Towards Long-term Fairness in Interactive Recommendation: A Maximum Entropy Reinforcement Learning Approach

Xiaoyu Shi[1,2],Quanliang Liu[1,2], Hong Xie[1,2,*], Mingsheng Shang[1,2]
1. Chongqing Key Laboratory of Big Data and Intelligent Computing,
Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences, Chongqing, China
2. Chongqing School, University of Chinese Academy of Sciences, Chongqing, China
Email:{xiaoyushi,xiehong,msshang}@cigit.ac.cn

***Abstract*—This paper considers the problem of maintaining the long-term fairness of item exposure in interactive recommendation systems under the dynamic setting that user preference and item popularity evolve over time. The challenge is that the evolving dynamics of user preference and item popularity in the feedback loop amplify the long-term "unfairness" of item exposure. To address this challenge, we first formulate a constrained Markov Decision Process (MDP) to capture evolving dynamics of user preference. The proposed constrained MDP imposes long-term fairness requirements via maximum entropy techniques. Moreover, to illuminate the "unfairness" amplifying effect caused by the evolving dynamic of item popularity in the feedback loop, we design a debiased reward function to eliminate popularity bias in the training data. To this end, the proposed framework can maintain acceptable recommendation accuracy while exposing items as randomly as possible, ensuring long-term benefits for users. Experiments on three datasets demonstrate the effectiveness and superiority of our proposed framework in terms of recommendation performance and fairness.**

***Index Terms*—Recommendation System, Long-term Fairness, Popularity Bias, Reinforcement Learning, Maximum Entropy Policy**

I. Introduction

Recommender system (RS) is committed to providing *personalized* web-based products and services (generally called *items*) for users via filtering massive information. It plays an essential role in various online platforms, such as product recommenders for online stores, playlist generators for video and music websites, and content pushers for social media and content-sharing platforms. RS usually faces the *popularity bias* in the observation data, which exhibits the extremely imbalanced or long-tailed distribution over items. The recommender trained on such skewed data leads to the item exposure fairness issue [1], [2]: a small number of popular items occupy a lot of exposure rate, while the majority of items receive little attention from users. As a result, it severely damages the visibility of unpopular items.

To address this issue, several approaches have been proposed to develop a fairness-aware recommender. For instance, eliminating the underlying bias or sensitive features on training data before the learning process [3], [4], introducing the additional regularization term on the training loss [5], [6], or performing a post-hoc ranking adjustment to re-ranking the recommendation list [7], [8]. However, most of them study fair decision-making in a one-shot setting, i.e., their recommendation decisions are made by maximizing fairness in static scenarios. In such situation, blindly trusting the static fair decisions may even exacerbate unfairness future [9], [10], since they do not consider the evolving dynamics of user preference in the environment and ignore the amplification effect of the feedback loop in the recommendation system. Thus, these methods can not satisfy the long-term fairness in dynamic environments.

In this paper, we focus on studying the *long-term fairness* of item exposure in the interactive recommendation. The long-term fairness views the recommendation as a sustainable interactive process instead of one-shot objective, which aims to satisfy fairness in the long run by meeting dynamic factors over time. An interactive recommendation system (IRS) formatted the recommendation process as a sequential decision-making process, allowing us convenient to learn recommendation policies that adapt to dynamic environments and reduce the amplification effects of the feedback loop in RS.

Toward this end, we seek to establish a relationship between item exposure rate and information entropy, to guide model training in a more fair direction. In RS, entropy can be used to measure the inconsistency of item exposure. The lower the value, the fairer the recommendations generated by a recommendation policy. Therefore, we draw on the entropy to observe and evaluate the current degree of fairness of a recommendation system.

We propose an offline reinforcement learning (RL) approach that leverages Soft Actor-Critic framework for Interactive Recommendation (named SAC4IR) to automatically planning

This work is supported in part by the National Natural Science Foundation of China under Grants 62072429, in part by the Chinese Academy of Sciences "Light of West China" Program, and in part by the Key Cooperation Project of Chongqing Municipal Education Commission (HZ2021008, HZ2021017), and the "Fertilizer Robot" project of Chongqing Committee on Agriculture and Rural Affairs. * is corresponding author.

the optimal recommendation policy under entropy constraint. The reasons for choosing offline RL can be summarized as: 1) the learning time and deploying cost of training RL policy online is huge and unacceptable, and 2) an immature policy hurts the user satisfaction on recommendation results [11]. Our offline RL learning framework consists of three main steps: 1) learning a debiasing-aware reward model to capture both user interest and item popularity bias, 2) leveraging the debiasing-aware reward model to guide the RL policy training under entropy constraint, and 3) evaluating the learned RL policy in terms of recommendation accuracy and fairness ability. Moreover, SAC4IR adopts a stochastic actor to generate the recommendation list by maximizing the defined debiasing-aware reward model with entropy constraints.

