Breaking Filter Bubble: A Reinforcement Learning Framework of Controllable Recommender System

Zhenyang Li*
Tsinghua University, China
lizy20@mails.tsinghua.edu.cn

Yancheng Dong*
Tsinghua University, China
dongyc20@mails.tsinghua.edu.cn

Chen Gao†
Tsinghua University, China
Huawei Noah's Ark Lab, China
chgao96@gmail.com

Yizhou Zhao
Carnegie Mellon University, United States
yizhouz@andrew.cmu.edu

Dong Li
Huawei Noah's Ark Lab, China
lidong106@huawei.com

Jianye Hao
Huawei Noah's Ark Lab, China
haojianye@huawei.com

Kai Zhang
Tsinghua Shenzhen International Graduate School, Tsinghua University
Pearl River Delta, Research Institute of Tsinghua, China
zhangkai@sz.tsinghua.edu.cn

Yong Li
Tsinghua University, China
liyong07@tsinghua.edu.cn

Zhi Wang†
Tsinghua-Berkeley Shenzhen Institute, Tsinghua Shenzhen International Graduate School
Peng Cheng Laboratory, China
wangzhi@sz.tsinghua.edu.cn

ABSTRACT

In the information-overloaded era of the Web, recommender systems that provide personalized content filtering are now the mainstream portal for users to access Web information. Recommender systems deploy machine learning models to learn users' preferences from collected historical data, leading to more centralized recommendation results due to the feedback loop. As a result, it will harm the ranking of content outside the narrowed scope and limit the options seen by users. In this work, we first conduct data analysis from a graph view to observe that the users' feedback is restricted to limited items, verifying the phenomenon of centralized recommendation. We further develop a general simulation framework to derive the procedure of the recommender system, including data collection, model learning, and item exposure, which forms a loop. To address the filter bubble issue under the feedback loop, we then propose a general and easy-to-use reinforcement learning-based method, which can adaptively select few but effective connections between nodes from different communities as the exposure list. We conduct extensive experiments in the simulation framework based on large-scale real-world datasets. The results demonstrate that our proposed reinforcement learning-based control method can serve as an effective solution to alleviate the filter bubble and the separated communities induced by it. We believe the proposed framework of controllable recommendation in this work can inspire not only the researchers of recommender systems, but also a broader community concerned with artificial intelligence algorithms' impact on humanity, especially for those vulnerable populations on the Web.

CCS CONCEPTS

• **Information systems** → **World Wide Web**; *Recommender systems*; *Personalization.*

KEYWORDS

Controllable Recommendation, Filter Bubble, Reinforcement Learning

ACM Reference Format:
Zhenyang Li, Yancheng Dong, Chen Gao, Yizhou Zhao, Dong Li, Jianye Hao, Kai Zhang, Yong Li, and Zhi Wang. 2023. Breaking Filter Bubble: A Reinforcement Learning Framework of Controllable Recommender System. In *The Web Conference 2023 (WWW'23), April 30–May 4, 2023, Austin, Texas, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3543507.3583856

1 INTRODUCTION

Recommender systems (RS) are currently one of the most popular Web applications in the information-overloaded era [10, 33, 36]. The recommendation algorithms infer user preferences from the collected behavioral log and generate personalized lists of content. Almost all recommender systems pursue the objective of high accuracy to improve user experience and platform profit. Due to the existence of the *feedback loop* [21, 28], the information that users can access is largely determined by the exposure list of recommendation algorithms, in turn narrowing the feedback that the recommender system can collect. As a result, the users' behaviors are more and more likely to be constrained by a small fraction

*Contribution equally
†Corresponding author

WWW'23, April 30–May 4, 2023, Austin, Texas, USA

ACM ISBN 978-1-4503-9416-1/23/04…$15.00
https://doi.org/10.1145/3543507.3583856

of homogeneous content, which is known as the *filter bubble* effect [2, 8, 11, 24]. For example, in short-video platforms such as TikTok, users may be exposed to only a specific group of short-videos, so they will not have the opportunity to interact with the videos outside the bubble, bringing concerns including fairness, discrimination, etc., especially for the underrepresented communities on the Web such as users from less-developed areas, elderly people, and so on. Therefore, it is essential to prevent the recommendation algorithms from dominating the system savagely, motivating our solution of a controllable recommender system.

Some existing works have used the concept of controllable recommendation [25, 26, 30], but it always refers to the user who can control the system. Parra *et al.* [25] investigated a new scenario under which users can control how different recommendation algorithms or recommendation strategies are fused. Rahdari *et al.* [26] developed an interactive recommender system where users can select keywords to filter and control recommendation results. Wang *et al.* [30] consider a scenario that the user aims to explore diverse items and can provide such feedback to the recommender systems. Another recent work [5] also used the term "controllable recommendation"; however, the definition of "control" refers to using a hyper-parameter to balance two recommendation algorithms. In short, the existing works for controllable recommendation pay more attention to let users have the right to control the recommendation list. In contrast, in this work we focus on the filter bubble issue caused by the feedback-loop, and study how to control the recommender system to alleviate this issue.

In this paper, we first analyze the issue of filter bubbles in existing recommenders when there is no control. To address the issue, we propose a general framework of controllable recommendation, under which we design a simulator following the standard pipeline of real-world recommender systems. Further experiments on the simulator also verify the filter bubble caused by feedback-loop of recommender systems. We then develop a reinforcement learning (RL) pipeline based on graph neural networks. Specifically, the graph neural network can learn node representations, which can be coupled to the recommendation model, and the RL method can adaptively select the possible interaction across communities with different interests and control the system to make auxiliary exposures. We conduct experiments on two large-scale benchmark datasets, and the results show that our exposure strategy based on RL can reduce the number or size of isolated communities, while keeping similar or even better recommendation accuracy. That is, the negative impact of recommender systems can be partly addressed. In short, the contribution of this paper is claimed as follows.

- We take the first step to approach the problem of addressing the filter bubble with a controllable recommender system, motivated by the analysis from real-world datasets and in a different manner compared with existing works of controllable recommendation.
- We propose a general framework supported by a simulator and a RL method based on graph neural networks, which can control the additional exposure to help break the filter bubble in existing recommenders.
- Extensive experiments on two real-world datasets verify the effectiveness of our method, having good accuracy and better community characteristics. The further studies of the proposed method explain its rationality.

