MITIGATING SAMPLING IMBALANCE IN GENERATIVE LISTWISE RECOMMENDATION VIA DISCREPANCY-GUIDED EXPLORATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Generative listwise recommender systems search over a combinatorial space of item lists, yet sparse interaction feedback often leads to highly imbalanced exploration: training increasingly concentrates on well-covered lists, while potentially high-utility lists remain rarely sampled and insufficiently evaluated. We identify this *sampling imbalance* as a key bottleneck and show that the discrepancy between a list's empirical reward and its model-assigned selection probability provides a direct signal of under-exploration. Building on this insight, we propose a Teacher–Student collaborative framework (TESCO) for discrepancy-guided exploration. An auxiliary Teacher combines the Student's discrepancy signal with list reward to preferentially sample under-explored yet promising lists, while the Student learns from both self-generated and Teacher-generated experiences through alternating sampling with a shared replay buffer. This mechanism enables targeted exploration beyond the Student's current sampling distribution without modifying the original reward definition or observed interaction data. Experiments on three real-world recommendation datasets under both simulated online and offline settings show that TESCO consistently improves list-level utility and ranking quality while maintaining competitive diversity and coverage, with average-reward gains of up to 4.5% over the strongest baseline. Extensive ablation and sensitivity analyses further validate the effectiveness and robustness of the proposed discrepancy-guided exploration mechanism. The code is available at https://anonymous.4open.science/r/Tesco-77DD/.

1 INTRODUCTION

Recommender systems are central to user-facing web services such as e-commerce, news, and short-video platforms. These systems generate ranked lists of items to satisfy personalized preferences and drive user engagement. In high-frequency consumption scenarios, including short videos and news feeds, recommendations are typically delivered as entire lists rather than single items to accommodate continuous browsing behavior. The quality of these lists directly influences user experience, making list-level optimization a critical objective. Early approaches often adopt pointwise or pairwise learning-to-rank paradigms (Cheng et al., 2016; Guo et al., 2017). However, evidence suggests that directly optimizing list-level utility is more effective (Burges, 2010), as it accounts for interactions among items within a list. For instance, multiple highly clickable items may compete with one another, whereas less popular items can enhance overall appeal through complementarity or diversity. These observations have motivated listwise ranking and recommendation methods (Ai et al., 2018; Burges, 2010; Jiang et al., 2019).

The key challenge in list recommendation is efficiently performing a combinatorial search over the exponentially large action space. Existing approaches can generally be divided into two categories: methods that learn list-level evaluators (Feng et al., 2021) and methods that learn list-level generators (Jiang et al., 2019). The former approximates a list-level utility function using an evaluator to guide list construction. This paradigm relies heavily on evaluator accuracy, which limits its effectiveness in practical recommendation tasks. The latter category consists of generative methods that model list generation directly, enabling them to capture both intra-list structural patterns and overall utility during generation. While the stochastic nature of these methods can substantially improve di-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

versity, it often leads to noticeable degradation in recommendation quality (Liu et al., 2023). During recommendation, the probability mass of generated lists heavily concentrates on already-discovered high-reward lists, leaving potentially high-reward but unexplored lists with vanishingly low generation probability. This issue is particularly pronounced when models are trained using standard cross-entropy objectives.

To address these challenges, recent studies reformulate the objective as generating recommendation lists that are both diverse and high-quality (Liu et al., 2023). Diversity expands the coverage of the combinatorial action space and facilitates the discovery of better recommendation lists, while high quality ensures that list-level exploration remains meaningful. Along this line, generative list recommendation methods explicitly model the distribution over entire recommendation lists, enabling the capture of position preferences and intra-list item correlations. Representative approaches include list-CVAE (Jiang et al., 2019) and its variants, such as Pivot-CVAE (Liu et al., 2021), which highlight the trade-off between accuracy and diversity, as well as GFN4Rec (Liu et al., 2023), which leverages flow-based training to align list generation probabilities with list-level utility and provides increased exposure to more diverse lists. Despite these advances, existing generative approaches still **suffer from insufficient exploration of lists that have never interacted with users.** Faithfully modeling the complete list's score distribution is crucial for generating diverse lists that align with user preferences; however, in real-world systems with highly sparse feedback, model training is confined to historically recommended user–list interactions. Parameters associated with lists that are rarely recommended or have never co-occurred with users in any recommendation list receive little to no effective gradient updates. Consequently, the learned generation policy increasingly favors popular and highly exposed lists, while **high-potential lists**, *i.e.*, **lists that have not been previously recommended but may align well with user interests, remain under-explored.** This self-reinforcing training dynamic traps the model in a local optimum defined by historical data, preventing it from perceiving or learning from **unobserved yet informative high-loss regions**, and ultimately exacerbating exposure bias.

To address this issue, we argue that the selection of training samples must be adaptive, prioritizing under-explored lists that may match the user's preferences. Conventional training procedures, which emphasize frequent and highly exposed lists, fail to capture informative regions of the preference distribution that lie outside historical records. To this end, we propose a Teacher–Student collaborative learning framework (termed TESCO) that actively guides exploration toward under-explored, high-potential lists. The Teacher uses the Student's GFlowNet discrepancy to identify regions of insufficient exploration (positive discrepancy) versus over-exposure (negative discrepancy), and adaptively samples from the former, *i.e.*, lists where the Student incurs high loss but may align with user preferences. As shown in Figure 1, this enables targeted exploration beyond historical data, avoiding popularity bias. Unlike fixed-buffer replay methods, our Teacher generalizes across the combinatorial space to discover novel high-value lists, which the Student then learns from to improve both coverage and preference alignment.

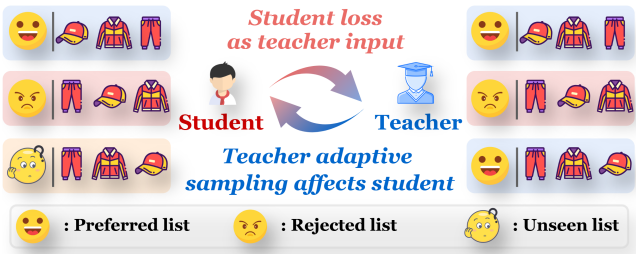

Figure 1: Illustration of the adaptive sampling framework, the Teacher explores high-loss, under-exposed lists; the Student learns from them to better align with user preferences.

The core contributions are summarized as follows:

- We characterize a *sampling imbalance* problem in generative listwise recommendation: training increasingly concentrates on well-covered lists, leaving potentially high-utility lists under-explored. We show that the flow discrepancy between empirical reward and model-assigned probability mass provides a direct signal for identifying such under-explored regions.
- We introduce TESCO, a Teacher–Student framework that turns this discrepancy into adaptive exploration. The Teacher combines the Student's discrepancy with list reward to prioritize trajectories that are both under-explored and potentially useful, while alternating Teacher–Student sampling incorporates these informative trajectories into Student learning, enabling targeted exploration beyond the Student's current sampling distribution.

- Across three real-world datasets and both simulated online and offline evaluations, TESCO consistently improves recommendation utility and ranking quality while maintaining competitive diversity and coverage, with average-reward gains of up to 4.5% over the strongest flow-based baseline. Its advantages persist in the 500K-item KuaiRand-27K setting, while ablation and sensitivity analyses further validate the effectiveness and robustness of discrepancy-guided exploration.

2 METHODOLOGY

TESCO is a Teacher–Student framework for listwise recommendation. The Student learns to generate high-reward lists via Generative flow networks (GFlowNets) (Bengio et al., 2021), while the Teacher adaptively guides exploration toward high-potential regions identified from the Student's training loss shown in Figure 2. Both models share a user encoder, but only the Student updates its parameters during training. They interact through alternating sampling to enrich the training data with informative trajectories.

2.1 PROBLEM FORMULATION

We consider the list-wise recommendation problem where the system recommends a list of items to users. Let $\mathcal{U}$ denote the set of users and $\mathcal{I}$ denote the set of items. For a user $u \in \mathcal{U}$, a recommendation request includes the user's profile features $\mathcal{X}_u$ and interaction history $\mathcal{H}_u$, which are encoded into a user state $\mathbf{s}_u \in \mathbb{R}^{d_s}$.

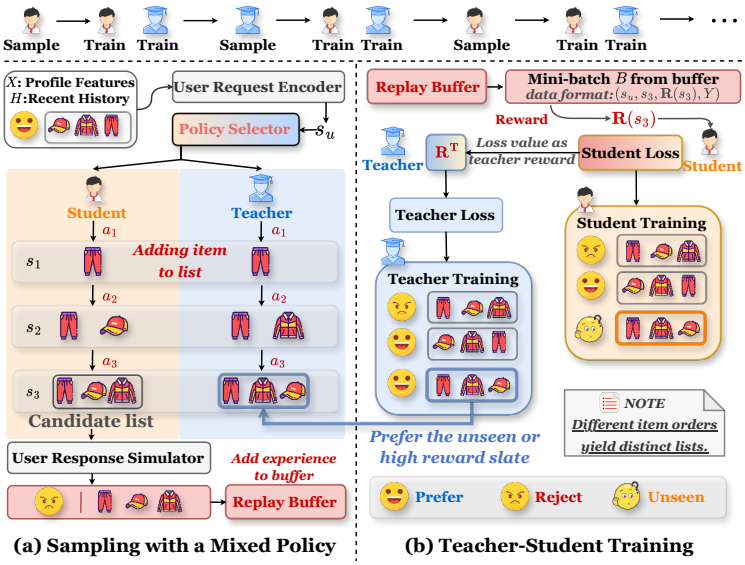

Figure 2: Workflow of the alternating Teacher-Student sampling for generating recommendation lists. (a) The behavior policy mixes Student and Teacher policies to generate lists, and stores experiences in the form of $(s_u, s_K, \mathcal{R}(s_K), Y)$ into a buffer. (b) The Teacher or Student training process. Specifically, the Teacher is updated based on the Student's loss, targeting uncovered lists.

