k, \hat{\lambda}_k)}{\partial \hat{P}_k},
\]

\[
\theta_B^{t+1} \leftarrow \theta_B^t - \frac{\delta_t}{|S|} \sum_{(\boldsymbol{g}, \alpha) \in S} \nabla_{\theta_B}P_B(\alpha; \theta_B) \frac{\partial L^t(\boldsymbol{g}, \alpha, \hat{P}_k, \hat{B}_k, \hat{\lambda}_k)}{\partial \hat{B}_k},
\]

\[
\theta_\lambda^{t+1} \leftarrow \theta_\lambda^t + \frac{\delta_t}{|S|} \sum_{(\boldsymbol{g}, \alpha) \in S} \nabla_{\theta_\lambda}P_\lambda(\alpha; \theta_\lambda) \frac{\partial L^t(\boldsymbol{g}, \alpha, \hat{P}_k, \hat{B}_k, \hat{\lambda}_k)}{\partial \hat{\lambda}_k}.
\]
10: end for

V. SIMULATION RESULTS

In this section, we evaluate the size generalization performance of the proposed mean-GNNs for learning the power and bandwidth allocation in URLLC.

A. System Setup

The cell radius is 250m. All users are randomly located in a road with the 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. The maximal transmit power of the BS and the number of antennas are respectively set as \( P_{\max} = 43 \text{ dBm} \) and \( N_t = 8 \). The single-sided noise spectral density is \( N_0 = -173 \text{ dbm/Hz} \). The required overall packet loss probability and the delay bound are
respectively \( \epsilon_{\text{max}} = 10^{-5} \) and \( D_{\text{max}} = 1 \) ms. After subtracting the downlink transmission delay and the decoding delay from the delay bound, the queueing delay is bounded by 0.8 ms \([23]\). As in \([25]\), the decoding error probability \( \epsilon^c_k \) and the queueing delay violation probability \( \epsilon^q_k \) are set as \( \epsilon^c_k = \epsilon^q_k = \epsilon_{\text{max}}/2 \). 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 \([26]\). The packets arrive at the BS according to Poisson process with average arrival rate \( a = 0.2 \) packets/frame. Then, \( S^E = \frac{a}{\theta} (e^\theta - 1) \) packets/frame and \( \theta = \ln[1 - \frac{\ln(\epsilon_{\text{max}}/2)}{aD_{\text{max}}/T_f}] \) \([25]\).

When the system parameters such as \( P_{\text{max}}, N_t, \tau, T_f, \epsilon_{\text{max}}, D_{\text{max}} \) and user locations change, similar results can be obtained and hence are not provided for conciseness.

B. Sample Generation and Fine-tuned Hyper-parameters

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

Since the loss function in \((21)\) is simpler than \((22)\) and the label for pre-training \( \mathcal{B}^v \) can be easily obtained by solving \((20b)\) with training samples, we use supervised learning to train this FNN. Denote one sample for pre-training \( \mathcal{B}^v \) as \([\alpha_k, K, \mathcal{B}^v] \), where \( \alpha_k \) is obtained from the path loss model with randomly generated locations, and \( \mathcal{B}^v \) is the label. We generate 4,000 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 \((g, \alpha)\), which is a realization of the small and large-scale channel gains of all users. In each realization, \( g \) is randomly generated from Rayleigh distribution and \( \alpha \) is obtained from the path loss model with randomly generated locations of all users. The training, validation and test sets are generated according to Table. II. In particular, 2000 samples are generated for \( K = 10 \) (i.e., all these samples for the vectors \( g \) and \( \alpha \) 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 \( g \) and \( \alpha \) are scalars, 100 samples for \( g \) and \( \alpha \) 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. These three datasets are used as follows unless otherwise specified.