Our contributions are summarized as follows:

- We introduce the information entropy to measure the inconsistency of item exposure in the interactive recommendation, which is important but neglected by existing approaches. We formulate the issue of achieving long-term fairness in dynamic recommendation scenario as a Markov Decision Process with entropy constraints.
- We propose SAC4IR, an offline RL approach to maintaining long-term fairness in the interactive recommendation, which leverages an entropy maximization policy to recommend items as fairly as possible. Moreover, to illuminate the unfairness amplifying effect in the feedback loop, we counteract the effect of popularity bias by explicitly modeling the popularity into the reward model.
- We conduct extensive experiments on three real-world recommendation datasets. Results demonstrate that the SAC4IR enjoys superiority in recommendation performance and item fairness, compared to other SOTA methods.

II. Problem Definition

A. *MDP for Recommendation*

In our recommendation scenario, we denote the user set as $\mathcal{U} = \{u_1, u_2, ..., u_N\}$ and item set as $\mathcal{I} = \{i_1, i_2, ..., i_M\}$. The set of all interaction sequences of a user $u \in \mathcal{U}$ can be denoted as $\mathcal{D}^u = \{\mathcal{S}_1^u, \mathcal{S}_2^u, ..., \mathcal{S}_{|\mathcal{D}^u|}^u\}$. The content of the k-th interaction record $\mathcal{S}_k^u = \{(u, i^k, f^k, t^k)\}^{1 \le k \le |\mathcal{S}_k^u|} \in \mathcal{D}^u$ includes a user u, the user's action on recommended items (i.e., clicked item) i^k, and the user's immediate feedback f^k and timestamps t^k.

The recommendation task aims to recommend a list to users to achieve user satisfaction and fairness metrics . This process can be cast as a reinforcement learning problem, whose key components are summarized as follows:

- **State.** The state s_t is defined as a user's historical interactions before time t, and sorted s_t in chronological order.
- **Action.** An action a_t is to recommend an item i to a user u at time t, and we let e_{a_t} denote the embedding vector of action a_t. In this paper, each action a_t means one item i, and multiple actions form a recommended list.
- **Reward.** The reward r_t is predicted by the debiasing-aware reward model.
- **Policy network.** The policy network is the parameterized target policy: $\pi_\varphi = \pi_\varphi(a_t|s_t)$ that represents the probability of sampling an action a_t in the state s_t.

B. *Entropy Fairness Constraint*

We introduce entropy to measure the degree of fairness of item exposure and help us achieve global-level fairness. The concrete form of entropy in our task can be defined as:

$$\mathcal{H}(\mathcal{I}) = \mathbb{E}\big[-\log p(i)\big] = -\sum_{i \in \mathcal{I}} p(i) \log p(i), \tag{1}$$

where $p(i)$ denotes the exposure of the item i (i.e., the probability calculated by the number of occurrences of item in recommendation lists). Thus, we can denote the limit of fairness as:

$$\begin{aligned} \mathcal{H}(\mathcal{I}) &= -\log(1/|\mathcal{I}|) \\ &= \mathcal{H}_0 \ge \alpha \mathcal{H}_0, \end{aligned} \tag{2}$$

where $\mathcal{H}_0$ is the given maximum entropy. The $\alpha \in (0, 1]$ allows the maximum entropy technique to be relaxed to a realistic scenario of fairness as the system can adjust the value of α, and (2) ensures $\mathcal{H}(\pi_\varphi) \ge \alpha \mathcal{H}_0$ at each iteration, which enforces a minimum degree of exploration.

C. *Fairness Constrained RL Objective*

In here, our RL-based method aims to learn a recommendation policy π via maximizing the expected cumulative rewards with entropy constraints. It is defined as follows:

$$\begin{aligned} &\max_{\pi_\varphi} \sum_{t=0}^{T} \mathbb{E}_{(s_t, a_t) \sim \rho_{\pi_\varphi}} \big[\gamma^t \mathcal{R}(a_t, s_t)\big], \\ &\text{s.t.} \quad \mathbb{E}_{(s_t, a_t) \sim \rho_{\pi_\varphi}} \big[-\log\big(\pi_\varphi(\cdot|s_t)\big)\big] \ge \alpha \mathcal{H}_0, \forall t \\ &\qquad\qquad 0 < \alpha \le 1, \end{aligned} \tag{3}$$

where ρ_π denotes the state-action distribution by policy π_φ. Actions entropy $-\log(\pi_\varphi(\cdot|s_t))$ is calculated by (1), and we mark it as $\mathcal{H}(\pi_\varphi(\cdot|s_t))$, where the policy $\pi_\varphi(\cdot|s_t)$ evaluates the probability distribution of the actions in the state s_t.

