ICMRec: Item Cluster-Wise Multi-Objective Optimization for Unbiased Recommendation

Yule Wang, Xin Xin, Yue Ding, Yunzhe Li, and Dong Wang

Abstract—Recommender system based on historical user-item interactions is of vital importance for web-based services. However, the observed data used to train the recommender model suffers from severe bias issues (e.g., exposure bias, popularity bias). Practically, the item frequency distribution of the dataset is a highly skewed power-law distribution. Interactions of a small fraction of head (popular) items account for almost the whole training data. The normal training paradigm from such biased data tends to repetitively generate recommendations from the head items, which further exacerbates the biases and affects the exploration of potentially interesting items from the niche (long-tail) set. In this work, distinct from existing methods, we innovatively explore the central theme of unbiased recommendation from an item cluster-wise multi-objective optimization perspective. Through an empirical study, we find that head items are highly likely to be recommended because the gradients coming from head items dominate the overall gradient update process, which further affects the optimization of niche items. Aiming to balance the learning on various item clusters that differ in popularity during the training process, we characterize the recommendation task as an item cluster-wise multi-objective optimization problem. To this end, we propose a model-agnostic framework namely Item Cluster-Wise Multi-Objective Recommendation (ICMRec) for unbiased recommendation. In detail, we define our item cluster-wise optimization target that the recommender model should balance all item clusters that differ in popularity. Thus we set the model learning on each item cluster as a unique optimization objective. To achieve this goal, we first explore items’ popularity levels from a novel causal reasoning perspective. Then, we devise popularity discrepancy-based bisecting clustering to separate the discriminated item clusters. Next, we adaptively find the overall harmonious gradient direction for multiple item cluster-wise optimization objectives from a Pareto-efficient solver. Finally, in the prediction stage, we perform counterfactual inference to further eliminate the impact of user conformity. We conduct experiments on three public datasets, instantiating ICMRec with three state-of-the-art backbone recommender models. Extensive experimental results demonstrate the superiorities of ICMRec on overall recommendation performance and biases elimination. Codes will be open-source upon acceptance.

Index Terms—Data mining, Recommender systems, Fairness and Debias.

1 INTRODUCTION

Recommender systems (RS) play a crucial role in online services and platforms to address the problem of information overload [1], [2], [3]. In conventional methods based on the most prevalent technique, Collaborative Filtering (CF) [2], [3], [4], a recommender model is trained using observed user-item historical feedbacks with the target of providing personalized recommendation items given the current user state. Since it’s difficult to collect explicit feedback from various real-world applications, in this work we focus on implicit feedbacks, which are easily collected by storing users’ interaction behaviors (e.g., click), generating the dataset for further training steps.

However, inherent bias issues emerge as great challenges for modern RS. Traditionally, the incremental training procedure of the recommender model in industrial applications of RS is a process of mutual dynamic evolution, where exists a severe bias amplification feedback loop [5], as shown in Figure 1. Specifically, first, since a fraction of users are likely to interact with the head items, these actions will be collected by the RS. Then, as the head items are much more frequently treated as positive samples, they will be pushed towards much higher ranking scores compared with other items (i.e., popularity bias). After the RS generates the recommended list, the exposure mechanism of the RS results in the fact that users could only be exposed to the top head items and only interact with them (i.e., exposure bias), further affecting the collection of user-item interactions. As the circle back of training data and recommend list in the RS continues, the biases get amplified, introducing the notorious Matthew effect [5]. In the end, a small fraction of head items accounts for almost the whole training dataset.

Hence, debiasing in recommendation plays an important role in improving the system’s performance. From the user’s perspective, he/she could be easily bored with the head items repetitively recommended by the system. There are potentially relevant items that will lead to larger user satisfaction in the niche set but they have never been exposed [6], [7]. As for service providers, the recommendation from niche items can embrace more marginal profit compared with head items [8]. Generally speaking, the recommendation task is a typical exploitation-exploration problem. The unbiased recommendation will benefit both users and service providers with better exploration, which finally turns into larger benefits in the long-run [9].

Several existing methods focused on unbiased recommendation are developed based purely on inverse propensity scoring (IPS) [10], [11], [12], which re-weights the data samples with certain theoretical analyses. However, it introduces high variance in the implicit feedback scenario [12], [13], the estimated weights fluctuate drastically in each step.
Cluster-Wise Multi-Objective Recommendation (ICMRec)

Motivated by the above observation, we propose Item Cluster-Wise Multi-Objective Recommendation (ICMRec) for unbiased recommendation. Note that ICMRec is a general model-agnostic framework and can be instantiated with various CF-based backbone models, such as Matrix Factorization (MF) [21], Neural matrix factorization (NeuMF) [3], etc.

More precisely, we define our item cluster-wise optimization target that the recommender model should perform fairly for all item clusters that differ in popularity, setting the model learning on each item cluster as a unique optimization objective. To achieve this goal, we first explore items’ popularity levels from a novel causal reasoning perspective, calculating their correlation score with the user’s global propensity. Then, we devise popularity discrepancy-based bisecting clustering to separate the discriminated item clusters, in which the representation and popularity information are both considered. Next, we adaptively find the weights of multiple item cluster-wise optimization objectives from a Pareto-efficient solver. As a result, the learning over the whole training data can be seen as a weighted aggregation of multiple cluster-wise objectives, getting the overall harmonious gradient direction. Finally, in the prediction stage, we perform counterfactual inference to further eliminate the impact of user conformity. We conduct extensive experiments on three public datasets, instantiating ICMRec with three state-of-the-art recommender models. The bias issues of RS are effectively mitigated in ICMRec. At the same time, experimental results show significant improvement in recommendation performance, especially on the niche item set. To the best of our knowledge, this is the first attempt to explore unbiased recommendation from an item cluster-wise multi-objective optimization perspective. To summarize, this work makes the following contributions:

— After investigating the training process of traditional methods, we observe that the bias issues keep getting amplified due to the gradient domination of head items, which provides a new direction to address the bias issues. Thus, we propose to eliminate the bias issues in recommender systems from an item cluster-wise optimization perspective based on gradients, which explicitly balance the learning on various item clusters that differ in popularity.

— We propose a unified unbiased recommendation framework ICMRec based on item cluster-wise multi-objective optimization, which is featured with popularity discrepancy-based clustering, Pareto-efficiency solver, and counterfactual inference.

— We instantiate ICMRec with three state-of-the-art recommender models and conduct experiments on three real-world datasets. Experimental results demonstrate that ICMRec significantly alleviates the popularity bias problem and improves recommendation performance in recommender systems.

Fig. 1: Biases Occurrence and Amplification Loop.

(a) Gradient Norm Distribution  
(b) Gradient Conflict Example

Fig. 2: Empirical study on Gowalla dataset.

Some other works introduce additional models [14], [15] for pseudo-label or propensity score calculation, resulting in the estimation overlap issue [16]. On the other hand, methods using causal reasoning [17], [18], [19] have explicitly modeled the impact of item/user popularity through causal graphs. Though these methods have achieved performance gain, they did not address the exposure bias in RS.

To eliminate the biases in a unified framework, in this article, we analyze the bias issues of recommender system from an optimization perspective. Specifically, we observe that the interaction frequencies of all the items in the training data follow an extreme power-law distribution [20]. Next, we conduct an empirical study on the Gowalla dataset, on which we train the state-of-the-art LightGCN model. Figure 2(a) visualizes the norm (i.e., \(L_2\)-norm \(\|\|_2\)) of gradients coming from different items. Figure 2(b) gives examples of the overall gradients of one head item \(i_{1706}\), two niche items \(i_{8698}, i_{13155}\), and the total effect of the above three terms in one training epoch. We can draw the following observation:

— Head items have much larger gradient norms than niche items, indicating that the overall gradient direction is dominated by head items.

