High-Order Graph Relationship-Based Sampling Strategies for Recommendation Systems

Zhiqiang Huang

***Abstract*—In recommendation systems, collaborative filtering typically leverages user-item interaction history to encode user preferences and item characteristics into dense vectors. Most existing research focuses on model architecture, such as LightGCN, which explicitly incorporates high-order graph neighbor relationships into the embedding function, significantly enhancing model performance. However, research on data-level strategies, particularly sampling methods, remains limited. Most approaches simply treat items interacted with by users as positive samples, while all other items are considered negative samples. Negative samples are then randomly selected or sampled based on popularity. In contrast, we argue that high-order graph relationships can be exploited at the data level to generate higher-quality positive and negative samples, and can also serve as a form of data augmentation. Furthermore, we introduce a curriculum learning-based sampling strategy block (SSB) as a novel training approach. Through extensive experiments, we analyze the impact of the proposed sampling strategies and validate the effectiveness of SSB.**

***Index Terms*—Recommendation systems, Collaborative filtering, Sampling strategies, Negative sampling, Curriculum learning**

I. Introduction

With the rapid development of the internet and the exponential growth of information, users are increasingly facing the issue of data overload. In this context, recommendation systems have become a key technology to mitigate this problem, playing an indispensable role in areas such as e-commerce, social media, and online entertainment [1] [2]. These systems help improve user experience and platform efficiency by accurately filtering and delivering content that aligns with users' interests and needs.

Collaborative filtering (CF) [3]–[5], one of the most widely used techniques in recommendation systems, has gained significant attention for its ability to predict users' preferences based primarily on interactions between users and items. The core assumption of CF is that users with similar behaviors tend to have similar preferences [4], [5]. Classic CF models, such as Matrix Factorization (MF) [3], LightGCN [6], and SGL [7], have shown promising results. MF factorizes the user-item interaction matrix into latent factors, capturing the relationship between users and items. LightGCN enhances traditional graph-based methods by encoding higher-order neighborhood relations into embeddings via graph convolutions. SGL, on the other hand, leverages contrastive learning by introducing auxiliary self-discrimination tasks, generating better representations. These advancements have significantly improved the ability to capture complex patterns in user-item interactions, leading to more accurate recommendations.

However, despite these advancements, current models still face several challenges. One major issue is the potential for bias in the learned representations because the used training data usually contains biased user-item interactions [3], [8]–[10], which can be further exacerbated by graph convolutions [8], [11], [12]. While various methods have been proposed to address bias, they often come with limitations. For example, adversarial methods like IRGAN [13] use generative adversarial networks (GANs) to select informative negative samples, but these approaches often suffer from high computational costs, unstable training, and long convergence times. Other debiasing methods [14]–[16] aim to reduce bias during model learning but struggle to eliminate all sources of bias effectively.

At the same time, research has also focused on improving sampling strategies at the data level [17]–[20], which are crucial for training effective recommendation models. Sampling methods can be broadly categorized into several types: random negative sampling [4], popularity-based sampling [21]–[23], hard negative sampling [24], [25], and adversarial sampling [13], [26]. Hard negative sampling prioritizes selecting the most challenging negatives based on the model's current predictions. While this accelerates convergence, it risks overfitting by focusing too much on negatives that are too similar to positives. Adversarial sampling incorporates GANs into the process, where a generator selects negative samples to confuse the discriminator (the recommendation model). While promising, this method faces challenges related to computational complexity and unstable training. In summary, there is a need for a more efficient sampling strategy, and existing methods also overlook the impact on bias.

In this paper, we propose a novel sampling strategy that explicitly incorporates higher-order neighborhood relationships from the graph at the data level. While most current methods focus on popularity-based sampling [21]–[23], we argue that integrating higher-order relationships can improve the learning of the loss function, thereby improving the performance of recommendation tasks. This is similar to the process where graph convolutions utilize higher-order relationships [6], [27] to improve the embedding function and, in turn, enhance recommendation task performance. This idea also serves as one of the inspirations for our proposed sampling strategy. Unlike graph convolutions [6], which aggregate neighbors and amplify bias [8], our approach incorporates higher-order relations in the sampling process without aggregation, thereby avoiding this issue. Additionally, by combining this with a popularity-based sampling technique, we reduce the model's bias toward popular items, encouraging more balanced and diverse learning.

Inspired by the idea of curriculum learning [28], [29], we also introduce a new block-based training strategy (referred to as the Sampling Strategy Block, or **SSB**). In this framework, a "block" consists of several training stages, each employing a different sampling strategy. The overall sample distribution progresses from easy to difficult, ensuring stable training and improved model performance.

The main contributions of this work are as follows:

1) We propose a novel sampling strategy that incorporates higher-order neighborhood relationships into the sampling process. Furthermore, we consider the impact on bias.
2) We introduce a block-based sampling strategy (SSB), inspired by curriculum learning, which ensures stable training and efficient performance.
3) Through extensive experiments, including benchmark and ablation studies, along with comprehensive visualizations, we demonstrate the effectiveness of the proposed sampling strategy and block-based training approach.

II. RELATED WORK

In this section, because we focus on sampling strategies in collaborative filtering-based recommendation systems. We review existing works on classical collaborative filtering models and commonly used sampling strategies in recommendation systems, which are most relevant to our work.

A. Collaborative filtering(CF) Models

CF [3]–[5] is a fundamental and popular topic in recommendation systems. It aims to uncover users' latent preferences and items' characteristics by mining user-item history interaction data and encodes these relationships into dense vectors to identify new user-item connections. These interactions are typically implicit (e.g., clicks, add-to-cart actions) rather than explicit (e.g., ratings, likes) [5].