Instead of learning the recommendation policy online, we focus on offline learning with a logged dataset of trajectories. Each trajectory $\tau = \{s_1, a_1, r_1 ..., s_T\}$ is drawn from log data. The expectation of (3) take over the trajectories, i.e. $\mathcal{R}(\tau) = \mathcal{R}(a_t, s_t) + \mathcal{H}(\pi_\varphi(\cdot|s_t))$. All user interaction trajectories are stored in the buffer before training, i.e., $\mathcal{D} = \{s_t^u, a_t^u, r_t^u, s_{t+1}^u\}_{u=1}^N$. For brevity, we will not consider specific users when we mention buffer again below, that is, we will omit the superscript u.

III. Methodology

A. *Offline RL-based Framework*

The framework of SAC4IR is illustrated in Fig. 1. It contains three stages (shown in three color blocks): pre-learning stage, RL planning stage, and RL evaluation stage. The functions of three stages correspond to the three objectives: 1) learning the debiasing-aware reward model, 2) leveraging the learned reward model to guide recommendation policy π_φ by training on the historical interaction data, and 3) evaluating policy π_φ in

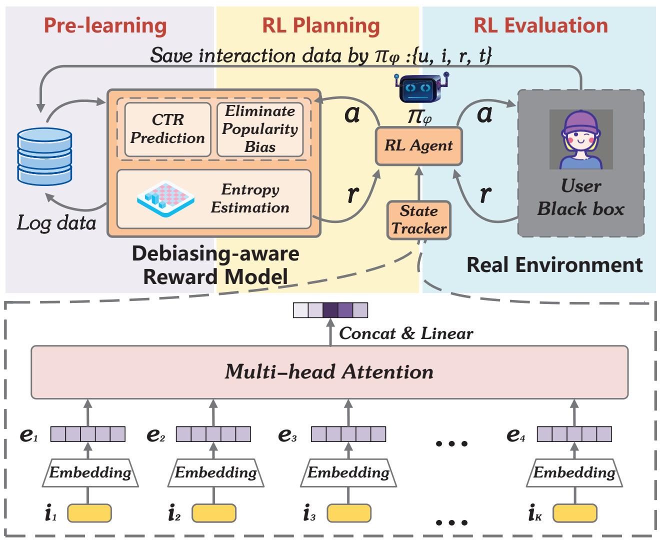

Fig. 1. Learning Framework of SAC4IR

the real environment. Next, we separately introduce the three main components in SAC4IR: state tracker module, debiasing-aware reward model, and RL agent (i.e., recommender).

B. Attention-based State Tracker

The state s_t in IRS should include all interacted actions $\{a_1, a_2, ..., a_t\}$ at time t for RL agent to make decisions. To automatically extract key information from these vectors, we use a state tracker to derive s_t.

In detail, the inputs of the state tracker are user interaction sequences, and the output $\boldsymbol{H}_t$ is state representation. We first utilize an embedding layer to transform item vector into a low-dimensional dense vector e_k by $e_k = \tanh(\boldsymbol{W}_e a_k + \boldsymbol{b}_e) \in \mathbb{R}^{D_e}$. Then we map all embedding vectors into three spaces: query matrix $\boldsymbol{Q}$, key matrix $\boldsymbol{K}$, and value matrix $\boldsymbol{V}$. For a specific sequence query $\boldsymbol{Q} = \{q_1, q_2, ..., q_K\}$, the state representation $\boldsymbol{H}$ can be calculated by:

$$\begin{aligned} h_k &= Attention\big((\boldsymbol{K},\boldsymbol{V}), q_k\big) \\ &= \sum_{i=1}^{K} softmax\big(score(k_i, q_k)\big) v_i, \end{aligned} \tag{4}$$

$$\begin{aligned} \boldsymbol{H} &= softmax\big(score(\boldsymbol{K},\boldsymbol{Q})\big)\boldsymbol{V} \\ &= [h_1, h_2, ..., h_K] \in \mathbb{R}^{D_e \times K}, \end{aligned} \tag{5}$$

Where $score(k, q)$ is calculated by the Dot-Product model.