At each recommendation step, the system generates a *list* $s_K = [i_1, i_2, \ldots, i_K]$ of size K, where each $i_k \in \mathcal{I}$ is an item. When list s_K is presented to user u, the user provides multi-behavior feedback $\mathbf{Y} \in \{0,1\}^{K \times M}$, where $y_{k,m} \in \{0,1\}$ indicates whether the user exhibited feedback type m (e.g., click, like) for item i_k. The *reward* $\mathcal{R}(s_K)$ for list s_K is computed as the average of item-wise rewards: $\mathcal{R}(s_K) = \frac{1}{K}\sum_{k=1}^{K}\sum_{m=1}^{M} w_m \cdot y_{k,m}$, where $\boldsymbol{w} = [w_1, w_2, \ldots, w_M]^T$ is a weight vector for different feedback types. Therefore, increasing the reward of a single item in the list does not necessarily improve the overall list reward. Instead, we should aim to identify more items with higher rewards within the same list. The objective is to learn a recommendation policy $\pi(s_K|u;\theta)$ that generates lists maximizing the listwise reward $\mathcal{R}(s_K)$.

2.2 PRELIMINARY ON GFLOWNETS

Generative flow networks (GFlowNets) (Bengio et al., 2021) are defined over a directed acyclic graph $\mathcal{G} = (\mathcal{S}, \mathcal{A})$, where $\mathcal{S}$ denotes the state space (containing a single initial state s_0 and terminal states $\mathcal{X} \subset \mathcal{S}$) and $\mathcal{A}$ represents the action space, *i.e.*, the set of edges. A *trajectory flow* is a nonnegative mapping $F : \mathcal{T} \to \mathbb{R}_{\geq 0}$ defined on the collection $\mathcal{T}$ of all complete trajectories. This induces *state flows* given by $F(s) = \sum_{\tau \ni s} F(\tau)$ and *edge flows* expressed as $F(s \to s') = \sum_{\tau=(\cdots \to s \to s' \to \ldots)} F(\tau)$. For states that are neither initial nor terminal, the flow

satisfies the *flow matching* condition:

$$F(s) = \sum_{(s'' \to s) \in \mathcal{A}} F(s'' \to s) = \sum_{(s \to s') \in \mathcal{A}} F(s \to s'). \tag{1}$$

The trajectory flow F yields a probability distribution over trajectories via $P(\tau) = F(\tau)/Z$, with $Z = F(s_0) = \sum_{\tau \in \mathcal{T}} F(\tau)$. The flow F is termed *Markovian* when P decomposes as:

$$P(\tau = (s_0 \to \cdots \to s_n)) = \prod_{t=1}^{n} P_F(s_t|s_{t-1}), \tag{2}$$

where $P_F(\cdot|s)$ designates the *forward policy*. An equivalent backward decomposition exists through a *backward policy* $P_B(\cdot|s)$:

$$P(\tau = (s_0 \to \cdots \to s_n)|s_n = x) = \prod_{t=1}^{n} P_B(s_{t-1}|s_t). \tag{3}$$

Forward and backward policies are connected to the flows by:

$$P_F(s'|s) = \frac{F(s \to s')}{F(s)}, \quad P_B(s|s') = \frac{F(s \to s')}{F(s')}, \tag{4}$$

which leads directly to the *detailed balance* condition:

$$F(s)P_F(s'|s) = F(s')P_B(s|s'). \tag{5}$$

2.3 Student Model for List Generation Based on User–List Preferences

We formulate list generation as a sequential decision process using GFlowNets. The generation process is modeled as a trajectory $\tau = (s_0, a_1, s_1, a_2, \ldots, a_K, s_K)$, where:

- **State** s_t: a partial list with t items, encoded as $[\mathbf{s}_u; \mathbf{E}_{1:t}]$ where $\mathbf{E}_{1:t}$ contains embeddings of the first t selected items. The initial state s_0 represents an empty list.
- **Action** a_t: selecting an item $i \in \mathcal{I}$ to append to the current partial list, transitioning from state s_{t-1} to s_t.
- **Terminal state** s_K: a complete list with K items.

The GFlowNets models a forward policy $P_F(s_t|s_{t-1})$ that generates trajectories. Since the next state s_t is deterministically constructed from s_{t-1} and the action a_t in slate recommendation, in this paper, we use the $P_F(a_t|s_{t-1})$ to denote the $P_F(s_t|s_{t-1})$. We consider two training objectives: Trajectory Balance (TB) (Malkin et al., 2022) and Detailed Balance (DB) (Bengio et al., 2021).

Trajectory Balance (TB). The TB objective ensures consistency across the entire trajectory:

$$F(s_0) \prod_{t=1}^{K} P_F(a_t|s_{t-1}) = \mathcal{R}(s_K) \prod_{t=1}^{K} P_B(s_{t-1}|s_t), \tag{6}$$

where $F(s)$ is the flow function representing unnormalized probability mass. In practice, we use a uniform backward policy, so the backward terms are omitted. The Student discrepancy $\delta(\tau;\theta)$ and TB loss $\mathcal{L}_{\text{TB}}(\tau;\theta)$ for a trajectory τ is:

$$\delta(\tau;\theta) = \log\left(\mathcal{R}(s_K) + \epsilon_R\right) - \left[\log F_{0_\theta} + \sum_{t=1}^{K} \log P_F(a_t|s_{t-1};\theta) + \log Z_\theta\right], \tag{7}$$

$$\mathcal{L}_{\text{TB}}(\tau;\theta) = \left(\delta(\tau;\theta)\right)^2, \tag{8}$$

where F_0 is a hyperparameter that represents the fixed global normalizing bias for the forward passes compared to observed rewards, $\log Z_\theta$ is a initial flow estimator, and ϵ_R is a hyperparameter that represent the global reward bias that it is introduced to control the smoothness of the loss landscape and avoids division by zero rewards. These settings follow the GFN4Rec (Liu et al., 2023). ***Remark***: The Teacher and Student models use the same user encoder architecture, but the Teacher model does not update the user encoder to prevent it from interfering with the Student's user encoder learning. However, they employ independent GFlowNets parameterizations: the Student's parameters are denoted by θ, and the Teacher's by ϕ. Ensure that each learns a distinct policy and flow estimation.

Detailed Balance (DB). The DB objective ensures consistency at each transition step. For non-terminal transitions, the DB loss is:

$$\mathcal{L}_{\text{DB}}(\tau;\theta) = \sum_{t=1}^{K-1} \left(\log F(s_{t-1};\theta) + \log P_F(a_t|s_{t-1};\theta) - \log F(s_t;\theta)\right)^2. \tag{9}$$

For the terminal state, we have:

$$\mathcal{L}_{\text{terminal}}(\tau;\theta) = \left(\log F(s_K) - \log(\mathcal{R}(s_K) + \epsilon_R)\right)^2. \tag{10}$$

The total DB loss is $\mathcal{L}_{\text{DB}}(\tau;\theta) + \mathcal{L}_{\text{terminal}}(\tau;\theta)$.

2.4 Teacher Model for Adaptive Exposure to Under-Explored Lists

In this section, we revisit the design philosophy of the Teacher model, *i.e.*, an auxiliary GFlowNets that adaptively improves the sampling efficiency of the Student in recommendation. Both Student and Teacher operate on the same user state and candidate list, yet are guided by distinct objectives: the Teacher's role is to target trajectories where the Student's loss is high, thereby exposing the Student to underexplored and hard-to-learn regions.

Motivated by Kim et al. (2025), the Teacher's reward function is closely tied to the Student's training signals. This $\delta(\tau;\theta)$ in Eq. 7 measures how much the Student underfits or overfits a trajectory. When $\delta(\tau;\theta) > 0$, the trajectory is underexplored by the Student (low predicted probability but high empirical reward); when $\delta(\tau;\theta) < 0$, it is already well-covered.

Favoring under-explored lists. Because the Teacher model should encourage the Student model to visit under-explored lists, it should favor trajectories where the $\delta(\tau;\theta) > 0$. Therefore, we increase the weight of the Teacher's reward for lists where the $\delta(\tau;\theta) > 0$, while adding a smoothing constant ϵ:

$$\mathcal{R}^{\text{weighted}}_{\text{Teacher}}(\tau) = \begin{cases} \epsilon + (1+C)(\delta(\tau;\theta)), & \text{if } \delta(\tau;\theta) > 0 \\ \epsilon + (\delta(\tau;\theta))^2, & \text{if } \delta(\tau;\theta) \leq 0 \end{cases} \tag{11}$$

The weighted term $(1+C)$ gives additional weight when the Student's discrepancy is positive, which indicates the Student discovers an under-explored list. Additionally, in our empirical experiments, we observed that the absolute values of the discrepancy are all small fractions less than 1. *To further amplify the weighting effect*, we leave positive discrepancies unchanged, while squaring negative discrepancies to make their contributions even smaller. This pragmatic design choice, while not theoretically optimal, yields stable training across datasets in practice. This design encourages the model to preferentially explore previously under-explored trajectories.