— There exists direction conflicts between gradients coming from head items and niche items. That is to say, updating model parameters based on gradients dominated by head items sacrifice the learning of niche items.

Motivated by the above observation, we propose Item Cluster-Wise Multi-Objective Recommendation (ICMRec)

1. https://snap.stanford.edu/data/loc-gowalla.html

The rest of this article is structured as follows. In Section 2 we present some preliminaries of this work, including our problem formulation and analysis of the items’ gradient influence. Then, Section 3 introduces technical parts of the ICMRec framework in detail and Section 4 presents the experimental results. Next, we introduce related work in Section 5 and conclude in Section 6.


2 PRELIMINARY

2.1 Problem Formulation

Let \( \mathcal{U} = \{u_1, u_2, \ldots, u_{|\mathcal{U}|}\} \) denote the set of users and items, respectively, where \(|\mathcal{U}|\) is the size of the user set, and \(|\mathcal{I}|\) is the size of the item set. In this paper, we consider training the recommender model from implicit feedback (i.e., observed interactions are considered as positive samples while negative samples are sampled from missing interactions). The positive interaction set between the users and items is defined as: \( \mathcal{R} = \{(u, i) | u \in \mathcal{U}, i \in \mathcal{I}, user\ u\ interacted\ with\ item\ i\}\). Formally, given the dataset \( \{(\mathcal{U}, \mathcal{I}, \mathcal{R})\} \), our task is to learn a predictive recommender model \( \mathcal{M} \) such that for each user, it can accurately rank all the candidate items in the user’s set according to his/her preference. After that, items with the Top-N prediction scores will constitute the final recommendation list to the user.

However, various biases occur in real-world RS, resulting in inferior model performance. As recent work has identified several types of biases in RS\cite{5}, in this work, we will focus on eliminating the following two classes of bias shown in Figure 1:

— Exposures Bias: This bias happens as users are constantly kept exposed to a small part of head items, so it’s hard for the recommender model to learn their preference on the less observed niche item set.

— Popularity Bias: The recommender model tends to overestimate the ranking scores of popular items while underestimating the ranking scores of niche items.

2.2 Bridging Item Popularity and Gradient Influence in Recommender Models

As for a traditional recommender model extracting individual user interest and item representations, under the normal training setting (i.e., each training sample has equal weight), the total loss function is defined on the whole training data, which is shown as

\[
\mathcal{L}^n = \mathcal{L}^+ + \mathcal{L}^- = \sum_{(u,i) \in \mathcal{R}^+} \mathcal{L}_{u,i} + \sum_{(u,i) \in \mathcal{R}^-} \mathcal{L}_{u,i}, \tag{1}
\]

where \(\mathcal{R}^+\) and \(\mathcal{R}^-\) denote the set of positive samples and the set of sampled negative samples, correspondingly, \(\mathcal{L}_{u,i}\) is the specific loss (i.e., cross-entropy) of \((u, i)\) pair. Then we expand the loss on positive samples as

\[
\mathcal{L}^+ = \sum_{(u,i) \in \mathcal{R}^+} \mathcal{L}_{u,i} = \sum_{i \in \mathcal{I}} \sum_{u \in \mathcal{R}^+_i} \mathcal{L}_{u,i} = \sum_{i \in \mathcal{I}} \mathcal{L}^+_i, \tag{2}
\]

where \(\mathcal{I}\) denotes the whole item set and \(\mathcal{R}^+_i\) is the set of users who have interacted with item \(i\) on positive samples.

In Equation (2), due to the popularity bias in training data, \(\mathcal{R}^+_i\) is extremely imbalanced. For head items, \(\mathcal{R}^+_i\) contains much more samples than niche items. As a result, given shared parameter \(\mathbf{W}_{sh}\), when we perform updates according to \(\frac{\partial \mathcal{L}^+_i}{\partial \mathbf{W}_{sh}}\), the majority of gradients would come from the loss on head items. When there are conflicts between gradients coming from head items and gradients coming from niche items, the normal training setting would scarrify the learning on niche items to achieve a lower overall loss. That is to say, the overall gradient direction is dominated by head items.

3 ICMRec: PROPOSED METHODOLOGY

To tackle the two challenges discussed in Section 2 and perform popularity debiasing for recommender systems, we propose to cluster the items with discrepancies in popularity and treat them as multiple optimization objectives with equal importance, resulting in our ICMRec framework. Specifically, it consists of three aspects: popularity discrepancy-based clustering, Pareto-efficiency solver, and counterfactual inference.

The workflow of ICMRec is provided in Figure 3. Firstly, according to the training target that the recommender model should perform equally for all clusters of items that differ in popularity, we explore items’ actual popularity level (via a specific neural model) from a novel cause-effect perspective and then devise popularity discrepancy-based bisecting clustering to separate the discriminated item clusters. Next, we acquire the non-conflict gradient direction for multiple item cluster-wise objectives from the Pareto-efficiency solver. Finally, we combine the above components in the training stage and perform counterfactual inference in the prediction stage to further eliminate the impact of item popularity. In this section, we will detail each component after the illustration of Item Cluster-Wise Multi-objective Optimization Principle.

3.1 Item Cluster-Wise Multi-Objective Optimization Principle

3.1.1 Definition of Item Cluster-Wise Multi-Objective Optimization

Unlike the traditional optimization process where the recommender model is treated as a unified objective, one intuitive idea for popularity debiasing is to consider the learning on each item as an objective and formulate the training over whole (positive) data as a weighted sum of these multiple item-wise objectives.

This method aims to find a set of weights for the item-wise objectives based on their gradients, so that they can be optimized without hurting the others. However, in practice, the size of the item set \(|\mathcal{I}|\) can be extremely large. As a result, naively considering each item-wise loss \(\mathcal{L}^+_i\) as an optimization objective mainly suffers from two problems: (1) the high variance of the reweighted loss \(\sum_i \mathcal{L}^+_i\); (2) the unacceptable computational cost of finding the weights.

To address these problems, ICMRec divides the whole item set into \(K\) clusters (\(K << |\mathcal{I}|\)) and treats the training target as an item cluster-wise multi-objective optimization problem, in which the model learning on each item cluster is set as a unique optimization objective.

Aiming that the learning on items that differ in popularity can be optimized fairly in the training process, we consider the item’s actual popularity factor during the identification of item clusters; items similar in popularity are likely to be put into the same cluster. Formally, we define the concept of Item Cluster-Wise Multi-Objective Optimization:
Definition 3.1. Given a popularity-unbiased recommender model $M$ with the loss function $L$, $u \in U$ is a user and $i \in I$ is an item belonging to cluster $c_i \in C$. The prediction probability $P(Y_{ui} = 1)$ from $M$ should be conditional independence from $i$'s popularity $|R^+_i|$ and cluster $c_i$:

$$P_M(Y_{ui} = 1) \perp |R^+_i| \iff P_M(Y_{ui} = 1) \perp c_i,$$

where $\perp$ denotes probabilistic independence. Hence, under the framework of ICMRec, during the training process of recommender model $M$, all the item cluster-wise objectives should be optimized with equal importance through a Pareto-efficient solver (no constraints like boundary settings are added). Through such a process, the gradients coming from niche items will obtain proper weights, updating the shared parameters of the model with a harmonious overall gradient direction.