To parallelize the computation of state tracker, we split the $\boldsymbol{Q}$ into multi parts, called $head_{1,...,H}$. Then, we simultaneously apply the attention model for each head and concatenate all heads to get the final state representation:

$$Attention\big((\boldsymbol{K},\boldsymbol{V}),\boldsymbol{Q}\big) = head_1 \oplus head_2 \oplus ... \oplus head_H. \tag{6}$$

C. Debiasing-aware Reward Model

The debiasing-aware reward is fetched from user feedback and provides the optimization target for the RL planning stage. It aims to achieve item local-level fairness, specifically, by correctly modeling the reward to guide the RL agent to find a policy that can promote user satisfaction (i.e., premising the recommendation accuracy yet eliminating the popularity bias). Therefore, there are two sub-modules we set up in our reward model: an interest estimation module r_{ctr} designed for estimating user's intrinsic interest; and a debiasing module r_{deb} to capture the popularity bias of item:

$$\mathcal{R}(s_t, a_t) = r_{ctr}(s_t, a_t) - \beta \cdot r_{deb}(a_t), \tag{7}$$

where β indicates the importance of debiasing module.

***Reward module of interest estimation* r_{ctr}.** To track the sparse reward problem in our RL-based recommender, we develop a CTR model to capture the user interest in items, which is a concise myopic model to predict user responses on items in IRS [12]. In detail, we employ a Multi-Layer Perceptron(MLP) network to train the CTR model, in which the input is the user state and action, and the output is the predicted CTR value. The model is defined as follows:

$$r_{ctr}(s_t, a_t) = \sigma\big(\boldsymbol{b}^\top [s_t; a_t] + \boldsymbol{c}\big), \tag{8}$$

where $\boldsymbol{b}$ and $\boldsymbol{c}$ refers to the weight matrix of network, and $\sigma(\cdot$ is the *ReLU* activation function.

***Reward module of popularity compensation* r_{deb}.** While interest estimation is important, it is always plagued by popularity bias. We consider that a fair recommendation should be to improve the exposure rate of the long-tail items and explore potential user interests. To achieve this, we model the item popularity explicitly into the reward model:

$$r_{deb}(a_t) = \frac{1}{-\log\big(pop(a_t)\big)}, pop(a_t) = \frac{|N(a_t)|}{|N|}, \tag{9}$$

where $pop(a_t)$ is the popularity of item a_t.

D. Maximum Entropy Recommendation Policy

According to the defined recommendation problem, our task is to learn a recommendation policy π_φ via maximizing the cumulative user satisfaction of the long-term interaction with entropy constraints.

To achieve this, we first design a Stochastic Actor for list recommendation. Given the user's current state s_t, the task of action generator is to appropriately generate an optimal recommendation list via maximizing the expected reward (7). When generating action for a specific user state s_t, we first calculate the Q-values of all candidate items through the policy π_φ and record it as the distribution $Q(s_t, \mathcal{I})$. Then we calculate the attractiveness score of each item to the user, and get a distribution $s(\mathcal{I}|s_t, \mathcal{I})$. Finally, we compute their inner product $\overline{Q}(s_t, \mathcal{I}) = s(\mathcal{I}|s_t, \mathcal{I}) \cdot Q(s_t, \mathcal{I})$, and normalize by the partition function Zs_t. At this time, it conforms to the Boltzmann distribution:

$$\pi_\varphi(\mathcal{I}|s_t) \propto \frac{\exp\big(\overline{Q}(s_t, \mathcal{I})\big)}{Zs_t}, \tag{10}$$

where the policy π_φ has a property of probability $\sum_i p(i) = 1, \forall p(i) \in [0, 1]$, that is, greater the probability of the item, the more likely it is to be sampled. It is useful in recommendation tasks because it gives similar exposure probabilities to multiple items with the same value instead of choosing the largest one by greedy, which is a fair policy for generating novel recommendation results.

E. Off-Policy Training and Evaluation Stage

In this section, we separately introduce the process of training stage and the evaluation stage of SAC4IR.