Reward mixing. However, such a loss-based Teacher reward alone may cause the Teacher to over-sample high-loss but low-utility regions. Therefore, the final Teacher reward $R^T(\tau)$ is the combination of the Student's discrepancy and reward:

$$\mathcal{R}^T(\tau) = \mathcal{R}^{\text{weighted}}_{\text{Teacher}}(\tau)\left(\mathcal{R}(s_K)\right)^{\alpha}. \tag{12}$$

Remark: We apply *stop-gradient* to $\delta(\tau;\theta)$ in the Teacher's reward and loss, so gradients from the Teacher do not flow back through the Student's parameters θ. This ensures the Teacher adapts to the Student's current policy without interfering with its training, maintaining stable and decoupled optimization. Here, $\alpha > 0$ trades off the contribution of the reward and the discrepancy. This encourages Teacher sampling to focus on both hard-to-predict and useful lists. The ablation study on our choice of C and α, see Section 3.5.1. The Teacher TB for a trajectory τ is:

$$\delta(\tau;\phi) = \log\left(\mathcal{R}^T(\tau) + \epsilon_R\right) - \left[\log F_{0_\phi} + \sum_{t=1}^{K} \log P_F(a_t|s_{t-1};\phi) + \log Z_\phi\right]. \tag{13}$$

$$\mathcal{L}_{\textbf{TB}}(\tau;\phi) = \left(\delta(\tau;\phi)\right)^2. \tag{14}$$

For the non-terminal transitions and terminal states, the DB loss is:

$$\mathcal{L}_{\text{DB}}(\tau;\phi) = \sum_{t=1}^{K-1} \left(\log F(s_{t-1};\phi) + \log P_F(a_t|s_{t-1};\phi) - \log F(s_t;\phi)\right)^2. \tag{15}$$

$$\mathcal{L}_{\text{terminal}}(\tau;\phi) = \left(\log F(s_K) - \log(\mathcal{R}^T(\tau) + \epsilon_R)\right)^2. \tag{16}$$

The total DB loss is $\mathcal{L}_{\text{DB}}(\tau;\phi) + \mathcal{L}_{\text{terminal}}(\tau;\phi)$. We minimize the above loss function to optimize the Teacher parameters ϕ.

Table 1: Overall performance of baselines and TESCO on **ML1M**. The left part reports online performance, and the right part reports offline performance. Best results are in **bold**, and sub-optimal results are underlined. Metrics with ↑ indicate higher is better, while ↓ indicates lower is better.

	Online				Offline					
Method	Avg. $\mathcal{R}$ (↑)	Max. $\mathcal{R}$ (↑)	Coverage (↑)	ILD (↑)	Avg. $\mathcal{R}$ (↑)	Max. $\mathcal{R}$ (↑)	R-NDCG (↑)	R-MRR (↑)	Coverage (↑)	ILD (↑)
CF	2.081 ± 0.018	2.929 ± 0.011	10.340 ± 0.374	0.531 ± 0.003	1.675 ± 0.015	2.694 ± 0.018	0.563 ± 0.029	0.0713 ± 0.001	12.217 ± 1.359	$\mathbf{0.729 \pm 0.001}$
ListCVAE	1.001 ± 0.175	2.267 ± 0.224	$\mathbf{310.60 \pm 7.162}$	$\mathbf{0.799 \pm 0.007}$	0.895 ± 0.016	2.565 ± 0.025	0.277 ± 0.037	0.037 ± 0.002	53.510 ± 0.890	0.704 ± 0.001
PRM	2.167 ± 0.124	2.967 ± 0.075	22.000 ± 1.414	0.534 ± 0.005	1.979 ± 0.013	2.933 ± 0.016	0.661 ± 0.033	0.084 ± 0.001	$\mathbf{107.31 \pm 3.780}$	0.606 ± 0.001
GFN4Rec-TB (Explore)	2.094 ± 0.023	2.967 ± 0.005	70.480 ± 1.217	0.600 ± 0.002	2.021 ± 0.014	2.918 ± 0.011	0.681 ± 0.018	0.088 ± 0.001	31.620 ± 1.184	0.603 ± 0.002
GFN4Rec-TB	2.219 ± 0.068	2.933 ± 0.091	12.600 ± 0.894	0.573 ± 0.005	2.106 ± 0.013	2.962 ± 0.007	0.708 ± 0.019	0.090 ± 0.001	16.480 ± 0.367	0.549 ± 0.001
GFN4Rec-DB (Explore)	1.780 ± 0.024	2.862 ± 0.030	34.720 ± 0.543	0.658 ± 0.002	1.964 ± 0.013	2.884 ± 0.015	0.664 ± 0.021	0.085 ± 0.001	39.850 ± 1.327	0.628 ± 0.003
GFN4Rec-DB	2.050 ± 0.059	3.000 ± 0.000	9.6000 ± 0.548	0.566 ± 0.006	2.054 ± 0.015	2.922 ± 0.013	0.692 ± 0.034	0.087 ± 0.002	21.770 ± 0.959	0.584 ± 0.008
TESCO-TB (Explore)	2.098 ± 0.033	2.945 ± 0.019	87.290 ± 2.003	0.602 ± 0.002	2.071 ± 0.017	2.944 ± 0.013	0.689 ± 0.020	0.089 ± 0.001	34.240 ± 1.562	0.608 ± 0.003
TESCO-TB	$\mathbf{2.297 \pm 0.037}$	2.967 ± 0.075	14.800 ± 1.643	0.587 ± 0.005	$\mathbf{2.143 \pm 0.018}$	$\mathbf{2.983 \pm 0.012}$	$\mathbf{0.713 \pm 0.031}$	$\mathbf{0.093 \pm 0.001}$	16.360 ± 2.128	0.559 ± 0.005
TESCO-DB (Explore)	2.177 ± 0.037	2.977 ± 0.070	56.770 ± 1.886	0.593 ± 0.004	2.032 ± 0.014	2.901 ± 0.019	0.676 ± 0.017	0.088 ± 0.001	32.810 ± 1.214	0.596 ± 0.002
TESCO-DB	2.257 ± 0.070	$\mathbf{3.000 \pm 0.000}$	8.2002 ± 1.095	0.542 ± 0.016	2.097 ± 0.015	2.913 ± 0.048	0.701 ± 0.016	0.091 ± 0.001	16.112 ± 1.032	0.551 ± 0.001

Table 2: Overall performance of baselines and TESCO on **KuaiRand-1K**. The left part reports online performance, and the right part reports offline performance. Best results are in **bold**, and sub-optimal results are underlined. Metrics with ↑ indicate higher is better, while ↓ indicates lower is better.

	Online				Offline					
Method	Avg. $\mathcal{R}$ (↑)	Max. $\mathcal{R}$ (↑)	Coverage (↑)	ILD (↑)	Avg. $\mathcal{R}$ (↑)	Max. $\mathcal{R}$ (↑)	R-NDCG (↑)	R-MRR (↑)	Coverage (↑)	ILD (↑)
CF	2.254 ± 0.086	4.002 ± 0.086	112.45 ± 12.25	0.539 ± 0.004	1.941 ± 0.016	3.860 ± 0.057	0.390 ± 0.025	0.0824 ± 0.001	17.275 ± 1.657	0.611 ± 0.002
ListCVAE	1.903 ± 0.045	3.600 ± 0.303	439.80 ± 23.446	$\mathbf{0.689 \pm 0.004}$	2.025 ± 0.011	3.942 ± 0.073	0.392 ± 0.028	0.0861 ± 0.002	2.0250 ± 0.011	$\mathbf{0.612 \pm 0.001}$
PRM	2.238 ± 0.055	4.017 ± 0.217	66.200 ± 6.648	0.526 ± 0.003	2.093 ± 0.022	3.952 ± 0.065	0.425 ± 0.015	0.0900 ± 0.001	$\mathbf{212.94 \pm 19.31}$	0.556 ± 0.003
GFN4Rec-TB (Explore)	2.232 ± 0.019	4.073 ± 0.078	347.95 ± 16.64	0.619 ± 0.005	2.101 ± 0.012	3.941 ± 0.051	0.417 ± 0.020	0.0898 ± 0.001	63.420 ± 2.714	0.574 ± 0.002
GFN4Rec-TB	2.327 ± 0.092	4.054 ± 0.190	18.000 ± 1.225	0.527 ± 0.002	2.140 ± 0.010	3.897 ± 0.047	0.422 ± 0.031	$\mathbf{0.0911 \pm 0.002}$	17.120 ± 0.796	0.533 ± 0.002
GFN4Rec-DB (Explore)	2.068 ± 0.016	3.965 ± 0.029	$\mathbf{743.23 \pm 5.089}$	0.601 ± 0.007	2.067 ± 0.011	3.972 ± 0.044	0.401 ± 0.018	0.0890 ± 0.001	94.830 ± 3.106	0.603 ± 0.002
GFN4Rec-DB	2.195 ± 0.077	4.033 ± 0.247	248.40 ± 11.93	0.591 ± 0.008	2.113 ± 0.010	3.993 ± 0.040	0.410 ± 0.019	0.0901 ± 0.001	30.200 ± 0.960	0.587 ± 0.001
TESCO-TB (Explore)	2.243 ± 0.030	4.037 ± 0.090	399.31 ± 9.640	0.627 ± 0.002	2.184 ± 0.013	3.988 ± 0.048	0.429 ± 0.016	0.0902 ± 0.001	71.560 ± 2.845	0.582 ± 0.002
TESCO-TB	$\mathbf{2.430 \pm 0.116}$	$\mathbf{4.100 \pm 0.091}$	18.600 ± 5.441	0.542 ± 0.008	$\mathbf{2.237 \pm 0.015}$	$\mathbf{4.018 \pm 0.063}$	$\mathbf{0.4398 \pm 0.014}$	0.0908 ± 0.001	18.400 ± 1.240	0.545 ± 0.001
TESCO-DB (Explore)	2.251 ± 0.031	4.081 ± 0.089	386.65 ± 10.15	0.554 ± 0.001	2.123 ± 0.010	3.986 ± 0.052	0.418 ± 0.017	0.0894 ± 0.001	68.340 ± 2.604	0.571 ± 0.001
TESCO-DB	2.341 ± 0.076	4.000 ± 0.118	34.800 ± 8.289	0.549 ± 0.005	2.182 ± 0.008	3.967 ± 0.079	0.4281 ± 0.0217	0.0903 ± 0.001	16.241 ± 0.978	0.559 ± 0.001

2.5 ALTERNATING TEACHER–STUDENT SAMPLING FOR LIST GENERATION

As shown in Figure 2, the framework of TESCO adopts an alternating sampling scheme between the Teacher and the Student, where experiences generated by both models are incorporated into the historical buffer. Both the Student and the Teacher are trained on data drawn from this shared buffer, and, as illustrated in Algorithm 1, we simultaneously minimize the loss functions of both the Teacher and the Student. Since the Teacher injects lists sampled from high-loss regions into the Student's history to actively guide exploration toward areas with both high loss and high list scores, the sampling ratio between the Teacher and the Student becomes a critical design choice. Excessive Teacher participation may introduce a large number of high loss trajectories, potentially destabilizing the Student's training, whereas insufficient Teacher involvement weakens the effect of guided exploration. We therefore determine suitable configurations through parameter sensitivity experiments under the given experimental setting in Section 2.5.