The fine-tuned hyper-parameters are shown in Table. II, where \([*_1, *_2, \ldots, *_L]\) denotes that a DNN is with \(*_l\) neurons in the \( l \)th hidden layer. Considering that the large-scale channel gains are usually small, we scale them as \( \ln(\alpha) + 30 \). In order to make the input variables being in the same order of magnitude when pre-training \( \mathcal{B}^v \), we also scale \( K \) as \( K/200 \).
TABLE I
GENERATING SAMPLES IN TRAINING, VALIDATION AND TEST SETS.

|                  | Sizes $K$ in samples | Number of samples generated for each $K$ | Total number of samples |
|------------------|----------------------|------------------------------------------|-------------------------|
| Training set     | [10]                 | 2,000                                    | 2,000                   |
| Validation set   | [10]                 | 200                                      | 200                     |
| Test set         | [1, 2, 5, 10, 50, 100, 200] | 100                                      | 700                     |

TABLE II
HYPER-PARAMETERS.

| Hyper-Parameters | GNN | FNN $B^v$ |
|------------------|-----|-----------|
| Number of neurons in hidden layers | $[4K]$ | $[200, 100, 100, 50]$ |
| Activation function of the hidden layers | Leaky ReLU (i.e., $y = x, x \geq 0$, $y = 0.01x, x < 0$) | |
| Activation function of the output layer | Softmax | Softplus |
| Learning rate | $0.01$ | $0.01$ |
| Batch size | 10 | 100 |
| Epochs | 5000 | 2500 |

C. Size Generalization Performance

The system performance is measured by the availability and total bandwidth achieved by $[\hat{P}_1, ..., \hat{P}_K]$ and $[\hat{B}_1, ..., \hat{B}_K]$. Availability is the percentage of the users with satisfied QoS among all the users in the system that is one of the key performance metrics for URLLC [27], and total bandwidth is the sum of the bandwidth of all users achieved by a learned policy. The two metrics are respectively defined as,

$$A_K \triangleq \frac{1}{K|\mathcal{N}_K|} \sum_{(g, \alpha) \in \mathcal{N}_K} \sum_{k=1}^{K} I(\epsilon_k < \epsilon_{\text{max}}), \quad W^v_K \triangleq \frac{1}{|\mathcal{N}_K|} \sum_{(g, \alpha) \in \mathcal{N}_K} \sum_{k=1}^{K} \hat{B}_k \text{ (MHz)},$$

where $\mathcal{N}_K$ is the subset in the test set containing the samples for the channel vectors with the same size, and $\epsilon_k$ is the packet loss probability of the $k$th user achieved by a learned policy.

To reflect the impact of 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 randomly initialized while the test set remains unchanged. The values of the two metrics are obtained by selecting the second worst test results with 10 well-trained DNNs, hence are with confidence level of 90%.

1) Reliability Controlling: The learned solution is unable to satisfy the QoS with probability one. As demonstrated in [23], the packet loss probability can be ensured by setting a more...
stringent requirement for reliability than the required reliability $\epsilon_{\text{max}}$ during training, i.e. $\epsilon_D < \epsilon_{\text{max}}$. With such a conservative design, even if the overall packet loss probability achieved by the learnt solution exceeds $\epsilon_D$, the probability of violating the original requirement $\epsilon_{\text{max}}$ is low (e.g., the availability is 1.00 for $\epsilon_D = 6 \times 10^{-6}$ as shown in the following table).

The impacts of different values of $\epsilon_D$ on the system performance are shown in Table. III. We can see that by reducing $\epsilon_D$, the availability can be remarkably improved with a slight increase of the total bandwidth. For $\epsilon_D = 6 \times 10^{-6}$, the availability achieves 1.00 with a bandwidth loss $53.14 - 51.81 \approx 2.57\%$ from $\epsilon_D = 10^{-5}$ when $K = 200$ in the test set. Recalling that $K = 10$ in training set as in Table I the results in Table. III indicate that the required reliability $\epsilon_{\text{max}}$ can be ensured even when the well-trained GNNs are tested in the scenario with different number of users from the training set.