Training stage. The training stage consists of two procedures. First, We initialize the replay buffer $\mathcal{D}$ and store the experience collected from the user's historical data. Then, we sample trajectories from the replay buffer and update the function approximators using the stochastic gradients, including Actor (i.e., policy network) and Critics (i.e., main network and target network). Furthermore, we set double soft Q-functions to mitigate overestimation [13]. We train the Critic main network by minimizing soft Bellman residual:

$$\begin{aligned} J_Q(\theta) = \mathbb{E}_{(s_t,a_t)\sim\mathcal{D}}\Big[\frac{1}{2}\big(\mathcal{R}(a_t, s_t) \\ + \gamma\mathbb{E}_{s_{t+1}\sim p}[V_{\overline{\theta}}(s_{t+1})] - Q_\theta(s_t, a_t)\big)^2\Big], \end{aligned} \tag{11}$$

where θ represents the parameters of the Critic main network, which is trained from samples in the replay buffer $\mathcal{D}$. The $V_{\overline{\theta}}$ is implicitly parameterized through parameter θ in the soft Q-function, and the parameter $\overline{\theta}$ is an exponential moving average of the θ. (11) can be optimized with stochastic gradients. And we utilize (12) to model KL-divergence minimizing process to learn Actor parameter φ:

$$\begin{aligned} J_\pi(\varphi) = \mathbb{E}_{s_t\sim\mathcal{D},\epsilon_t\sim\mathcal{N}}\big[\log\pi_\varphi\big(f_\varphi(\epsilon_t; s_t)|s_t\big) \\ - Q_\theta\big(s_t, f_\varphi(\epsilon_t; s_t)\big)\big]. \end{aligned} \tag{12}$$

Evaluation stage. Our evaluation method's intuition is that the RA filters the items from available item space for a given sequence of interaction data. If SAC4IR works efficiently, the interaction items in this recommended task will emerge in recommendation list. RA only considers items in available item space rather than the whole action space because we must mask items already used during training to prevent over-fitting during evaluation stage.

IV. EXPERIMENTS

We conduct extensive offline experiments to explore the SAC4IR performs as compared with other SOTA models in static scenario.

A. Experimental Setup

We introduce the experimental settings concerning the environments, evaluation metrics, and SOTA methods.

Datasets. We conduct experiments on three datasets: MovieLens-1m[1], Ciao[2] and Amazon Musical Instruments (i.e., Amazon-MI)[3]. From [14], we convert the original 5-star ratings into two categories, according to the threshold rating 3. We apply a multi-core filter to ensure that users have enough interacted records in the three datasets.

Baselines. We compare SAC4IR with the following six baselines.

- **MostPop** [15]. It recommends the most popular items for all users without considering personalization.
- **BPRMF** [16]. It optimizes the Matrix Factorization (MF) model with Bayesian Personalized Ranking loss.
- **C^{2}UCB** [17]: It is a diverse recommendation method based on contextual bandit, which defines an entropy regularizer.
- **MF-IPS** [3]. It is a SOAT method for item exposure fairness. It adjusts the data distribution to be even by reweighting the interaction examples for model training.
- **DICE** [18]. It is a SOAT method that uses causal embeddings to deal with the unfairness of item exposure.
- **SAC-Rec** [19]. It is a SOAT MDP-based RL recommendation method that utilizes entropy maximization techniques for extensive action exploration to achieve robust recommendations.

Evaluation protocols. We use precision@K [20], HR@K [21] and NDCG@K [22] to measure the recommendation accuracy. We use Gini Index [23] to evaluate the ability of fairness of recommendation model at item individual level.

Following [24], we sorted each user's interactions chronologically and regraded the recent 20% interactions as the test set, the subsequent 20% interactions as the validation set, and left as the training set.

Parameter setting. The fine-tuned hyper-parameters of all mentioned methods are tuned on the validation data to be a fair comparison. In SAC4IR, we set the capacity of replay memory $B = 300000$ and the size of minibatch $N = 8$. We utilize the Adam optimizer to optimize the network. The default learning rate for the network for all the experiments. For MDP, we set state size as 10 in each timestamp. The discount factor $\gamma = 0.99$.

B. Overall Performance in Static Scenario

In this part, we analyze the effectiveness of recommendation accuracy and individual fairness of SAC4IR and its rationality. For the performance of fairness, we compare SAC4IR with all baselines on Gini@K. Table I shows the Top-K recommendation results of SAC4IR and other competitors. Moreover, we set ablation experiments conducted on SAC4IR to explore the influence of different components on the experimental results. Here, we benchmarked against the SAC framework (i.e., SAC-Rec), meaning that accuracy reward is the basis for testing. Then we test the effects of debiasing-aware reward (i.e., SAC-Deb) and different state representation models (i.e., SAC-GRU and SAC-Att) on recommendation accuracy, respectively. We can find that:

- The proposed method achieves the most satisfactory performance. The results verify the effectiveness of SAC4IR in accuracy, which is attributed to the correct representation of user preferences, and positive rewards from the interest estimation module.

