IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier

Exploration Degree Bias: The Hidden Influence of Node Degree in Graph Neural Network-based Reinforcement Learning

PETER TARÁBEK[1], DÁVID MATIS[1]
[1]Faculty of Management Science and Informatics, University of Žilina, 010 26 Žilina, Slovakia
Corresponding author: P. Tarábek (e-mail: peter.tarabek@fri.uniza.sk).

This research was supported by the Ministry of Education, Science, Research and Sport of the Slovak Republic under the contract No. VEGA 1/0525/23.

ABSTRACT Graph Neural Networks (GNNs) have demonstrated remarkable performance in tasks involving graph-structured data, but they also exhibit biases linked to node degrees. This paper explores a specific manifestation of such bias, termed Exploration Degree Bias (EDB), in the context of Reinforcement Learning (RL). We show that EDB arises from the inherent design of GNNs, where nodes with high or low degrees disproportionately influence output logits used for decision-making. This phenomenon impacts exploration in RL, skewing it away from mid-degree nodes, potentially hindering the discovery of optimal policies. We provide a systematic investigation of EDB across widely used GNN architectures—GCN, GraphSAGE, GAT, and GIN—by quantifying correlations between node degrees and logits. Our findings reveal that EDB varies by architecture and graph configuration, with GCN and GIN exhibiting the strongest biases. Moreover, analysis of DQN and PPO RL agents illustrates how EDB can distort exploration patterns, with DQN exhibiting EDB under low exploration rates and PPO showing a partial ability to counteract these effects through its probabilistic sampling mechanism. Our contributions include defining and quantifying EDB, providing experimental insights into its existence and variability, and analyzing its implications for RL. These findings underscore the need to address degree-related biases in GNNs to enhance RL performance on graph-based tasks.

INDEX TERMS Exploration Degree Bias, Graph Neural Networks, Message-passing mechanism, Node degree bias, Reinforcement learning

I. INTRODUCTION

Deep machine learning has boosted the research on data mining and pattern recognition. While the data in many machine learning problems is in the form of a sequence or a matrix, there is an increasing number of applications in which the data is represented as a graph. Graph structured data, characterized by nodes and edges, gives us natural versatility to model complex relationships between entities.

Graph neural networks (GNNs) [1]–[4] is a novel machine learning method for graph structures. They have found their usefulness in many different tasks such as recommendation systems [5], [6], natural language processing [7], molecular chemistry [8], [9], drug discovery [10], combinatorial optimization [11]–[13], to name a few, with many of these studies reporting state-of-the-art performance. The fundamental mechanism of GNNs is the message passing [9]. It allows GNNs to capture complex relationships in the graph data structure by iteratively aggregating and updating information

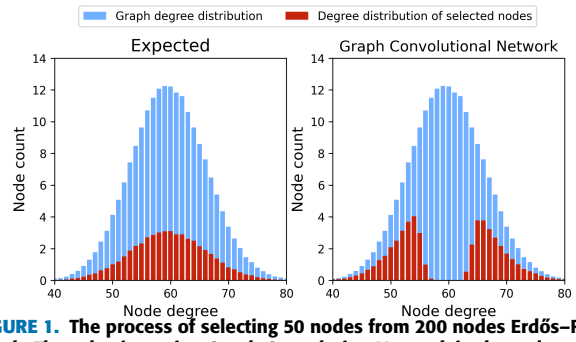

FIGURE 1. The process of selecting 50 nodes from 200 nodes Erdős–Rényi graph. The selection using Graph Convolution Network is skewed towards nodes with lower and higher degrees (right) and does not follow the expected distribution of uniformly selecting all nodes (left).

from neighboring nodes.

We are interested in using GNNs in combination with Reinforcement Learning (RL). RL is a powerful framework for learning complex decision-making policies through interactions with an environment. Modern RL agents frequently employ neural networks to learn a policy, which in many cases defines a probability distribution over possible actions. During training, the agent samples actions from this distribution to *explore* the environment. A central challenge in RL is the trade-off between *exploration* and *exploitation* [14]: the agent must *exploit* its current knowledge to maximize rewards, while also *explore* new actions to discover better policies. Effective exploration is crucial for the agent to find an optimal policy in the long run.

Even though GNNs achieve state-of-the-art results in many tasks, they also have their shortcomings, which include over-smoothing caused by deep networks [15] or by node degree shift between training and real-world data [16]. There are many papers [16]–[19], that analyze degree-related bias in message-passing GNNs. Most of the degree-related bias papers show that low-degree nodes contain insufficient or noisy information [20] and on the other hand high-degree nodes have a larger influence on message passing because they have more links with other nodes [20].

In this paper, we contribute to existing studies on GNN node degree-related biases by examining a specific manifestation of this bias in combination with RL. In node-level decision tasks the last layer of GNN assigns a scalar value (i.e. logit) to every node, which are used in exploration. We show that in GNNs, there is a tendency for the output logit to be either positively or negatively correlated with node degree. Specifically, nodes with higher or lower degrees disproportionately influence the output logits. This degree-based skew (Fig. 1) is a result of the inherent structure of GNNs message passing mechanisms, where node embeddings are aggregated based on neighboring node information.

In RL, where exploration is crucial for discovering optimal policies, this bias has serious implications. RL agents need to explore unknown environments to receive rewards and, subsequently, exploit this knowledge to improve decision-making. However, the degree-based skew in GNNs can result in certain nodes being explored far less frequently. In the early phases of learning—when the neural networks are randomly initialized—this bias can make it almost impossible for the agent to visit specific nodes in the graph, especially those with mid-range degrees. If these under-explored nodes are critical to optimal policies or reward-rich paths, the RL agent may never receive the necessary reward signals to guide it toward those solutions. From the RL point of view, the bias is "hidden" because the skew is not explicitly caused by the exploration policy or reward structure but by the internal characteristics of the GNN architecture itself.

We refer to this phenomenon as Exploration Degree Bias (EDB) to emphasize the imbalance in the agent's exploration caused by node degree during the learning process. We chose this name because it highlights the source of the bias (node degree) and its direct impact on exploration, which is fundamental to RL. By skewing exploration towards high- and low-degree nodes, the bias prevents the agent from sufficiently investigating parts of the environment. EDB bears similarities to primacy bias [21], where early experiences disproportionately influence the learning process, leading to a focus on familiar actions or states. By reinforcing the exploration of high- and low-degree nodes while neglecting mid-range nodes, EDB can exacerbates the effects of primacy bias in RL, limiting the agent's ability to fully explore the environment and discover optimal policies.

Overall, we make the following key contributions:

- We examine and define Exploration Degree Bias (EDB), a specific manifestation of node degree-related bias in GNNs, as it affects systems combined with RL. This highlights a previously unrecognized challenge in GNN-based RL systems, where degree-dependent skew impacts exploration efficiency and policy optimization.
- We demonstrate the existence of correlations between node degree and output logits to varying degrees across widely used GNN architectures [5]: GCN [2], GraphSAGE [3], GAT [22] and GIN [4]. This existence confirms that EDB is a natural property of the message-passing mechanisms in GNNs. We also provide aggregated experimental results in histograms showing how the probability distribution of selecting nodes during exploration is skewed, favoring high-degree or low-degree nodes.
- We examine the effect of EBD on the exploration in DQN [23] and PPO [24] RL agents. By evaluating two distinct RL algorithms, we reveal how degree bias manifests differently based on exploration mechanisms.

II. PRELIMINARIES

In the following sections, we will first present the fundamental definitions of graphs, followed by the basic principles of Graph Neural Networks (GNNs) and the core concepts of Reinforcement Learning (RL). We analyzed the GCN [2], GraphSAGE [3], GAT [22] and GIN [4], which according to Wu *et al.* [25] taxonomy are all part of the Convolutional Graph Neural Network family.