3 EXPERIMENTS

In this section, we aim to address four research questions (**RQ**):

- **RQ1:** Do the recommendation lists generated by our model better align with user preferences?
- **RQ2:** How diverse are the recommendation lists produced by our model?
- **RQ3:** How do different components of our model affect recommendation performance?
- **RQ4:** How robust is the model, and how sensitive is it to different parameter configurations?

3.1 EXPERIMENTAL SETUPS

Datasets. In this paper, we use three real-world datasets: **ML1M** (Harper & Konstan, 2015), **KR1K** (Gao et al., 2022), and **KR27K** (Gao et al., 2022). ML1M is the one-million version of the MovieLens dataset, which consists of users' movie rating histories with original ratings in $\{1, \ldots, 5\}$. We convert the ratings into implicit feedback signals, including clicks (rating ≥ 3), likes (rating ≥ 4), and stars (rating ≥ 5).

Baselines and Models. We compare **TESCO** with both ranking and re-ranking models, including **CF** (Kang & McAuley, 2018), **ListCVAE** (Jiang et al., 2019), **PRM** (Pei et al., 2019), and **GFN4Rec** (Liu et al., 2023).

Evaluation Metrics. We employ both online and offline metrics to comprehensively evaluate the recommendation performance. **Online Metrics:** In the online user environment, we evaluate each recommendation list O through the following metrics: average reward, max reward, coverage, and ILD. **Offline Metrics:** For offline evaluation, we evaluate each recommendation list O through the following metrics: average reward, max reward, R-NDCG, R-MRR, coverage, and ILD.

Table 3: Overall performance of baselines and TESCO on **KuaiRand-27K**.

	Online				Offline					
Method	Avg. $\mathcal{R}$ (↑)	Max. $\mathcal{R}$ (↑)	Coverage (↑)	ILD (↑)	Avg. $\mathcal{R}$ (↑)	Max. $\mathcal{R}$ (↑)	R-NDCG (↑)	R-MRR (↑)	Coverage (↑)	ILD (↑)
CF	2.248 ± 0.016	3.998 ± 0.015	$1{,}284.6 \pm 95.3$	0.541 ± 0.002	1.937 ± 0.003	3.856 ± 0.009	0.388 ± 0.004	0.0822 ± 0.0002	198.5 ± 14.2	0.613 ± 0.0004
ListCVAE	1.895 ± 0.008	3.592 ± 0.042	$\mathbf{5{,}127.3 \pm 124.5}$	$\mathbf{0.691 \pm 0.001}$	2.021 ± 0.002	3.938 ± 0.011	0.390 ± 0.005	0.0859 ± 0.0003	23.4 ± 1.2	$\mathbf{0.614 \pm 0.0002}$
PRM	2.232 ± 0.010	4.009 ± 0.038	752.4 ± 58.2	0.528 ± 0.001	2.089 ± 0.004	3.948 ± 0.010	0.423 ± 0.003	0.0898 ± 0.0002	$\mathbf{2{,}458.3 \pm 165.4}$	0.558 ± 0.0005
GFN4Rec-TB (Explore)	2.228 ± 0.004	4.068 ± 0.012	$3{,}984.2 \pm 98.7$	0.621 ± 0.001	2.096 ± 0.002	3.942 ± 0.008	0.414 ± 0.004	0.0897 ± 0.0002	621.4 ± 17.3	0.579 ± 0.0003
GFN4Rec-TB	2.324 ± 0.013	4.051 ± 0.031	203.8 ± 15.6	0.529 ± 0.001	2.137 ± 0.002	3.893 ± 0.007	0.420 ± 0.005	$\mathbf{0.0910 \pm 0.0003}$	197.2 ± 6.8	0.535 ± 0.0003
GFN4Rec-DB (Explore)	2.065 ± 0.003	3.961 ± 0.005	$\underline{8{,}642.1 \pm 89.3}$	0.603 ± 0.001	2.071 ± 0.002	3.971 ± 0.007	0.401 ± 0.003	0.0892 ± 0.0002	918.6 ± 24.5	0.604 ± 0.0002
GFN4Rec-DB	2.193 ± 0.011	4.030 ± 0.039	$2{,}852.6 \pm 148.3$	0.593 ± 0.001	2.110 ± 0.002	$\underline{3.989 \pm 0.006}$	0.408 ± 0.003	0.0900 ± 0.0002	$\underline{348.6 \pm 8.2}$	$\underline{0.589 \pm 0.0002}$
TESCO-TB (Explore)	2.241 ± 0.005	4.033 ± 0.014	$4{,}582.9 \pm 112.4$	$\underline{0.629 \pm 0.001}$	2.173 ± 0.002	3.986 ± 0.009	0.427 ± 0.003	0.0899 ± 0.0002	688.2 ± 19.6	0.586 ± 0.0002
TESCO-TB	$\mathbf{2.428 \pm 0.021}$	$\mathbf{4.097 \pm 0.015}$	214.2 ± 62.3	0.544 ± 0.001	$\mathbf{2.234 \pm 0.003}$	$\mathbf{4.014 \pm 0.010}$	$\mathbf{0.438 \pm 0.002}$	$\underline{0.0907 \pm 0.0002}$	212.3 ± 10.6	0.547 ± 0.0002
TESCO-DB (Explore)	2.248 ± 0.006	$\underline{4.078 \pm 0.014}$	$4{,}453.8 \pm 118.7$	0.556 ± 0.001	2.145 ± 0.001	3.978 ± 0.010	0.419 ± 0.004	0.0895 ± 0.0002	652.7 ± 18.4	0.572 ± 0.0002
TESCO-DB	$\underline{2.338 \pm 0.014}$	3.997 ± 0.019	401.3 ± 95.6	0.551 ± 0.001	$\underline{2.179 \pm 0.001}$	3.963 ± 0.012	$\underline{0.426 \pm 0.004}$	0.0902 ± 0.0002	187.5 ± 8.4	0.561 ± 0.0002

3.2 BETTER ALIGN WITH USER PREFERENCES (RQ1)

We evaluate TESCO in both online and offline environments. As shown in Tables 1–3, our model consistently achieves superior average and max reward online, clearly outperforming all baselines. Among compared variants, TESCO (TB) performs the best. On ML1M, TESCO (TB) improves average reward by approximately 3.515% over GFN4Rec-TB, with more pronounced gains on KuaiRand-1K and KuaiRand-27K, reaching about 4.427% and 4.475%, respectively. Notably, consistent gains on KuaiRand-27K (27K users, 500K items) demonstrate effective scaling to large combinatorial action spaces. These results indicate more effective policies than baselines, with greater potential at scale. The advantage on KuaiRand-1K and KuaiRand-27K can be attributed to their wider and more dispersed reward ranges, where the Teacher more effectively guides the Student toward high-loss, high-reward regions. On KuaiRand-27K, the expanded item space exacerbates insufficient exploration in conventional methods, making discrepancy-guided exploration especially valuable for uncovering under-explored, high-potential lists. Offline, our model also outperforms all baselines, improving average reward by approximately 1.757% on ML1M, 4.533% on KuaiRand-1K, and 4.539% on KuaiRand-27K over GFN4Rec-TB. These consistent improvements, especially on large-scale KuaiRand-27K, highlight the critical role of the Teacher in learning effective policies across diverse scales.

3.3 GAIN DIVERSE LISTS WHILE MAINTAINING RECOMMENDATION QUALITY (RQ2)

3.3.1 ABILITY TO GENERATE MORE DIVERSE RECOMMENDATION LISTS

As shown in Tables 1–3, despite significant improvements in average and max reward, the coverage of our model does not exhibit a sharp decline. Across datasets and settings, TESCO achieves coverage close to GFN4Rec, and even exceeds it on certain datasets. Notably, on KuaiRand-27K, TESCO maintains competitive coverage ($4{,}582.9$ for TESCO-TB (Explore) online and 688.2 offline) while delivering substantial reward improvements, demonstrating scalable exploration without sacrificing diversity. Furthermore, ILD results indicate effectively diverse discovered lists, with TESCO-TB (Explore) achieving an ILD of 0.629 on KuaiRand-27K, among the highest across all methods. This suggests GFN4Rec may not sufficiently explore high-reward regions, whereas the Teacher role effectively encourages the Student to sample trajectories with both high loss and high reward. As a result, the model better balances exploitation and coverage, yielding more diverse results. Table 3 also shows superior R-NDCG (0.438 for TESCO-TB) and R-MRR on KuaiRand-27K offline. These results indicate better reward distribution learning and more diverse lists, even at large scale.

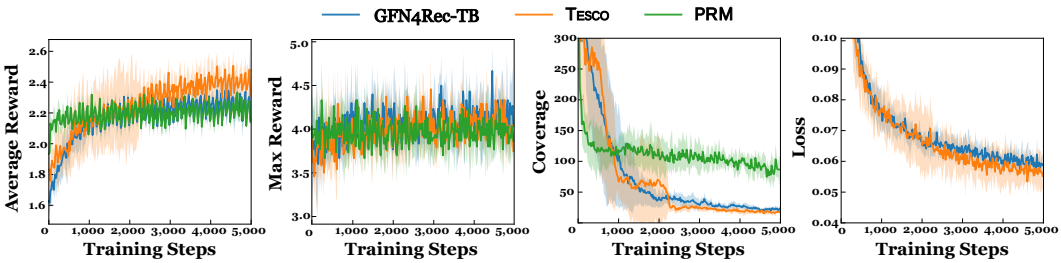

Figure 3: Comparison of performance between TESCO and the two strongest baselines on KuaiRand-1K in online environment, as well as a comparison of loss between TESCO and GFN4Rec.