[1]https://files.grouplens.org/datasets/movielens
[2]https://www.cse.msu.edu/∼tangjili/datasetcode/
[3]https://jmcauley.ucsd.edu/data/amazon

TABLE I
RECOMMENDATION PERFORMANCE AND ABLATION ANALYSIS OF ALL METHODS ON THE THREE DATASETS. FOR RECOMMENDATION PERFORMANCE, WE COMPARE SAC4IR WITH ALL BASELINES. *%improv.* REPRESENTS THE RELATIVE IMPROVEMENT OF SAC4IR OVER THE CORRESPONDING BASELINE, AND WE HIGHLIGHT THE BEST RESULTS IN BOLD AND SUB-OPTIMAL RESULTS WITH UNDERLINE.

Dataset	Methods	Top-20				Top-50			
		Precision↑	HR↑	NDCG↑	Gini↓	Precision↑	HR↑	NDCG↑	Gini↓
MovieLens-1m	MostPop	0.0496	0.4690	0.0824	0.9898	0.0409	0.7055	0.1148	0.9784
	BPRMF	0.1827	0.6267	0.2214	0.9161	0.1296	0.7672	0.1428	0.8413
	C^{2}UCB	0.1523	0.5146	0.1842	0.8795	0.1460	0.7281	0.127	0.7806
	MF-IPS	0.1531	0.4942	0.1883	0.7217	0.1452	0.7992	0.1119	0.7141
	DICE	0.1601	0.5823	0.2034	0.8031	**0.1483**	0.8215	0.1424	0.6881
	SAC-Rec	0.1563	0.7339	0.3282	0.8668	0.1026	0.8123	0.3833	0.7942
	SAC-Deb	0.0983	0.6667	0.2791	0.6636	0.0740	0.7670	0.3852	0.6125
	SAC-GRU	0.1917	0.8402	0.328	0.8114	0.1493	0.9611	0.529	0.7215
	SAC-Att	0.1950	0.8781	0.3624	0.8819	0.1573	0.9686	0.5292	0.8355
	SAC4IR	**0.1895**	**0.8670**	**0.3292**	**0.6863**	0.1453	**0.9530**	**0.4290**	**0.6608**
	%improv.	3.72	18.13	0.30	12.72	-	16.00	11.92	8.75
Ciao	MostPop	0.0496	0.4920	0.0813	0.9822	0.0494	0.6660	0.1228	0.9488
	BPRMF	0.1599	0.6246	0.1220	0.9752	0.0842	0.8611	0.1601	0.9450
	C^{2}UCB	0.1171	0.5589	0.1421	0.9471	0.1081	0.7179	0.1482	0.8111
	MF-IPS	0.1584	0.5769	0.1095	0.8720	0.0897	0.8261	0.1326	0.7795
	DICE	0.1712	0.6734	0.1099	0.8762	0.0971	**0.8908**	0.1207	0.7483
	SAC-Rec	0.1799	0.6733	0.1214	0.8939	0.1068	0.7448	0.1559	0.8241
	SAC-Deb	0.1490	0.5670	0.0514	0.7939	0.0947	0.8197	0.1318	0.7021
	SAC-GRU	0.1993	0.7166	0.1612	0.9141	0.1163	0.8846	0.1693	0.8347
	SAC-Att	0.2083	0.7191	0.1651	0.9545	0.1162	0.8898	0.1967	0.8652
	SAC4IR	**0.1917**	**0.6830**	**0.1609**	**0.8538**	**0.1135**	0.8652	**0.1638**	**0.7136**
	%improv.	6.56	1.43	13.23	14.22	5.00	-	2.31	13.79
Amazon-MI	MostPop	0.0030	0.0589	0.0212	0.9998	0.0020	0.0984	0.0298	0.9996
	BPRMF	0.0078	0.1435	0.1073	0.9848	0.0034	0.1878	0.0539	0.9785
	C^{2}UCB	0.0041	0.1811	0.1657	0.9910	0.0015	0.1858	0.1097	0.9834
	MF-IPS	0.0059	0.1189	0.1501	0.9720	0.0028	0.1376	0.0953	0.9312
	DICE	0.0032	0.1627	0.1927	0.9472	0.0021	0.1637	0.0305	0.9466
	SAC-Rec	0.0071	0.1726	0.1643	0.9773	0.0070	0.1763	0.0986	0.9638
	SAC-Deb	0.0038	0.1465	0.1492	0.9034	0.0034	0.1630	0.0837	0.9032
	SAC-GRU	0.0133	0.2118	0.1992	0.9478	0.0111	0.2169	0.1507	0.9380
	SAC-Att	0.0172	0.2291	0.2049	0.9493	0.0121	0.2389	0.1531	0.9487
	SAC4IR	**0.0107**	**0.2039**	**0.1993**	**0.9266**	**0.0094**	**0.2105**	**0.1286**	**0.9161**
	%improv.	37.18	12.59	3.43	39.02	34.29	12.09	17.23	21.95