### 3.1.2 Optimization Target of ICMRec

Formally, in the total parameter learning space $\theta$ of recommender model $M$, there exist certain parameters $\theta^{sh}$ which are shared between item cluster-wise objectives and some parameters $\theta^{ss}$ which are only correlated to a specific item cluster objective. ICMRec focuses on the solution of objective weights on $\theta^{ss}$, since $\theta^{ss}$ would not affect the learning of other item clusters. Taking all the above factors, we formulate the loss function of ICMRec to train $\theta^{sh}$ on positive samples as

$$L^{sh+} = \sum_{k=1}^{K} w_k L^+_{(k)}, \quad \text{where } L^+_{(k)} = \sum_{i \in \mathcal{N}_k} L^+_i,$$

$$\text{s.t. } \sum_{k=1}^{K} w_k = K, w_k \geq 0, \forall k \in \{1, \ldots, K\}$$

(4)

where $w_k$ is the weight for the $k$-th item cluster $c_k$ and $\mathcal{N}_k$ denotes the set of items in $c_k$. When $K = 1$, we can recover the normal training weight setting. $L_k$ is the learning objective for cluster $k$. For the training of objective-specific parameters $\theta^{ss}$, we still use $L^+$ as shown in Eq. (1).

In the following, we will describe the three main components of ICMRec in detail.

### 3.2 Popularity Discrepancy-based Clustering

Considering the item popularity during clustering, rustically clustering the items through the absolute ranking of interacted times is overly empirical and fails to consider the feature information of items. Here we first explore measuring the level of popularity discrepancy between item subsets and set it as a clustering criterion. In specific, for item subset popularity level measurement, we calculate its contained items' correlations with the global propensity of the user set $U$ and then design a novel adaptive item clustering algorithm finding item subsets with more severe popularity discrepancy.
non-linear feature interaction functions \[ f_2(\cdot) \]. However, the impact of global propensity on the candidate item is ignored by these methods. In the above causal framework, we can calculate each item’s correlation with the global propensity through \( f_2(\cdot) \).

3.2.2 Adaptive Item Clustering

We propose an unsupervised clustering algorithm Popularity Discrepancy-based Bisecting AE-KMeans (described in Algorithm 1) for discovering items clusters differ in popularity while incorporating their representation information. Firstly, in each iteration \( t \), the algorithm performs a bisection on the cluster with the largest variance of score \( S^g \) calculated in Eq. (6) by the Bisecting AE-KMeans [24], in which an autoencoder is introduced for high-level feature extraction from the original latent space of items representation \( X_T \). We denote the fetched original cluster as \( c_t \) and the bisected new clusters as \( c_{t1} \) and \( c_{t2} \).

**Popularity Discrepancy Measurement.** Aiming to find item clusters differ in popularity, we define a metric \( \hat{D} \) to measure the level of popularity discrepancy of \( c_{t1}, c_{t2} \) and \( c_{t2}, \) which is calculated by:

\[
\hat{D}_{(N_k)} = \frac{1}{|N|} \sum_{i \in N} f_d(x_i), \quad \text{given item set } N.
\]

In the above equations, \( f_d(\cdot) \) is the learned underlying causal mechanism of global propensity in Eq. (6), given cluster \( c_{t} \), \( S^g_{(N_k)} \) and \( S^g_{(N \setminus N_k)} \) are the average correlation score with the global propensity of items in and out of \( c_k \), respectively; \( \hat{D}_{(N_k)} \) is the L1-norm of the difference between the prior two terms, measuring the level of popularity discrepancy between items in and out of \( c_k \).

If the popularity discrepancy of any split new cluster \( \{c_{t1}, c_{t2}\} \) is greater than that of the original cluster \( c_t \), the split is performed; otherwise, \( c_t \) is kept. This bisection step repeats until the desired number of clusters \( K \) is reached.

**Algorithm 1 Popularity Discrepancy-based Clustering**

**Input:** Maximum cluster number \( K \);

- Item set \( I \) and embeddings \( X_T = \{x_1, x_2, \ldots, x_t\} \)

**Output:** Item clusters \( \mathcal{C} \)

1: Initialize item cluster: \( c_0 \leftarrow I \)
2: \( h \leftarrow \max\_\text{heap}([\sigma^2_{c_0}, c_0]) \) \( \sigma^2 \) : the key of \( h \), denoting cluster’s variance of \( S^g \) in Eq. (6)
3: while \( h.\text{size}() < K \) and not \( h.\text{isEmpty}() \) do
4: \( (\sigma^2_{c_1}, c_{t1}) \leftarrow h.\text{peek}() \) \( \Rightarrow \) Cluster \( c_{t1} \) with the largest variance of \( S^g \)
5: Perform Bisecting AE-KMeans on \( c_t \), get \( c_{t1}, c_{t2} \)
6: if \( \max(\hat{D}_{(N_{t1})}, \hat{D}_{(N_{t2})}) \geq \hat{D}_{(N_{t})} \) then
7: \( h.\text{pop}() \)
8: \( h.\text{push}((\sigma^2_{c_{t1}}, c_{t1}), (\sigma^2_{c_{t2}}, c_{t2})) \)
9: end if
10: end while
11: return \( h.\text{clusters} \)

3.3 Pareto-efficiency Solver

Obtaining the item clusters \( \mathcal{C} \), in the following, we will describe how to find \( w_k \) for each cluster objective. Firstly, we
provide a brief introduction to Pareto-efficiency and some related concepts.

Given a system that aims to minimize a series of objective functions \( L^+_1, \ldots, L^+_K \), Pareto-efficiency is a state when it is impossible to improve one objective without hurting other objectives. Formally, we provide the following definition:

**Definition 3.2.** For a minimization task of multiple objectives, let \( s_m \) and \( s_n \) denote two solutions as \( s_m = (f_1^m, \ldots, f_K^m) \) and \( s_n = (f_1^n, \ldots, f_K^n) \), \( s_m \) dominates \( s_n \) if and only if \( f_1^m \leq f_1^n, \ldots, f_K^m \leq f_K^n \).

Then the concept of Pareto-efficiency is defined as:

**Definition 3.3.** Solution \( s = (f_1, \ldots, f_K) \) is Pareto-efficient if and only if there is no other solution that dominates \( s \).

In ICMRec, we aim to find \( w_k \) so that the solution of each cluster-wise objective is Pareto-efficient, aka Item Cluster-Wise Pareto-efficiency. It is worth mentioning that Pareto-efficient solutions are not unique and the set of all such solutions is named as Pareto Frontier. For popularity debiasing, all the item cluster-wise objectives should be optimized with equal importance, thus we aim to find the Pareto-efficient solution that mostly balances all the objectives from the Pareto Frontier.

Formally, according to Definitions 3.3, we use the Karush-Kuhn-Tucker (KKT) conditions \([25]\) to describe the property of \( w_k \):

- \( w_1, \ldots, w_K \geq 0 \) and \( \sum_{k=1}^K w_k = K \).
- For shared parameters \( \theta^sh \), \( \sum_{k=1}^K w_k \nabla_{\theta^sh} L^+_k = 0 \).

As a result, the task of finding \( w_k \) can be formulated as

\[
\min_{w_1, \ldots, w_K} \left\{ \sum_{k=1}^K w_k \nabla_{\theta^sh} L^+_k \right\}
\]

\[s.t. \sum_{k=1}^K w_k = K, w_k \geq 0, \forall k \in \{1, \ldots, K\}\]

The optimization problem defined in Eq.(9) is equivalent to finding a minimum-norm point in the convex hull of the set of input points (i.e., \( w_k \)) \([25]\). Given the \( L^+_k \) in each training batch, we use the Frank-Wolfe algorithm \([27]\) to solve the convex optimization problem of Eq.(9), obtaining the balanced solution of \( w = (w_1, \ldots, w_K) \in \mathbb{R}^K \). Algorithm 2 shows the detail of this process.