### TABLE III
**IMPACTS OF THE VALUES OF $\epsilon_D$ ON SYSTEM PERFORMANCE.**

| $\epsilon_D$ | $K = 1$ | $K = 2$ | $K = 5$ | $K = 10$ | $K = 50$ | $K = 100$ | $K = 200$ |
|--------------|---------|---------|---------|---------|---------|---------|---------|
|              | $A_K$   | $W_K$   | $A_K$   | $W_K$   | $A_K$   | $W_K$   | $A_K$   |
| $10^{-5}$    | 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   |
| $8 \times 10^{-6}$ | 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   |
| $6 \times 10^{-6}$ | **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** |
| $5 \times 10^{-6}$ | 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   |

2) **Impact of Small $K$:** In previous analytical analysis, $K$ is assumed approaching infinity. In what follows, we reduce the value of $K$ in training set to show that the proposed method can still be applied for small number of users. In Table. IV we show the minimal number of samples and the time required to train the GNNs (i.e., sample complexity and time complexity) 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 size of 1, 2, 5, 10, 50, 100 or 200 in each scenario. It is shown that the learned policies by the proposed GNNs can achieve good generalization performance for large size (e.g., $K = 200$) even trained with samples of small size (e.g., $K = 2$), despite that the number of training samples and training time increase with the decreasing sizes of training samples. The increased training complexity comes from a fact that the output of the pre-trained FNN, $B_v$, is not an accurate approximate of $B_k^*$ when $K$ is small, which calls for more training samples to make $P_B$ to learn the policy accurately.

3) **Performance Comparison:** To evaluate the performance of the proposed GNN in terms of size generalization, we compare the system performance of the following three DNNs.
TABLE IV
TRAINING COMPLEXITY FOR ACHIEVING THE AVAILABILITY OF 1.00, $\epsilon_D = 6 \times 10^{-6}$.

| Size $K$ of training samples | Number of training samples | Training time (s) |
|-----------------------------|---------------------------|-------------------|
| [2]                         | 5,000                     | 1,290.32          |
| [5]                         | 4,000                     | 1,289.91          |
| [8]                         | 3,000                     | 1,185.11          |
| [10]                        | 2,000                     | 875.43            |

- “P-GNN”: This is the proposed GNNs, where the power allocation policy is learned by $P_P$ with $\text{softmax}$ in output layer and the bandwidth allocation policy is learned by $P_B$ with the pre-trained “activation function” at the output layer, i.e., $\hat{B}_k = \hat{B}_k^f \hat{B}_k^v$.
- “M-GNN”: This is existing mean-GNNs, where the power allocation policy is learned by $P_P$ with $\text{softmax}$ in output layer, but the bandwidth allocation policy is learned by a mean-GNN with SI-AFs in all layers (which is $P_B$ without pre-trained “activation function” at the output layer, i.e., $\hat{B}_k = \hat{B}_k^f$).
- “FNN”: The power and bandwidth allocation policies are learned by two FNNs. Each FNN is with one hidden layer consisting of 300 neurons. Both the input size and output size 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 that of the GNNs in Table II.

We first provide the minimal number of training samples and the values of $\epsilon_D$ required by each DNN to converge in Table. We where both the availability and total bandwidth can be improved less than 1% by further increasing the training samples and epoches. We also provide the training time to reflect the time complexity of each DNN. Since “FNN” can only be with fixed-size inputs, we use the samples with $K_{\max} = 200$ to train the FNN such that it can be used for the test samples with variable sizes of $K \leq 200$, where the test samples with $K < 200$ are padded with zeros. For “M-GNN”, we respectively use the training samples of sizes $K = 10$ and $K = 200$ for comparison. 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 $\epsilon_D$ chosen for different DNNs are also provided in Table.