Classical CF models evolved from early matrix factorization (MF) [3] approaches, which merely parameterize user and item embeddings and computes the relevance score to reconstruct historical interactions. Subsequently, graph-based methods such as NGCF [27] introduced explicit encoding of high-order neighbor relationships and collaborative signals into the embedding process. Building on this, the lightweight yet effective LightGCN [6] model retained only the essential graph convolution operations, achieving great results. However, studies have shown that graph convolutions can amplify popularity bias [8], [11], [12], increasing the influence of popular items and deviating from the real interests of the user.

Recent advancements include contrastive learning-based graph models [7], [8], [30]–[32], which incorporate auxiliary contrastive tasks to generate embeddings with desired properties, such as mitigating popularity bias or improving noise robustness. Notable examples include SGL [7], SimGCL [31], and RocSE [32]. Bias, a critical factor in recommendation systems, has also been extensively studied. For instance, the IPS [14] model balances accuracy and coverage by weighting loss inversely with item popularity scores. The state-of-the-art AdvDrop [8] model leverages adversarial learning to measure bias, generating bias-aware and bias-mitigated subgraphs that are then integrated into a contrastive learning framework. In our work, we also examine how proposed sampling strategies influence bias.

B. Sampling Strategies

In recommendation systems, negative samples are typically employed in loss functions to contrast with positive samples (e.g., BPR [4] and InfoNCE [33] loss functions), enabling the model to learn ranking or preference relationships. Sampling strategies play a pivotal role in mitigating data sparsity and enhancing model performance. Existing sample methods can be broadly classified into three categories: static negative sampling [4], [21]–[23], hard negative sampling [24], [25], and adversarial sampling [13], [26].

1) Static Negative Sampling: Static methods assume that the probability of sampling negative examples remains constant throughout training. These methods include:1)Random Negative Sampling [4]: Negative samples are uniformly drawn from missing data. This straightforward approach is commonly adopted due to its simplicity.2)Popularity-Based Sampling [21]–[23]: Negative samples are selected according to popularity distributions. For instance, in [23], Word2Vec applied to Recommendation: random walk sequences are generated using this strategy to capture graph structures.

2) Hard Negative Sampling: Hard negative sampling [24], [25] dynamically adjusts the distribution of candidate samples during training, prioritizing the selection of the most challenging negatives based on the current model state. For example, items with the highest predicted scores among a random set of candidates are chosen as hard negatives. While this approach accelerates model convergence, it risks overfitting by overemphasizing negatives that closely resemble positives, thereby compromising sample diversity.

3) Adversarial Sampling: Adversarial sampling [13], [26] incorporates generative adversarial networks (GANs) into the sampling process, engaging in a minimax game between a generator and a discriminator to identify more informative negatives. For instance, IRGAN [13] employs a generator to select negatives that confuse the discriminator, which acts as the recommendation model. Despite their potential, adversarial sampling methods often face practical challenges, including high computational complexity, unstable training, and long convergence times, limiting their usability in real-world scenarios.

Our proposed sampling strategy takes a novel approach by leveraging higher-order neighbor information. Specifically, certain negative samples can be reclassified as positives during training based on their relationships with high-order neighbors. This strategy alleviates data sparsity [3], mitigates overfitting risks, and enhances the model's ability to learn higher-order neighbor relationships. From both data and model perspectives, this approach contributes to more robust and effective recommendations.

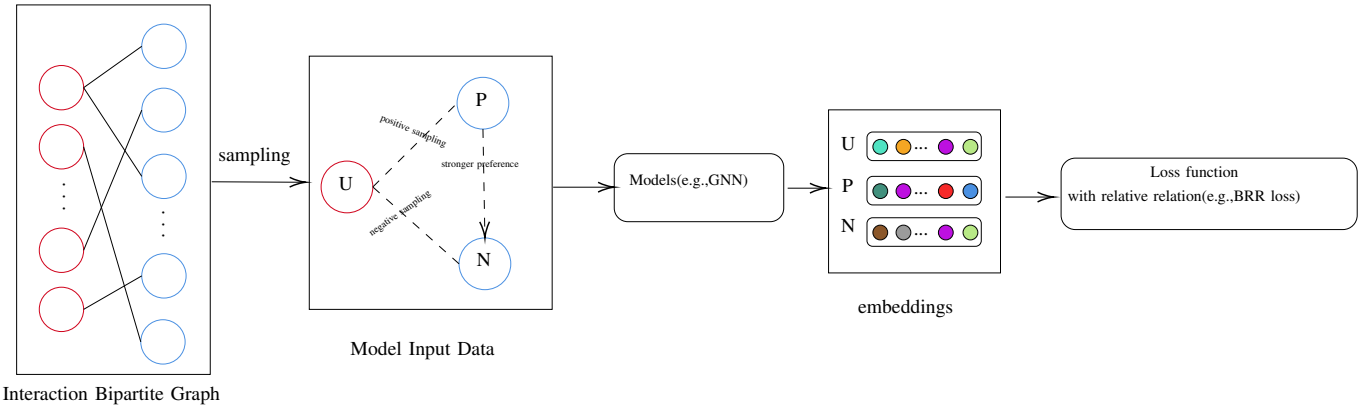


Fig. 1: Common training process of graph-based model architecture in CF.

III. Preliminary

To better understand the role of sampling strategies in recommendation systems and provide context for our work, this section focuses on introducing graph convolution frameworks that achieve excellent results in collaborative filtering and the fundamentals of negative sampling.

A. Graph Convolution Framework

Collaborative filtering (CF) [3]–[5] is a fundamental and active area in recommendation systems. Its core assumption is that users with similar behavior patterns exhibit similar preferences [4], [5]. Most CF-based models aim to infer user preferences and item features from historical user-item interaction data, encoding them into dense vectors for downstream tasks.