- SAC4IR improved to varying degrees on the three datasets relative to the baseline. Specifically, SAC4IR outperforms all baselines by an average of 8.94%, 7.07%, and 22.22% on MovieLens-1m, Ciao, and Amazon-MI, respectively. The difference in the improvement on different datasets is because there are different properties on these three datasets. For example, the degree of sparsity in the data varies significantly. High sparsity means that most niche items rarely appear in long-tail training data, resulting in failure of MostPoP and MF-based models. That is why achieving fairness makes the proposed SAC4IR significantly outperform the baseline methods on the Amazon-MI dataset. Hence, we believe that improving fairness could give the recommender more advantages, especially in the case of high sparsity.
- Regarding of RL recommendation method, we can see that SAC-Rec and SAC4IR surpass C^{2}UCB comprehensively, because they model the recommendation process as a stochastic model of sequential decisions based on the Markov Decision Process (MDP). Compared with the static model, such as the bandit algorithm, the MDP model is more suitable for interactive recommendation. Furthermore, SAC4IR reasonably applies entropy in training to explore potential items that users prefer, instead of just using it as a regularizer like C^{2}UCB.
- Our method achieves much better Gini. Compared to the baseline, SAC4IR gets an average improvement of 10.74% in MovieLens-1m, 14.01% in Ciao, and 30.49%

in Amazon-MI. To put it simply, our method can achieve much better effectiveness of item individual fairness. These improvements are attributed to the cascading effects of entropy constraints acting on global-level, and debiasing-aware reward models acting on item individual-level.

- For ablation experiments, the Gini metric of SAC-Deb is improved at the expense of accuracy compared to naive SAC (i.e., SAC-Rec). This result means that the debiasing-aware reward model leads to fairer recommendation results. Meanwhile, SAC-ATT shows better recommendation accuracy than SAC-GRU, which demonstrates the attention mechanism can capture the dynamic transfer of user preference more accurately. After combining state representation and debiasing-aware reward model, SAC4IR performs best with trade-off recommendation accuracy and fairness.

V. Conclusion

This work studies how to model and achieve long-term fairness of item exposure in interactive systems from a maximum entropy perspective. By analyzing current fairness-aware recommender, we find that current fairness methods focus on the static recommendation scenario, with ignoring the evolving dynamics of user preference and the amplification effects of the feedback loop in the recommendation system. Thus, it is necessary to study fairness in the dynamic recommendation scenario. To overcome these issues, We propose SAC4IR, a maximum entropy-based offline Reinforcement Learning for eliminating exposure unfairness in the interactive recommendation. It leverages a soft actor-critic framework to generate a list of recommendations with a stochastic actor. An attention-based state representation model is designed to capture the user's dynamic preference. Meanwhile, the debiasing-aware reward model is proposed to eliminate the item popularity bias in a fine-grained level. We conduct experiments on three real-world datasets, and supply analyses on the rationality of SAC4IR.

References

[1] M. Mansoury, H. Abdollahpouri, M. Pechenizkiy, B. Mobasher, and R. Burke, "Feedback loop and bias amplification in recommender systems," in *Proceedings of the 29th ACM international conference on information & knowledge management*, 2020, pp. 2145–2148.

[2] A. Lin, J. Wang, Z. Zhu, and J. Caverlee, "Quantifying and mitigating popularity bias in conversational recommender systems," in *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, 2022, pp. 1238–1247.

[3] T. Schnabel, A. Swaminathan, A. Singh, N. Chandak, and T. Joachims, "Recommendations as treatments: Debiasing learning and evaluation," in *international conference on machine learning*. PMLR, 2016, pp. 1670–1679.

[4] T. Calders, F. Kamiran, and M. Pechenizkiy, "Building classifiers with independency constraints," in *2009 IEEE international conference on data mining workshops*. IEEE, 2009, pp. 13–18.