A. GRAPH

We use $\mathcal{G} = (\mathcal{V}, \mathcal{E}, X)$ to denote a graph, where $\mathcal{V} = \{v_1, v_2, ..., v_{|\mathcal{V}|}\}$ is a set of nodes, $\mathcal{E}$ is a set of edges which can be represented by adjacency matrix $A \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$, where $A_{ij} = 1$ if $e_{ij} = (v_i, v_j) \in \mathcal{E}$, otherwise $A_{ij} = 0$. Each node v_i has a feature vector $x_i \in X \subset \mathbb{R}^{|\mathcal{V}|\times d}$, where d represents the dimensions of the features.

B. GRAPH NEURAL NETWORKS

Graph Neural Networks (GNNs) [2], [3], [22] represent a group of neural network architectures specifically designed to learn representations of graph's components - its nodes, edges, and even the entire graph. They use the graph structure and node's features $x_i \in X$ to learn an inner representation of a

node, denoted by h_v, where $v \in \mathcal{V}$. Latest GNNs use message passing [9] also referred to as neighborhood aggregation. During this aggregation, we iteratively update the node's embedding by aggregating embeddings of its neighbors. After k iterations of aggregation, the node's embedding captures the information of its k-hop neighborhood. Formally, the aggregation and update of k-th layer of a GNN is:

$$g_i^{(k)} = \text{AGGREGATE}^{(k)}\Big(h_j^{(k-1)}, \forall j \in \mathcal{N}(i)\Big), \quad (1)$$

$$h_i^{(k)} = \text{UPDATE}^{(k)}\Big(h_i^{(k-1)}, g_i^{(k)}\Big), \quad (2)$$

where $h_i^{(k)}$ is the embedding of node i at the k-th iteration/layer, and $\mathcal{N}(i)$ are the neighbors of node i. To initialize the message passing the $h_i^{(0)}$ is set to the node's features x_i.

The $AGGREGATE^{(k)}(\cdot)$ and $UPDATE^{(k)}(\cdot)$ are arbitrary differentiable functions. The aggregation function must be permutation invariant so that the ordering of nodes does not change the output. *Sum*, *mean*, or *max* are often used. Choosing the right *AGGREGATE* and *UPDATE* function is crucial.

C. GRAPH CONVOLUTIONAL NETWORK

The power of convolution lies in the sharing of learnable weights, thus reducing model size and enhancing efficiency. Graph Convolutional Neural Networks generalize the operation of convolution from grid data to graph data. One of the most basic GNNs based on the operation of convolution is Graph Convolutional Network (GCN) [2], which uses the weighted sum of the neighboring node features. A single layer of GCN is defined as

$$H^{(k+1)} = \sigma\Big(D^{-\frac{1}{2}}\hat{A}D^{-\frac{1}{2}}H^{(k)}W^{(k)}\Big), \quad (3)$$

where $H^{(k)}$ is the matrix of all node features at layer k, $\hat{A} = A + I$ is adjacency matrix with added self-loops, D is the diagonal degree matrix of $\hat{A}$, $W^{(k)}$ is the layer specific learnable weight matrix and $\sigma(\cdot)$ denotes the nonlinear activation function (e.g. ReLU). The node-wise form is

$$h_i^{(k+1)} = \sigma\Big(\sum_{j \in \mathcal{N}(i)\cup\{i\}} \frac{1}{\sqrt{d_i d_j}} h_j^{(k)} W^{(k)}\Big), \quad (4)$$

where d_i denotes the degree of node i, i.e., the count of node's links. The normalization factor $\frac{1}{\sqrt{d_i d_j}}$ ensures that the contributions from neighboring nodes are appropriately scaled, preventing the feature magnitudes from growing too large.

D. GRAPHSAGE

Many real-world problems work with graphs that have millions of nodes. Hamilton et al. [3] propose an aggregation-based inductive representation learning model named GraphSAGE. The key idea is that it learns to generate embedding by sampling and aggregating feature information from the node's local neighborhood of fixed size. Fixed-size neighborhood sampling improves scalability and efficiency, allowing the method to learn on large-scale graphs.

Their method offers several choices for the aggregation function, including the mean aggregator, LSTM aggregator, and the pooling aggregator. When using the mean aggregator, GraphSAGE approximately resembles the GCN [2] model. A layer of GraphSAGE using the mean aggregator can be formulated as:

$$g_i^{(k)} = \text{MEAN}\Big(\{h_i^{(k-1)}\} \cup \{h_j^{(k-1)}, \forall j \in \mathcal{N}(i)\}\Big), \quad (5)$$

$$h_i^{(k)} = \sigma\Big(W^{(k)} g_i^{(k)}\Big), \quad (6)$$

where the node's current representation $h_i^{(k-1)}$ is concatenated with the aggregated fixed-sizes neighbor vector and fed through a fully connected layer with activation function $\sigma(\cdot)$.

E. GRAPH ATTENTION NETWORK

The benefit of attention mechanisms is that they allow dealing with variable-sized inputs, focusing on the most relevant parts of the input. Veličković et al. [22] have designed a Graph Attention Network (GAT) that can learn the weighting function (attention) for the neighboring nodes. Instead of using a non-parametric weighting function like GCN [2], GAT implicitly learns the weights, so that more important nodes receive larger weight. A single layer is defined as:

$$h_i^{(k)} = \sigma\Big(\sum_{j \in \mathcal{N}(i)\cup\{i\}} \alpha_{ij} W^{(k)} h_i^{(k-1)}\Big), \quad (7)$$

where α_{ij} is the attention coefficient, i.e. the importance of node j's features to node i. These coefficients are computed as:

$$e_{ij} = a^{(k)}\big(W^{(k)} h_i^{(k-1)}, W^{(k)} h_j^{(k-1)}\big), \quad (8)$$

$$\alpha_{ij} = \text{softmax}_j\big(e_{ij}\big) = \frac{\exp\big(e_{ij}\big)}{\sum_{v \in \mathcal{N}(i)\cup\{i\}} \exp\big(e_{iv}\big)}, \quad (9)$$

where $a(\cdot)$ is a single-layer feed-forward neural network used to perform self-attention on the nodes.

F. GRAPH ISOMORPHISM NETWORK

Many existing GNNs use a 1-layer perceptron, a linear mapping followed by a non-linear activation function. While it is sufficient in many node classification tasks, there are indeed network neighborhoods that models with 1-layer perceptrons can never distinguish [4]. They show that GCN [2] and GraphSAGE [3] are incapable of distinguishing different graph structures. Xu et al.'s [4] goal was to design maximally powerful GNN while being simple. Graph Isomorphism Network (GIN) has the ability to distinguish different graph structures by mapping them to different representations. A single layer of GIN is defined as

$$h_i^{(k)} = \text{MLP}^{(k)}\Big(\big(1 - \epsilon^{(k)}\big)h_i^{(k-1)} + \sum_{j \in \mathcal{N}(i)} h_j^{(k-1)}\Big), \quad (10)$$

where MLP denotes a multi-layer perceptron and ϵ is either a learnable parameter or a fixed scalar.

G. REINFORCEMENT LEARNING

The RL is fundamentally built upon the Markov Decision Process (MDP). MDPs provide a formal mathematical model, where the formal definition is the tuple $\{\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma\}$ [14]. An agent chooses action $a \in \mathcal{A}$ based on the environment state $s \in \mathcal{S}$ and receives reward r according to a mapping $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. The environment transitions into state s' based on the transition function $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \Delta(\mathcal{S})$ and the interaction continues. The goal is to find policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ that maximizes the expected sum of discounted rewards called returns:

$$\mathbb{E}_\pi \Big[\sum_{t=0}^{\infty} \gamma^t r_t \Big| s_t = s, a_t = a \Big], \tag{11}$$

where the discount factor $\gamma \in [0, 1]$ determines the strength of long-term rewards.