Let U and I denote the sets of users and items, respectively, and $O^+ = \{y_{ui} \mid u \in U, i \in I\}$ represents the observed historical interactions between users and items. These interactions are the primary training data for CF. A bipartite graph $G = (V, E)$ can be constructed based on the interaction history, where $V = U \cup I$ is the set of nodes (users and items) and E is the set of observed edges. If u has interacted with i, then $e_{ui} \in E$. The adjacency matrix of G is denoted as A. The goal of recommendation systems can be seen as predicting new connections in A [3].

Graph Convolutional Networks(GCNs) [6], [27] have shown great potential in collaborative filtering by encoding high-order neighbor relationships into embeddings. GCNs typically consist of three major components:

1) **Embedding Initialization Layer:** Each user and item has an initial embedding, which are trainable parameters denoted as Z_U and Z_I. For example, the initial embedding of a user $u \in U$ is $\mathbf{z}_u^0$, and similarly for items.
2) **Embedding Convolution Layers:** These layers utilize graph structure to propagate and aggregate information from neighbors. For a specific user u, the embedding at the l-th layer is given by:

$$\mathbf{z}_u^{(l)} = f_{\text{combine}}\left(\mathbf{z}_u^{(l-1)}, f_{\text{aggregate}}(\{\mathbf{z}_i^{(l-1)} \mid i \in \mathcal{N}_u\})\right), \tag{1}$$

where $\mathcal{N}_u$ denotes the neighbors of u, $f_{\text{aggregate}}$ combines structural information from neighbors, and f_{combine} determines how the previous layer's representation of u is updated.
3) **Prediction Layer:** After n layers of convolution, the model aggregates embeddings $\mathbf{z}_u^0, \mathbf{z}_u^1, \ldots, \mathbf{z}_u^n$ to produce the final representation:

$$\mathbf{z}_u = f_{\text{readout}}(\mathbf{z}_u^0, \mathbf{z}_u^1, \ldots, \mathbf{z}_u^n), \tag{2}$$

where f_{readout} could be a simple concatenation, weighted sum, or a more complex function. Similarly, the final item embedding $\mathbf{z}_i$ is obtained. Finally, a similarity function $s(\mathbf{z}_u, \mathbf{z}_i)$, such as inner product or a neural network, calculates the user-item score $\hat{y}_{ui}$, which is used for ranking or other recommendation tasks.

B. Bayesian Personalized Ranking and Negative Sampling

To learn model parameters, the Bayesian Personalized Ranking (BPR) [4] loss is widely used due to its ability to capture preference ordering effectively:

$$\mathcal{L}_{\text{BPR}} = -\sum_{(u,i,j)\in O} \ln \sigma(\hat{y}_{ui} - \hat{y}_{uj}) + \lambda \|\Theta\|_2^2, \tag{3}$$

where σ is the sigmoid function, O is the set of triplets (u, i, j) such that (u, i) is an observed interaction and (u, j) is not, and Θ represents model parameters. BPR loss encourages higher scores for observed interactions compared to unobserved ones.

Negative sampling [4], [6], [13], [21]–[27] plays a critical role in learning effective representations, as it provides contrastive signals to distinguish observed from unobserved interactions. Negative samples are typically drawn from unobserved user-item interactions. For example, the (u, j) pairs in BPR loss above. Common strategies include static negative sampling, hard negative sampling and adversarial sampling, which are elaborated in RELATIVE WORK partII-B.

In a nutshell, effective negative sampling is crucial, as it directly impacts the quality of learned embeddings and the model's ability to generalize. The graph-based model architecture in collaborative filtering can be referred to Fig. 1. It is worth mentioning that Graph Contrastive Learning [7], [8], [31], [32] has recently gained attention for its ability to

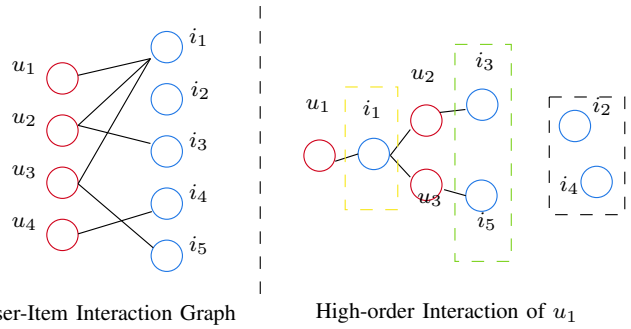
improve node representations by introducing the node self-discriminating task as a auxiliary task. These tasks involve contrasting positive and negative samples, and losses like InfoNCE [33] is employed. However, due to time and resource constraints, our current work focuses on sampling strategies in the main task of recommendation rather than auxiliary tasks, leaving the exploration of sample strategy in auxiliary task for future work.

IV. Methodology

In this section, we first review the training framework for model parameters in collaborative filtering(CF) [3]–[5], where the loss function is based on preference relationships (e.g., BPR [4] loss function), as shown in Fig. 1. Initially, the training data can be viewed as a bipartite graph, where, for each user, we perform positive and negative sampling. During model training, this is done in batches. The model then outputs embedding vectors for the users or items, which are used to compute the loss through a loss function. The model parameters are optimized via stochastic gradient descent or other optimization methods.

Let us focus on the sampling process. The positive and negative samples used for learning the loss function essentially correspond to samples with stronger and weaker preference signals, respectively [4]. These samples are relative: an item that a user has interacted with is assumed to have a stronger preference signal for that user, which is a natural assumption. Almost all methods treat items that a user has interacted with as positive samples. Additionally, an item that a user has not interacted with but is associated with similar users is also assumed to have some preference signal for the user under collaborative filtering assumptions [4], [5]. Moreover, compared to other items with no significant relationship, these items naturally carry stronger preference signals. This observation justifies the sampling strategies we propose. Below, we will describe two types of sampling strategies we introduce: a higher-order interaction-based [6], [27] sampling strategy and a curriculum learning-based [28], [29] sampling strategy block (SSB).