### 3.4 Optimization and Counterfactual Inference

#### 3.4.1 Weighted Binary Cross-Entropy

We use binary cross-entropy (BCE) as the basement loss to optimize parameters, which has been intensively adopted in recommendation \([28, 29]\). More precisely, the specific loss \( L_{u,i} \) for a \((u, i)\) pair is formulated as

\[
L_{u,i} = -y_{ui} \log(\sigma(y_{ui})) + (1-y_{ui}) \log(1-\sigma(y_{ui})),
\]

where \( y_{ui} \) is the ground-truth label for the \((u, i)\) pair, \( \sigma(\cdot) \) is the sigmoid function. \( y_{ui} \) is produced according to Eq.(7).

The optimization function of ICMRec regarding the shared parameter \( \theta^sh \) is formulated as:

\[
L^{sh} = L^{sh^+} + L^{-} + \lambda_r \|\theta^{sh}\|_2^2
\]

where \( L^{sh^+} \) is calculated according to Eq.(4), \( L^- \) is calculated according to Eq.(1), \( \lambda_r \) is regularization coefficient.

For objective-specific parameter \( \theta^{os} \), we maintain the normal setting, the optimization function is formulated as

\[
L^{os} = L^n + \lambda_r \|\theta^{os}\|_2^2
\]

Algorithm 2 illustrates the overall training procedure of ICMRec. In each training batch, as shown in line 5–7, ICMRec first calculates the score function of the user individual interest causal path and global propensity causal path, which are combined into the total prediction score. Next, the items are clustered through the Popularity Discrepancy-based Clustering in line 8. ICMRec then performs the PE-solver to obtain the weight \( w_k \) for each item cluster \( c_k \) in line 9. While in line 10–13 we calculate the loss functions for shared and objective-specific parameters, respectively, then perform the update.

#### 3.4.2 Counterfactual Inference

Considering the causal graph in Figure 4(a), node \( P \) (denoting global propensity) has two edges pointing to node \( I \) and \( Y \), respectively. The path \( P \rightarrow I \) refers to the inherent exposure mechanism in recommender models \([19]\) while path \( P \rightarrow Y \) means the global propensity directly impacts the user behaviors, which increases observed interactions of head items conforming the propensity even though they may not match users’ true interest. For answering the counterfactual question: “what the prediction score would be if the framework only considers user individual interest?”, we remove the bad effect of popularity bias on model inference through cutting off the path \( P \rightarrow Y \) (shown in Figure 4(b)), making the prediction scores free from the global propensity we modeled in Equation (4). To this end, we perform counterfactual inference as follows:

\[
\tilde{Y}_{u,i} := S^n_{u,i}
\]

where \( \tilde{Y}_{u,i} \) is the final prediction score. Algorithm 4 describes the procedure of inference in ICMRec.

#### 3.5 Complexity Analyses of ICMRec

For the space complexity, ICMRec just introduces an MLP for modeling the item’s correlation with the global propensity in Eq.(6). Compared to the large number of item and
Algorithm 3 The Training Procedure of ICMRec

Input: Dataset \( \mathcal{R}^+, \mathcal{R}^- \); Backbone recommender model \( \mathcal{M} \); Maximum item cluster number \( K \); Learning rate \( \eta \) and all other hyperparameters;

Output: Parameters in the whole learning space \( \theta: \{\theta^{sh}, \theta^{os}\} \);

1: Initialize \( \theta \);
2: while not converge do
3: \((u, i) \leftarrow \) Sample a mini-batch from \( \mathcal{R}^+ \) and \( \mathcal{R}^- \);
4: \(x_u, x_i \leftarrow \) Lookup user, item representations from \( \mathcal{M} \);
5: \(S_{u,i}^u \leftarrow \) Calculate score function of user individual interest causal path using Equation (5);
6: \(S_{p,i}^g \leftarrow \) Calculate score function of global propensity causal path using Equation (6);
7: \(\tilde{Y}_{u,i} \leftarrow \) Calculate total prediction score function based on \(S_{u,i}^u \) and \(S_{p,i}^g \) using Equation (7);
8: \(C \leftarrow \) \{c_1, c_2, \ldots, c_K\} \leftarrow \) Generate \( K \) item clusters using Algorithm 1
9: \(w \leftarrow \) \{w_1, w_2, \ldots, w_K\} \leftarrow \) Update cluster weights using Algorithm 2
10: \(L^{sh} \leftarrow \) Compute loss for shared parameters based on \(\tilde{Y}_{u,i} \) and \(w \) according to Eq.(11);
11: \(L^{os} \leftarrow \) Compute loss for objective-specific parameters based on \(\tilde{Y}_{u,i} \) according to Eq.(12);
12: \(\theta^{sh} \leftarrow \theta^{sh} - \eta \cdot \partial L^{sh} / \partial \theta^{sh} ;
13: \theta^{os} \leftarrow \theta^{os} - \eta \cdot \partial L^{os} / \partial \theta^{os} ;
14: \) end while
15: return \( \theta \)

Algorithm 4 Inference of ICMRec

Input: User \( u \); Item \( i \); Backbone recommender model \( \mathcal{M} \);

Output: Prediction score \( \tilde{y}_{u,i} \);

1: \(x_u, x_i \leftarrow \) Lookup representations of \( u, i \) from \( \mathcal{M} \);
2: \(S_{u,i}^u \leftarrow \) Calculate score function of user individual interest causal path using Equation (5);
3: \(\tilde{y}_{u,i} \leftarrow \) Assign final score function directly based on \(S_{u,i}^u \) according to Equation (13);
4: return \( \tilde{y}_{u,i} \)

user embeddings in the parameter set, the additional space cost of ICMRec is negligible in practice.

In the following part, we will analyze the time complexity of ICMRec. Compared to the normal training setting of the backbone model, the training process of ICMRec contains two additional terms in each step: Item Clustering and PE-Solver. Suppose the embedding size of the backbone model is \( d \), by iterating \( K \) times, the total complexity of the item clustering process takes \( O(K \times |L| \times d) \); For the PE-Solver in Algorithm 2, its complexity mainly depends on the matrix multiplications in line 2, the dimension of the shared parameters mainly comes from the user embeddings, whose size is \(|U| \times d \). Therefore, this part costs \( O(K^2 \times |U| \times d) \). As a result, the additional time complexity of ICMRec is \( O(K \times |L| \times d + K^2 \times |U| \times d) \).

It should be noticed that in the inference stage of ICMRec, since the rating scores are calculated from the user individual interest causal path alone (in Eq.(5)), the time complexity of inference keeps the same as the backbone model.

4 EXPERIMENTAL SETUP

In this section, we conduct experiments aiming to answer the following research questions:

**RQ1**: How does the proposed ICMRec perform compared with normal training and other debiasing methods in terms of overall accuracy and unbiasedness?

**RQ2**: How do each component and different hyperparameter settings of ICMRec affect the recommendation performance?

**RQ3**: What about the solution of weights assigned for multiple item cluster-wise optimization objectives found by ICMRec?

**RQ4**: How is the interpretability of ICMRec?

4.1 Experimental Settings

4.1.1 Datasets and Pre-processing

We conduct experiments on three public accessible datasets: Last.Fm [1] Gowalla and Yelp2018 [2]. The datasets vary in scale, domain, and sparsity. Table 1 summarizes the statistics of the three datasets.

**Last.Fm**: This is a widely used dataset that contains 1 million ratings between users and movies. We binarize the ratings into implicit feedback. Interacted items are considered as positive samples. Due to the sparsity of the dataset, we use the 10-core setting, i.e., retaining users and items which have at least 10 interactions.

**Gowalla**: This is the check-in dataset obtained from Gowalla, where users share their locations by checking-in behavior [30]. To ensure the quality of the dataset, we use the 20-core setting.