One of the basic methods is Q-Learning [26], which learns action-value function $Q_\pi(s, a) = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r_t | s_t = s, a_t = a]$. Q-learning is an off-policy algorithm, i.e., it improves the policy π through another policy (e.g. ϵ-greedy) that interacts with the environment. These algorithms are able to learn policies from data that another policy has collected. On-policy algorithms such as Asynchronous Actor-Critic [27], or Proximal Policy Optimization (PPO) [24] on the other hand optimize the policy π directly. In both cases, the policy π or its part is approximated using a deep neural network.

An important concept of learning from experience is *exploration* and *exploitation*. This unique challenge does not arise in supervised or unsupervised learning. To maximize rewards, RL agent must prioritize actions that have previously yielded high rewards. However, finding these actions requires experimenting with actions that have not been selected before. The agent must balance *exploitation* of known rewarding actions to accumulate rewards with *exploration* of new actions to improve future decision-making [14].

III. RELATED WORK

Despite the state-of-the-art performance of GNNs across many tasks, they also exhibit performance degradation due to imbalance in data distribution, presence of noise in data, or generalization capabilities for out-of-distribution data [28]. Several studies [16]–[19] have analyzed degree-related biases in GNNs [20], highlighting how low-degree nodes tend to have inadequate and noisy information, making them less influential during the message-passing process [17], while high-degree nodes exhibit better classification performance due to their extensive connections [20].

Liu et al. [19] have studied the bias stemming from varying neighborhood structures and node degrees. They proposed a novel framework DegFairGNN, that works with any neighborhood aggregation-based GNNs. It addresses this bias by learning a debiasing function that generates contexts that balance the structural differences between high-degree and low-degree nodes. As a result, the drastic differences in contextual structures across nodes can be balanced to reach a structurally fair state, enabling the generation of fair node representations. Chen et al. [16] have designed a framework called BA-GNN that deals with the bias caused by the distribution shift between training and testing node distributions. The aim is to learn invariant node representations across different distributions, making the model robust to unknown distribution shifts.

While numerous papers studied degree bias, they predominantly focus on supervised learning. In contrast, our research addresses a different challenge: the exploration imbalance in reinforcement learning, which introduces a unique degree-related bias. To the best of our knowledge, no other work analyzes the direct connection between the node's degree and GNN output in relation to exploration in reinforcement learning.

RL methods have achieved remarkable success across various complex tasks [29]–[31]. These successes are mostly attributed to advances in deep neural networks that approximate the value function or policies, enabling the handling of high-dimensional state and action space [32]. The usage of GNNs gives RL the possibility to solve many more problems in domains such as manufacturing, where they can be applied for controlling interconnected systems of machines efficiently [33], in optical networking to search for better routes [34], or in combinatorial optimization in solving the Vehicle Routing problem [35], [36], or the Traveling Salesman problem [12], [37], [38]. This powerful combination has also been applied to control graph dynamics in scenarios like epidemic spread and power network failures [39].

RL also has its problems. One of the most known is the dilemma between exploration and exploitation. The challenge is to balance between exploration and exploitation, where agents must explore new actions to improve future decision-making while exploiting known rewarding actions to maximize immediate rewards [14]. Another related issue is the primacy bias [21], where agents tend to overfit early experiences, which can negatively impact their learning process and ability to discover optimal policies. This concept is particularly relevant in the context of EDB, where the inherent structure of GNNs skews an agent's exploration towards high- and low-degree nodes, thereby limiting the exploration of mid-range nodes. While EDB stems from the architectural characteristics of GNNs, primacy bias emerges from the agent's tendency to prioritize early experiences. Together, these biases can reinforce each other, further hindering the agent's ability to explore and learn effectively.

IV. EXPERIMENTS

The goal of the experiments is to show the existence of EDB and its influence on the exploration in RL. RL agents use deep neural networks to approximate policy functions. In general, the network assigns a value to every action of how good it is (e.g. Q-value or probability). The higher the value, the better the action. If the network is biased towards certain actions while neglecting others, it can hinder proper exploration. To investigate the EDB we have designed two sets of experiments. The first set focuses on GNN networks in isolation, where we use the correlation between the node's degree and

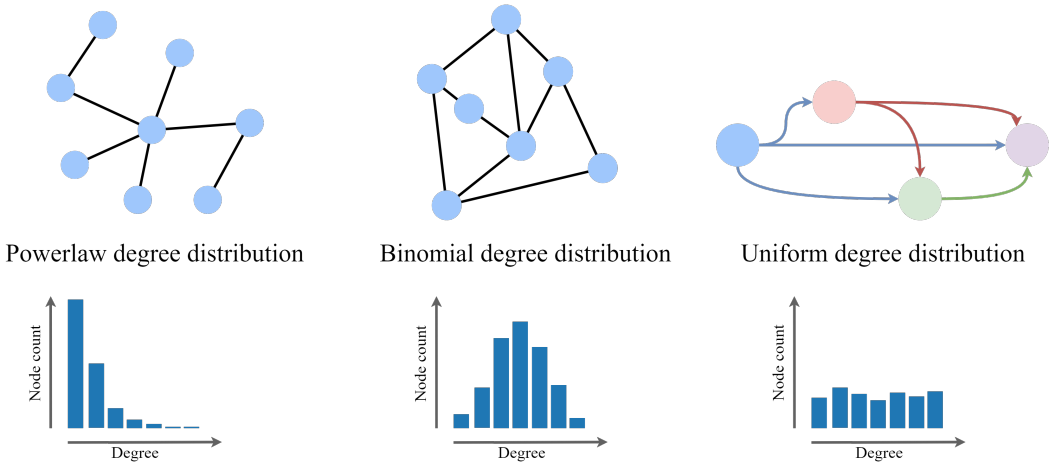

FIGURE 2. Three different graph structures used in experiments. Graphs with power-law degree distribution, binomial degree distribution, and uniform degree distribution.

the network's output to indicate the presence of EDB. To further illustrate this, we employ histograms that visualize the node selection process based solely on the GNN output. The second set of experiments extends this investigation on the exploration policy of DQN [23] and PPO [24] agents.

We have tested GCN, GAT, GraphSAGE, and GIN networks which are commonly used GNNs. Every experiment uses a randomly initialized neural network that represents the early stage of the exploration phase. These networks are composed of 4 GNN layers with ReLU used as the activation function. Each network assigns a scalar value (i.e. logit) to every node, which in terms of RL is interpreted either as Q-value or probability of selecting the node.

To show the existence of EDB we calculated the Pearson's correlation coefficient between the node's degree of randomly generated graph and the logits of a randomly initialized GNN. Using a randomly initialized neural network is crucial for two reasons. First, it allows us to observe the influence of EDB without the biases introduced by a model that has undergone training. This ensures that any variations in the logit values can be directly attributed to the EDB rather than learned behaviors. Second, training in RL typically starts with a randomly initialized network. This setup is essential to accurately assess how EDB manifests in the early stages of learning, which is when exploration is particularly critical. Big positive correlation would suggest that nodes with higher degrees would get bigger Q-values or probability and, thus would be more preferred over the other nodes. We report the mean absolute correlation value along with the standard deviation. To further demonstrate the effect of EDB, we use a histogram visualization of selecting 50 nodes based on their logit values out of a 200-node graph. We perform two types of node selection: (1) randomly sampling 50 nodes to ensure equal probability for all nodes, thus creating the expected distribution and (2) selecting 50 nodes based on the highest output logits by a randomly initialized neural network, as the value of these logits directly influence the Q-values or probabilities in RL. Each histogram displays the node degree distribution of the graph, the expected degree distribution for 50 randomly selected nodes, and the degree distribution for 50 selected nodes by their highest logits. To make the results statistically significant we repeated the experiments 1000 times both for correlation and node selection.