A. Higher-Order Relationship-Based Sampling Strategy



User-Item Interaction Graph High-order Interaction of u_1

Fig. 2: An simple example of the user-item interaction graph and the high-order Interaction.

Let's begin with a simple example, as shown in Fig. 2. On the left, we have the user-item bipartite interaction graph, and on the right, we have the higher-order interaction graph for user u_1, which illustrates the general relationships between u_1 and other users/items. We define the shortest path from u_1 to any node as the number of hops from u_1 to that node. For example, if the shortest path from u_1 to i_3 is $u_1 \rightarrow i_1 \rightarrow u_2 \rightarrow i_3$, then i_3 is a 3-hop neighbor of u_1. It is important to note that in a bipartite graph, odd-numbered hops correspond to items, and even-numbered hops correspond to users. Since we sample items in positive-negative sampling, we consider odd-numbered hops.

Existing sampling methods, such as random sampling [4], popularity-based sampling [21]–[23], hard negative sampling [24], [25], and adversarial sampling [13], [26], treat interacted items as positive samples, with other un-interacted items being considered negative samples. The primary difference lies in how negative samples are selected. However, we propose leveraging the graph structure to more finely partition the items. On the one hand, we exploit the higher-order neighborhood relationships of users in the data, and on the other hand, richer positive and negative sample choices can help alleviate issues like data sparsity and overfitting [3] [6].

Our sampling strategy involves dividing all items into three categories: (1) items directly interacted with by the user (e.g., the nodes inside the yellow box in Fig. 2), (2) 3-hop items (e.g., the nodes inside the green box in Fig. 2), and (3) other items (e.g., the nodes inside the black box in Fig. 2). We refer to these as easy positive items, uncertain items, and easy negative items. The two basic versions of our higher-order sampling strategies are as follows:

1) **Easy positive items as positive samples, uncertain items as negative samples, without considering easy negative items.** We argue that uncertain items, as 3-hop neighbors of the user, are likely to be items the user would be interested in. Intentionally using these items as negative samples helps the model learn to better distinguish between user preferences.
   This strategy is similar to hard negative sampling [24], [25], where negative samples are selected with scores close to the positive ones to provide stronger contrastive signals. However, our approach is based on a static graph structure and directly utilizes the graph's structural information as a strong supervisory signal, independent of the current model training results. Nevertheless, like hard negative sampling, this strategy can accelerate model convergence but may also lead to overfitting. Moreover, using this strategy alone misses out on valuable information, such as the contrast between easy positive and easy negative items. If we force the model to distinguish between similar items too early, it may destabilize training and lead to suboptimal solutions. Therefore, we do not recommend using this strategy alone.
   The primary motivation for this strategy is twofold: it helps the model fine-tune its ability to distinguish user preferences and accelerates convergence. We apply this strategy in the sampling strategy blocks discussed later.
2) **Easy positive items and uncertain items as positive samples, easy negative items as negative samples.** As

mentioned earlier, uncertain items, being 3-hop neighbors, are assumed to carry preference signals for the user. Therefore, it is natural to treat uncertain items as positive samples.

This can be seen as explicitly using the higher-order neighborhood relationships in the graph [6], [27] as a contrastive signal. This strategy also serves as a form of data augmentation, which helps alleviate data sparsity [3]. However, it is important to note that the number of uncertain items is typically very large, and we explain this later. To prevent an overemphasis on uncertain items, we propose a refined sampling rule that limits the number of uncertain items selected as positive samples. Specifically, the number of uncertain items selected as positive samples is constrained by the user's average interaction count, the number of uncertain items, and the number of easy positive items. The mathematical expression is as follows:

$$\min\left(\max\left(\frac{n_{\text{inter}}}{n_{\text{user}}} - n_{\text{ep}}, 0\right), n_{\text{un}}, \lambda n_{\text{ep}}\right) \tag{4}$$

where n_{inter} is the total number of interactions (edges in the bipartite graph), n_{user} is the total number of users, n_{ep} is the number of easy positive items for a user, n_{un} is the number of uncertain items for a user, and λ controls the maximum proportion of uncertain items relative to easy positive items. The first constraint focuses on augmenting data for users with fewer interactions, the second limits the number of times each uncertain item can be selected as a positive sample, preventing overfitting to the graph's structure, and the third constraint ensures that the main contrast signal comes from direct interactions rather than from the graph structure.

Based on the above sampling strategy, we can further apply random sampling, popularity-based sampling, or Inverse Propensity Scoring (IPS) sampling to the negative sample set to achieve different effects. Random sampling is the most common approach [4]. Popularity-based sampling and IPS sampling [21]–[23] help the model learn the differences between positive samples and specific sets of items based on their popularity. Popularity-based sampling tends to focus on negative samples with higher popularity, which may be more challenging for the model to distinguish.

B. Curriculum Learning-Based Sampling Strategy Block (SSB)

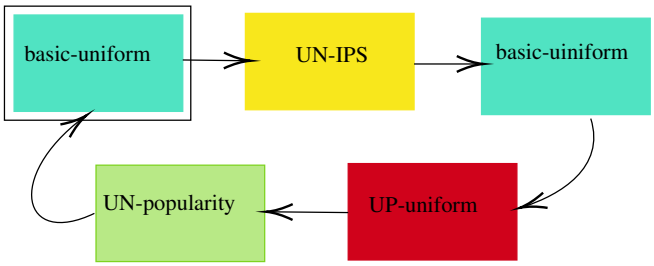


Fig. 3: A specific Sampling Strategy Block(SSB).