3.3.2 ABILITY TO IMPROVE THE QUALITY OF LEARNED LIST SCORES DISTRIBUTION

As shown in Figure 3, we compare TESCO with the two strongest baselines (GFN4Rec and PRM) on average reward, max reward, coverage, and TB loss. Since PRM uses a different loss, we only report TB loss for the two flow-based models. TESCO converges to 0.0533, whereas GFN4Rec converges to 0.0578, indicating more accurate learning of the list-wise reward distribution. Lower converged loss correlates with higher average reward, while coverage and max reward remain sta-

ble. This suggests that accurate reward distribution learning enables better preference alignment and improved recommendation quality. To verify whether discovered high-reward lists are incorporated into the learned policy, we analyze offline metrics in Tables 1–3. On ML1M, TESCO-TB improves R-NDCG by +0.7% and R-MRR by +3.3%. On KuaiRand-1K, R-NDCG gains +4.2%, while R-MRR stays stable (-0.3%). On KuaiRand-27K, average reward improves by +4.5% and R-NDCG by +4.3%, with R-MRR unchanged. These consistent gains across datasets demonstrate that the Teacher enables more faithful approximation of the underlying reward distribution, improving ranking quality even without online feedback.

3.3.3 Reward-Aware Exploration Rather Than Random Diversification

Both online and offline performance improve simultaneously, demonstrating that the Teacher facilitates directed exploration toward under-explored, reward-informative regions rather than simply adding noise. On KuaiRand-27K, TESCO-TB achieves the highest average reward (2.428 online, 2.234 offline) and max reward (4.097 online, 4.014 offline) while maintaining competitive ILD and coverage. These gains across a vastly expanded action space confirm that the Teacher mechanism scales robustly to large item spaces. The slight yet consistent increases in ILD without significant reward loss further validate reward-aware exploration.

Overall, TESCO demonstrably expands the effective exploration space while maintaining or improving reward efficiency. It successfully directs the model toward under-explored, high-reward regions and enhances the completeness of the reward distribution modeling. The consistent performance gains across ML1M, KuaiRand-1K, and the large-scale KuaiRand-27K datasets collectively confirm that the proposed Teacher mechanism achieves under-explored and high-reward aware exploration, a key advancement over prior generative flow-based recommender models.

3.4 Impact of Key Model Components (RQ3)

3.4.1 Teacher–Student Framework *vs.* Student-Only Learning

In Table 4, we observe that our model consistently outperforms the strongest baseline GFN4Rec by a substantial margin in both offline and online evaluations. This confirms that our approach is not merely optimized for static historical data but generalizes effectively to interactive, online environments. Notably, the improvements in **Avg.** $\mathcal{R}$ and **Max.** $\mathcal{R}$ are even more pronounced under online evaluation, suggesting that our method better leverages real-time user feedback to refine its recommendations.

Table 4: Relative improvement (%) of TESCO-TB over the best baseline GFN4Rec-TB in online and offline evaluations. Positive values indicate gains.

Setting	Metric	ML1M	KuaiRand-1K
Online	Avg. $\mathcal{R}$ (↑)	+3.5%	+4.4%
	Max. $\mathcal{R}$ (↑)	+1.2%	+1.1%
	Coverage (↑)	+17.5%	+3.3%
	ILD (↑)	+2.4%	+2.9%
Offline	Avg. $\mathcal{R}$ (↑)	+1.8%	+4.5%
	Max. $\mathcal{R}$ (↑)	+0.7%	+3.1%
	R-NDCG (↑)	+0.7%	+4.2%
	R-MRR (↑)	+3.3%	-0.3%
	Coverage (↑)	-0.7%	+7.5%
	ILD (↑)	+1.8%	+2.3%

3.4.2 Effect of Teacher Reward Design

As shown in Table 5, We observe that the *Reward-Only* configuration achieves slightly higher average reward than the *Loss-Only* variant, but performs worse in terms of maximum reward, coverage, and ILD—with coverage decreasing by approximately 22.9%. This indicates that relying solely on external rewards provides insufficient exploration over the list space and tends to overlook certain lists with high scores. In contrast, the *Full Configuration* exhibits notable improvements in both average and maximum rewards compared with the other two variants, while also achieving higher coverage and ILD than the *Reward-Only* setup. The incorporation of the loss component enables exploration towards a broader set of under-explored list spaces. Compared to a purely reward-driven Teacher configuration, this design yields greater diversity in the generated lists, which is instrumental for producing varied yet high-quality recommendation lists.

Table 5: Results of different Teacher reward configurations for TESCO in the online environment on the KuaiRand-1K dataset.

Configuration	KuaiRand-1K			
	Avg. $\mathcal{R}$ (↑)	Max. $\mathcal{R}$ (↑)	Coverage (↑)	ILD (↑)
Reward-Only	2.355 ± 0.110	3.933 ± 0.089	16.600 ± 3.027	0.540 ± 0.028
Loss-Only	2.353 ± 0.041	3.944 ± 0.091	21.533 ± 1.963	0.556 ± 0.011
Full Configuration	2.430 ± 0.116	4.100 ± 0.091	18.600 ± 5.441	0.542 ± 0.008

3.4.3 Exploration vs. Greedy

From the results presented in Table 2, we can compare the behavior of TESCO (Explore) and TESCO. Both configurations reflect a consistent trade-off: TESCO (Explore) yields higher coverage and ILD,

Table 6: Impact of varying Teacher-Student sampling ratios on online recommendation performance, with all results reported on the KuaiRand-1K dataset. Best results are in bold. Metrics with ↑ indicate higher is better, while ↓ indicates lower is better.

Ratio	Avg. $\mathcal{R}$ (↑)	Max. $\mathcal{R}$ (↑)	Coverage (↑)	Loss (↓)
1.0	$\mathbf{2.4849 \pm 0.0632}$	3.9667 ± 0.1394	16.60 ± 1.14	$\mathbf{0.0525 \pm 0.0023}$
2.0	2.3750 ± 0.0383	3.9667 ± 0.0745	$\mathbf{22.40 \pm 1.34}$	0.0591 ± 0.0031
3.0	2.4768 ± 0.0598	$\mathbf{4.0000 \pm 0.1179}$	12.00 ± 1.22	0.0539 ± 0.0029

indicating superior diversity, while TESCO produces recommendation lists of higher average reward and max reward. This suggests that the exploration mode broadens list exposure at the cost of some accuracy, whereas the greedy mode focuses on optimizing immediate reward.

3.4.4 VALIDITY OF ONLINE SIMULATOR

While offline metrics such as NDCG and MRR are widely used, reliable online simulation has become an active research direction (Ie et al., 2019). As shown in Tables 1–3, the performance rankings induced by our online simulator and offline test splits are largely aligned across methods. On the large-scale KuaiRand-27K dataset (27K users, 500K items), online and offline rankings remain highly consistent, suggesting the simulator scales robustly and captures key user behavior patterns even in massive action spaces. Nevertheless, the pretrained environment may converge to behaviors resembling the offline distribution and remains susceptible to mode collapse without proper regularization (Metz et al., 2017).

3.5 ROBUSTNESS AND SENSITIVITY ANALYSIS (RQ4)

3.5.1 IMPACT OF DIFFERENT TEACHER REWARD DESIGN

As shown in Figure 4, we denote model variants as Tesco_C_α, where parameters C and α govern the Teacher reward function. We evaluate $C \in \{0, 9, 19, 29, 39, 49\}$ and $\alpha \in \{2.0, 2.5, 2.6, 2.7, 2.8, 2.9, 3.0, 4.0, 6.0, 8.0\}$, and report only those configurations that achieve an average reward greater than 2.40. The figure reveals that across different values of C and α, the final model performance remains consistently high. This suggests that the method is not sensitive to specific hyperparameter choices, reflecting a degree of robustness in the proposed approach.

3.5.2 IMPACT OF DIFFERENT TEACHER-STUDENT SAMPLING RATIO

In this section, we study how the Teacher-Student sampling ratio affects model performance. It determines Student sampling steps between each Teacher step. A ratio of 1 produces alternating steps (Student → Teacher → ...), while a ratio of 2 has two Student steps per Teacher step.

As shown in Table 6, all variants attain average reward over 2.37, substantially outperforming baseline GFN4Rec-TB with similar maximum rewards, which indicates stable discovery of high-reward lists. Adopting different Teacher ratios consistently improves baseline performance. We set the ratio to 1 in main experiments for balanced guidance and stable training.

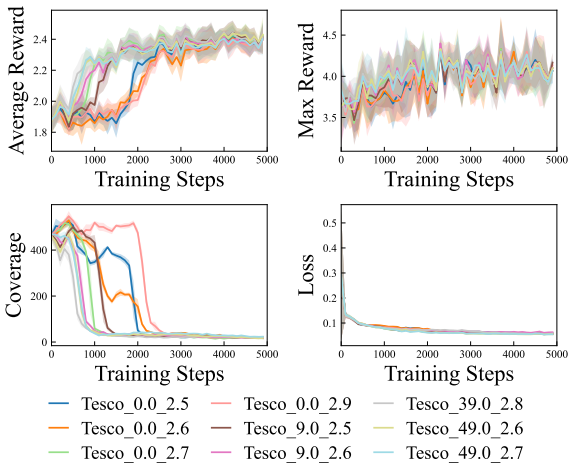

Figure 4: Online performance on the KuaiRand-1K dataset for different variants of TESCO.

4 CONCLUSION

In this work, we propose TESCO, a Teacher–Student framework that leverages discrepancy signals guide sampling toward informative yet under-explored regions of the list space. Evaluated on two real-world datasets in both offline and simulated online settings, TESCO consistently outperforms strong baselines in terms of user-aligned utility and diversity. Ablation studies further confirm the robustness of the approach across a range of hyperparameter configurations and training protocols.