To further show the implications we show the EDB on the DQN and PPO training policies. We use the same network architectures as in the previous experiment, with only the difference of either interpreting the logits as Q-Values in DQN or applying the *softmax* on the output in case of PPO policy. In DQN the network assigns a Q-value (i.e. logit) to every action. The bigger the Q-value, the better the action. DQN is an off-policy agent, meaning it can learn from actions that are not determined by its current policy. During learning, it uses the ϵ-greedy policy, which balances exploration and exploitation. In this policy, $\epsilon \in [0, 1]$ represents the probability of choosing a random action. When $\epsilon = 1$, actions are chosen uniformly at random. In contrast, when $\epsilon = 0$, actions are selected based on the output of the GNN, where the nodes with the highest Q-values (logits) are chosen. In general ϵ-greedy policy starts with $\epsilon = 1$ and gradually lowers the probability of selecting random action. That is why we show experiments of the DQN agent for $\epsilon \in \{0, 0.5, 1\}$. The PPO on the other hand is an on-policy agent and its exploration fully relies on the GNN output. It uses the *softmax* function to convert output logits of actor network into probabilities, which are then used to choose actions.

We demonstrate the EDB on three types of graph structures (Fig. 2) frequently encountered in various problems and show that different GNN methods result in different bias strength across these graph structures. The degree distribution of most real-world graphs (e.g. citation graphs, networking) are power-law [17], [40], [41] where a few nodes have a very high degree, while most nodes have a low degree. According to [42] 49% of graphs have other distribution than power-law. Therefore, for the second type of graph, we used the Erdős-

TABLE 1. Absolute mean correlation with standard deviation between node's degree and the output of randomly initialized networks on graphs with 20, 50, 80, 100, 200, and 500 nodes. The color gradient is generated using the mean correlation value.

GCN	20	50	80	100	200	500
PL-1	0.995 ± 0.0	0.99 ± 0.0	0.986 ± 0.01	0.983 ± 0.01	0.976 ± 0.01	0.961 ± 0.02
BI-1	0.992 ± 0.03	0.998 ± 0.0	0.998 ± 0.0	0.999 ± 0.0	0.999 ± 0.0	1.0 ± 0.0
UN-1	0.969 ± 0.01	0.951 ± 0.02	0.94 ± 0.03	0.935 ± 0.03	0.915 ± 0.04	0.892 ± 0.06
PL-6	0.616 ± 0.28	0.586 ± 0.27	0.559 ± 0.26	0.574 ± 0.26	0.547 ± 0.26	0.544 ± 0.25
BI-6	0.603 ± 0.28	0.787 ± 0.25	0.867 ± 0.21	0.893 ± 0.19	0.949 ± 0.14	0.975 ± 0.11
UN-6	0.889 ± 0.14	0.831 ± 0.17	0.788 ± 0.19	0.757 ± 0.2	0.696 ± 0.23	0.593 ± 0.27

GraphSAGE	20	50	80	100	200	500
PL-1	0.351 ± 0.23	0.27 ± 0.2	0.26 ± 0.19	0.244 ± 0.2	0.238 ± 0.2	0.21 ± 0.18
BI-1	0.366 ± 0.22	0.358 ± 0.23	0.44 ± 0.24	0.474 ± 0.25	0.569 ± 0.27	0.685 ± 0.27
UN-1	0.615 ± 0.24	0.557 ± 0.24	0.517 ± 0.23	0.493 ± 0.22	0.456 ± 0.2	0.402 ± 0.19
PL-6	0.177 ± 0.14	0.108 ± 0.08	0.083 ± 0.06	0.076 ± 0.06	0.053 ± 0.04	0.037 ± 0.03
BI-6	0.21 ± 0.15	0.116 ± 0.09	0.094 ± 0.07	0.083 ± 0.06	0.056 ± 0.04	0.036 ± 0.03
UN-6	0.359 ± 0.22	0.256 ± 0.18	0.237 ± 0.17	0.237 ± 0.17	0.189 ± 0.14	0.149 ± 0.12

GAT	20	50	80	100	200	500
PL-1	0.321 ± 0.23	0.25 ± 0.2	0.225 ± 0.19	0.215 ± 0.19	0.184 ± 0.17	0.158 ± 0.17
BI-1	0.241 ± 0.18	0.192 ± 0.14	0.166 ± 0.12	0.159 ± 0.12	0.124 ± 0.09	0.097 ± 0.07
UN-1	0.294 ± 0.16	0.274 ± 0.15	0.206 ± 0.14	0.237 ± 0.14	0.207 ± 0.14	0.182 ± 0.13
PL-6	0.077 ± 0.06	0.036 ± 0.03	0.028 ± 0.02	0.025 ± 0.02	0.018 ± 0.01	0.012 ± 0.01
BI-6	0.221 ± 0.16	0.118 ± 0.09	0.094 ± 0.07	0.082 ± 0.06	0.058 ± 0.04	0.035 ± 0.03
UN-6	0.867 ± 0.2	0.821 ± 0.21	0.822 ± 0.2	0.805 ± 0.2	0.779 ± 0.2	0.723 ± 0.22

GIN	20	50	80	100	200	500
PL-1	0.528 ± 0.23	0.379 ± 0.25	0.326 ± 0.24	0.318 ± 0.24	0.291 ± 0.23	0.255 ± 0.22
BI-1	0.535 ± 0.15	0.326 ± 0.07	0.272 ± 0.05	0.251 ± 0.04	0.193 ± 0.03	0.136 ± 0.02
UN-1	0.59 ± 0.21	0.398 ± 0.18	0.31 ± 0.14	0.281 ± 0.13	0.208 ± 0.09	0.128 ± 0.06
PL-6	0.476 ± 0.21	0.345 ± 0.22	0.306 ± 0.22	0.292 ± 0.23	0.273 ± 0.22	0.263 ± 0.22
BI-6	0.489 ± 0.15	0.315 ± 0.08	0.261 ± 0.06	0.244 ± 0.05	0.19 ± 0.03	0.135 ± 0.02
UN-6	0.558 ± 0.2	0.36 ± 0.16	0.291 ± 0.13	0.257 ± 0.12	0.186 ± 0.09	0.113 ± 0.06

Renyi model [43] to generate random graphs with binomial distribution. The last kind of graphs have uniform degree distribution, naturally occurring when nodes denote sequential processes dependent on time. In our experiments, each node is linked to all future nodes, facilitating the transition from any completed process to any subsequent one. Tasks like vehicle scheduling [44], [45], or in general any task, project, or job scheduling problems [46] work with such graphs. Throughout this paper, we will use the abbreviations PL, BI, and UN to refer to randomly generated graphs with power-law, binomial, and uniform degree distributions, respectively. For simplicity, we are working with graphs that have attributes (features) only in nodes. In our experiments the nodes of the graphs have either one constant feature set to a value of 1, which eliminates randomness and solely focuses on the message-passing process, or six random-value features that better model real-world scenarios. Having a constant feature and eliminating the randomness makes it possible to understand better and show the hidden bias stemming from the individual GNNs. The 6 random features were generated using a normal distribution $\mathcal{N}(\mu = 0, \sigma^2 = 1)$. We denote these two graph configurations with suffixes 1, and 6, e.g. PL-1 for power-law degree distribution with one constant feature, and PL-6 with six random features.

V. RESULTS

Table 1 shows the mean absolute correlation with standard deviation between nodes' degrees and output logits for GCN, GraphSAGE, GAT, and GIN. We used the absolute value because the correlation oscillated between negative and positive values. The next experiment better explains this phenomenon. Results of the GCN network show a very strong correlation across all graph structures for both one and six features. In most cases (PL-1, UN-1, PL-6, UN-6), the correlation decreased with increasing graph size. The only case where the correlation increased with graph size was for binomial graphs (BI-1, BI-6). The results for GraphSAGE, indicate lower overall correlation values compared to GCN. However, GraphSAGE still shows a noticeable degree of correlations mainly on uniform graphs and on graphs with one feature. These findings suggest that while GraphSAGE is more resistant to correlation than GCN, the issue persists, particularly in smaller graphs. The GAT model that uses the attention mechanism generally exhibits the lowest degree of correlation among the models tested. The most notable correlation is on UN-6. The GIN model shows a moderate degree of correlation between node degree and output logits. Adding more features did not decrease the correlation as much as in other GNNs.