The system performance achieved by the learned policies in the test scenarios where the test samples are generated with size of 1, 2, 5, 10, 50, 100 or 200 in each scenario is shown in Table. where “M-GNN$^K$” denotes the “M-GNN” trained by the samples with size of $K$. 
TABLE V

| Methods ($\epsilon_D$) | $K$ in training samples | Training samples | Training time (s) |
|------------------------|--------------------------|------------------|-------------------|
| P-GNN ($6 \times 10^{-6}$) | [10] | 2,000 | 875.43 |
| M-GNN ($5 \times 10^{-6}$) | [10] | 2,000 | 503.81 |
| M-GNN ($3 \times 10^{-6}$) | [200] | 300 | 429.71 |
| FNN ($2 \times 10^{-6}$) | [200] | 100,000 | 49,422.77 |

The number of samples used for training and the corresponding values of $\epsilon_D$ used in training are shown in Table V. Hence, “M-GNN$^{200}$” and “FNN” are trained solely on the maximal-sized samples. 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 generalize, their availability is 1.00 and the required total bandwidth 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 required by “M-GNN$^{10}$” 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 required by “M-GNN$^{200}$” 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 in the test scenarios where $K < 200$, much larger total bandwidth is required than “P-GNN”. Similarly, “FNN” only performs close to “P-GNN” when $K = 200$ in the test set, but is either with lower availability or much larger total bandwidth than “P-GNN” for other values of $K$. While the availability of the policies learned by “M-GNN$^{10}$” and “FNN” can achieve 1.00 for all the test scenarios by setting more conservative value of $\epsilon_D$, much large total bandwidth are required than those shown in the table (not shown due to the lack of space).

These results show that “P-GNN” achieves good size generalization performance. This 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.

VI. CONCLUSION

In this paper, we revealed the size generalization mechanism of GNNs when learning resource allocation policies. We showed that GNNs are not always size generalizable, and solely satisfying
the PE property does not ensure a GNN generalizable to input sizes. Our findings suggest that aggregators and activation functions are key design choices for size generalization with GNNs, which impose an inductive bias for learning the function with respect to the input sizes. We used the mean-GNN with size-invariant activation functions as an example to show a size generalization condition: a GNN can generalize to the size of a PE policy if the size-scaling law of the GNN is the same as the size-scaling law of the PE policy, from which we interpreted why several GNNs considered in literature can generalize well to different problem scales. To show how to make GNNs generalizable to the scales of resource allocation problems, we considered power and bandwidth allocation policies. For the power allocation policy, we showed that it can be learnt by mean-GNN with size generalization ability if selecting appropriate activation function at the output layer. For the bandwidth allocation policy, we proposed a method to make the GNN generalizable to size by scaling its output with a pre-trained “activation function”. The activation function is selected or pre-trained based on the prior knowledge for the power and bandwidth problem, which is the sum constraint or the KKT condition. Simulation results showed that the proposed GNN can be applied for allocating power and bandwidth to different numbers of users when it is trained with samples for given number of users.

APPENDIX A

PROOF OF PROPOSITION 1

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

When \(K \to \infty\) and the elements in \(x\) are i.i.d., each element in \([x^d_j]_{j\neq k}\) approaches to a deterministic function of \(K\) [28]. Therefore, \(\sum_{j\neq k} \phi(x^d_j)\) can be asymptotically expressed as a
function of $K$, i.e., $\sum_{j \neq k} \phi(x^d_j) \rightarrow \bar{f}(K)$. Then, from (I) we obtain

$$y_k = \bar{f}(x_k, \sum_{j \neq k} \phi(x^d_j)) \rightarrow \bar{f}(x_k, \bar{f}(K)) \triangleq f(x_k, K).$$  \hspace{1cm} (A.1)

This completes the proof.

**APPENDIX B**

**PROOF OF PROPOSITION 2**

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

When $K \rightarrow \infty$ and $x_k, k = 1 \cdots, K$ are i.i.d., we first prove a statement that $h^l_k, k = 1, \cdots, K$ are i.i.d. is true for $l = 0, 1, \cdots, L$ using mathematical induction.