**Yelp2018**: This dataset is adopted from the 2018 edition of the Yelp challenge. Wherein, the local business shops like restaurants and bars are viewed as the items. Similarly, we use the 20-core setting to ensure that each user and item have at least 20 interactions.

All the samples in the above datasets are binarized into interacted or not. To reflect the average recommendation performance over the item set and get rid of the power-law distribution of the raw dataset [20], we need a debiased test set for a fair evaluation. To this end, we follow the data preprocessing methods of previous approaches [17], [31] to construct an unbiased test set where the interactions are sampled from a uniform distribution over items (all the samples will be re-weighted by the debiasing algorithms).

4.1.2 Evaluation Protocols

We adopt cross-validation to evaluate the performance. The ratio of training / validation / test set is 80% / 10% / 10%. The ranking is performed among the whole item set. Each

---

3. https://files.grouplens.org/datasets/hetrec2011/hetrec2011-lastfm-2k.zip
4. https://www.kaggle.com/yelp-dataset/yelp-dataset/version/7
experiment is repeated 5 times and the average performance is reported. The recommendation quality is measured both in terms of overall accuracy and unbiasedness.

The overall accuracy is measured with two metrics: Recall@N and Normalized Discounted Cumulative Gain@N (NDCG@N). Recall@N measures how many ground-truth items are included in the top-N positions of the recommendation list. NDCG@N is a rank-sensitive metric that assigns higher weights to top positions in the top-N recommendation list [32].

For the evaluation of recommendation unbiasedness, we first split the whole item set \( I \) into \( I_h \) and \( I_n \) according to the ratio of 20% / 80% Pareto Principle. \( I_h \) represents the set of head items and \( I_n \) denotes the set of niche items. Here 20% means the top 20% of the total item numbers, other than 20% of the total interactions. We then adopt the following six metrics.

**Recall@N and NDCG@N:** Recall@N measures how many head items belong to \( I_h \) are included in the top-N positions of the recommendation list and interacted with the user. Similarly, NDCG@N only considers head items.

**Recall-Niche@N and NDCG-Niche@N:** Recall-Niche@N measures how many niche items belong to \( I_n \) are included in the top-N positions of the recommendation list and interacted with the user. Similarly, NDCG-Niche@N only considers niche items.

**Coverage@N and APT@N:** Coverage measures the coverage of all the top-N recommendation lists to the whole item set \( I \). Average Percentage of niche items (APT) among the recommendation list is another more readily interpretable but closely related metric we use for evaluation. Precisely, Coverage@N and APT@N are defined as:

\[
\text{Coverage}@N = \frac{|\bigcup_{u \in \text{test}} \text{list}_{@N}(u)|}{|I|} \tag{14}
\]

\[
\text{APT}@N = \frac{1}{|U|} \sum_{u \in \text{test}} \frac{|\text{list}_{@N}(u) \cap I_n|}{|\text{list}_{@N}(u)|} \tag{15}
\]

where \( \text{list}_{@N}(u) \) represents the list of top-N recommended items for each user \( u \) in the test set.

### 4.1.3 Baselines

We instantiate the proposed ICMRec with three renowned recommender models, including classic matrix factorization, neural-network-based model, and graph-based model:

- **Matrix Factorization (MF)** [33]: The most widely-used recommender model using linear user and item embeddings for production-based prediction score.

- **Neural Matrix Factorization (NeuMF)** [3]: NeuMF is one notable deep learning-based recommender model. It combines matrix factorization and multi-layer perceptrons (MLP) to learn high-order interaction signals.

- **LightGCN** [4]: LightGCN is a state-of-the-art graph-based model that learns user and item representations by linearly propagating them on the interaction graph. The user and item embedding is formulated as the aggregation of hidden vectors in all layers.

| Backbone Method | R@20 | NG@20 | R@20 | NG@20 | R@20 | NG@20 |
|----------------|------|-------|------|-------|------|-------|
| Normal         | 0.0396 | 0.0383 | 0.1594 | 0.1375 | 0.0423 | 0.0312 |
| UBPR           | 0.0404 | 0.0382 | 0.1652 | 0.1426 | 0.0479 | 0.0356 |
| DICE           | 0.0407 | 0.0382 | 0.1699 | 0.1439 | 0.0514 | 0.0369 |
| MACR           | 0.0413 | 0.0391 | 0.1762 | 0.1544 | 0.0546 | 0.0415 |
| SIPW           | 0.0406 | 0.0386 | 0.1719 | 0.1542 | 0.0531 | 0.0415 |
| CPR            | 0.0421 | 0.0399 | 0.1873 | 0.1616 | 0.0580 | 0.0437 |
| ICMRec         | 0.0409* | 0.0387* | 0.1754* | 0.1489* | 0.0513* | 0.0386* |

* denotes significance p-value <0.01 compared with the best baseline.

Each model is trained with the following model-agnostic debiasing frameworks:

- **Normal Training (Normal):** This is the normal training procedure with vanilla BCE loss, which is shown in Eq.(1).

- **Unbiased Bayesian Personalized Ranking (UBPR)** [12]: UBPR is an inverse-propensity-score based method that modifies the conventional pairwise loss. It uses a non-negative loss function for reducing the variance of the estimator.

- **Disentangling Interest and Conformity with Causal Embedding (DICE)** [31]: This is a state-of-the-art debiasing method that learns disentangled embeddings for the two causes, user interest and conformity.

- **Model-Agnostic Counterfactual Reasoning (MACR)** [17]: It is a causal-reasoning method that utilizes counterfactual inference to remove the direct effect of item properties on rank scores, mitigating the popularity bias.

- **Self-Inverse Propensity Weighting (SIPW)** [10]: It is a method gradually mitigating the exposure bias of the recommender system without incurring high computational costs. It reuses the model’s prior prediction as the sample’s propensity score of the current training step.

- **Cross Pairwise Ranking (CPR)** [22]: It is a method forming the loss term as the combination of multiple observed interactions at once, removing the influence of data biases caused by the exposure mechanism.

### 4.1.4 Parameter Settings

All methods are learned with the Adam optimizer [34] except using RMSprop optimizer in NeuMF-based models.
The batch size is set as 1024. The learning rate is set as $1e^{-3}$. We evaluate the validation set every 3000 batches of updates. For a fair comparison, the embedding size is set as 64 for all models. For NeuMF and LightGCN, we utilize a three-layer structure. The node-dropout and message-dropout in LightGCN are set as 0.1 on all datasets. For hyperparameters of ICMRec, $\alpha$, and $\lambda$, are searched between $\{1e^{-4}, 1e^{-3}, 2e^{-3}, 3e^{-3}, 1e^{-2}\}$ on all three datasets. The maximum item cluster number $K$ is searched between $\{3, 4, 5, 6\}$ on various datasets. Note that model hyperparameters keep the same across all different training frameworks for a fair comparison.

4.2 Performance Comparison (RQ1)

Experimental results show the superiority of ICMRec in terms of both accuracy and unbiasedness.

4.2.1 Overall Accuracy Comparison.

Table 2 shows the overall accuracy performance of top-N recommendation on all three datasets: Last.Fm, Gowalla, and Yelp2018, respectively. Although ICMRec is proposed to tackle the bias issues in RS, in all cases, it outperforms normal training and other debiasing frameworks in terms of overall accuracy. It demonstrates that ICMRec achieves a better trade-off between unbiasedness and overall accuracy compared with other frameworks. The performance gain regarding the overall accuracy metrics NDCG@20 / Recall@20 is 4.03% / 2.98%. This improvement mainly comes from promoting high-quality niche items while downgrading irrelevant head items.