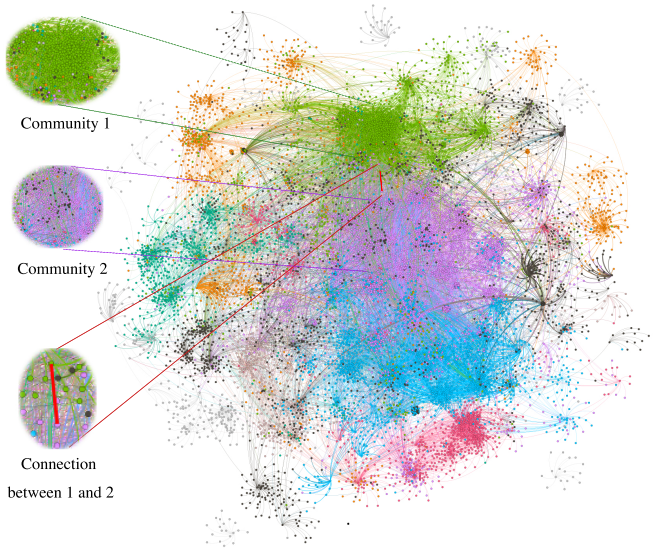


Figure 1: Community detection visualization of Gowalla. There are few connections between communities.

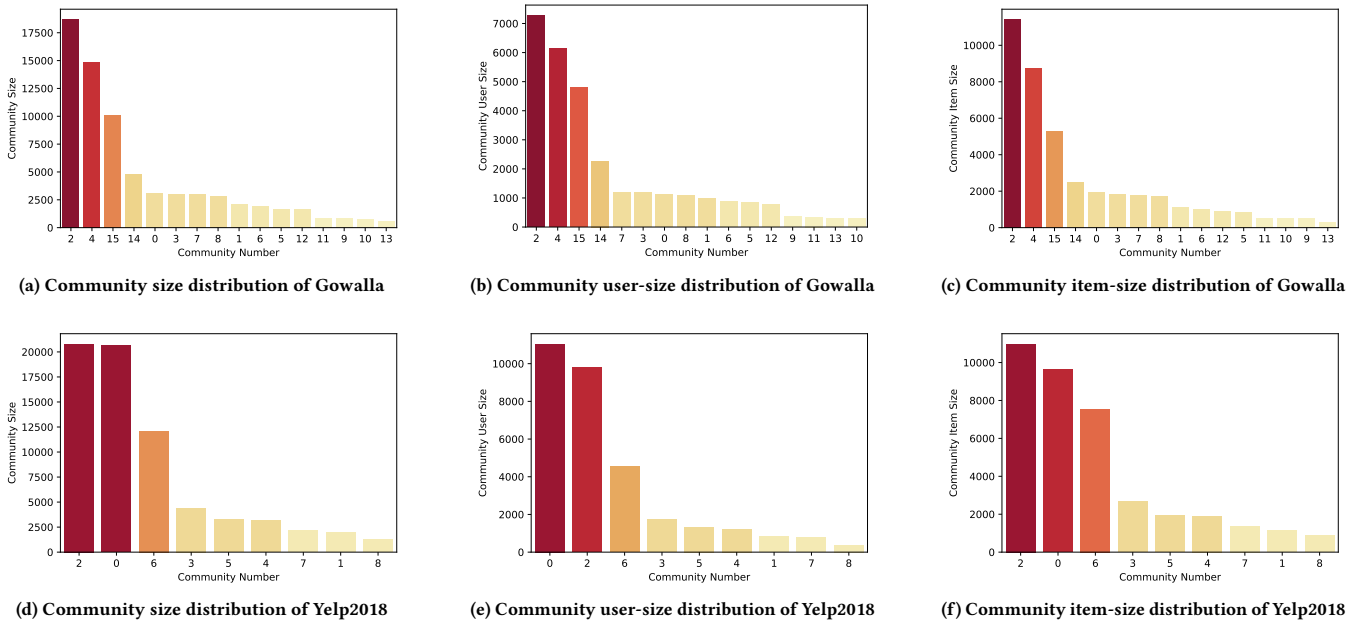
2 MOTIVATION

In this section, we perform analysis on existing recommenders to confirm the issues (mentioned in the introduction). We aim to understand the filter bubble, *i.e.*, the extent to which certain users are stuck in a "bubble" of specific recommended content from the real-world datasets.

Specifically, the user-item interaction data in recommender system can be viewed as a bipartite graph, in which users and items are treated as nodes and interactions are treated as edges. We utilize the widely-used community detection algorithm [4] on the bipartite graph to obtain communities refer to the user-item clusters with dense interactions. A community with few interactions with other user/item nodes means that the users in this community mainly interacts with the items inside, with constrained user accessibility, the detected communities can be regarded as *bubbles*.

We illustrate the communities in Figure 1 of Gowalla dataset. Here different colors refer to users or items belonging to different communities, where we do not distinguish user or item. As we can observe, the whole user-item graph is separated into multiple subgraphs *i.e.*, communities. For example, the largest and the second-largest community are marked with green and purple, in which the connections (user-item interactions) are very dense and the cross-community interactions are far sparser. We also present the distribution of the community size of Gowalla dataset and Yelp2018 dataset in Figure 2. We can observe that the quantitative results also show that users and items can be divided into multiple communities.

To alleviate the effect of the filter bubble, it is essential to introduce a control strategy into the system. Imagine we can control the recommender system to add an exposure edge connecting Community-1 and Community-2, as shown in Figure 1, the users



(a) Community size distribution of Gowalla (b) Community user-size distribution of Gowalla (c) Community item-size distribution of Gowalla

(d) Community size distribution of Yelp2018 (e) Community user-size distribution of Yelp2018 (f) Community item-size distribution of Yelp2018

Figure 2: Community size distribution of Gowalla dataset and Yelp2018 dataset.

in a community can access the content from another one, which provides users opportunities to break the bubble.

3 PROBLEM DEFINITION

In this section, based on the motivation supported by the analysis of real-world data, we define the studied problem. We provide the lists of all used symbols along with their definitions in Table 1.