The correlation is observed across all tested GNNs and graph configurations at varying rates. The correlation is reduced when multiple random features are present. The only

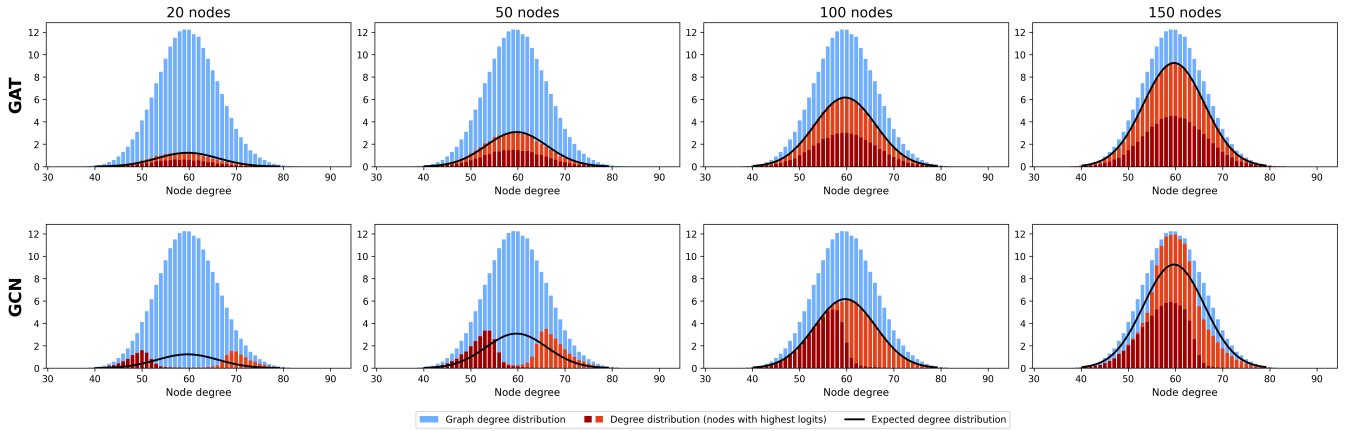

FIGURE 3. **The process of choosing 150 nodes from 200-node Erdős–Rényi graphs based on the logit output of a randomly initialized GAT and GCN networks. We show the intermediate steps for 20, 50, 100, and 150 nodes.**

instance where the correlation increased with more features was when using the GAT network with UN graphs. Overall the most correlated results are on uniformly structured graphs.

Fig. 4 demonstrates the effect of EDB using histogram visualization of selecting 50 nodes based on their logit values across different GNNs and graph configurations. Selected nodes based on the highest logits are depicted by the red color which is the sum of dark red and light red which is explained below. The tail of the Power-law graphs is cut out to the first 25 degrees for better viewing. We use Kullback–Leibler (KL) divergence to measure how much does the observed distribution differs from the expected distribution. All histograms (Fig. 4, 5, 6) are accompanied with the KL divergence shown in Table 2.

We found that, depending on the network's initialization, the selection process tends to favor either high-degree or low-degree nodes. This is caused by logit's sign that is determined by the weights of the last GNN layer. When the logits have a negative sign, their order reverses, leading to a preference reversal from high-degree to low-degree nodes. This causes the correlation to oscillate between high positive and high negative values. The difference between logits' signs is depicted by the shade of red. We use dark red to depict the selection process by a network that generated negative logits and light red when the selection occurred using a network that generated positive logits.

On average the selection should follow the expected distribution but the results show that this is not the case. The GCN and GIN exhibit the biggest difference compared to the expected distribution across all graph structures and feature configurations. This is further supported by the observed correlation from Table 1.

Fig. 3 shows the process of selecting 20 nodes all up to 150 nodes out of 200 node graph. Here, the distinction between dark and light red is crucial as it shows two different ways of selecting nodes. The results of GAT show that on BI-6 all degrees are equally preferred and it's not dependent on the logit's sign. On the other hand, GCN exhibits skewed

TABLE 2. **KL divergence between expected and observed distribution when selecting 50 nodes out of 200 node graph.**

Experiment	KL-divergence					
	PL-1	BI-1	UN-1	PL-6	BI-6	UN-6
GCN	0.603	0.508	0.523	0.016	0.13	0.094
GraphSAGE	0.017	0.006	0.277	0.001	0.0	0.002
GAT	0.012	0.002	0.047	0.007	0.001	0.14
GIN	0.03	0.384	0.523	0.023	0.211	0.203
DQN-GCN $\epsilon = 1$	0.0	0.0	0.0	0.0	0.0	0.0
DQN-GCN $\epsilon = 0.5$	0.035	0.044	0.048	0.011	0.035	0.027
DQN-GCN $\epsilon = 0$	0.603	0.508	0.523	0.016	0.13	0.094
PPO-GCN	0.0	0.0	0.0	0.0	0.0	0.0
PPO-GIN	0.081	0.001	0.016	0.037	0.001	0.014

selection towards high and low-degree nodes. The results of selecting 100 nodes using the GCN network seem to fit the expected distribution but that interpretation is incorrect. Even though the selection resembles the expected distribution the shades of red are skewed towards the sides. Therefore, after you initialize a GCN the selection process will follow either the dark or light red pattern based on the logits' sign.

Implications for DQN and PPO

Further, we explored the impact of integrating GNNs with DQN and PPO agents, to investigate how the exploration is affected by EDB.

Fig. 5 and Fig. 6 visualize the distributions when using DQN and PPO agents respectively. Training a DQN agent relies on using another policy like the ϵ-greedy policy. Fig. 5 shows results for the DQN agent using a GCN network with three different exploration settings. In the case of $\epsilon = 1$, the agent explores the environment with a distribution that closely matches the expected distribution. Conversely, with $\epsilon = 0$, the degree distribution diverges from the expected pattern exhibiting EDB in exactly the same way as reported in Fig. 4. The reduced randomness limits the agent's ability to adequately explore diverse nodes, showing the EDB. In general, $\epsilon = 0$ shows the results of the previous experiment