4.2.2 Unbiasedness Comparison.

Figure 5 and Figure 6 visualize the performance of ICMRec and other competing methods in terms of Recall-Head@20, Recall-Niche@20, NDCG-Head@20, and NDCG-Niche@20 on the state-of-the-art LightGCN backbone model. A similar trend can also be observed in MF and NeuMF. Table 3 shows the performance of ICMRec and other competing methods in terms of Coverage@20 and APT@20. We have the following observations:

(1). According to Table 3, ICMRec achieves the best unbiasedness performance among all methods. This observation confirms that the proposed ICMRec is effective to alleviate the bias issues and generate better recommendations on the niche item set. Meanwhile, the item coverage rate gets greatly improved, demonstrating the fairness of ICMRec. Specifically, the backbone model with ICMRec achieves the average absolute Coverage@20 / APT@20 gain of 15.45% / 33.23%.

(2). In Figure 5 and Figure 6, the points which fall into the top-right area indicate better performance with both higher head and niche accuracy. It’s obvious that the proposed ICMRec achieves the highest niche accuracy while only sacrifices a little head accuracy compared with normal training. However, other methods cannot achieve such a performance. In most cases, these methods tend to lead to a larger decrease in head accuracy while obtaining a smaller gain in niche accuracy. This result demonstrates that ICMRec achieves a better balance between head items and niche items, compared with other methods. The performance gain of ICMRec mainly comes from the growth in niche items without the loss across head items.

(3). We conduct one-sample t-tests and the obtained results (i.e., p-value < 0.01) indicate that the improvement regarding both recommendation unbiasedness metrics and overall accuracy metrics of ICMRec is statistically significant.

4.3 Ablation Study and Hyper-parameter Study (RQ2)

4.3.1 Ablation Study

In this part, we conduct an ablation study to analyze the functionality of the main components of ICMRec (i.e., item cluster-wise re-weighting (CR), popularity discrepancy-based bisecting clustering (PD), and counterfactual inference (CI)). Table 4 shows the performance of ICMRec and its variants on all three datasets using LightGCN as the backbone recommender model. We introduce the variants and analyze the component effects respectively:

(1). Remove Cluster-wise Re-weighting (w/o CR): The most significant unbiasedness and overall accuracy degradation occurred without the re-weighting strategy. This proves the effectiveness of our proposed method: clustering the items and treating the recommendation target as an item cluster-wise multi-objective optimization problem can greatly alleviate the inherent bias issues of the model and improve its performance.

(2). Remove Popularity Discrepancy-based Bisecting Clustering (w/o PD): In this variant, we replace the item clustering method with AE-KMeans. After ignoring the item popularity factor and performing clustering only in the item representation space, we find that the overall performance gets dropped. This result validates that item popularity should be a necessarily concerned during its clustering process and our popularity discrepancy-based clustering method

| Backbone Method | Cov@20 | AT@20 | Cov@20 | AT@20 | Cov@20 | AT@20 |
|-----------------|--------|-------|--------|-------|--------|-------|
| **Normal**      | 0.3508 | 0.1065| 0.3547 | 0.0920| 0.2955 | 0.0545|
| UBPR            | 0.3785 | 0.1333| 0.3710 | 0.1178| 0.3035 | 0.0639|
| DICE            | 0.4115 | 0.1790| 0.3912 | 0.1190| 0.3767 | 0.0708|
| MACR            | 0.4211 | 0.1851| 0.3710 | 0.1391| 0.3841 | 0.0923|
| SIPW            | 0.4248 | 0.1807| 0.4195 | 0.1257| 0.4745 | 0.0743|
| CPR             | 0.4273 | 0.1835| 0.4307 | 0.1331| 0.3923 | 0.0919|
| **ICMRec**      | 0.4418 | 0.2043| 0.4860 | 0.1600| 0.4677 | 0.1168|

* denotes significance p-value < 0.01 compared with the best baseline.
The general observation is that overall recommendation trend can also be observed on other datasets and backbones. A similar illustration, NDCG@20 and NDCG-Niche@20 under different cluster numbers on Last.Fm and Yelp2018 dataset. A similar trend can also be observed on other datasets and backbones. The general observation is that overall recommendation accuracy basically maintains the same level while the unbiasedness performance and accuracy on the niche item set manifest bell-shaped curves. Increasing the cluster number from 3 to 4 leads to the largest performance gain. Later on, the model’s unbiasedness and performance on the niche item set keep diminishing along with the increase of cluster number. Such experimental results indicate that adaptively can help distinguish different item cluster-wise optimization objectives.

(3) Remove Counterfactual Inference (w/o CI): If the user global propensity causal path is kept, we find that the unbiasedness metrics get harmed heavily, indicating that using the combined causal model for inference is biased by the item popularity. On the other hand, with the participation of counterfactual inference, we perform inference purely based on the user’s individual interest, which can better reflect the user’s true preference.

To sum up, the combination of the above strategies (i.e., ICMRec) yields the best performance, proving that all the components of ICMRec are effective and work collaboratively to improve the recommendation unbiasedness performance and overall accuracy.

4.3.2 Hyper-parameter Study

(1) Effect of Item Cluster Number $K$: In this part, we use LightGCN as the backbone recommender model since it has the start-of-the-art overall accuracy performance. Here we choose $K \in \{3, 4, 5, 6\}$ to conduct experiment. Figure 7 illustrates, NDCG@20 and NDCG-Niche@20 under different cluster numbers on Last.Fm and Yelp2018 dataset. A similar trend can also be observed on other datasets and backbones. The general observation is that overall recommendation accuracy basically maintains the same level while the unbiasedness performance and accuracy on the niche item set manifest bell-shaped curves. Increasing the cluster number from 3 to 4 leads to the largest performance gain. Later on, the model’s unbiasedness and performance on the niche item set keep diminishing along with the increase of cluster number. Such experimental results indicate that adaptively

### TABLE 4: Ablation study of method components on three datasets. Boldface denotes the highest scores. w/o denotes without. ↓ indicates a severe performance drop (more than 10%).

| Dataset Variant | Unbiasedness Metrics | Overall Metrics |
|-----------------|----------------------|-----------------|
|                 | $R_T$@20 | $N_T$@20 | Cov@20 | $AT_T$@20 | $R_T$@20 | $NG_T$@20 |
| Default         | 0.0177   | 0.0136   | 0.4359 | 0.1784   | 0.1845   | 0.1573   |
| w/o CR          | 0.0171   | 0.0128   | 0.4359 | 0.1784   | 0.1845   | 0.1573   |
| w/o PD          | 0.0171   | 0.0128   | 0.4359 | 0.1784   | 0.1845   | 0.1573   |
| w/o CI          | 0.0171   | 0.0128   | 0.4359 | 0.1784   | 0.1845   | 0.1573   |
| Default         | 0.0177   | 0.0128   | 0.4359 | 0.1784   | 0.1845   | 0.1573   |
| w/o CR          | 0.0171   | 0.0136   | 0.4359 | 0.1784   | 0.1845   | 0.1573   |
| w/o PD          | 0.0171   | 0.0136   | 0.4359 | 0.1784   | 0.1845   | 0.1573   |
| w/o CI          | 0.0171   | 0.0136   | 0.4359 | 0.1784   | 0.1845   | 0.1573   |
| Default         | 0.0177   | 0.0136   | 0.4359 | 0.1784   | 0.1845   | 0.1573   |

Fig. 5: Recall-Head@20 and Recall-Niche@20 on LightGCN

Fig. 6: NDCG-Head@20 and NDCG-Niche@20 on LightGCN

(c) Yelp2018
assigning the items into 4 clusters leads to the most satisfactory performance. However, with more item clusters, the ability of ICMRec in debiasing gets compromised. The reason could be that ICMRec would put more focus on the balance between clusters that differ in representation information rather than the balance between head and niche items when K is too large.