The basic unit of our sampling strategy block is a sampling strategy, which includes common strategies (e.g., treating interacted items as positive samples and others as negative samples) and our proposed rules for classifying positive and negative samples, along with weight rules for sampling from the negative sample set. We define the most basic strategy as `basic`, the previously proposed strategy (1) in IV-A part as `UN` (uncertain negative sample), which focuses on negative sampling of uncertain items, and strategy (2) as `UP` (uncertain positive sample), which focuses on positive sampling of uncertain items. We also append `-uniform`, `-IPS`, and `-popularity` to indicate whether the negative sample set is sampled with uniform, IPS-based, or popularity-based weights. For example, `UP-popularity` means that easy positive items and uncertain items are used as positive samples, while easy negative items are used as negative samples, with negative samples being sampled based on their popularity.

A specific SSB, used in our experiments, is shown in Fig. 3. Each small box represents a sampling strategy, and each epoch uses one strategy, with the strategy switching in the direction of the arrows for the next epoch. At the beginning of training, the sampling strategy corresponds to the one outlined in the black-bordered box in Fig. 3. The overall design of the sampling strategy block follows the idea of curriculum learning [28], [29]: starting with simple strategies and gradually increasing the complexity. For instance, `basic-uniform` ensures model stability and prevents bias, while `UP-uniform` serves as a form of data augmentation, helping alleviate data sparsity and reduce overfitting risks. `UN-popularity` biases the sampling toward high-popularity and strongly collaborative negative samples, which can be considered very hard negatives. When the model is performing well, this module helps fine-tune the model and enables it to distinguish user preferences more precisely.

C. Justification for the Positive-Negative Sample Partitioning Strategy

In the proposed sampling strategies, we classify items into three categories: 1-hop, 3-hop, and other items. Higher-order hops (e.g., 5-hop, 7-hop) are not further considered for two reasons. First, the collaborative signal from 5-hop items as positive samples would have significantly diminished. Second, the number of 3-hop items is already very large and can provide ample structural information. The number of n-hop items grows exponentially as n increases. Suppose each node in the user-item interaction graph has an average of k edges. After 1 hop, we obtain k nodes; after 2 hops, we obtain k^2 nodes; and after n hops, we obtain k^n nodes. For instance, if $k = 10$, then after 3 hops, we can get 1000 nodes, which is a 100-fold increase. Although some of these 1000 nodes may overlap, the total number of nodes is still on the order of $O(k^3)$, which is why we consider 3-hop neighbors sufficient for capturing the graph's structural information.

V. EXPERIMENTS

We conducted extensive experiments on real-world datasets to validate the proposed method and address the following research questions:

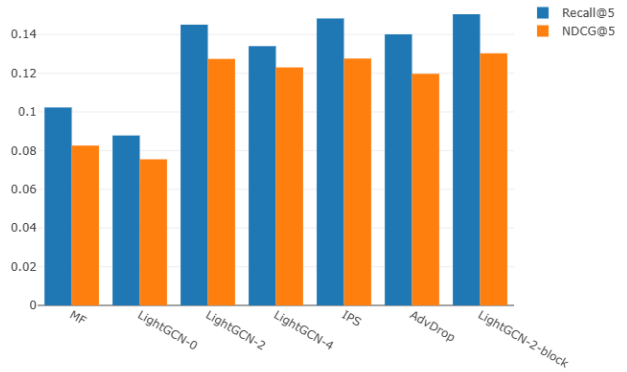


Fig. 4: Performance on Coat test_id

- **RQ1:** How does the SSB-based curriculum learning approach perform compared to other baseline models?
- **RQ2:** How does the proposed high-order neighbor relationship-based sampling strategy perform when used in isolation?
- **RQ3:** Can SSB mitigate biases in the recommendation process?

A. Experimental Setup

Datasets: We used the small *Coat* [8], [34] dataset, which contains user ratings of outerwear items during shopping sessions. The dataset includes a biased dataset and an unbiased random dataset. The biased dataset was split into training, validation, and test sets in a 7:1:2 ratio (with `test_id` representing the in-distribution test set), while the unbiased dataset was used as an out-of-distribution (OOD) test set, referred to as `test_ood`.

Baselines: We compared our method with several classic and state-of-the-art models, including Matrix Factorization (MF) [3], LightGCN [6] models with 0, 2, and 4 layers, as well as bias-correction models like IPS-CN [14] and the adversarial learning-based model AdvDrop [8], which has recently shown excellent results.

Evaluation Metrics: To assess model performance, we used NDCG@K [8] and Recall@K [8] as evaluation metrics.

B. Comparison w line Performance (RQ1)

Due to time and resource constraints, we applied the SSB-based curriculum learning strategy only to the 2-layer LightGCN model, which we call LightGCN-2-block. We then compared its performance with that of the baselines. It is important to note that the SSB-based strategy is related to the sampling method and is independent of the model architecture; we will investigate its effect on other models in future research.

In Fig. 4, we can see the approximate result comparison of each model on Coat test_id. The detailed results on the biased `test_id` dataset are shown in Table I. From the table, we draw the following conclusions:

- The LightGCN model outperforms the traditional MF model by a significant margin. The 2-layer LightGCN shows a 46% improvement in Recall@3 and a 41.8% improvement in Recall@5, further validating the effectiveness of graph convolution.
- However, increasing the number of layers does not always lead to better performance. The 4-layer LightGCN actually performs worse than the 2-layer version, indicating that the model suffers from over-smoothing in this small dataset.
- Both the bias-correction models IPS and AdvDrop achieved good performance, with IPS being the best among all baselines.
- The LightGCN-2-block model, which incorporates the SSB strategy, achieved a 2.45% improvement in Recall@3 and a 3.72% improvement in Recall@5 compared to the standard LightGCN-2, surpassing IPS and becoming the best model on all metrics in the `test_id` set.