Traditional recommender system Based on the above notations and definitions, the traditional recommender system's goal can be described as follows. The given input D_{train} is the historically-collected user-item interaction data. The output is a model that can estimate the probability that a user will interact with a given item, and the recommendation list is generated by ranking the items in candidate pools based on the probability.

Controllable recommender system Different from the traditional recommendation, the controllable recommendation's goal in our studied problem has another output: the controlled exposure *i.e.* the auxiliary recommendation list that tries to 1) satisfy user interests, 2) collect useful feedback data for the further model learning and 3) alleviate the filter bubble.

Table 1: Frequently used notations in this paper.

Notations	Descriptions
$\mathcal{U}, \mathcal{I}$	The user set and the item set
$\mathcal{I}_i^t$	The item set recommended to user u_i at time step t
D_{train}^t	The training set at the time step t
D_{test}	The testing set at all time steps
f_{sim}	The simulator that models user-item interactions
$\mathcal{E}_t$	The additional exposure edges set at time step t
$\mathcal{N}_i$	Neighbors of node i
$\mathbf{x}_i^{(l)}$	The embedding of node i at layer l
$\mathbf{z}_i \in \mathbb{R}^{p\times 1}$	The encoded p-dimension representation of node i
$\mathbf{W}^{(l)}, \mathbf{b}^{(l)}$	The parameters of the lth convolution layer
$\mathbf{W}_0, \mathbf{W}_1$	The parameters of linear layer
G_k	The graph structure after executing k action
$\lvert\mathcal{B}\rvert = M$	The memory replay buffer of size M
$s_i \in \mathcal{S}$	A state and the state space
$A_{s_i} = \{a_i\}$	An action and the action space for state s_i
$r_i, r_{i,i+n}$	A reward and the summed reward of n step.
Θ	The parameters of evaluation Q-network
$\hat{\Theta}$	The parameters of target Q-network
γ	The reward factor
T	The maximum number of game episodes
K	The maximum steps in one game
K_{init}	The heuristic initialization steps

4 METHODOLOGY

In this section, we first introduce the overall framework of controllable recommender system with the proposed simulation environment, and we then elaborate on the carefully-developed controllable exposure strategy based on reinforcement learning.

4.1 Simulator environment and framework

As the problem proposed in Section 3, how to simulate the centralized recommendation phenomenon and evaluate the effectiveness of control strategies remains a challenge. Here we provide a

framework for discussing this scenario, namely the user interaction modeling and RS policy evolution.

To collect the user feedback on the recommendation, we utilize a simulator f_{sim} to model the interaction between the user and RS. We leverage the historical interaction data to train the simulator, capturing the user's preference and interest. Given the recommended items set $\mathcal{I}_i^t$ to user u_i at time step t, the simulator acts as a real user and outputs the selected item set as $f_{sim}(\mathcal{I}_i^t, u_i)$, which contains user behavior and explicit interests of goods. In reality, the recommendation system is always updated and evolved based on user feedback. Considering the accumulation of real-time feedback from users into historical training data, we construct trainset D_{train}^t at time step t.

After detailedly describing the two components, the entire simulating flow as:

- **Step1**: At the beginning of RS, there is no historical data in the database and thus the parameters of RS are randomly initialized. The exposure items $\mathcal{I}_i^0$ to user u_i at time step $t = 0$ is randomly chosen, and the training dataset $D_{train}^0 = \emptyset$.
- **Step2**: At the current time step t, we use the RS to expose L items for every user u_i time step t, denoting $\mathcal{I}_i^t$ as the recommended item set. Then we make use of the simulator to get the user feedback as $f_{sim}(\mathcal{I}_i^t, u_i)$. The RS perceives the user's interaction with the exposed items, and generates the accumulated training dataset as $D_{train}^{t+1} = D_{train}^t \cup \{(user_j, item_j)|user_j \in \mathcal{U}, item_j \in f_{sim}(\mathcal{I}_j^t, user_j)\}$. If we introduce the exposure strategy, then the additional exposure edges set $\mathcal{E}_t$ is also added to the D_{train}^{t+1}.
- **Step3**: We evaluate the performance of RS and the community-aware metrics, which will have a further discussion next.
- **Step4**: We train the RS model with the new interactions D_{train}^{t+1}, and update the parameters to get the state-of-art RS model. Then, we repeat **Step2** to **Step4**.

Furthermore, to more accurately evaluate the performance of different methods in **Step3**, we randomly select an independent test set D_{test}, where the user-item pairs in D_{test} remain unchanged for evaluating the precision or recall metrics at a different time step. In this way, we can make a fair comparison between time steps. Note that our simulation system is different from the existing recommendation simulators such as Virtual-Taobao [27], since our work mainly focuses on the user-item feedback.

4.2 Exposure strategy

Next, we introduce the controlling component into the simulation process. According to motivation in Sec.2, we design the controlled exposure strategy to consider the intervention, aiming at controlling the RS towards a direction of decentralized recommendation.

There are various communities in the user-item graph, which indicates the shared interests and strong connections among the user and item groups. Without the loss of generality, we use Louvain [4], an unsupervised approach that has been widely used for community detection.

In order to break the closeness of different communities, we introduce our heuristic exposure method to increase the connectivity of the entire graph by exposing items in one community to users in another community. Specifically, we consider users with the most diverse set of interacted items as the representatives of their community. These users have a wide range of interests, and are more likely to be interested in items from other communities. What's more, the item set interacted by these users is more diverse and probably includes unpopular products on the fringes of the community.

Here we use the intra-list similarity (ILS) as the diversity score of one user's item set. If a recommendation system is recommending lists of very similar items, the ILS score will be high.

$$\mathrm{ILS}(\mathcal{I}) = \frac{2}{k(k-1)} \sum_{i \in \mathcal{I}} \sum_{j \neq i \in \mathcal{I}} \mathrm{Sim}(i, j) \tag{1}$$

where the similarity between two items is calculated by the inner product of the two item embedding. Therefore, the exposure strategy first chooses several communities to get their most diverse users. For each selected user, we randomly choose their historical items to get user-item pairs in different communities. These pairs are heuristically generated by exposure strategy and added to the training set for the next time step as described in **Step 2**.

4.3 Our reinforcement learning-based methodology

We further design an RL based exposure method to mine the promising edges between communities. The edges can be regarded as the environment defined in RL. Moreover, RL follows a Markov Decision Process while interactions can construct a user-item graph. In this case, our RL approach can learn from the sequential edges exposure to produce more effective recommendations.

