Responsible RecSys by Design: Approximation Algorithms for Calibrated Recommendations with Sponsored Items

Jing Yuan [1], Shaojie Tang[2], Shuzhang Cai [3], Yao Wang [4]

[1] Department of Computer Science and Engineering, University of North Texas, Texas, USA
[2]Department of Management Science and Systems, University at Buffalo, New York, USA
[3]Naveen Jindal School of Management, University of Texas at Dallas, Texas, USA
[4] Center of Intelligent Decision-making and Machine Learning,
School of Management, Xi'an Jiaotong University, Xi'an, China
jing.yuan@unt.edu, shaojiet@buffalo.edu, sxc200125@utdallas.edu, yao.s.wang@gmail.com

Abstract

Calibrated Recommendation Systems (CRS) balance user preferences with constraints like diversity, fairness, and novelty to create inclusive recommendation lists. However, existing research often overlooks the mandatory inclusion of sponsored items, assuming unrestricted product selection. In practice, sponsored items, paid for by advertisers, must be included, which can conflict with CRS goals when advertisers' priorities misalign with system objectives. This paper addresses this gap by formulating CRS with sponsored items as a combinatorial optimization problem. We develop efficient approximation algorithms to generate the most calibrated recommendation lists while meeting sponsorship requirements.

Introduction

Calibrated Recommendation Systems (CRS) aim to balance user preferences with diversity, fairness, and responsibility in recommendations. For example, while recommending books, the CRS will include works from different authors, including those from under-represented backgrounds or other niche categories to ensure inclusiveness.

Meanwhile, sponsored placements have grown significantly within these systems (Malthouse et al. 2019; Tang et al. 2024). For example, sponsored items constitute 76% of overall Amazon ad spend among sellers[1]. Figure 1 gives an example of recommendation systems that accommodate sponsored items. Those sponsored items are promoted or paid for by advertisers or partners, and their inclusion in the recommendation list is crucial to fulfill contractual obligations and achieve monetization goals (Liao et al. 2022; Yan et al. 2020). However, incorporating sponsored items may run counter to the CRS's objectives because advertisers may prioritize promoting particular brands or goods, which may misalign with the CRS's mission to provide broad and well-balanced suggestions. This raises a significant research issue: *"Can we build a calibrated recommendation list while ensuring the inclusion of sponsored items?"*

While a large body of existing work has been done on CRS (Steck 2018; Kleinberg, Ryu, and Éva Tardos 2023; Wang, Tu, and Gionis 2025), it often assumes the absence of sponsored items and, hence, their impacts. For instance, these studies ignore the obligation to include sponsored items in the recommendation lists. This literature gap motivates us to revisit the CRS problem. The main contributions of this paper are summarized as follows.

1. We are the first to propose and investigate the CRS problem in the presence of sponsored items. Our primary objective is to generate inclusive and responsible recommendation lists that mandate the inclusion of sponsored items. This work addresses a significant gap in the CRS literature, as incorporating sponsored items is a common practice on most platforms.

2. We formulate the problem as a combinatorial optimization task. Building on the framework introduced by (Steck 2018), our objective function comprises two key components: A quality term, reflecting the overall relevance of recommendations to user preferences, and a calibration term, which balances user preferences with diversity, fairness, and responsibility in recommendations, ensuring alignment with the target distribution. A unique challenge in our study is the mandatory inclusion of all sponsored items, which complicates the optimization process as these items may conflict with traditional CRS objectives. Our key contribution lies in developing efficient and effective algorithms to address this problem, along with providing theoretical performance bounds for these algorithms.

3. We perform extensive experiments on two real-world datasets to evaluate the performance of our solutions. The results demonstrate the effectiveness of our approach in achieving high user satisfaction and a high degree of calibration. Additionally, we discuss practical strategies for balancing the trade-off between these two criteria.

Related Works

In this section, we review existing studies on two topics closely related to our work. The first stream focuses on calibrated recommendation but does not account for the inclusion of sponsored items. The second stream considers the presence of sponsored items but does not impose constraints on the distribution of recommended items.

Calibrated Recommendation. The concept of calibrated recommendation was initially proposed by (Steck 2018).

[1]https://www.junglescout.com/blog/amazon-sponsored-product-ads/

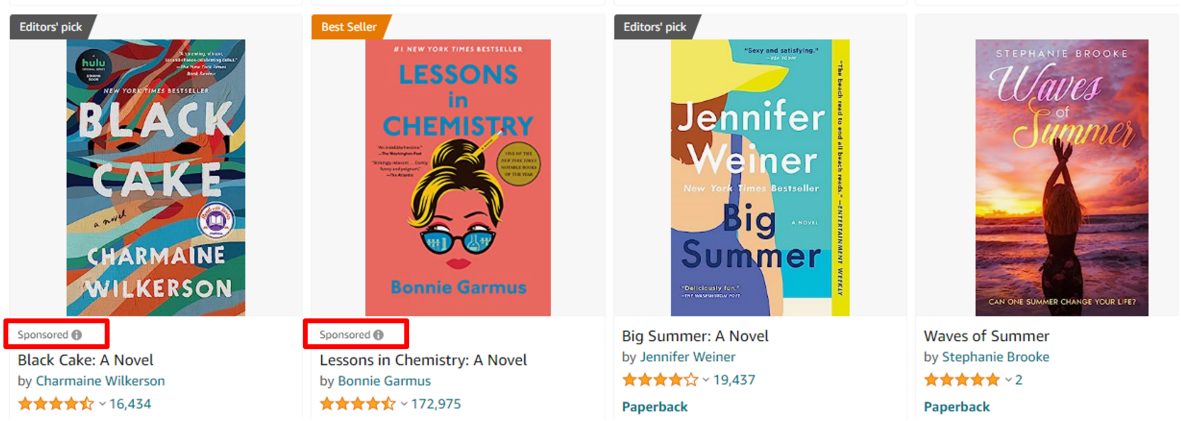

Figure 1: An illustration of a recommendation system from Amazon presenting a lineup of books, with sponsored items emphasized on the left-hand side.