TABLE I: Performance on Coat test_id

Metrics	Recall@3	NDCG@3	Recall@5	NDCG@5
MF	0.0725	0.0719	0.1023	0.0826
LightGCN-0	0.0630	0.0680	0.0878	0.0755
LightGCN-2	0.1060	0.1156	0.1451	0.1274
LightGCN-4	0.1066	0.1170	0.1340	0.1230
IPS	0.1084	0.1153	0.1483	0.1276
AdvDrop	0.1011	0.1072	0.1401	0.1197
LightGCN-2-block	0.1086	0.1170	0.1505	0.1303

TABLE II: Performance on Coat test_ood

Metrics	Recall@3	NDCG@3	Recall@5	NDCG@5
MF	0.3124	0.6266	0.4552	0.6928
LightGCN-0	0.2891	0.5822	0.4210	0.5824
LightGCN-2	0.3362	0.6430	0.5070	0.6552
LightGCN-4	0.3419	0.6410	0.5073	0.6525
IPS	0.3440	0.6484	0.5114	0.6584
AdvDrop	0.3559	0.6708	0.5091	0.6706
LightGCN-2-block	0.3430	0.6443	0.5091	0.6706

The results on the unbiased `test_ood` dataset are shown in Table II. From these results, we draw the following conclusions:

- The 2-layer LightGCN shows only a 7.6% improvement in Recall@3 and an 11.4% improvement in Recall@5 compared to MF, which is much smaller than the improvement on the biased test set (`test_id`). This suggests that while LightGCN can better leverage high-order neighbor relationships through graph convolution, it also amplifies popularity bias and learns biased representations, which limits its improvement on unbiased data.
- Both AdvDrop and IPS effectively mitigate bias and achieved the best performance in Recall@3 and Recall@5, respectively.
- The LightGCN-2-block model, which incorporates the SSB strategy, shows improvement on all metrics in `test_ood`, indicating that SSB helps the model learn less biased representations. This may be due to one of the sampling strategies in SSB mitigating popularity bias during training.

TABLE III: Ablation study about sampling strategies on Coat

Metrics	test_id				test_ood			
	Recall@3	NDCG@3	Recall@5	NDCG@5	Recall@3	NDCG@3	Recall@5	NDCG@5
basic-uniform	0.1060	0.1156	0.1451	0.1274	0.3362	0.6430	0.5070	0.6552
basic-popularity	0.1054	0.1166	0.1440	0.1282	0.3458	0.6489	0.5081	0.6579
Basic-IPS	0.1063	0.1160	0.1453	0.1288	0.3460	0.6491	0.5068	0.6575
UP-uniform	0.0973	0.1030	0.1388	0.1173	0.3457	0.6484	0.5107	0.6587
UP-popularity	0.1023	0.1111	0.1351	0.1203	0.3423	0.6422	0.5047	0.6517
UP-IPS	0.1059	0.1166	0.1420	0.1269	0.3451	0.6414	0.5100	0.6546
UN-uniform	0.1008	0.1118	0.1472	0.1277	0.3483	0.6544	0.5096	0.6621
UN-popularity*	0.1070	0.1162	0.1501	0.1304	0.3443	0.6479	0.5012	0.6529
UN-IPS	0.1021	0.1116	0.1466	0.1265	0.3425	0.6477	0.5140	0.6614

C. Impact of Single Sampling Strategies (RQ2)

We also conducted a series of experiments to evaluate the impact of using individual sampling strategies, specifically our proposed high-order neighbor-based sampling strategies. We replaced the sampling strategy in the 2-layer LightGCN model and treated it as an extreme ablation study, where only one sampling unit from SSB was used. We compared different sampling strategies, including `basic-uniform`, `basic-popularity`, `basic-IPS`, `UP-uniform`, `UP-popularity`, `UP-IPS`, `UN-uniform`, `UN-popularity`, and `UN-IPS` (detailed descriptions of these terms can be found in the METHODOLOGYIV-B section).

The results are shown in Table III. From these experiments, we observe the following:

- Using a single sampling strategy generally leads to a decrease in performance on the `test_id` dataset, but an increase in performance on the `test_ood` dataset. This suggests that leveraging high-order neighbor relationships at the data level can help mitigate the bias amplification issue caused by graph convolution.
- The `UN-popularity` strategy improved performance on `test_id`, but its performance decreased on `test_ood`. This may be because `UN-popularity` focuses on sampling popular items, which amplifies popularity bias and leads to more biased representations.

D. Visualizing the Impact of SSB on Bias (RQ3)

To further investigate the effect of SSB on bias, we visualized the embeddings learned by the 0-layer, 2-layer, and 4-layer LightGCN models, as well as the 2-layer LightGCN model trained with SSB. We used t-SNE [35] to reduce the embedding dimensions to 2D and considered specific categories: user gender and item popularity. Item popularity was categorized into three groups based on the number of interactions. The results are shown in Figs. 5.

From Fig. 5a-5c and 5e-5g, we can see that graph convolutions tend to exacerbate bias, which is the same result as [8]. However, Figures 5d and 5h show that the SSB-trained model (i.e., LightGCN-2-block mitigates some of this bias, generating more unbiased representations compared to the original LightGCN with two layers. This suggests that SSB indeed helps in learning more balanced and less biased embeddings.

VI. LIMITATIONS

Although training based on SSB achieves some performance improvements, there are several limitations to the current approach. First, the design of SSB is still largely reliant on manual crafting, depending heavily on the designer's intuition and experience. Furthermore, the effectiveness of SSB may be dataset-specific. The underlying principle of SSB is rooted in curriculum learning, where the difficulty of training varies across different datasets. As a result, the sampling units may need to be adjusted in order to maintain stable for different models.

Another limitation is related to the high-order neighbor-based sampling strategy we proposed, which requires considering items up to the 3-hop level. The number of items grows exponentially as the hop distance increases. For example, if each node in the graph has an average of 100 connections, a user's 3-hop items could amount to approximately 10^6 items—an extremely large number. In large-scale datasets, this can result in significant computational costs and memory usage, making these sampling strategies potentially impractical for very large datasets. Therefore, further work is needed to explore how these strategies can be adapted or optimized for scalability in large-scale recommendation systems.