Note that graph neural network-based methods have become the state-of-the-art approaches in multiple recommendation tasks [9, 34, 35]. We leveraged GCN [16] to encode the graph structural information into the p-dimensional embedding space.

Initialized by RS user and item embedding, the message passing between nodes and their neighbors as:

$$\mathbf{x}_i^{(l+1)} = \mathrm{ReLU}\left(\sum_{j \in \mathcal{N}_i \cup \{i\}} \frac{1}{\sqrt{\deg(i)} \cdot \sqrt{\deg(j)}} \left(\mathbf{x}_j^{(l)} \mathbf{W}^{(l)} + \mathbf{b}^{(l)} \right) \right) \tag{2}$$

where $x_i^{(l)}$ is the embedding of the node i at layer l, $\mathcal{N}_i$ is the neighbors of the node i, $\deg(i)$ is the degree number of the node i, and $\mathbf{W}^{(l)}$ and $\mathbf{b}^{(l)}$ are the parameters of convolution layer l. Instead of simply average pooling all nodes' embeddings, we add a virtual node s that connects all the nodes to capture the complex information of the entire graph. After updating several iterations, the GCN aggregates the embedding of multi-hop neighbors and we get the representation $\mathbf{z}_i$ for each node i.

Since we have already represented every node in the graph with their representations, we can formalize the RL method with quadruple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R})$ as follows:

- **State** The state s_i is defined as the current user-item graph at step i. After propagation on the graph, the representation of the virtual node $\mathbf{z}_s$ can be utilized as the state representation.
- **Action** The action $a_i = (u, v)$ exposes an edge between user u and item v, and corresponding action space A_{s_i} for a state s_i is the set of candidate exposure edges in the graph. Due to

the large action space of all user-item pairs, we select top N similar items for every user as the candidate exposure edges.

- **Transition** The transition $\mathcal{P}$ is the probability that the state changes from s_i to s_{i+1}. The policy network encodes the state to output a probability distribution $\mathcal{P}(s_{i+1}|s_i, a_i)$, where $a_i \in A_{s_i}$.
- **Reward** The reward $r_i = R(s_i, a_i)$ is defined as the increase of the community-aware metric. Given the exposure edge (u, v), we calculate the community number before and after adding the edge. Then the reward is the difference between the reciprocal of the number of communities.

We use the Q-network to estimate the cumulative rewards from the current state-action pair (s, a). It takes state s and action a as input and outputs the Q-value $Q(s, a)$. Each action is an exposure edge (u, v) between user node u and item node v. Our Q-network has two parts: the encoding part and the estimation part. The encoding part is parameterized by the GCN as $\mathbf{W}^{(l)}$ and $\mathbf{b}^{(l)}$. And the estimation part is the mapping from state-action pair (s, a) to a scalar value $Q(s, a)$ by using MLP.

$$Q(s, a) = \mathbf{W}_0^T \text{ReLU}(\text{CONCAT}(\mathbf{z}_u, \mathbf{z}_v, \mathbf{z}_s) \cdot \mathbf{W}_1) \tag{3}$$

where $\mathbf{W}_0 \in \mathbb{R}^{p\times 1}$, $\mathbf{W}_1 \in \mathbb{R}^{3p\times 1}$ are weight parameters of MLPs, $\mathbf{z}_u, \mathbf{z}_v, \mathbf{z}_s \in \mathbb{R}^{p\times 1}$ are the representations of user node, item node and virtual node respectively.

For every time step t in **Step2**, we build the game environment with the initial user-item graph G_0, where every edge is in D^t_{train}. Beginning with the initial graph, the agent uses ϵ-greedy strategy to take the highest Q-value action with probability $1 - \epsilon$ or take a random action. Each action a_k will add an edge to the current graph G_{k-1} and retrieve the next graph G_k. For better mining the exposure edges, we initialize the action by the heuristic method at the beginning of the game. Along with the game processing, we gradually increase the exposure edge set step by step until the game ends, which means whether the game exceeds the max step or the rewards continue to decline in consecutive steps.

Because adding few edges has little effect on the community detection, we adopt the n-step DQN [29] for better approximation. When the game step exceeds the n, we get the transitions $(s_i, a_i, r_{i,i+n}, s_{i+n}, G_i, G_{i+n})$, where $r_{i,i+n} = \sum_{k=0}^{n-1} R(s_{i+k}, a_{i+k})$ is the truncated n-step reward from given state s_i. These collected transitions are stored in the experience replay buffer $\mathcal{B}$.

What's more, the DQN [22] use two Q-network as the target network $\hat{Q}(s, a; \hat{\Theta})$ and evaluation network $Q(s, a; \Theta)$. And the $\hat{\Theta}$, the parameters of target network, is periodically copied from the evaluate network parameters Θ. It will enable a stable learning Q-value for better performance.

Hence the loss function is Eq. 4 and we perform stochastic gradient descents (SGD) over it to update the model parameters.

$$L(\Theta) = E_{U(\mathcal{B})}\left[\left(\left(r_{i,i+n} + \gamma \max_a \hat{Q}\left(s_{i+n}, a; \hat{\Theta}\right)\right) - Q\left(s_i, a_i; \Theta\right)\right)^2\right] \tag{4}$$

where γ denotes the reward factor.

The overall framework of our proposed method is shown in Algorithm 1, for a better understanding.