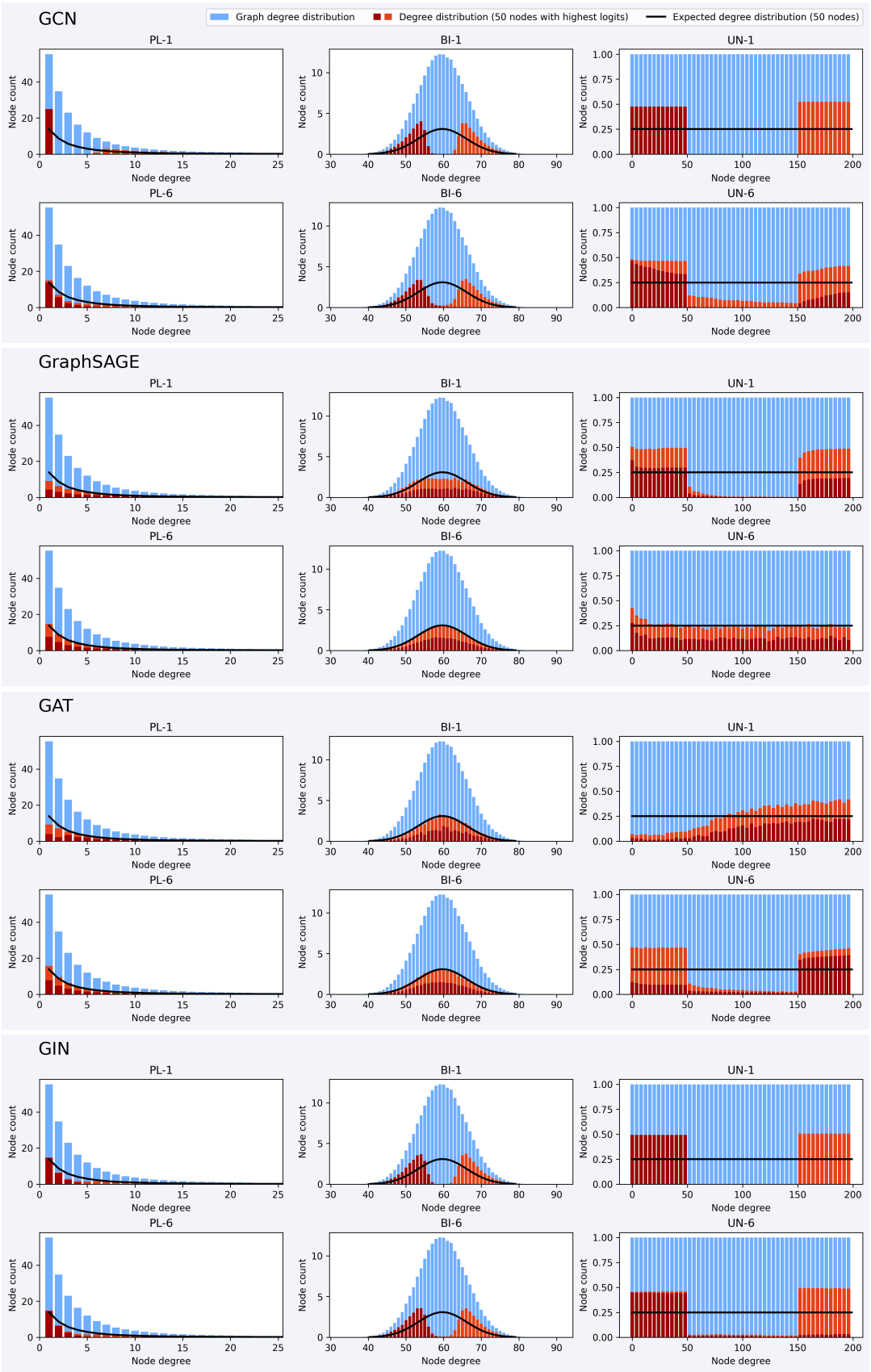

FIGURE 4. **Degree distributions of selecting 50 nodes out of 200 node graph using highest logits across different graph structures and GNNs. Darker distribution represents selections made with negative logits, and brighter distribution represents selections made with positive logits.**

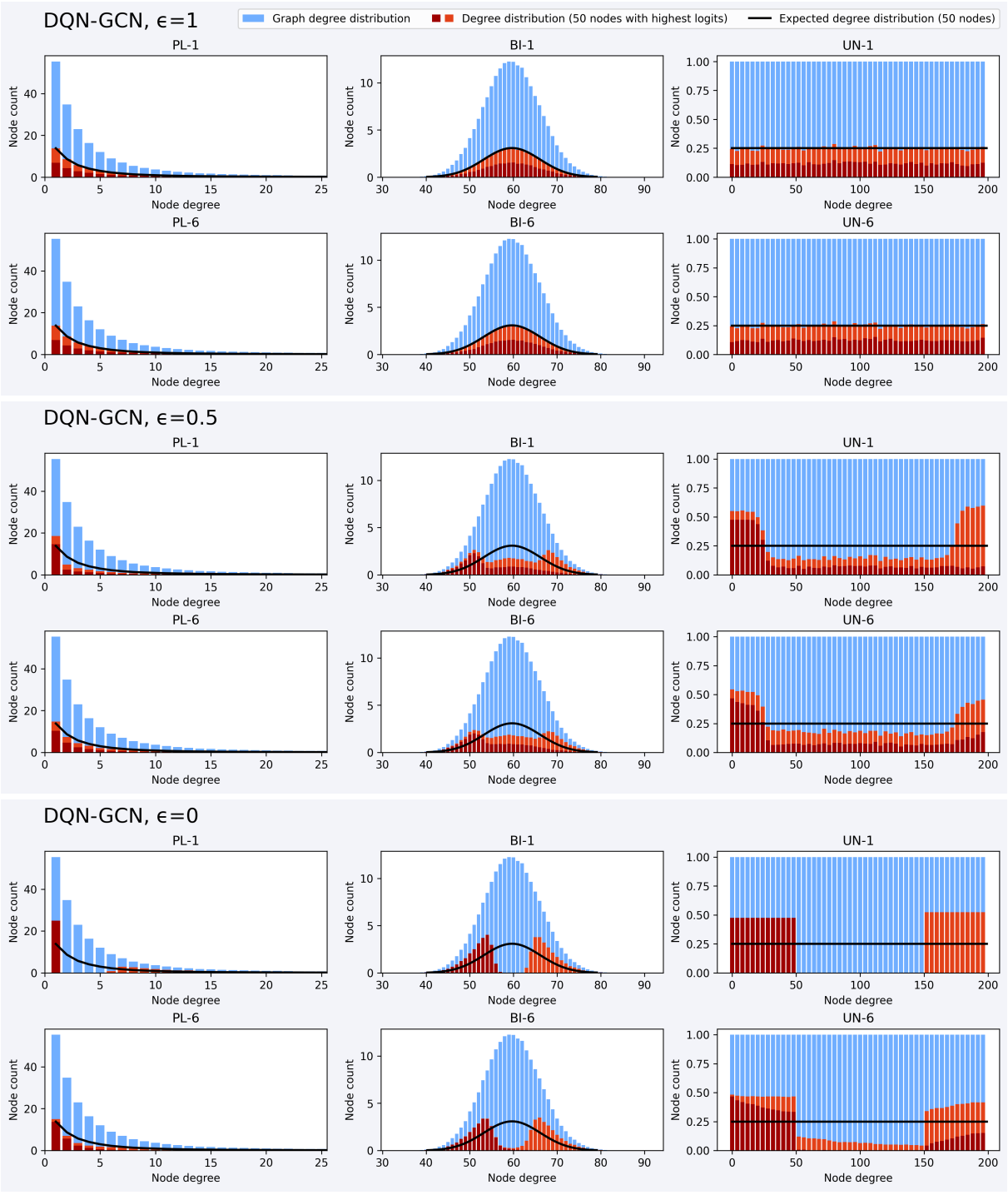

FIGURE 5. **Degree distribution of picking 50 nodes out of 200 node graph with a DQN agent with $\epsilon \in \mathbf{0}, \mathbf{0.5}, \mathbf{1}$ using a GCN network. Darker distribution represents selections made with negative logits, and brighter distribution represents selections made with positive logits.**