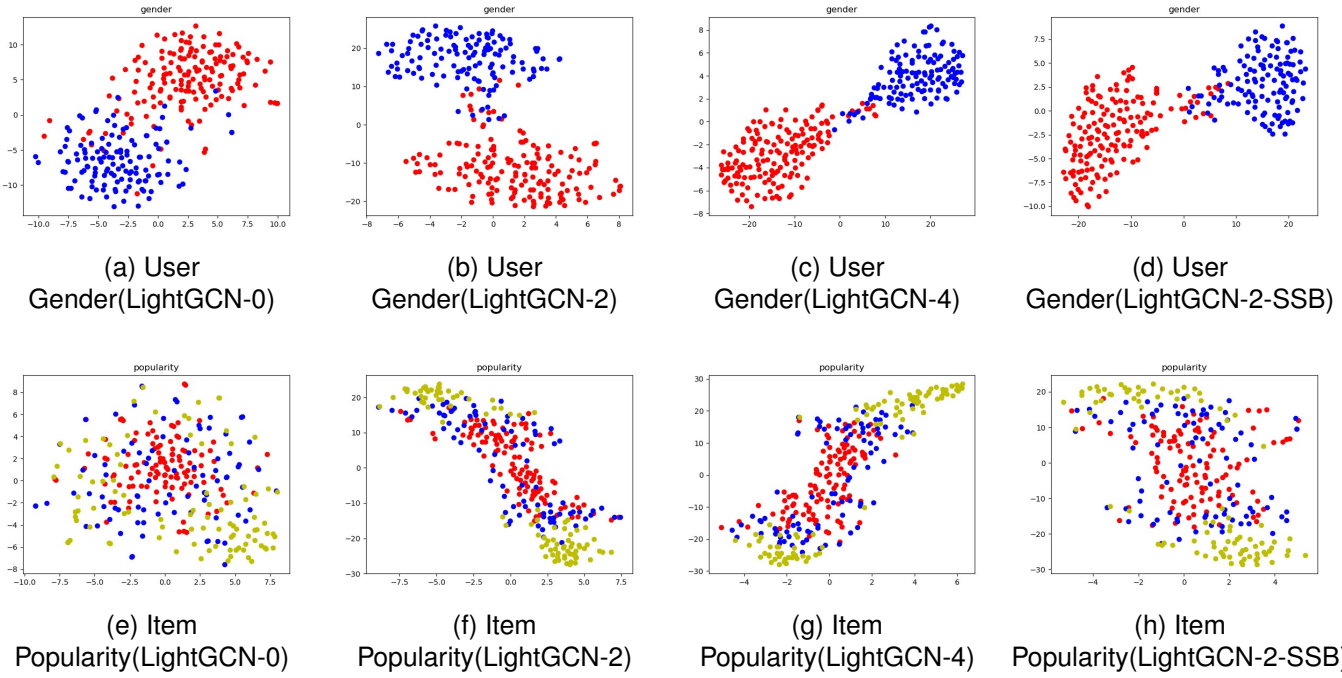
VII. CONCLUSION

In this work, we introduced high-order neighbor relationship-based sampling strategies which aim to leverage graph structural information at the data level, and the SSB(Sampling Strategy Block) training framework. The proposed sampling strategies can alleviate the sparsity problem to some extent and helps the model learn more unbiased representations. Through extensive experiments, we validated the effectiveness of SSB and investigated how the proposed sampling strategies impact model performance.

This work may provide insights into related areas such as data augmentation and training stability in recommendation systems. We believe there is substantial potential for future research in improving how sampling strategies can explicitly utilize high-order graph neighbor relationships, as well as designing more efficient and scalable SSB frameworks.

REFERENCES

[1] P. Covington, J. Adams, and E. Sargin, "Deep Neural Networks for YouTube Recommendations," in *Proceedings of the 10th ACM Conference on Recommender Systems*, ser. RecSys '16. New York, NY, USA: Association for Computing Machinery, 2016, pp. 191–198.



(a) User Gender(LightGCN-0) (b) User Gender(LightGCN-2) (c) User Gender(LightGCN-4) (d) User Gender(LightGCN-2-SSB)

(e) Item Popularity(LightGCN-0) (f) Item Popularity(LightGCN-2) (g) Item Popularity(LightGCN-4) (h) Item Popularity(LightGCN-2-SSB)

Fig. 5: T-SNE [35] visualizations of user and item representations learned by LightGCN with zero, two, four layers respectively and 2-layers LightGCN with SSB we proposed.

[2] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, "Graph Convolutional Neural Networks for Web-Scale Recommender Systems," 2018.

[3] Y. Koren, R. Bell, and C. Volinsky, "Matrix Factorization Techniques for Recommender Systems," *Computer*, pp. 30–37, 2009.

[4] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "BPR: Bayesian Personalized Ranking from Implicit Feedback," 2012.

[5] S. Rendle, "Item Recommendation from Implicit Feedback," 2021.

[6] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, and M. Wang, "LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation," in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '20. New York, NY, USA: Association for Computing Machinery, 2020, pp. 639–648.

[7] 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*. Virtual Event Canada: ACM, 2021, pp. 726–735.

[8] A. Zhang, W. Ma, P. Wei, L. Sheng, and X. Wang, "General Debiasing for Graph-based Collaborative Filtering via Adversarial Graph Dropout," in *Proceedings of the ACM Web Conference 2024*. Singapore Singapore: ACM, 2024, pp. 3864–3875.

[9] J. Chen, H. Dong, X. Wang, F. Feng, M. Wang, and X. He, "Bias and Debias in Recommender System: A Survey and Future Directions," *ACM Trans. Inf. Syst.*, pp. 67:1–67:39, 2023.