(2). Effect of User Global Propensity Causal Path Coefficient $\alpha$: To evaluate the impact of coefficient $\alpha$, we vary its value in the range of $\{0, 1e^{-4}, 1e^{-3}, 2e^{-3}, 3e^{-3}, 5e^{-3}, 1e^{-2}\}$. The experimental results are summarized in Figure 8. We can observe that the overall NDCG@20 reaches its peak when $\alpha = 2e^{-3}$, thus demonstrating that properly decomposing the factors behind user interactions can truly extract users’ individual interest. Meanwhile, with the increment of $\alpha$, the model’s unbiasedness and performance on the niche item set keep rising swiftly. This implies that more niche items are excavated when we put more emphasis on the user global propensity. To sum up, for higher overall accuracy, we choose $\alpha = 2e^{-3}$ as our default setting.

4.4 Investigation of Item Cluster-wise Objective Optimization Solution (RQ3)

In this part, we conduct experiments to demonstrate whether ICMRec generates a debiased Pareto-efficient solution and attaches proper importance to the niche item set.

4.4.1 Pareto Frontier and the Searched PE Point

On the Gowalla dataset with all the three recommender models, we mark our ICMRec solution and generate the Pareto Frontiers of head-set and niche-set losses by running the Pareto MTL algorithm $[28]$ with various trade-off preference vectors, shown in Figure 9(a) and 10(a). It can be observed that the obtained Pareto Frontiers under different constraints follow Pareto-efficiency, i.e., no point achieves both lower head and niche losses than other points. When the model focuses more on the head item set, the head-set loss is lower while the niche-set loss increases, and vice versa.

When it comes to the found solution of ICMRec, we can observe that on all recommender models, those points mainly lie in the middle part of the Pareto Frontiers. This observation indicates that the Pareto-efficiency solver component of ICMRec coincides with our aim of balancing the trade-off between head items and niche items. Furthermore, the solutions of ICMRec obtain fewer losses on both of the two objectives than the Pareto Frontier generated by Pareto MTL, manifesting its superiority.

4.4.2 Weights Learning Curves

To be clear of the training process and weight assignment, we further plot the curves of the average weights assigned to niche items until convergence, as shown in Figure 9(b). We use LightGCN as the backbone recommender model on all three datasets. We can observe that the found weight
solutions of ICMRec mainly focus on the niche item set. After varying at the early training stage, the weight for the niche set becomes flattened and then converges to a value around $[1.04, 1.12]$. The reason draws on the fact that the gradient norm divergence between head and niche items is highly large at the initial training stage; later on, when the model parameters have been updated towards the gradient direction of head items, the gradients’ norm of head items reduced while those of niche items get amplified in scale. On the other hand, the normal training setting neglects these Pareto-efficient weights and treats all items in a unified optimization target, leading to the bias toward head items and pre-exposed items. Hence, the proposed ICMRec can effectively eliminate the bias issues of RS by assigning adaptive weights to head and niche cluster-wise objectives.

4.5 Case Study (RQ4)
To interpret the effectiveness of ICMRec, on the Last.Fm dataset, we randomly select a user u715 and retrieve his two top-5 recommendation lists from LightGCN-Normal and LightGCN-ICMRec given the same historical interactions, visualized in Figure 10. We observe that the recommendation list derived from LightGCN-Normal contains popular ground-truth items but does not contain any niche items. LightGCN-ICMRec contains two niche items, Farben Lehre and Plavi Orkestar thanks to the proper weights assigned to niche items in his interaction history. One of the two recommended niche items belongs to the ground-truth test set, which improves the model’s unbiasedness and overall accuracy. Note that the two recommendation lists have several items in common (e.g. Erasure), which indicates that LightGCN-ICMRec can also capture users’ preferences for head items.

5 RELATED WORK
In this section, we summarize the related work into the following two categories: unbiased recommendation, and multi-objective optimization in recommendation.

5.1 Methods for Unbiased Recommendation
Inverse Propensity Weighting (IPW) is a widely-used debiasing method for RS [35], which re-weights each sample loss to recover the unbiased distribution. IF4UREc [7] calculates the influence score of each sample based on its Hessian matrix to the parameters in the neural network. AutoDebias [14] combines IPW with a meta-learning paradigm to find a universal solution for debiasing. Jae-wonoo et al. [10] designed BISER that reuses the model’s prior prediction as the sample’s propensity score of the current training step. Nonetheless, the high variance of estimated propensity score is the main drawback of these IPW-based methods. By contrast, in the cluster-wise optimization setting of ICMRec, such an issue has been greatly mitigated.

Based on causal graphs and operators, causal reasoning methods could also eliminate the negative effect of item popularity in recommendation. DICE [31] assigns two independent representations to each user and item, respectively modeling the separate effects of relevance and conformity. However, assigning each item a unique conformity embedding may lead to model over-parameterization and overfitting. PDA [19] removes the confounding popularity bias and adjust the recommendation score during training. MACR [17] removes the direct causal effect of user conformity and item conformity through counterfactual inference alone. However, the above methods could not address the exposure bias caused by the self-loop of RS (Figure 1).

Besides, some empirically designed methods relieve the bias factors based on auxiliary knowledge input such as side-information, user feedback, and niche item clustering. [36], [37]; CPR [22] forming the unbiased loss term as the combination of multiple observed interactions at once; Some works [38], [39] introduce domain adaptation by gaining unbiased domain knowledge from a small but unbiased dataset and apply it to the main-biased dataset for training.

In summary, none of the existing methods emphasized solving the bias issues in RS during the training process from item gradient or cluster-wise objective optimization perspective. In this work, we define recommendation as an item cluster-wise objective optimization problem, guiding the model to balance the gradient update on all item clusters that differ in popularity. We assign adaptive weights to the gradient of cluster-wise objectives according to Pareto-efficiency. Through such approach, biases in RS can be effectively eliminated.

5.2 Multi-Objective Optimization in Recommendation
In the research field of RS, despite that overall accuracy is always set as the main objective of recommendation, some researches have also been done focusing on other objectives such as availability, profitability, and usefulness [40], [41]. Besides, metrics about diversified recommendations such as diversity, novelty, and fairness are also considered as objectives [42], [43]. Recently, user-oriented objectives such as user sentiment are considered for better recommendation [44], [45]. As for a commercial RS, CTR (Click Through Rate) and GMV (Gross Merchandise Volume) are included in the objectives [28], [46] to gain higher profits.

The optimization methods for multiple objectives can be categorized into two categories: heuristic search [47] and scalarization [48], [49]. Evolutionary algorithms are popular choices for heuristic search which deal simultaneously with a series of possible solutions in a single run [50]. But these algorithms usually depend heavily on the heuristic experience [47], [51]. Scalarization methods transform multiple objectives into a single one with a weighted sum of all objective functions [49]. Then the overall objective function is optimized to be Pareto-efficient, where no single objective can be further improved without hurting the others [28].

In this paper, we aim to address the bias issues in recommendation from a multi-objective optimization perspective. Unlike existing methods which introduce new metrics as the objectives [28], [42], [43], we consider the learning on each cluster of items as an objective. After that, we focus on finding an optimal weight solution for the cluster-wise objectives so that an overall harmonious gradient direction can be obtained.