AI USE STATEMENT

Generative AI tools were used solely for language polishing and improving the clarity of the manuscript. They were not used to generate research ideas, design the methodology or experiments, analyze the results, or produce scientific claims. The authors reviewed and take full responsibility for all content in the paper.

ETHICS STATEMENT

This work uses existing benchmark datasets and does not involve new data collection or human-subject experiments. We follow the ICLR Code of Ethics and do not identify any specific ethical concerns associated with this study.

REPRODUCIBILITY STATEMENT

The methodology and training procedure are described in Section 2, and the experimental setup is provided in Section 3. Further implementation details, dataset processing, evaluation protocols, and additional results are included in the appendix. Anonymous source code is also provided.

REFERENCES

Qingyao Ai, Keping Bi, Jiafeng Guo, and W Bruce Croft. Learning a deep listwise context model for ranking refinement. In *The 41st international ACM SIGIR conference on research & development in information retrieval*, pp. 135–144, 2018.

Emmanuel Bengio, Moksh Jain, Maksym Korablyov, Doina Precup, and Yoshua Bengio. Flow network based generative models for non-iterative diverse candidate generation. *Advances in neural information processing systems*, 34:27381–27394, 2021.

Christopher JC Burges. From ranknet to lambdarank to lambdamart: An overview. *Learning*, 11 (23-581):81, 2010.

Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th international conference on Machine learning*, pp. 129–136, 2007.

Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems*, pp. 7–10, 2016.

Yufei Feng, Binbin Hu, Yu Gong, Fei Sun, Qingwen Liu, and Wenwu Ou. Grn: Generative rerank network for context-wise recommendation. *arXiv preprint arXiv:2104.00860*, 2021.

Chongming Gao, Shijun Li, Yuan Zhang, Jiawei Chen, Biao Li, Wenqiang Lei, Peng Jiang, and Xiangnan He. Kuairand: An unbiased sequential recommendation dataset with randomly exposed videos. In *Proceedings of the 31st ACM International Conference on Information and Knowledge Management*, CIKM '22, pp. 3953–3957, 2022. doi: 10.1145/3511808.3557624. URL https://doi.org/10.1145/3511808.3557624.

Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. Deepfm: a factorization-machine based neural network for ctr prediction. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pp. 1725–1731, 2017.

F Maxwell Harper and Joseph A Konstan. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)*, 5(4):1–19, 2015.

Eugene Ie, Chih-wei Hsu, Martin Mladenov, Vihan Jain, Sanmit Narvekar, Jing Wang, Rui Wu, and Craig Boutilier. Recsim: A configurable simulation platform for recommender systems. *arXiv preprint arXiv:1909.04847*, 2019.

Ray Jiang, Sven Gowal, Yuqiu Qian, Timothy Mann, and Danilo J. Rezende. Beyond greedy ranking: Slate optimization via list-CVAE. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=r1xX42R5Fm.

Wang-Cheng Kang and Julian McAuley. Self-attentive sequential recommendation. In *2018 IEEE international conference on data mining (ICDM)*, pp. 197–206. IEEE, 2018.

Minsu Kim, Sanghyeok Choi, Taeyoung Yun, Emmanuel Bengio, Leo Feng, Jarrid Rector-Brooks, Sungsoo Ahn, Jinkyoo Park, Nikolay Malkin, and Yoshua Bengio. Adaptive teachers for amortized samplers. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=BdmVgLMvaf.

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

Matevž Kunaver and Tomaž Požrl. Diversity in recommender systems–a survey. *Knowledge-based systems*, 123:154–162, 2017.

Dawen Liang, Rahul G Krishnan, Matthew D Hoffman, and Tony Jebara. Variational autoencoders for collaborative filtering. In *Proceedings of the 2018 world wide web conference*, pp. 689–698, 2018.

Shuchang Liu, Fei Sun, Yingqiang Ge, Changhua Pei, and Yongfeng Zhang. Variation control and evaluation for generative slate recommendations. In *Proceedings of the Web Conference 2021*, pp. 436–448, 2021.

Shuchang Liu, Qingpeng Cai, Zhankui He, Bowen Sun, Julian McAuley, Dong Zheng, Peng Jiang, and Kun Gai. Generative flow network for listwise recommendation. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 1524–1534, 2023.

Weiwen Liu, Yunjia Xi, Jiarui Qin, Fei Sun, Bo Chen, Weinan Zhang, Rui Zhang, and Ruiming Tang. Neural re-ranking in multi-stage recommender systems: A review. In Luc De Raedt (ed.), *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022*, pp. 5512–5520. ijcai.org, 2022. doi: 10.24963/IJCAI.2022/771. URL https://doi.org/10.24963/ijcai.2022/771.

Nikolay Malkin, Moksh Jain, Emmanuel Bengio, Chen Sun, and Yoshua Bengio. Trajectory balance: Improved credit assignment in gflownets. *Advances in Neural Information Processing Systems*, 35:5955–5967, 2022.

Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein. Unrolled generative adversarial networks. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=BydrOIcle.

Changhua Pei, Yi Zhang, Yongfeng Zhang, Fei Sun, Xiao Lin, Hanxiao Sun, Jian Wu, Peng Jiang, Junfeng Ge, Wenwu Ou, et al. Personalized re-ranking for recommendation. In *Proceedings of the 13th ACM conference on recommender systems*, pp. 3–11, 2019.

Steffen Rendle. Factorization machines. In *2010 IEEE International conference on data mining*, pp. 995–1000. IEEE, 2010.

Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proceedings of the 28th ACM international conference on information and knowledge management*, pp. 1441–1450, 2019.

Jun Wang, Lantao Yu, Weinan Zhang, Yu Gong, Yinghui Xu, Benyou Wang, Peng Zhang, and Dell Zhang. Irgan: A minimax game for unifying generative and discriminative information retrieval models. In *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*, pp. 515–524, 2017.

Fen Xia, Tie-Yan Liu, Jue Wang, Wensheng Zhang, and Hang Li. Listwise approach to learning to rank: theory and algorithm. In *Proceedings of the 25th international conference on Machine learning*, pp. 1192–1199, 2008.

CONTENTS

1 Introduction **1**

2 Methodology **3**

2.1 Problem Formulation 3

2.2 Preliminary on GFlowNets 3

2.3 Student Model for List Generation Based on User–List Preferences 4

2.4 Teacher Model for Adaptive Exposure to Under-Explored Lists 5

2.5 Alternating Teacher–Student Sampling for List Generation 6

3 Experiments **6**

3.1 Experimental Setups 6

3.2 Better Align with User Preferences (RQ1) 7

3.3 Gain Diverse Lists While Maintaining Recommendation Quality (RQ2) 7

3.3.1 Ability to Generate More Diverse Recommendation lists 7

3.3.2 Ability to Improve the Quality of Learned List Scores Distribution 7

3.3.3 Reward-Aware Exploration Rather Than Random Diversification 8

3.4 Impact of Key Model Components (RQ3) 8

3.4.1 Teacher–Student Framework *vs.* Student-Only Learning 8

3.4.2 Effect of Teacher Reward Design 8

3.4.3 Exploration vs. Greedy 8

3.4.4 Validity of Online Simulator 9

3.5 Robustness and Sensitivity Analysis (RQ4) 9

3.5.1 Impact of Different Teacher Reward Design 9

3.5.2 Impact of Different Teacher-Student Sampling Ratio 9

4 Conclusion **9**

A Related Work **13**

A.1 List-wise Recommendation 13

A.2 Classical Generative List Recommendation 13

B Pseudocode **14**

C Details of Datasets **14**

D Details of Baselines **15**

E Details of Experimental Settings **15**

E.1 Simulated User Environment 15

E.2 Implementation Details 16

F Details of Evaluation Metrics **16**

G More Experimental Results and Analysis **17**

G.1 Component Study: Trajectory Balance vs. Detailed Balance 17

G.2 Impact of Different Teacher Reward Design 17

G.3 Computational Complexity 17

H Discussion **18**

H.1 Limitations 18

H.2 Broader Impacts 18

A Related Work

A.1 List-wise Recommendation

Real-world recommendation services typically present users with a list of items at once, rather than scoring items in isolation. In such contexts, user responses to individual items are shaped by how the entire list is organized. Some users may prefer diversity, while others benefit from multiple similar options within the same list (Kunaver & Požrl, 2017). This observation motivates the list-wise recommendation formulation, which explicitly accounts for mutual influences among items in the exposed list (Ai et al., 2018; Burges, 2010; Cao et al., 2007; Pei et al., 2019; Xia et al., 2008). The underlying idea is to learn from the impact of adding or removing a specific item, evaluated against list-level objectives such as NDCG or more complex reward signals defined over the whole list. In contrast, standard pointwise and pairwise learning-to-rank methods (Cheng et al., 2016; Kang & McAuley, 2018; Koren et al., 2009; Rendle, 2010; Sun et al., 2019) aim to learn an item-wise scoring function for a given user request, casting recommendation as a classification problem to enable efficient supervised learning. Their expressiveness largely depends on the design of the user request encoder include DNN (Cheng et al., 2016) and Transformers (Kang & McAuley, 2018). During inference, items are ranked by this learned function, and the top K are selected. Yet this paradigm fails to capture inter-item dynamics inherent in real deployments. Further evidence shows that in multi-stage systems, reranking models can better model item correlations thanks to a drastically reduced candidate set, allowing the use of more powerful neural architectures (Feng et al., 2021; Liu et al., 2022; Pei et al., 2019).