Algorithm 1 RL-based exposure strategy

```
Initialize experience replay buffer B with M
Initialize the evaluate Q-network with random weights Θ
Initialize the target Q-network with Θ̂ = Θ
while Given the exposure time step t ∈ {1, 2, ...} do
    Build the game env with initial graph G_0 using D^t_train
    for episode ∈ {1, 2, ..., T} do
        Reset the game environment and get the initial state s_0
        for every step k ∈ {1, 2, ..., K} do
            if k ≤ K_init then
                Choose action a_k by heuristic method
            else
                Select action a_k by ε-greedy w.r.t Q-value
            Exposure corresponding user-item pair of a_k and get graph G_k
            Receive the reward r_k
            Get s_{t+1} by GCN Eq. 2 on new graph G_K
            if k ≥ n then
                Store (s_{k-n}, a_{k-n}, r_{k-n,k}, s_k, G_{k-n}, G_k) into B
            if |B| ≥ M then
                Uniformly sample mini-batch of transitions from (s_i, a_i, r_{i,i+n}, s_{i+n}, G_i, G_{i+n}) ~ U(B)
                Update Θ via SGD w.r.t the loss function Eq.4
                Every C step update Θ̂ = Θ
    Use the target network Q̂ to run the game and get the exposure edges set E_t as the sequence of action
```

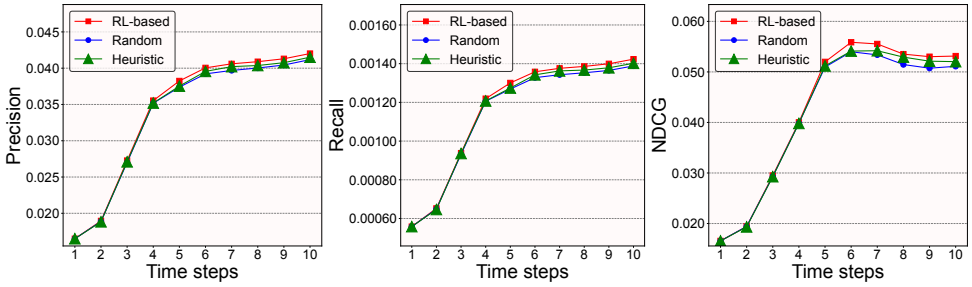
5 EXPERIMENTS

5.1 Experimental settings

In order to reduce the workload of implementation and adjuration in experiments' parameters, we use LightGCN [13] with the model that was retrained by ourselves according to the settings in LightGCN as the simulator of our exposure initialization. We utilize the officially released datasets (train and test lists) from LightGCN, and the quantity details are shown in Table 2. The Gowalla and Yelp2018 are the same as those used in LightGCN [1], so we take the advantage of the statistical results in the paper. To compare the effevtiveness with related works, we introduce precision and connectivity-aware metric to evaluate the model performances and the extent our method can control the RS.

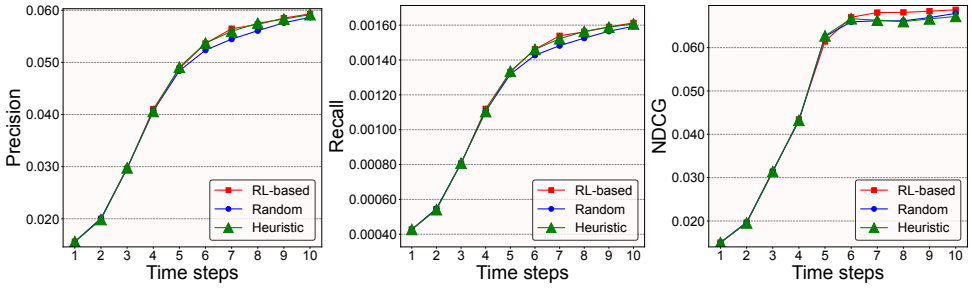
5.1.1 Datasets. We employ two widely used datasets, Gowalla [7] and Yelp2018, in our experiments. Gowalla is a US friend relationship dataset, containing 950,327 friend relationships among 196591 users, and each record represents a pairwise friend relationship. Yelp, the largest review website in US, published its internal dataset Yelp2018, including 4.7 million user reviews, more than 150,000 merchant profiles, 200,000 images, and 12 metropolitan areas. In addition, there are 1 million tips from 1.1 million users and over 1.2 million merchant properties. To accurately compare with LightGCN in our study, we completely adhere to its settings.

[1] https://github.com/gusye1234/LightGCN-PyTorch



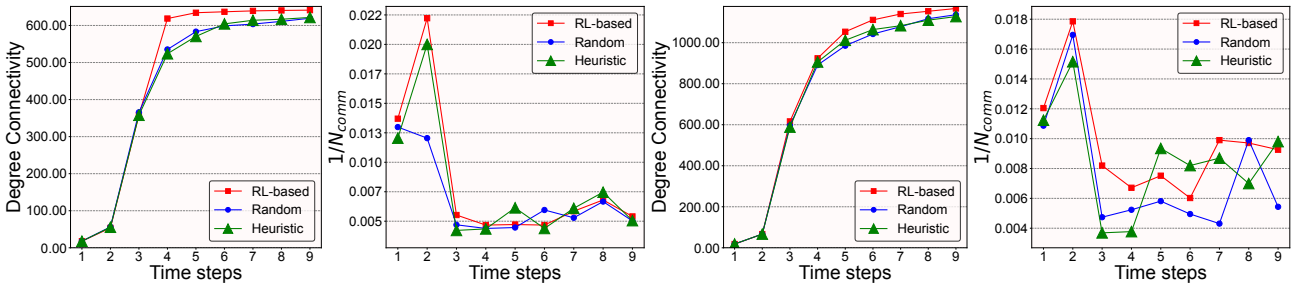
(a) Precision of Gowalla (b) Recall of Gowalla (c) NDCG of Gowalla

Figure 3: Accuracy performance on the Gowalla dataset.



(a) Precision of Yelp2018 (b) Recall of Yelp2018 (c) NDCG of Yelp2018

Figure 4: Accuracy performance on the Yelp2018 dataset.



(a) Average degree connectivity of Gowalla (b) $1/N_{comm}$ of Gowalla (c) Average degree connectivity of Yelp2018 (d) $1/N_{comm}$ of Yelp2018

Figure 5: Community-aware measures on Gowalla and Yelp2018 datasets.

Table 2: Statistics of the two datasets used in experiments.

Dataset	Users	Items	Interaction	Density
Gowalla	29,858	40,981	1,027,370	0.00084
Yelp2018	31,668	38,048	1,561,406	0.00130

5.1.2 Evaluation Metrics. We evaluate the overall performance in terms of commonly used precision metrics in addition to our proposed connectivity-aware and diversity metrics.