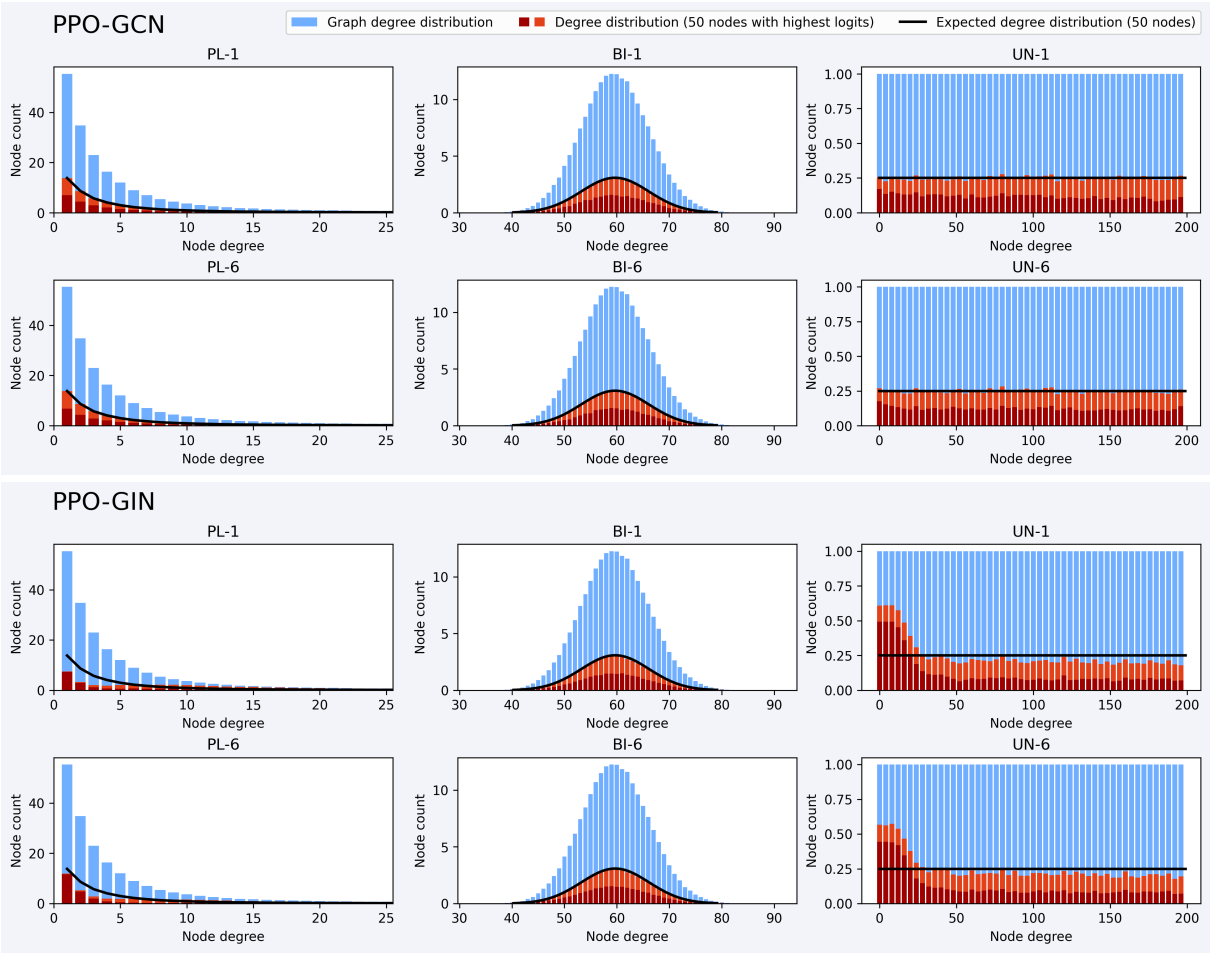

FIGURE 6. Degree distribution of picking 50 nodes out of 200 node graph with a PPO agent using GCN and GIN networks. Darker distribution represents selections made with negative logits, and brighter distribution represents selections made with positive logits.

(Fig. 4), and results for $\epsilon \in (0, 1)$ are combinations of distributions shown in Fig. 4 and uniform distribution. Therefore we did not include other figures for the GraphSAGE, GAT, and GIN network.

The results (Fig. 6) show that when using a PPO agent with a GCN network, the *softmax* function of PPO responsible for sampling actions effectively smooths out the selection process. For our testing setup, logits are very similar, around 0 with a standard deviation of 0.03, resulting in *softmax* probabilities to have a uniform distribution. This leads to a behavior where the selected nodes align with the expected distribution. However, in the case of the GIN network, the standard deviation of logits is in the hundreds or thousands (depending on the graph structure), making the probabilities have the same EDB as the original logits before the *softmax* layer. This causes a slight offset from the expected distributions in the PL and UN graphs. In the case of GAT and GraphSAGE, the standard deviation is 0 and 0.02 respectively, thus probabilities formed a uniform distribution similar to GCN. Therefore, we did not include figures for these two graph networks.

VI. CONCLUSION AND FUTURE WORK

This paper investigates **Exploration Degree Bias** (**EDB**), a newly identified form of bias related to node degrees in **Graph Neural Networks** (**GNNs**), which arises when these architectures are used within **Reinforcement Learning** (**RL**) systems. In RL, effective exploration is crucial to discovering optimal policies, as agents must investigate various parts of their environment to identify rewarding states and actions. Our findings show that EDB disrupts this exploration process by skewing action selection towards nodes with extreme degrees, thereby compromising the agent's ability to explore the environment evenly.

We evaluated four widely used GNN architectures: GCN, GraphSAGE, GAT, and GIN, across different graph structures. Our results highlight that **GCN** and **GIN** are the most susceptible to EDB, showing strong correlations between node degree and output logits. These models tend to **overprioritize high-degree or low-degree nodes**, depending on initialization. This bias is influenced by the GNN's **message-passing mechanism**, where node embeddings are aggregated based on neighboring nodes. As a result, nodes with higher degrees receive more aggregated information, leading to a

skewed representation and influencing the action selection process in RL agents.

In contrast, **GraphSAGE** and **GAT** demonstrated greater resistance to EDB. GraphSAGE's sampling-based neighbor aggregation limits the overemphasis on high-degree nodes by selecting a fixed number of neighbors, thereby reducing the degree-dependent bias. GAT introduces additional randomness during the random initialization of its weights, which influences the message-passing process. This randomness allows the attention mechanism to assign varying importance to neighboring nodes, rather than strictly adhering to their degrees. We hypothesized that GAT has the potential to mitigate EDB during training, as the dynamic weighting may reduce the model's reliance on high-degree nodes for decision-making. This property suggests that the model could foster a more balanced exploration strategy, potentially leading to more effective learning in reinforcement learning contexts. To validate this hypothesis, future research should investigate the temporal dynamics of GAT in relation to EDB, assessing how the attention mechanism's adaptability evolves throughout training and its impact on exploration behaviors. Overall, architectural differences in GraphSAGE and GAT suggest that models with more flexible or adaptive aggregation methods can lessen the effect of EDB, making them more suitable for RL tasks that demand unbiased exploration.

When examining RL agents, we found that **DQN**, particularly under low-exploration conditions (e.g., $\epsilon = 0$), is highly susceptible to EDB, reflecting the same skew seen in the GNN logits. This highlights how EDB interferes with exploration when randomness is minimized, potentially leading to suboptimal policy learning. We showed that the **PPO agent** benefits from the use of the **softmax function**, which smooths the logit differences and leads to a more uniform selection of nodes. However, the effectiveness of this mitigation is contingent on the scale of the logits - while GCN produced logits close to zero, resulting in a uniform selection distribution, the larger logit variations in GIN undermined the smoothing effect of the softmax function, allowing the bias to persist.

Future work should focus on analyzing how various message-passing mechanisms influence the training dynamics related to EDB. Additionally, investigating the interplay between EDB and primacy bias in reinforcement learning scenarios could yield valuable insights into their combined effects on exploration strategies. Exploring whether alternative training procedures, such as different initialization schemes, could mitigate EDB is another promising direction. These methods may complement architectural adaptations like GAT's dynamic weighting via its attention mechanism, providing additional strategies to foster balanced exploration and improve learning outcomes in RL tasks. Such investigations could enhance our understanding of GNN-driven agent behavior and contribute to the development of more effective approaches for decision-making systems.