A.2 Classical Generative List Recommendation

To address the combinatorial complexity of list output spaces, generative approaches model the full joint distribution over recommended lists and produce entire lists using generative models. This perspective has been explored across pointwise (Liang et al., 2018; Wang et al., 2017), listwise (Jiang et al., 2019; Liu et al., 2021), and list recommendation settings. One representative method, ListCVAE (Jiang et al., 2019), leverages Conditional Variational Autoencoders (CVAE) to capture positional biases and interdependencies among items in list distributions. While conceptually appealing, subsequent work (Liu et al., 2021) demonstrates that ListCVAE suffers from a tension between accuracy and diversity, reflecting the broader difficulty of balancing exploitation and exploration in generative list recommendation. More recently, GFN4Rec (Liu et al., 2023) introduced a flow matching framework (Bengio et al., 2021) that directly aligns list generation probability with its utility, showing potential to deliver high-quality recommendations with substantially improved diversity—thereby supporting more effective online exploration. Nevertheless, GFN4Rec relies solely on historically recommended lists for training, which inherently limits its ability to explore under-explored high-potential lists absent from past lists. Our approach also falls within the generative paradigm and similarly couples list generation probability with utility scores. Crucially, it incorporates a Teacher model that adaptively steers the Student model to probe under-explored regions of the entire list space. This mechanism significantly boosts exposure for under-explored high-potential lists, yielding lists that are simultaneously more diverse and of higher quality.

Algorithm 1 Process of TESCO Training

Ensure: Trained policy $\pi(s_K|u)$

Initialize replay buffer $\mathcal{A} = \emptyset$
while not converged **do**
  # Apply behavior policy in running episodes
  Select behavior policy :

$$P_\beta(\tau) = \begin{cases} P_F(\tau;\theta), & \text{if select Student,} \\ P_F(\tau;\phi), & \text{if select Teacher,} \end{cases} \tag{17}$$

  for each user u in batch **do**
    Encode user state $\mathbf{s}_u$
    Initialize empty list s_0
    # Online sampling phase
    for $t = 1$ to K **do**
      Sample item $a_t \sim P_\beta(\tau)$
      Update list $s_t = s_{t-1} \oplus a_t$
    end for
    Obtain user feedback $\mathbf{Y}$ and compute reward $\mathcal{R}(s_K)$
    # Add experiences to buffer
    Store trajectories $\tau_(\mathbf{s}_u, s_K, \mathcal{R}(s_K), \mathbf{Y})$ in $\mathcal{A}$
  end for
  # Train on the mini-batch sampled from buffer
  if $|\mathcal{A}| \geq$ minimum buffer size **then**
    Sample mini-batch $\mathcal{B}$ from $\mathcal{A}$
    for each trajectory $\tau(\mathbf{s}_u, s_K, \mathcal{R}(s_K), \mathbf{Y})$ in $\mathcal{B}$ **do**
      Compute Student discrepancy: $\delta(\tau_i;\theta)$ (Eq. 7)
      Compute Teacher reward: $\mathcal{R}_T$ (Eq. 12)
      Compute Teacher discrepancy: $\delta(\tau_i;\phi)$ (Eq. 13)
    end for
    # Student training
    Update Student parameters:
    $\theta \leftarrow \text{Optimizer}\left(\mathcal{L}_{TB}(\tau;\theta)\right)$ (Eq. 8)
    # Teacher training
    Update Teacher parameters:
    $\phi \leftarrow \text{Optimizer}\left(\mathcal{L}_{TB}(\tau;\phi)\right)$ (Eq. 14)
  end if
end while

B PSEUDOCODE

This section describes the detailed algorithmic flow of our framework, as shown in the Algorithm 1.

C DETAILS OF DATASETS

KR1K and KR27K are two versions of the KuaiRand dataset, an unbiased sequential recommendation dataset collected from the recommendation logs of the Kuaishou mobile app with randomly exposed videos inserted into standard recommendation feeds. Among them, KR27K is the complete version containing over 27K users, while KR1K is a smaller version constructed by randomly sampling 1,000 users from KR27K and then removing irrelevant videos. User feedback in KuaiRand includes clicks, views, likes, comments, forwards, follows, and hates, where all behavior types are treated as binary signals. By including both KR1K and the larger-scale KR27K, we evaluate our method on KuaiRand under two different scales, and further verify its effectiveness in a more challenging large-scale recommendation setting.

For all datasets, we filter the data to 20-core users and segment each user's interaction history into sequences of length $K = 6$, treating each segment as an observed recommendation list. For sim-

Table 7: Dataset Summary. The records are used for offline training of policies and online user environment, but not used for online training of policies.

Dataset	$\lvert\mathcal{U}\rvert$	$\lvert I\rvert$	#Record	$\lvert\mathcal{B}\rvert$	Range of $\mathcal{R}$
ML1M	6,400	3,706	1,000,208	3	[0,3]
KR1K	1,000	69,219	2,597,865	7	[-1,6]
KR27K	27,285	507,493	71,487,342	7	[-1,6]

plicity, we set the item-wise reward weight $w_b = 1$ for all behavior types, except that the hate signal in KuaiRand is assigned $w_{\text{hate}} = -1$. As a result, the item-wise reward ranges are $\mathcal{R}(u, i) \in [0, 3]$ for ML1M and $\mathcal{R}(u, i) \in [-1, 6]$ for both KR1K and KR27K. Dataset statistics are summarized in Table 7.

D Details of Baselines

We compare **TESCO** with both ranking and re-ranking models, including:

- **CF** (Kang & McAuley, 2018) is a pointwise model that scores user–item interactions via the dot product of user and item embeddings.
- **ListCVAE** (Jiang et al., 2019) is a generative model that captures the distribution over recommendation lists using a conditional VAE, where the reward is incorporated as a conditioning signal.
- **PRM** (Pei et al., 2019) is a re-ranking model that employs CF as the initial ranker and applies a transformer-based re-ranker to encode the candidate set.
- **GFN4Rec** (Liu et al., 2023) is a list-wise recommendation method that uses a generative flow network to represent the probabilistic list generation process.
- **TESCO**, which is our proposed model with TB loss. Comparison between TB and DB objectives will be further discussed in Section G.1. We use the same simulator design as GFN4Rec, which helps ensure a fair comparison in our experiments.

E Details of Experimental Settings

E.1 Simulated User Environment

To simulate complex multi-type user behaviors observed in real-world data, we construct a stochastic user response model $\mathcal{E} : \mathcal{U} \times \mathcal{C}^K \to \mathcal{B}^K$, which predicts the probability that a user u positively engages with item i under behavior type b. The base neural model $g(u, O)$ outputs the initial behavior likelihood and consists of two components: (i) a Transformer-based user history encoder, similar to the user request encoder, and (ii) a state-to-behavior predictor that estimates the user response probability for a given recommendation list O.

We train this base model using binary cross-entropy loss on the ground-truth labels $y_{u,i,b}$. The trained model achieves AUC scores ranging from 0.7 to 0.9 across different behaviors on both datasets. When the pretrained user response model is deployed in the online environment, we further incorporate an item-influence module that suppresses each item's initial ranking score based on its similarity to other items in the recommendation list, thereby simulating users' preference for recommendation diversity. A significant factor $\rho > 0$ is introduced to ensure the presence of item influence, and we set $\rho = 0.2$ for ML1M and $\rho = 0.1$ for KR1K.

The final user response $y_{u,i,b}$ is uniformly sampled according to the modified behavior likelihood, in order to model the inherent uncertainty in user feedback. For the data sampling strategy used by all online learning methods, each mini-batch consists of newly collected instances from online inference and samples uniformly drawn from the entire replay buffer, in equal proportion, to mitigate catastrophic forgetting.

E.2 Implementation Details

We conduct all experiments on a cluster equipped with 8 NVIDIA A800-SXM4-80GB GPUs (each with 80GB memory and Compute Capability 8.0). In terms of hyperparameter settings, we perform a grid search to find optimal configurations for both baselines and our proposed method. During evaluation, we report the average results over five runs with different random seeds. The training procedure of TESCO is detailed in Algorithm 1. Specifically, we can find the online sampling phase with stochastic action sampling in Algorithm 1. At test time, when the goal is to produce the highest-quality recommendation slate, we turn off the stochastic action sampling and instead perform greedy selection based on the scores provided by the item selection model. To further analyze the performance of TESCO, we observe the results under greedy selection and using sampling with the stochastic action sampling turned on, we denote the latter as TESCO (Explore). For our adaptive Teacher-Student GFlowNets method (Kim et al., 2025; Bengio et al., 2021), the default configuration uses the TB loss (The training of DB is analyzed in Section G.1). We follow the experimental setup of the representative baseline GFN4Rec in both online and offline environment settings exactly, including data splits, candidate set size, batch configuration, and evaluation protocols, ensuring a fair comparison. **Data Split.** We adopt a user-level sequential holdout strategy for dataset partitioning. Specifically, for each user, we sort their interaction records in chronological order and split them as follows: the earliest $n - 10$ records form the training set, the subsequent 5 records form the validation set, and the last 5 records form the test set, where n denotes the total number of interactions for that user. To ensure sufficient training signals, we exclude users whose training records account for less than 60% of their total interactions. This per-user fixed-holdout approach naturally adapts the train/val/test ratio to each user's activity level, rather than imposing a global fixed split ratio. For both the KuaiRand and MovieLens datasets, we consistently set the holdout size to 5 for both the validation and test sets.

F Details of Evaluation Metrics

We employ both online and offline metrics to comprehensively evaluate the recommendation performance. **Online Metrics:** In the online user environment, we evaluate each recommendation list O through the following metrics: average reward, max reward, coverage, and ILD. **Offline Metrics:** For offline evaluation, we evaluate each recommendation list O through the following metrics: average reward, max reward, R-NDCG, R-MRR, coverage, and ILD.