- **Precision.** To evaluate the model's predict items for several certain users, we adopt top-N metrics, including Precision, Normalized Discounted Cumulative Gain (NDCG) and Recall, following existing works [14, 18]. We set $N = 20$ in our experiments. To be more specific, we produce predictions on the full test dataset [37] and rank the candidate items for each user, thereby enhancing the solidity of our experimental results.
- **Connectivity-aware Metric and Diversity.** To describe the connectivity of the whole user-item graph, we assumed that the number of communities that are involved in the graph is a direct way to demonstrate it. The phenomenon of users' centralized

recommendation can be referred to as the compacting of the items around specific users. Consequently, the complete graph is tend to be divided into several sub-graphs, that raising up the number of communities N_{comm} and decline the connectivity. Furthermore, we define the diversity metric on the community numbers. For the reason that the growth of centralization can aggregate the top-k similar items around one user, and that can block the user from getting in touch with other types of items, reducing the diversity of items for each user. We simply propose $1/N_{comm}$ as the diversity metric. To fairly compare the performance, we calculate the average community number of last three time steps.

- **Aggregate degree connectivity.** Another proper dimension to measure the connectivity is the degree of nodes from the whole graph. The degree connectivity can be regarded as a more accurate version of the diversity, that can only illustrate the feature of the micro aspect of the graph. However, we formulate the centralization problem as directly related to the connectivity of each node, which refers to the calculation of degree. By increasing the nodes' degree via our methods, more edges will be established between the communities, resulting the boost of connectivity in the whole graph. In detail, we define the degree of i^{th} node is k_i with its neighbor nodes set $\mathcal{N}_i$, the aggregate degree d_i can be described as follows,

$$d_i = \frac{1}{s_i} \sum_{j \in \mathcal{N}_i} w_{ij} s_j \tag{5}$$

where s_i represents the degree of i^{th} node, and w_{ij} refer to the edge weight between i^{th} and j^{th} nodes. Lastly, d_i of the whole graph $D^t(G)$ at time step t can be defined as

$$D^t(G) = \sum_{k \in D^t_{train}} d_k \tag{6}$$

5.1.3 Simulator. The basic assumption of collaborative filtering is that similar users will show similar preferences for items [17, 31]. Since the entry of deep learning into the field, it is generally important to first learn the embedding of users and items in the hidden space, and then reconstruct the interaction between them. LightGCN [13] followed the definition above, and it is known as a highly efficient and simple algorithm. In light of the features of LightGCN and their effectiveness in evaluating the effectiveness of our framework, we choose it as our simulator to model users and items for better explicit representations.

5.1.4 Hyper-parameters Settings. The agent we have set to choose edges has similar structures with DQN [23], which is stacked by several MLPs. The size of embedding-use MLPs is fixed to 64 for evaluation and target net in DQN. All the user and item embeddings are obtained from the parameters in the pretrained LightGCN simulator weights file. We optimize DQN parameters with Adam [15] and use the default initial learning rate 0.001 with a batch size of 8 for all datasets. The reward coefficient γ is set to 0.9. The agent chooses $N = 100$ items for each user while the maximum capacity of memory buffer is $M = 100$. The optimization steps of the target net and accumulated rewards for back propagation are $C = 5$ and $n = 15$, respectively.

5.1.5 Baselines. We implement basic models as baselines to make comparisons and to better emphasize the performance of our proposed methods.

At first, the baselines are mainly constructed into train and test parts. Beginning with an empty training graph, the test starts to generate new user-item edges, which will be added to the training graph after every 20 epochs of the train process. In the training stage, we randomly sample a set of edges uniformly to train our recommender model. As for testing, the purpose is to generate new edges and add them to the train graph.

Overall, the baselines include the framework without extra exposed edges, random and heuristic exposure methods. The strategies of test process depend on different methods that we have produced; for instance, the baselines only contain the selection of top-k items for each user while the additional edges are exposed in our heuristic and RL frameworks. Random and heuristic exposure methods both supply more edges to the above top-k items. As for the the rest of methods, we constantly choose two from all the communities which are derived from the Louvain community detection algorithm [4]. After the selection of communities, we separately utilize random and heuristic method to choose users in one community and items in another, which refer to the random selection and top-k ILS diversity-based selection, respectively.

5.2 Overall performance comparison

We compare the proposed method on two large-scale benchmark datasets, based on the above simulator framework. We first present the performance of recommendation precision on the Gowalla dataset and the Yelp2018 dataset in Figure 3 and Figure 4, respectively. We also present the community-aware measures and degree connectivity on two datasets in Figure 5, in which we evaluate the user-item graph with the community detection algorithms mentioned in Section 2. Based on the experimental results, we make the following observations and conclusions.

- Our method achieves the best recommendation accuracy. As time evolves, the proposed method outperforms the other two baselines on all five metrics (Precision, Recall, NDCG, $1/N_{comm}$, Aggregate degree connectivity) and two utilized datasets. For a detailed comparison, we further present the performance at the last hop on the two datasets in Table 3 and Table 4, respectively.
- Our methods achieve good performance on community-aware measures. With the controlled exposure strategies based on the proposed reinforcement learning algorithms, the results of community detection find that the user-item interaction graph (triggered and induced by user-item exposure and users' decision process) is less centralized. In other words, the filter bubble issue is partly alleviated. Note that in this paper, our exposure strategy only takes into account a few parts of exposure traffic compared with the original recommendation algorithms, and thus the performance improvement will be more significant if we increase the traffic of controlled recommendation.

5.3 Ablation study of our proposed method

To further evaluate the rationality of the designed components in our method, we conduct ablation studies for a deeper analysis as follows.

Table 3: The overall performance among baseline and our methods on the Gowalla dataset.

Method	Precision	Recall	NDCG	Avg N_{comm}	$1/N_{comm}$	Degree
w/o expo.	0.0477	0.00160	0.0620	203	0.00492	634.45
Random expo.	0.0411	0.00138	0.0511	180	0.00556	619.42
Heuristic expo.	0.0415	0.00140	0.0520	165	0.00606	621.63
RL expo.	0.0420	0.00142	0.0531	167	0.00599	641.36

Table 4: The overall performance among baseline and our methods on the Yelp2018 dataset.

Method	Precision	Recall	NDCG	Avg N_{comm}	$1/N_{comm}$	Degree
w/o expo.	0.0586	0.00159	0.0680	132	0.00757	1158.56
Random expo.	0.0586	0.00159	0.0679	173	0.00578	1136.35
Heuristic expo.	0.0591	0.00160	0.0671	120	0.00833	1127.34
RL expo.	0.0593	0.00161	0.0687	104	0.00962	1166.73

Table 5: Ablation study of only heuristic exposure, heuristic and random initialized DQN on Gowalla and Yelp2018 datasets.