[5] A. Beutel, J. Chen, T. Doshi, H. Qian, L. Wei, Y. Wu, L. Heldt, Z. Zhao, L. Hong, E. H. Chi *et al.*, "Fairness in recommendation ranking through pairwise comparisons," in *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, 2019, pp. 2212–2220.

[6] S. Yao and B. Huang, "Beyond parity: Fairness objectives for collaborative filtering," *Advances in neural information processing systems*, vol. 30, 2017.

[7] H. Abdollahpouri, R. Burke, and B. Mobasher, "Managing popularity bias in recommender systems with personalized re-ranking," in *The thirty-second international flairs conference*, 2019.

[8] N. Mehrabi, U. Gupta, F. Morstatter, G. V. Steeg, and A. Galstyan, "Attributing fair decisions with attention interventions," *arXiv preprint arXiv:2109.03952*, 2021.

[9] Y. Li, H. Chen, S. Xu, Y. Ge, J. Tan, S. Liu, and Y. Zhang, "Fairness in recommendation: A survey," *arXiv preprint arXiv:2205.13619*, 2022.

[10] Y. Ge, S. Liu, R. Gao, Y. Xian, Y. Li, X. Zhao, C. Pei, F. Sun, J. Ge, W. Ou *et al.*, "Towards long-term fairness in recommendation," in *Proceedings of the 14th ACM international conference on web search and data mining*, 2021, pp. 445–453.

[11] R. Jagerman, I. Markov, and M. de Rijke, "When people change their mind: Off-policy evaluation in non-stationary recommendation environments," in *Proceedings of the twelfth ACM international conference on web search and data mining*, 2019, pp. 447–455.

[12] E. Ie, V. Jain, J. Wang, S. Narvekar, R. Agarwal, R. Wu, H.-T. Cheng, T. Chandra, and C. Boutilier, "Slateq: A tractable decomposition for reinforcement learning with recommendation sets," 2019.

[13] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *International conference on machine learning*. PMLR, 2018, pp. 1587–1596.

[14] H. Wang, F. Zhang, J. Wang, M. Zhao, W. Li, X. Xie, and M. Guo, "Ripplenet: Propagating user preferences on the knowledge graph for recommender systems," in *Proceedings of the 27th ACM international conference on information and knowledge management*, 2018, pp. 417–426.

[15] Y. Ji, A. Sun, J. Zhang, and C. Li, "A re-visit of the popularity baseline in recommender systems," in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2020, pp. 1749–1752.

[16] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "Bpr: Bayesian personalized ranking from implicit feedback," *arXiv preprint arXiv:1205.2618*, 2012.

[17] L. Qin, S. Chen, and X. Zhu, "Contextual combinatorial bandit and its application on diversified online recommendation," in *Proceedings of the 2014 SIAM International Conference on Data Mining*. SIAM, 2014, pp. 461–469.

[18] Y. Zheng, C. Gao, X. Li, X. He, Y. Li, and D. Jin, "Disentangling user interest and conformity for recommendation with causal embedding," in *Proceedings of the Web Conference 2021*, 2021, pp. 2980–2991.

[19] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*. PMLR, 2018, pp. 1861–1870.

[20] Z. Fayyaz, M. Ebrahimian, D. Nawara, A. Ibrahim, and R. Kashef, "Recommendation systems: Algorithms, challenges, metrics, and business opportunities," *applied sciences*, vol. 10, no. 21, p. 7748, 2020.

[21] S. Hors-Fraile, F. J. N. Benjumea, L. C. Hernández, F. O. Ruiz, and L. Fernandez-Luque, "Design of two combined health recommender systems for tailoring messages in a smoking cessation app," *arXiv preprint arXiv:1608.07192*, 2016.

[22] K. Järvelin and J. Kekäläinen, "Ir evaluation methods for retrieving highly relevant documents," in *ACM SIGIR Forum*, vol. 51, no. 2. ACM New York, NY, USA, 2017, pp. 243–250.

[23] W. Sun, S. Khenissi, O. Nasraoui, and P. Shafto, "Debiasing the human-recommender system feedback loop in collaborative filtering," in *Companion Proceedings of The 2019 World Wide Web Conference*, 2019, pp. 645–651.

[24] Z. Meng, R. McCreadie, C. Macdonald, and I. Ounis, "Exploring data splitting strategies for the evaluation of recommendation models," in *Fourteenth ACM conference on recommender systems*, 2020, pp. 681–686.