- **Average Reward (Avg. $\mathcal{R}$):** The average listwise reward across user requests, indicating the overall recommendation quality.
- **Max Reward (Max. $\mathcal{R}$):** The maximum reward obtained by a recommendation list, representing the best-case performance.
- **Coverage:** The number of distinct items exposed in a mini-batch, measuring the exploration breadth of the recommender.
- **Intra-List Diversity (ILD):** The average dissimilarity between items within each list, computed as:

$$\text{ILD}(O) = \frac{1}{K(K-1)} \sum_{a_i \in O} \sum_{a_j \in O \setminus \{a_i\}} (1 - \text{sim}(a_i, a_j))$$

- **Reward-based NDCG (R-NDCG):** Generalizes NDCG by using listwise reward $\mathcal{R}(s_K)$ as relevance scores:

$$\text{R-NDCG} = \frac{1}{K} \sum_{k=1}^{K} \text{R-NDCG}(k)$$

$$\text{R-NDCG}(k) = \frac{\sum_{a_k} \mathcal{R}(a_k) \cdot 2^{1-\text{Rank}(a_k)}}{\text{IDCG}}$$

- **Reward-based MRR (R-MRR):** Extends MRR by weighting reciprocal rank with rewards:

$$\text{R-MRR} = \frac{1}{K} \sum_{k=1}^{K} \text{R-MRR}(k)$$

$$\text{R-MRR}(k) = \sum_{a_k} \frac{\mathcal{R}(a_k)}{\text{Rank}(a_k)}$$

G More Experimental Results and Analysis

G.1 Component Study: Trajectory Balance vs. Detailed Balance

To investigate the impact of different GFlowNets objectives in the model, we conducted experiments with two objectives (TB and DB) under both online and offline settings on the ML1M, KuaiRand-1K, and KuaiRand-27K datasets. As shown in Table 1, Table 2, and Table 3, in the online setting, on the ML1M dataset, the TESCO-TB's average reward, coverage, and ILD all showed some improvement compared to TESCO-DB, while the change in max reward was not significant. This is because the maximum reward in the ML1M dataset is only 3, and the reward range is limited, so both TESCO-TB and TESCO-DB can find items close to the maximum reward. However, the improvement in average reward, coverage, and ILD indicates that TESCO-DB fails to discover some suboptimal solutions that TESCO-TB can identify, providing a more diverse list of items that better match user preferences.

On the KuaiRand-1K and KuaiRand-27K datasets, TESCO-TB showed more substantial improvements. Both average reward and max reward increased significantly, while coverage and ILD decreased slightly. It is worth noting that improvements in reward are naturally accompanied by some decrease in diversity, as it is not always possible to simultaneously enhance both quality and diversity of the recommendation list. The results suggest that, on large-scale datasets, TESCO-TB not only finds more suboptimal solutions but also discovers lists that TESCO-DB misses, leading to significant gains in both average and maximum reward. Notably, on the large-scale KuaiRand-27K dataset, TESCO-TB achieves the highest average reward (2.428) and max reward (4.097) online, further confirming the advantage of the TB objective in massive combinatorial action spaces. Although diversity metrics decrease slightly, our coverage and ILD remain comparable to those of GFN4Rec-TB, which also uses the TB objective.

In Table 1, Table 2, and Table 3, the TESCO-TB outperforms TESCO-DB in terms of average reward, max reward, R-NDCG, and R-MRR across all three datasets. This indicates that TB can discover more high-reward lists in offline datasets and recommend lists that better align with the user preference, further demonstrating its stronger ability to learn the list scores distribution.

In summary, TESCO-TB and TESCO-DB are both competitive on small datasets, but TESCO-TB outperforms TESCO-DB on large datasets. The superior performance of the TB objective stems from its design, which directly links the final recommendation quality to the initial state, allowing the reward to propagate more efficiently. In contrast, the DB objective focuses more on intermediate states, resulting in slower reward propagation and slower learning of the final recommendation quality from the initial state. Therefore, in practice, we recommend using the TESCO-TB.

G.2 Impact of Different Teacher Reward Design

Furthermore, to validate our design choice of multiplying by $(1 + C)$ for positive discrepancies in Eq. 11, we conduct an ablation study by replacing $(1+C)$ with $(1+C)^n$ and varying n from 1 to 5. As shown in Table 8, $n = 1$ (our default configuration) consistently achieves the best performance on both ML1M and KuaiRand-27K. While $n = 2$ remains competitive, increasing n further leads to noticeable performance degradation, eventually dropping below the baseline GFN4Rec-TB for $n \geq 4$. This indicates that overly amplifying the weighting for under-explored lists introduces excessive noise and destabilizes training. The results confirm that the simple linear scaling by $(1+C)$ provides a balanced and effective guidance signal for the Teacher model.

G.3 Computational Complexity

Finally, we analyze the computational overhead introduced by the Teacher model in terms of training time and GPU peak memory. Regarding training time, on ML1M, ListCVAE requires approximately 3,800 seconds for 5,000 training steps, while GFN4Rec takes about 4,200 seconds, and TESCO completes training in around 4,700 seconds. On KuaiRand-1K, ListCVAE requires approximately 4,600 seconds, while GFN4Rec takes about 5,100 seconds, and TESCO completes training

Table 8: Ablation study on the power of $(1 + C)^n$ in teacher reward. Best results are in **bold**, suboptimal results are underlined. Metrics with ↑ indicate higher is better.

	ML1M				KuaiRand-27K			
Method	Avg. $\mathcal{R}$ ↑	Max. $\mathcal{R}$ ↑	Coverage ↑	ILD ↑	Avg. $\mathcal{R}$ ↑	Max. $\mathcal{R}$ ↑	Coverage ↑	ILD ↑
GFN4Rec-TB	2.219 ± 0.068	2.933 ± 0.091	12.600 ± 0.894	0.573 ± 0.005	2.324 ± 0.013	4.051 ± 0.031	203.8 ± 15.6	0.529 ± 0.001
TS-TB (n=1)	$\mathbf{2.297 \pm 0.037}$	$\mathbf{2.967 \pm 0.075}$	$\mathbf{14.800 \pm 1.643}$	$\mathbf{0.587 \pm 0.005}$	$\mathbf{2.428 \pm 0.021}$	$\mathbf{4.097 \pm 0.015}$	$\mathbf{214.2 \pm 62.3}$	$\mathbf{0.544 \pm 0.001}$
TS-TB (n=2)	2.281 ± 0.042	2.961 ± 0.078	16.520 ± 1.752	0.592 ± 0.005	2.408 ± 0.025	4.088 ± 0.018	238.5 ± 58.7	0.551 ± 0.001
TS-TB (n=3)	2.256 ± 0.051	2.948 ± 0.082	14.180 ± 1.895	0.598 ± 0.006	2.365 ± 0.032	2.065 ± 0.022	198.3 ± 71.2	0.558 ± 0.001
TS-TB (n=4)	2.201 ± 0.063	2.935 ± 0.091	11.240 ± 2.124	0.606 ± 0.007	2.285 ± 0.048	4.021 ± 0.031	156.7 ± 85.6	0.567 ± 0.002
TS-TB (n=5)	2.152 ± 0.078	2.918 ± 0.098	8.960 ± 2.356	0.615 ± 0.008	2.178 ± 0.065	3.958 ± 0.045	98.4 ± 102.3	0.581 ± 0.003
CF	2.081 ± 0.018	2.929 ± 0.011	10.340 ± 0.374	0.531 ± 0.003	2.248 ± 0.016	3.998 ± 0.015	$1{,}284.6 \pm 95.3$	0.541 ± 0.002
PRM	2.167 ± 0.124	2.967 ± 0.075	22.000 ± 1.414	0.534 ± 0.005	2.232 ± 0.010	4.009 ± 0.038	752.4 ± 58.2	0.528 ± 0.001
ListCVAE	1.001 ± 0.175	2.267 ± 0.224	310.60 ± 7.162	0.799 ± 0.007	1.895 ± 0.008	3.592 ± 0.042	$5{,}127.3 \pm 124.5$	0.691 ± 0.001

in around 5,700 seconds. On the larger-scale KuaiRand-27K, ListCVAE requires approximately 18,500 seconds, GFN4Rec takes about 20,200 seconds, and TESCO takes around 22,800 seconds. These results reflect the additional cost of updating the Teacher's GFlowNet parameters.

In terms of GPU peak memory, although TESCO introduces an auxiliary Teacher model that functions as an additional GFlowNet, the GPU memory footprint increases by only approximately 12%–15% compared to GFN4Rec across all three datasets. This modest overhead is primarily due to two factors. First, the Teacher shares the entire backbone with the Student, including the embedding layers, Transformer encoder, feature mappings, and state compression layers. Only the final decision heads (the forward policy encoder and the initial flow estimator) are maintained independently for the Teacher. Second, since the Student and Teacher forward-backward passes are executed strictly sequentially, their intermediate activations do not coexist in memory. Consequently, the activation peak remains roughly the same as that of a single GFN4Rec model, and the additional GPU consumption stems solely from the Teacher head's parameters and its optimizer states. Overall, the moderate increases in both training time and GPU memory are outweighed by the substantial performance improvements achieved by our method.

H DISCUSSION

H.1 LIMITATIONS

While the simulated online environment aligns well with offline rankings across all datasets, we note that the pretrained user response model may theoretically converge to behavior patterns resembling the offline distribution, and remains susceptible to mode collapse without careful regularization—though we did not observe this in practice. Additionally, TESCO introduces modest computational overhead (∼12–15% GPU memory increase) due to the auxiliary Teacher model, which may require consideration in resource-constrained deployment scenarios.

H.2 BROADER IMPACTS

TESCO offers a principled and scalable framework for mitigating sampling imbalance in generative recommendation, with far-reaching potential across the recommender systems ecosystem. By explicitly guiding exploration toward under-explored yet high-utility lists, TESCO mitigates popularity bias and exposure bias—long-standing challenges that disproportionately favor already-popular items while burying high-quality niche content. This has the potential to reshape platform economics by improving fairness for long-tail creators and enabling more diverse content ecosystems. Furthermore, the discrepancy-guided exploration mechanism introduced here is not limited to recommendation; it provides a general paradigm for balancing exploitation and exploration in any large-scale sequential decision-making system with sparse reward signals, such as drug molecule design, automated experiment selection, or personalized education. In high-frequency consumption scenarios like short-video and news feeds, TESCO's ability to maintain recommendation quality while expanding coverage could directly enhance billions of user sessions, making list-level optimization both more effective and more equitable.