Dataset	only heur	heur init.	rand init.	Precision	Recall	NDCG	Avg N_{comm}	$1/N_{comm}$	Degree
Gowalla	✓	✗	✗	0.0415	0.00140	0.0520	165	0.00606	621.63
	✗	✗	✓	0.0418	0.00141	0.0528	240	0.00416	613.80
	✗	✓	✗	0.0420	0.00142	0.0531	167	0.00599	641.36
Yelp2018	✓	✗	✗	0.0591	0.00160	0.0671	120	0.00833	1127.34
	✗	✗	✓	0.0591	0.00161	0.0683	145	0.00689	1161.26
	✗	✓	✗	0.0593	0.00161	0.0687	104	0.00962	1166.73

Specifically, we compare our full framework with a degenerated model in which the main RL module or the edges initialization are removed, which refers to the first line of each dataset, showing that only the heuristic exposure method is in use. The experimental settings keep the same as above and the results shown in Tab.5, illustrating various prior methods to initial RL actions. We can observe from the results that heuristic action initially outperforms others. The third row performs better than the first row indicate that the effectiveness of the RL decision model. Comparing the second and third row, we can see that heuristic initial a more accurate action space to RL, which promotes its outcomes.

6 RELATED WORK

We would like to discuss the related work from the following three aspects, the diversity issue in recommender system, controllable recommender system, and reinforcement learning-based recommender system.

6.1 Low-diversity Issue and Filter Bubble in Recommendation

The recommender systems largely determine the exposure, *i.e.*, what the users can access is the ranking list generated by the recommendation algorithms, and in turn the user provides feedback on the recommended items, also known as a *feedback loop* in recommender system. Most recommender systems are optimized towards higher accuracy, and due to feedback loop, the recommended items always only cover a small fraction of item categories, which leads to a low-diversity recommendation [3, 19, 29]. The low-diversity and centralized recommendation will narrow the horizons of users, and users are trapped in the so-called filter bubble [8, 24]. Nguyen *et al.* [24] quantify the filter bubble using the diversity of user ratings, and the results show that the item sets become more and more narrow as time proceeds. Cinelli *et al.* [8] further found the filter bubble caused by recommendation algorithms to have a long-term negative impact on the Web platform of social media, leading to homophilic opinion clusters about several controversial topics. In short, the existing works on low-diversity issue or filter bubble in recommendation aims to improve the accuracy-oriented algorithm, while our work approaches the problem from a more general perspective by controlling and guiding the recommendation algorithms.

6.2 Controllable Recommender System

Existing works of controllable recommendation [25, 26, 30] typically refer to mechanisms in which the user can control the system. Parra *et al.* [25] investigated the role of user controllability in recommender systems by letting the user control how different recommendation algorithms or strategies are fused. Rahdari *et al.* [26] developed an interactive recommender system in which the users can select some keywords to filter and control the recommendation results. Wang *et al.* [30] considers a scenario wherein users aim to explore diverse items and can provide such feedback to the recommender system. A recent work [5] also used the term "controllable recommendation", but the definition of "control" refers only to using a hyper-parameter to balance two recommendation algorithms. Different from these existing works, in this paper, we aim to consider a more general platform-side framework to control the algorithms in recommender systems and to avoid the creation of isolated communities and the filter bubble effect.

6.3 Reinforcement Learning-based Recommendation

RL methods have been exploited in recommendation based on two aspects as follows [1, 6, 12, 20, 32]. The first category of RL research considers the sequential modeling of user behaviors in recommendation, and then designs sequential-decision models based on reinforcement learning. The second category of RL research for recommendation considers the multiple objectives, especially for those long-term optimization goals, and then develops reinforcement learning methods to maximize rewards that cannot be used in traditional supervision-learning methods. In this paper, we approach a different problem of controlling recommendation algorithms from a third-party view, and design a reinforcement learning-based framework based on graph neural networks. We aim to achieve the long-term controlling goal of alleviating the filter bubble in the user-item graph, while preserving similar or even better performance in terms of recommendation accuracy.

7 CONCLUSION AND FUTURE WORK

In this paper, we consider the filter bubble issue that arises due to the feedback loop in a traditional recommender systems, and we illustrate its existence via empirical analysis of real-world datasets. To address this issue, we propose a general and easy-to-use framework for controllable recommendation, under which the simulator based on offline datasets explains how the filter bubble is formed. We develop a reinforcement learning-based method combined with

a graph neural network. The graph neural network module can learn from user-item interaction, well coupled with recommendation models. The reinforcement learning method can determine which kind of auxiliary exposure should be made to alleviate the isolated and large-sized communities. Results on two datasets show that our method can break the bubble while keeping or even boosting the recommendation performance.

As for the future plan, the first follow-up work we plan to do is to test the proposed approach's performance via online A/B test on real-world Web service providers. Finally, we plan to conduct experiments on more datasets, especially for those in which users are more active but the interactions are more centralized, such as user-video interaction datasets in short-video platforms where users spend plenty of time every day.

ACKNOWLEDGMENTS

This work is supported by Shenzhen Science and Technology Program (Grant No. RCYX20200714114523079), the Key-Area Research and Development Program of Guangdong Province under the grant 2020B0909050003, the National Key Research and Development Program of China under 2022YFB3104702, the National Natural Science Foundation of China under 62272262 and the Fellowship of China Postdoctoral Science Foundation under 2021TQ0027 and 2022M710006.

REFERENCES

[1] M Mehdi Afsar, Trafford Crump, and Behrouz Far. 2021. Reinforcement learning based recommender systems: A survey. *ACM Computing Surveys (CSUR)* (2021).

[2] Guy Aridor, Duarte Goncalves, and Shan Sikdar. 2020. Deconstructing the filter bubble: User decision-making and recommender systems. In *Fourteenth ACM Conference on Recommender Systems*. 82–91.

[3] Azin Ashkan, Branislav Kveton, Shlomo Berkovsky, and Zheng Wen. 2015. Optimal greedy diversity for recommendation. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

[4] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. 2008. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment* 2008, 10 (2008), P10008.