When $l = 0$, we have $h^0_k = x_k$. Since $x_k, k = 1 \cdots, K$ 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

$$h^{n+1}_k = \sigma(U^{n+1}h^n_k + \frac{V^{n+1}}{K} \left( \sum_{j \neq k} h^n_j \right) + c^{n+1}).$$  \hspace{1cm} (B.1)

Since $h^n_k, k = 1, \cdots, K$ are i.i.d., $\frac{1}{K} \left( \sum_{j \neq k} h^n_j \right) = \mathbb{E}^K(h^n_k) = \mathbb{E}^K(h^n)$ when $K \rightarrow \infty$ [29], where $\mathbb{E}^K(\cdot)$ denotes expectation. Then, by substituting $\mathbb{E}^K(h^n)$ into (B.1), we have

$$h^{n+1}_k = \sigma(U^{n+1}h^n_k + V^{n+1}\mathbb{E}^K(h^n) + c^{n+1}),$$  \hspace{1cm} (B.2)

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

$$h^{l}_k = \sigma(U^l h^{l-1}_k + V^l \mathbb{E}^K(h^{l-1}) + c^l).$$  \hspace{1cm} (B.3)

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

$$\hat{y}_k = \sigma\left(U^{L+1}\sigma\left(\ldots U^2\sigma(U^1 x_k + V^1 \mathbb{E}^K(x) + c^1) + V^2 \mathbb{E}^K(h^1) + c^2\ldots\right) + V^{L+1}\mathbb{E}^K(h^L) + c^{L+1}\right)$$

$$\triangleq \hat{q}(x_k, K).$$  \hspace{1cm} (B.4)

This completes the proof.
APPENDIX C

PROOF OF PROPOSITION 3

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

\[
p^K(h^l) = p^{K'}(h^l)
\]  
(C.1)

is true for \( l = 0, 1, ..., L \) using mathematical induction.

When \( l = 0 \), we have \( h^0 = x \), hence \( \text{(C.1)} \) is true.

Assume that \( \text{(C.1)} \) is true for \( l = n \). In order to prove \( p^K(h^{n+1}) = p^{K'}(h^{n+1}) \), we need to prove that \( \text{Pr}^K \{ h^{n+1} \leq h \} = \text{Pr}^{K'} \{ h^{n+1} \leq h \} \) is true for any \( h \), where \( \text{Pr}^K \{ \cdot \} \) denotes probability. In other words, we need to prove \( \text{Pr}^K \{ h^{n+1}_k \leq h \} = \text{Pr}^{K'} \{ h^{n+1}_k \leq h \} \). When \( K, K' \to \infty \), from \( \text{(B.2)} \) we can derive that

\[
\text{Pr}^K \{ h^{n+1}_k \leq h \} = \text{Pr}^{K'} \{ h^{n+1}_k \leq A^K(h) \},
\]  
(C.2)

\[
\text{Pr}^{K'} \{ h^{n+1}_k \leq h \} = \text{Pr}^K \{ h^{n}_k \leq A^{K'}(h) \},
\]  
(C.3)

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

Since \( p^K(h^n) = p^{K'}(h^n) \), we have \( \mathbb{E}^K(h^n) = \mathbb{E}^{K'}(h^n) \). When \( \sigma(\cdot) \) does not dependent on \( K \), we have \( A^K(h) = A^{K'}(h) \). Hence, we can obtain

\[
\text{Pr}^K \{ h^n_k \leq A^K(h) \} = \text{Pr}^{K'} \left\{ h^n_k \leq A^{K'}(h) \right\},
\]

which indicates that \( \text{Pr}^K \{ h^{n+1}_k \leq h \} = \text{Pr}^{K'} \{ h^{n+1}_k \leq h \} \) (i.e., \( p^K(h^{n+1}) = p^{K'}(h^{n+1}) \)) according to \( \text{(C.2)} \) and \( \text{(C.3)} \).

Hence, \( \text{(C.1)} \) is true for \( l = 0, ..., L \). Thus, \( \mathbb{E}^K(h^l) = \mathbb{E}^{K'}(h^l) \). This completes the proof.