REFERENCES

[1] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE transactions on neural networks*, vol. 20, no. 1, pp. 61–80, 2008.
[2] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.
[3] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *Advances in neural information processing systems*, vol. 30, 2017.
[4] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" *arXiv preprint arXiv:1810.00826*, 2018.
[5] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, "Graph convolutional neural networks for web-scale recommender systems," in *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, 2018, pp. 974–983.
[6] S. Wu, Y. Tang, Y. Zhu, L. Wang, X. Xie, and T. Tan, "Session-based recommendation with graph neural networks," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, no. 01, 2019, pp. 346–353.
[7] L. Yao, C. Mao, and Y. Luo, "Graph convolutional networks for text classification," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, no. 01, 2019, pp. 7370–7377.
[8] D. K. Duvenaud, D. Maclaurin, J. Iparraguirre, R. Bombarell, T. Hirzel, A. Aspuru-Guzik, and R. P. Adams, "Convolutional networks on graphs for learning molecular fingerprints," *Advances in neural information processing systems*, vol. 28, 2015.
[9] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *International conference on machine learning*. PMLR, 2017, pp. 1263–1272.
[10] A. Fout, J. Byrd, B. Shariat, and A. Ben-Hur, "Protein interface prediction using graph convolutional networks," *Advances in neural information processing systems*, vol. 30, 2017.
[11] I. Bello, H. Pham, Q. V. Le, M. Norouzi, and S. Bengio, "Neural combinatorial optimization with reinforcement learning," *arXiv preprint arXiv:1611.09940*, 2016.
[12] E. Khalil, H. Dai, Y. Zhang, B. Dilkina, and L. Song, "Learning combinatorial optimization algorithms over graphs," *Advances in neural information processing systems*, vol. 30, 2017.
[13] W. Kool, H. Van Hoof, and M. Welling, "Attention, learn to solve routing problems!" *arXiv preprint arXiv:1803.08475*, 2018.
[14] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.
[15] G. Li, M. Muller, A. Thabet, and B. Ghanem, "Deepgcns: Can gcns go as deep as cnns?" in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 9267–9276.
[16] Z. Chen, T. Xiao, and K. Kuang, "Ba-gnn: On learning bias-aware graph neural network," in *2022 IEEE 38th International Conference on Data Engineering (ICDE)*. IEEE, 2022, pp. 3012–3024.
[17] X. Tang, H. Yao, Y. Sun, Y. Wang, J. Tang, C. Aggarwal, P. Mitra, and S. Wang, "Investigating and mitigating degree-related biases in graph convoltuional networks," in *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, 2020, pp. 1435–1444.
[18] J. Wu, X. Wang, F. Feng, X. He, L. Chen, J. Lian, and X. Xie, "Self-supervised graph learning for recommendation," in *Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval*, 2021, pp. 726–735.
[19] Z. Liu, T.-K. Nguyen, and Y. Fang, "On generalized degree fairness in graph neural networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 4, 2023, pp. 4525–4533.
[20] A. Subramonian, J. Kang, and Y. Sun, "Theoretical and empirical insights into the origins of degree bias in graph neural networks," *arXiv preprint arXiv:2404.03139*, 2024.
[21] E. Nikishin, M. Schwarzer, P. D'Oro, P.-L. Bacon, and A. Courville, "The primacy bias in deep reinforcement learning," in *International conference on machine learning*. PMLR, 2022, pp. 16 828–16 847.
[22] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio *et al.*, "Graph attention networks," *stat*, vol. 1050, no. 20, pp. 10–48 550, 2017.
[23] V. Mnih, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.
[24] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[25] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE transactions on neural networks and learning systems*, vol. 32, no. 1, pp. 4–24, 2020.
[26] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, pp. 279–292, 1992.
[27] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*. PMLR, 2016, pp. 1928–1937.
[28] W. Ju, S. Yi, Y. Wang, Z. Xiao, Z. Mao, H. Li, Y. Gu, Y. Qin, N. Yin, S. Wang *et al.*, "A survey of graph neural networks in real world: Imbalance, noise, privacy and ood challenges," *arXiv preprint arXiv:2403.04468*, 2024.
[29] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton *et al.*, "Mastering the game of go without human knowledge," *nature*, vol. 550, no. 7676, pp. 354–359, 2017.
[30] A. Kendall, J. Hawke, D. Janz, P. Mazur, D. Reda, J.-M. Allen, V.-D. Lam, A. Bewley, and A. Shah, "Learning to drive in a day," in *2019 international conference on robotics and automation (ICRA)*. IEEE, 2019, pp. 8248–8254.
[31] J. Degrave, F. Felici, J. Buchli, M. Neunert, B. Tracey, F. Carpanese, T. Ewalds, R. Hafner, A. Abdolmaleki, D. de Las Casas *et al.*, "Magnetic control of tokamak plasmas through deep reinforcement learning," *Nature*, vol. 602, no. 7897, pp. 414–419, 2022.
[32] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.
[33] J. Huang, J. Zhang, Q. Chang, and R. X. Gao, "Integrated process-system modelling and control through graph neural network and reinforcement learning," *CIRP Annals*, vol. 70, no. 1, pp. 377–380, 2021.
[34] P. Almasan, J. Suárez-Varela, K. Rusek, P. Barlet-Ros, and A. Cabellos-Aparicio, "Deep reinforcement learning meets graph neural networks: Exploring a routing optimization use case," *Computer Communications*, vol. 196, pp. 184–194, 2022.
[35] M. Nazari, A. Oroojlooy, L. Snyder, and M. Takác, "Reinforcement learning for solving the vehicle routing problem," *Advances in neural information processing systems*, vol. 31, 2018.
[36] H. Lu, X. Zhang, and S. Yang, "A learning-based iterative method for solving vehicle routing problems," in *International conference on learning representations*, 2019.
[37] I. Drori, A. Kharkar, W. R. Sickinger, B. Kates, Q. Ma, S. Ge, E. Dolev, B. Dietrich, D. P. Williamson, and M. Udell, "Learning to solve combinatorial optimization problems on real-world graphs in linear time," in *2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 2020, pp. 19–24.
[38] Q. Cappart, T. Moisan, L.-M. Rousseau, I. Prémont-Schwarz, and A. A. Cire, "Combining reinforcement learning and constraint programming for combinatorial optimization," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 5, 2021, pp. 3677–3687.
[39] E. Meirom, H. Maron, S. Mannor, and G. Chechik, "Controlling graph dynamics with reinforcement learning and graph neural networks," in *International Conference on Machine Learning*. PMLR, 2021, pp. 7565–7577.
[40] R. Albert and A.-L. Barabási, "Statistical mechanics of complex networks," *Reviews of modern physics*, vol. 74, no. 1, p. 47, 2002.
[41] A. Clauset, C. R. Shalizi, and M. E. Newman, "Power-law distributions in empirical data," *SIAM review*, vol. 51, no. 4, pp. 661–703, 2009.
[42] A. D. Broido and A. Clauset, "Scale-free networks are rare," *Nature communications*, vol. 10, no. 1, p. 1017, 2019.
[43] P. Erdos, A. Rényi *et al.*, "On the evolution of random graphs," *Publ. math. inst. hung. acad. sci*, vol. 5, no. 1, pp. 17–60, 1960.
[44] S. Bunte and N. Kliewer, "An overview on vehicle scheduling models," *Public transport*, vol. 1, no. 4, pp. 299–317, 2009.
[45] D. T. Eliiyi, A. Ornek, and S. S. Karakütük, "A vehicle scheduling problem with fixed trips and time limitations," *International Journal of Production Economics*, vol. 117, no. 1, pp. 150–161, 2009.
[46] M. L. Pinedo, *Scheduling*. Springer, 2012, vol. 29.

PETER TARÁBEK received his Ph.D. degree from the University of Žilina in applied informatics. Currently, he serves as an Assistant Professor in the Department of Mathematical Methods and Operations Research at the University of Žilina.

He has participated in several research and applied projects, in which he developed solutions in areas of computer vision, intelligent transportation systems, and Industry 4.0. His research interests include deep machine learning in computer vision and reinforcement learning, medical image analysis and explainability in deep machine learning.

DÁVID MATIS received B.S. and M.S. degrees in informatics from the University of Žilina, Žilina, Slovakia, in 2020 and 2022, respectively, where he is currently pursuing the Ph.D. degree in applied informatics.

His research interests are reinforcement learning and deep machine learning, which he uses to address complex optimization problems.