[5] Yukuo Cen, Jianwei Zhang, Xu Zou, Chang Zhou, Hongxia Yang, and Jie Tang. 2020. Controllable multi-interest framework for recommendation. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2942–2951.

[6] Xinshi Chen, Shuang Li, Hui Li, Shaohua Jiang, Yuan Qi, and Le Song. 2019. Generative adversarial user model for reinforcement learning based recommendation system. In *International Conference on Machine Learning*. PMLR, 1052–1061.

[7] Eunjoon Cho, Seth A Myers, and Jure Leskovec. 2011. Friendship and mobility: Friendship and mobility: User movement in location-based social networks-friendship and mobility. In *User Movement in Location-Based Social Networks ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*.

[8] Matteo Cinelli, Gianmarco De Francisci Morales, Alessandro Galeazzi, Walter Quattrociocchi, and Michele Starnini. 2021. The echo chamber effect on social media. *Proceedings of the National Academy of Sciences* 118, 9 (2021), e2023301118.

[9] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. 2019. Graph neural networks for social recommendation. In *The world wide web conference*. 417–426.

[10] Chen Gao, Yu Zheng, Nian Li, Yinfeng Li, Yingrong Qin, Jinghua Piao, Yuhan Quan, Jianxin Chang, Depeng Jin, Xiangnan He, and Yong Li. 2021. A Survey of Graph Neural Networks for Recommender Systems: Challenges, Methods, and Directions. *ACM Transactions on Recommender Systems (TORS)* (2021).

[11] Zhaolin Gao, Tianshu Shen, Zheda Mai, Mohamed Reda Bouadjenek, Isaac Waller, Ashton Anderson, Ron Bodkin, and Scott Sanner. 2022. Mitigating the Filter Bubble while Maintaining Relevance: Targeted Diversification with VAE-based Recommender Systems. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2524–2531.

[12] Yingqiang Ge, Xiaoting Zhao, Lucia Yu, Saurabh Paul, Diane Hu, Chu-Cheng Hsieh, and Yongfeng Zhang. 2022. Toward Pareto Efficient Fairness-Utility Trade-off in Recommendation through Reinforcement Learning. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*. 316–324.

[13] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. 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*. 639–648.

[14] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*. 173–182.

[15] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[16] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations (ICLR)*.

[17] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009), 30–37.

[18] Walid Krichene and Steffen Rendle. 2022. On sampled metrics for item recommendation. *Commun. ACM* 65, 7 (2022), 75–83.

[19] Matevž Kunaver and Tomaž Požrl. 2017. Diversity in recommender systems–A survey. *Knowledge-based systems* 123 (2017), 154–162.

[20] Feng Liu, Ruiming Tang, Xutao Li, Weinan Zhang, Yunming Ye, Haokun Chen, Huifeng Guo, and Yuzhou Zhang. 2018. Deep reinforcement learning based recommendation with explicit user-item interactions modeling. *arXiv preprint arXiv:1810.12027* (2018).

[21] Masoud Mansoury, Himan Abdollahpouri, Mykola Pechenizkiy, Bamshad Mobasher, and Robin Burke. 2020. Feedback loop and bias amplification in recommender systems. In *Proceedings of the 29th ACM international conference on information & knowledge management*. 2145–2148.

[22] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *nature* 518, 7540 (2015), 529–533.

[23] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *nature* (2015).

[24] Tien T Nguyen, Pik-Mai Hui, F Maxwell Harper, Loren Terveen, and Joseph A Konstan. 2014. Exploring the filter bubble: the effect of using recommender systems on content diversity. In *Proceedings of the 23rd international conference on World wide web*. 677–686.

[25] Denis Parra and Peter Brusilovsky. 2015. User-controllable personalization: A case study with SetFusion. *International Journal of Human-Computer Studies* 78 (2015), 43–67.

[26] Behnam Rahdari, Peter Brusilovsky, and Alireza Javadian Sabet. 2021. Connecting students with research advisors through user-controlled recommendation. In *Fifteenth ACM Conference on Recommender Systems*. 745–748.

[27] Jing-Cheng Shi, Yang Yu, Qing Da, Shi-Yong Chen, and An-Xiang Zeng. 2019. Virtual-taobao: Virtualizing real-world online retail environment for reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 4902–4909.

[28] Wenlong Sun, Sami Khenissi, Olfa Nasraoui, and Patrick Shafto. 2019. Debiasing the human-recommender system feedback loop in collaborative filtering. In *Companion Proceedings of The 2019 World Wide Web Conference*. 645–651.

[29] Richard S Sutton. 1988. Learning to predict by the methods of temporal differences. *Machine learning* 3, 1 (1988), 9–44.

[30] Wenjie Wang, Fuli Feng, Liqiang Nie, and Tat-Seng Chua. 2022. User-controllable Recommendation Against Filter Bubbles. *arXiv preprint arXiv:2204.13844* (2022).

[31] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural graph collaborative filtering. In *Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval*. 165–174.

[32] Xiting Wang, Kunpeng Liu, Dongjie Wang, Le Wu, Yanjie Fu, and Xing Xie. 2022. Multi-level recommendation reasoning over knowledge graphs with reinforcement learning. In *Proceedings of the ACM Web Conference 2022*. 2098–2108.

[33] Le Wu, Xiangnan He, Xiang Wang, Kun Zhang, and Meng Wang. 2022. A survey on accuracy-oriented neural recommendation: From collaborative filtering to information-rich recommendation. *IEEE Transactions on Knowledge and Data Engineering* (2022).

[34] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. 2019. Session-based recommendation with graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 33. 346–353.

[35] Mengqi Zhang, Shu Wu, Xueli Yu, Qiang Liu, and Liang Wang. 2022. Dynamic graph neural networks for sequential recommendation. *IEEE Transactions on Knowledge and Data Engineering* (2022).

[36] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. 2019. Deep learning based recommender system: A survey and new perspectives. *ACM Computing Surveys (CSUR)* 52, 1 (2019), 1–38.

[37] Wayne Xin Zhao, Junhua Chen, Pengfei Wang, Qi Gu, and Ji-Rong Wen. 2020. Revisiting Alternative Experimental Settings for Evaluating Top-N Item Recommendation Algorithms. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*.