6 Conclusion
In this paper, we propose to tackle the bias issues in recommendation from a multi-objective optimization per-
spective. We first find that head items are repetitively recommended due to the fact that head items tend to have larger gradient norms and thus dominate the gradient updates. Learning parameters based on such gradients could sacrifice the model’s performance on niche items. To alleviate such a phenomenon, we propose a model-agnostic framework namely ICMRec, modeling the recommendation task as an item cluster-wise multi-objective optimization problem. Specifically, ICMRec is featured with popularity discrepancy-based clustering, Pareto-efficiency solver, and counterfactual inference. We instantiate ICMRec with three state-of-the-art recommender models and conduct extensive experiments on three real-world datasets. The results demonstrate the effectiveness of ICMRec. Future work includes generalizing ICMRec for other similar tasks such as multi-class classification and long-tail document retrieval.

REFERENCES

[1] S. Rendle, “Factorization machines,” in 2010 IEEE International Conference on Data Mining. IEEE, 2010, pp. 995–1000.
[2] A. Mnih and R. R. Salakhutdinov, “Probabilistic matrix factorization,” Advances in neural information processing systems, vol. 20, pp. 1257–1264, 2007.
[3] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, “Neural collaborative filtering,” in WWW, 2017, pp. 173–182.
[4] X. He, “Lightgcn: Simplifying and powering graph convolution network for recommendation,” in SIGIR, 2020, pp. 639–648.
[5] J. Chen, H. Dong, X. Wang, F. Feng, M. Wang, and X. He, “Bias and debias in recommender system: A survey and future directions,” arXiv preprint arXiv:2010.03240, 2020.
[6] J. Li, K. Lu, Z. Huang, and H. T. Shen, “Two birds one stone: on both cold-start and long-tail recommendation,” in Proceedings of the 25th ACM international conference on Multimedia, 2017, pp. 896–906.
[7] G. Adomavicius and Y. Kwon, “Improving aggregate recommendation diversity using ranking-based techniques,” IEEE Transactions on Knowledge and Data Engineering, vol. 24, no. 5, pp. 896–911, 2011.
[8] C. Anderson, The Long Tail. Hachette Books, 2006.
[9] D. Jannach, L. Lerche, K. Mamekhkhos, and M. Jugovac, “What recommenders recommend: an analysis of recommendation biases and possible countermeasures,” UMUI, vol. 25, no. 5, pp. 427–491, 2015.
[10] J.-w. Lee, S. Park, J. Lee, and J. Lee, “Bilateral user-unbiased learning from biased implicit feedback.” 2022.
[11] Y. Saito, S. Yaginuma, Y. Nishino, H. Sakata, and K. Nakata, “Unbiased recommender learning from missing-not-at-random implicit feedback,” in Proceedings of the 13th International Conference on Web Search and Data Mining, 2020, pp. 501–509.
[12] Y. Saito, “Unbiased pairwise learning from biased implicit feedback,” in Proceedings of the 2020 ACM SIGIR on International Conference on Theory of Information Retrieval, 2020, pp. 5–12.
[13] A. Swaminathan and T. Joachims, “The self-normalized estimator for counterfactual learning,” advances in neural information processing systems, vol. 28, 2015.
[14] J. Chen, H. Dong, Y. Qiu, X. He, X. Xin, L. Chen, G. Lin, and K. Yang, “Autodebias: Learning to debias for recommendation,” in Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2021, pp. 21–30.
[15] Z. Zhu, Y. He, Y. Zhang, and J. Caverlee, “Unbiased implicit recommendation and propensity estimation via combinational joint learning,” in Fourteenth ACM Conference on Recommender Systems, 2020, pp. 551–556.
[16] M. Ren, W. Zeng, B. Yang, and R. Urtsasun, “Learning to reweight examples for robust deep learning,” in International conference on machine learning. PMLR, 2018, pp. 4334–4343.
[17] T. Wei, F. Feng, J. Chen, Z. Wu, J. Yi, and X. He, “Model-agnostic counterfactual reasoning for eliminating popularity bias in recommender system,” in Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, 2021, pp. 1791–1800.
[40] T. Jambor and J. Wang, “Optimizing multiple objectives in collaborative filtering,” in RecSys, 2010, pp. 55–62.
[41] S. M. McNee, J. Riedl, and J. A. Konstan, “Being accurate is not enough: how accuracy metrics have hurt recommender systems,” in CHI’06, 2006, pp. 1097–1101.
[42] M. T. Ribeiro, A. Lacerda, A. Veloso, and N. Ziviani, “Pareto-efficient hybridization for multi-objective recommender systems,” in Proceedings of the sixth ACM conference on Recommender systems, 2012, pp. 19–26.
[43] M. T. Ribeiro, N. Ziviani, E. S. D. Moura, I. Hata, A. Lacerda, and A. Veloso, “Multiobjective pareto-efficient approaches for recommender systems,” TIST, vol. 5, no. 4, pp. 1–20, 2014.
[44] C. Musto, M. de Gemmis, G. Semeraro, and P. Lops, “A multi-criteria recommender system exploiting aspect-based sentiment analysis of users’ reviews,” in Proceedings of the eleventh ACM conference on recommender systems, 2017, pp. 321–325.
[45] M. Rodriguez, C. Posse, and E. Zhang, “Multiple objective optimization in recommender systems,” in Proceedings of the sixth ACM conference on Recommender systems, 2012, pp. 11–18.
[46] P. Nguyen, J. Dines, and J. Krasnodebski, “A multi-objective learning to re-rank approach to optimize online marketplaces for multiple stakeholders,” arXiv preprint arXiv:1708.00651, 2017.
[47] E. Zitzler, M. Laumanns, and L. Thiele, “Spea2: Improving the strength pareto evolutionary algorithm,” TIK-report, vol. 103, 2001.
[48] J.-A. Desideri, “Multiple-gradient descent algorithm (mgda),” 2009.
[49] L. Xiao, Z. Min, Z. Yongfeng, G. Zhaquan, L. Yiqun, and M. Shaoping, “Fairness-aware group recommendation with pareto-efficiency,” in Proceedings of the Eleventh ACM Conference on Recommender Systems, 2017, pp. 107–115.
[50] S. Wang, M. Gong, H. Li, and J. Yang, “Multi-objective optimization for long tail recommendation,” Knowledge-Based Systems, vol. 104, pp. 145–155, 2016.
[51] A. Kendall, Y. Gal, and R. Cipolla, “Multi-task learning using uncertainty to weigh losses for scene geometry and semantics,” in CVPR, 2018, pp. 7482–7491.

Yule Wang is currently working towards the Ph.D. degree in the School of Computational Science and Engineering, Georgia Institute of Technology (Gatech), United States. He received his M.S. and B.S. degree at Shanghai Jiao Tong University (SJTU). His major research interests include neural latent discovery, deep generative models and data mining.

Xin Xin is now a tenure-track assistant professor in the school of computer science and technology, Shandong University. His research interests include machining learning and reinforcement learning for recommender systems and information retrieval. He has published more than 15 papers in top-ranking conferences, including SIGIR, IJCAI, ACL, WSDM, etc. He also serves as the program committee member and invited reviewer for tire-1 conferences and journals, such as SIGIR, IJCAI, WSDM, ACL, ACM Multimedia, TKDE, TOIS.

Yunzhe Li is a Ph.D. student in Department of computer science, University of Illinois, Urbana-Champaign/UIUC). His research interests involve information retrieval, natural language generation and recommender system.

Dong Wang received the bachelor’s and master’s degrees from Xi’an Jiaotong University and Chongqing University, in 1991 and 1994, respectively, and the Ph.D. degree in mechanical manufacturing and automation from Shanghai Jiao Tong University (SJTU), in 2001. He is currently a professor with the School of Software, SJTU. He was in charge of many projects supported by programs like the National High Technology Research and Development Program. His main research interests include RFID, Internet of Things, pervasive computing, wireless sensing, data mining, and recommender systems. He has many publications in top journals and conferences, like Ubicomp/IMWUT, Sensys, WWW, IJCAI, Percom and CIKM.