APPENDIX D

PROOF OF PROPOSITION 4

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

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

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

Analogous to Appendix B, we can derive that $h^l_k, k = 1, \cdots, K$ are i.i.d. for $l = 0, ..., L$ when $K \to \infty$ since $x_k, k = 1, \cdots, K$ are i.i.d.. Then, we have $\max_{j \neq k}^K h^{l-1}_j = D_{h^{l-1}}^{-1}(\frac{K}{K+1})$ according to [28], where $D_{h^{l-1}}^{-1}(\cdot)$ denotes the inverse of the probability distribution function of $h^{l-1}$. Then, the relation between the output and the input of the GNN can be expressed as

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

Since probability distribution function of continuous random variable is a strictly monotonically increasing function, $D_{h^{l-1}}^{-1}(\frac{K}{K+1})$ also strictly monotonically increases. Then, we have $D_{h^{l-1}}^{-1}(\frac{K}{K+1}) \neq D_{h^{l-1}}^{-1}(\frac{K'}{K'+1})$ 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 sum-aggregator is not invariant to the input size. For this GNN, the relation between $h^l_k$ and $h^{l-1}_k$ can be expressed as

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

When $K \to \infty$, again considering that $x_k$ is i.i.d. among $k$, we can derive that $h^l_k$ is also i.i.d. across $k$. According to the central-limit theorem, $\sum_{j \neq k}^K h^{l-1}_j$ follows a normal distribution, whose mean and variance increase with $K$. This suggests that $\sum_{j \neq k}^K h^{l-1}_j$ depends on $K$, which can be expressed as $\tilde{h}^{l-1}(K)$. Then, the relation between the output and the input of the GNN can be expressed as

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

which is no longer equal to $\hat{q}(x_k, K')$. This completes the proof.
REFERENCES