[10] Y. Saito, S. Yaginuma, Y. Nishino, H. Sakata, and K. Nakata, "Unbiased Recommender Learning from Missing-Not-At-Random Implicit Feedback," 2020.

[11] H. Abdollahpouri, M. Mansoury, R. Burke, and B. Mobasher, "The Unfairness of Popularity Bias in Recommendation," 2019.

[12] Z. Zhu, Y. He, X. Zhao, Y. Zhang, J. Wang, and J. Caverlee, "Popularity-Opportunity Bias in Collaborative Filtering," in *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, ser. WSDM '21. New York, NY, USA: Association for Computing Machinery, 2021, pp. 85–93.

[13] J. Wang, L. Yu, W. Zhang, Y. Gong, Y. Xu, B. Wang, P. Zhang, and D. Zhang, "IRGAN: A Minimax Game for Unifying Generative and Discriminative Information Retrieval Models," 2018.

[14] A. Gruson, P. Chandar, C. Charbuillet, J. McInerney, S. Hansen, D. Tardieu, and B. Carterette, "Offline Evaluation to Make Decisions About PlaylistRecommendation Algorithms," in *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, ser. WSDM '19. New York, NY, USA: Association for Computing Machinery, 2019, pp. 420–428.

[15] A. J. Bose and W. L. Hamilton, "Compositional Fairness Constraints for Graph Embeddings," 2019.

[16] Z. Chen, J. Wu, C. Li, J. Chen, R. Xiao, and B. Zhao, "Co-training Disentangled Domain Adaptation Network for Leveraging Popularity Bias in Recommenders," in *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '22. New York, NY, USA: Association for Computing Machinery, 2022, pp. 60–69.

[17] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, "Graph Convolutional Neural Networks for Web-Scale Recommender Systems," 2018.

[18] Z. Yang, M. Ding, X. Zou, J. Tang, B. Xu, C. Zhou, and H. Yang, "Region or Global? A Principle for Negative Sampling in Graph-Based Recommendation," *IEEE Transactions on Knowledge and Data Engineering*, pp. 6264–6277, 2023.

[19] M. Zhao, X. Huang, L. Zhu, J. Sang, and J. Yu, "Knowledge Graph-Enhanced Sampling for Conversational Recommendation System," *IEEE Transactions on Knowledge and Data Engineering*, pp. 9890–9903, 2023.

[20] C. Wang, J. Chen, S. Zhou, Q. Shi, Y. Feng, and C. Chen, "SamWalker++: Recommendation With Informative Sampling Strategy," *IEEE Transactions on Knowledge and Data Engineering*, pp. 2004–2018, 2023.

[21] N. Chawla and W. Wang, Eds., *Proceedings of the 2017 SIAM International Conference on Data Mining*. Philadelphia, PA: Society for Industrial and Applied Mathematics, 2017.

[22] T. Chen, Y. Sun, Y. Shi, and L. Hong, "On Sampling Strategies for Neural Network-based Collaborative Filtering," 2017.

[23] H. Caselles-Dupré, F. Lesaint, and J. Royo-Letelier, "Word2Vec applied to Recommendation: Hyperparameters Matter," 2018.

[24] W. Zhang, T. Chen, J. Wang, and Y. Yu, "Optimizing top-n collaborative filtering via dynamic negative item sampling," in *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '13. New York, NY, USA: Association for Computing Machinery, 2013, pp. 785–788.

[25] S. Rendle and C. Freudenthaler, "Improving pairwise learning for item recommendation from implicit feedback," in *Proceedings of the 7th ACM*

International Conference on Web Search and Data Mining, ser. WSDM '14. New York, NY, USA: Association for Computing Machinery, 2014, pp. 273–282.

[26] L. Cai and W. Y. Wang, "KBGAN: Adversarial Learning for Knowledge Graph Embeddings," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, M. Walker, H. Ji, and A. Stent, Eds. New Orleans, Louisiana: Association for Computational Linguistics, 2018, pp. 1470–1480.

[27] X. Wang, X. He, M. Wang, F. Feng, and T.-S. Chua, "Neural Graph Collaborative Filtering," in *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR'19. New York, NY, USA: Association for Computing Machinery, 2019, pp. 165–174.

[28] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proceedings of the 26th Annual International Conference on Machine Learning*, ser. ICML '09. New York, NY, USA: Association for Computing Machinery, 2009, pp. 41–48.

[29] S. Bian, W. X. Zhao, K. Zhou, J. Cai, Y. He, C. Yin, and J.-R. Wen, "Contrastive Curriculum Learning for Sequential User Behavior Modeling via Data Augmentation," in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, ser. CIKM '21. New York, NY, USA: Association for Computing Machinery, 2021, pp. 3737–3746.

[30] M. Jing, Y. Zhu, T. Zang, and K. Wang, "Contrastive Self-supervised Learning in Recommender Systems: A Survey," *ACM Transactions on Information Systems*, pp. 1–39, 2024.

[31] J. Yu, H. Yin, X. Xia, T. Chen, L. Cui, and Q. V. H. Nguyen, "Are Graph Augmentations Necessary? Simple Graph Contrastive Learning for Recommendation," 2022.

[32] H. Ye, X. Li, Y. Yao, and H. Tong, "Towards Robust Neural Graph Collaborative Filtering via Structure Denoising and Embedding Perturbation," *ACM Trans. Inf. Syst.*, pp. 59:1–59:28, 2023.

[33] A. van den Oord, Y. Li, and O. Vinyals, "Representation Learning with Contrastive Predictive Coding," 2019.

[34] T. Schnabel, A. Swaminathan, A. Singh, N. Chandak, and T. Joachims, "Recommendations as Treatments: Debiasing Learning and Evaluation," 2016.

[35] L. van der Maaten and G. Hinton, "Visualizing Data using t-SNE," *Journal of Machine Learning Research*, pp. 2579–2605, 2008.