[1] L. Liang, H. Ye, G. Yu, and G. Y. Li, “Deep-learning-based wireless resource allocation with application to vehicular networks,” Proceedings of the IEEE, vol. 108, no. 2, pp. 341–356, Feb. 2020.
[2] H. Sun, X. Chen, Q. Shi, M. Hong, X. Fu, and N. D. Sidiropoulos, “Learning to optimize: training deep neural networks for interference management,” IEEE Trans. on Signal Processing, vol. 66, no. 20, pp. 5438–5453, Oct. 2018.
[3] M. Eisen and A. Ribeiro, “Optimal wireless resource allocation with random edge graph neural networks,” IEEE Trans. on Signal Processing, vol. 68, no. 10, pp. 2977–2991, April 2020.
[4] D. Liu, C. Sun, C. Yang, and L. Hanzo, “Optimizing wireless systems using unsupervised and reinforced-unsupervised deep learning,” IEEE Netw., vol. 34, no. 4, pp. 270–277, July/Aug. 2020.
[5] A. Zappone, M. Di Renzo, and M. Debbah, “Wireless networks design in the era of deep learning: Model-based, AI-based, or both?” IEEE Trans. on Commun., vol. 67, no. 10, pp. 7331–7376, Oct. 2019.
[6] F. Restuccia and T. Melodia, “Deep learning at the physical layer: System challenges and applications to 5G and beyond,” IEEE Commun. Mag., vol. 58, no. 10, pp. 58–63, Oct. 2020.
[7] N. Kato, B. Mao, F. Tang, Y. Kawamoto, and J. Liu, “Ten challenges in advancing machine learning technologies toward 6G,” IEEE Wireless Commun., vol. 27, no. 3, pp. 96–103, June. 2020.
[8] T. M. Mitchell, “The need for biases in learning generalizations.” Tenical Report, Rutgers University, New Brunswick, 1980.
[9] S. Geman, E. Bienenstock, and R. Doursat, “Neural networks and the bias/variance dilemma,” Neural Computation, vol. 4, no. 1, pp. 1–58, 1992.
[10] Y. Shen, Y. Shi, J. Zhang, and K. B. Letaief, “Graph neural networks for scalable radio resource management: Architecture design and theoretical analysis,” IEEE J. Sel. Areas Commun., vol. 39, no. 1, pp. 101–115, Jan. 2021.
[11] C. Sun, J. Wu, and C. Yang, “Improving learning efficiency for wireless resource allocation with symmetric prior,” IEEE Wireless Commun. Mag., Early access.
[12] J. Guo and C. Yang, “Learning power allocation for multi-cell-multi-user systems with heterogeneous graph neural network,” IEEE Trans. Wireless Commun., vol. 21, no. 2, pp. 884–897, Feb. 2022.
[13] N. Keriven and G. Peyrè, “Universal invariant and equivariant graph neural networks,” Advances in Neural Information Processing Systems, 2019.
[14] Z. Manzil, K. Satwik, R. Siamak, P. Barnabas, S. Ruslan, and S. Alexander, “Deep sets,” Advances in Neural Information Processing Systems, 2017.
[15] M. Lee, G. Yu, and G. Y. Li, “Graph embedding-based wireless link scheduling with few training samples,” IEEE Trans. Wireless Commun., vol. 20, no. 4, pp. 2282–2294, April 2021.
[16] T. Chen, X. Zhang, M. You, G. Zheng, and S. Lambotharan, “A GNN based supervised learning framework for resource allocation in wireless IoT networks,” IEEE Internet of Things J., vol. 9, no. 3, pp. 1712–1724, Feb. 2022.
[17] G. Yehudai, E. Fetaya, E. Meirom, G. Chechik, and H. Maron, “From local structures to size generalization in graph neural networks,” International Conference on Machine Learning, 2021.
[18] A. Sannai, Y. Takai, and M. Cordonnier, “Universal approximations of permutation invariant/equivariant functions by deep neural networks,” arXiv:1903.01939, 2019.
[19] D. Wu and R. Negi, “Effective capacity: A wireless link model for support of quality of service,” IEEE Trans. Wireless Commun., vol. 2, no. 4, pp. 630–643, Jul. 2003.
[20] J. Tang and X. Zhang, “Quality-of-service driven power and rate adaptation over wireless links,” IEEE Trans. on Wireless Commun., vol. 6, no. 8, pp. 3058–3068, Aug. 2007.
[21] F. Kelly, “Notes on effective bandwidths,” *Stochastic networks: theory and applications*, 1996.

[22] L. Liu, P. Parag, J. Tang, and et al., “Resource allocation and quality of service evaluation for wireless communication systems using fluid models,” *IEEE Trans. on Inf. Theory*, vol. 53, no. 5, pp. 1767–1777, May 2007.

[23] C. Sun and C. Yang, “Learning to optimize with unsupervised learning: Training deep neural networks for URLLC,” *IEEE PIMRC*, 2019.

[24] W. Yang, G. Durisi, T. Koch, and et al., “Quasi-static multiple-antenna fading channels at finite blocklength,” *IEEE Trans. Inf. Theory*, vol. 60, no. 7, pp. 4232–4264, July 2014.

[25] C. She, C. Yang, and T. Q. S. Quek, “Joint uplink and downlink resource configuration for ultra-reliable and low-latency communications,” *IEEE Trans. Commun.*, vol. 66, no. 5, pp. 2266–2280, May 2018.

[26] 3GPP, *Study on Scenarios and Requirements for Next Generation Access Technologies*. Technical Specification Group Radio Access Network, Technical Report 38.913, Release 14, Oct. 2016.

[27] P. Popovski and et al., “Deliverable d6.3 intermediate system evaluation results,” [https://metis2020.com/wp-content/uploads/deliverables/METIS_D6.3_v1.pdf](https://metis2020.com/wp-content/uploads/deliverables/METIS_D6.3_v1.pdf), 2014.

[28] H. A. David and H. N. Nagaraja, *Order Statistics*. John Wiley and Sons, 2003.

[29] J. A. Rice, *Mathematical statistics and data analysis*. Cengage Learning, 2006.