The goal is to generate recommendations that precisely reflect a user's interests, at the same time in a calibrated manner. Since then, a growing interest in improving calibration in recommendation systems has been witnessed, including the application of greedy selection using statistical divergences, the development of calibration metrics (Naghiaei et al. 2022; da Silva and Durão 2022), and the utilization of LP-based heuristics (Seymen, Abdollahpouri, and Malthouse 2021). However, much of this research has primarily focused on the empirical evaluation of calibration heuristics. Very recently, (Kleinberg, Ryu, and Éva Tardos 2023) develop the first rigorous approximation algorithms that guarantee the quality of calibrated lists with provable performance bound. Similar to (Agrawal et al. 2009; Ashkan et al. 2015), they formulate their problem as a submodular optimization problem. However, all aforementioned studies assume the absence of sponsored items. We address this gap in the literature by proposing to study the calibrated recommendation problem in the presence of sponsored items. This problem is particularly challenging because incorporating sponsored items may conflict with the original goal of calibrated recommendations.

Recommendation with Sponsored Items. In the broader field of recommendation systems, there exist a few studies that examine the problem of how to jointly optimize the allocation of sponsored and organic items. For example, (Liao et al. 2022) investigated the problem of determining ad placements within the feed, with ads representing sponsored items and the feed consisting of organic items. Recently, (Tang et al. 2024) studied the problem of assortment planning in the presence of sponsored items. However, our problem setting differs significantly from theirs in two key ways: 1. Neither of the aforementioned studies considers a calibrated setting, meaning their frameworks do not impose constraints on the distribution of recommended items. 2. Their objective functions differ from ours. For instance, their focus is on maximizing metrics like revenue or engagement, driven by complex choice models. To the best of our knowledge, our work is the first to provide non-trivial approximation guarantees for calibration in the presence of sponsored items.

Notation	Description
$\mathcal{I}$	Set of n items
$\mathcal{G}$	Set of m classes
$\pi = \{\pi_1, \ldots, \pi_t\}$	Recommendation list of t items
p	Target distribution
$q(\pi)$	Recommendation distribution of π
w_j	Weight (visibility) of position j
$H^2(p, q(\pi))$	Squared Hellinger distance
$h(p, q(\pi))$	Closeness: $1 - H^2(p, q(\pi))$

Table 1: Summary of Notations

Preliminaries and Problem Statement

Throughout the remainder of this paper, we denote the set $\{1, 2, \ldots, i\}$ as $[i]$ for any positive integer i. When referring to a set A and an element a, we use $A + a$ to represent the union $A \cup \{a\}$. The key notations used in this paper are summarized in Table 1.

Preliminary: Target and Recommendation Distributions

The input to our problem includes a *target distribution* that the recommendation system aims to approximate. This target distribution can be interpreted in various ways depending on the context- a user preference distribution or a fairness-aware distribution across classes. For instance, consider a target distribution in which the user rated approximately 80% fiction and about 20% nonfiction books. Herein, one would expect the list of suggested books to have about 80% fiction titles, while the remaining 20% should be nonfiction works. Another example target distribution, which may be targeting fairness between book recommendations coming from different genders of authors, could be as follows: Suppose the desire is to ensure 30% recommended books are authored by women, and the remainder - 70% - by men. In

these cases, the target distribution would drive the recommendation system to guarantee proportional representation of books authored by people of different genders. In both cases, the target distribution serves as a reference or goal that the recommendation system aims to approximate. That is, by pulling the recommendations towards the target distribution, the system shall make recommendations that can potentially be more personalized to suit the taste of a user or meet the explicit criteria of fairness.

More formally, consider a set $\mathcal{I}$ of n *items* (e.g., movies), our objective is to provide a recommendation sequence of items $\pi = \{\pi_1, \pi_2, ..., \pi_t\}$ whose *classes* (e.g., genres) align with the target distribution. Here, $\pi_i \in \mathcal{I}$ denotes the item positioned in slot i of the recommendation list π. We represent the target distribution of classes $g \in \mathcal{G}$ as $p(g)$, where $\mathcal{G}$ denotes the set of m classes we want to calibrate (e.g., $\mathcal{G} = \{\text{Action}, \text{Comedy}, \text{Drama}\}$). Assume each item $i \in \mathcal{I}$ has a class distribution q_i (e.g., $q_i = \{\text{Action} : 0.8, \text{Comedy} : 0.1, \text{Drama} : 0.1\}$), we define the *recommendation distribution* over classes, denoted as $q(\pi)$, for a recommendation list $\pi = \{\pi_1, \pi_2, ..., \pi_t\}$ as follows:

$$q(\pi)(g) = \sum_{j=1}^{t} w_j \cdot q_{\pi_j}(g) \tag{1}$$

where w_j is the weight of position j, and we assume that $\sum_{j=1}^{t} w_j = 1$. Here, the weight w_j can be interpreted as the visibility of position j, representing the likelihood that a user will browse the item placed in that position. Intuitively, $q(\pi)(g)$ captures the aggregated contribution of class g across all items in the recommendation list π.

To quantify the dissimilarity between the target distribution p and the recommendation distribution $q(\pi)$, we follow the formulation of (Kleinberg, Ryu, and Éva Tardos 2023) and introduce the squared Hellinger distance, denoted as $H^2(p, q(\pi))$, which can be calculated using the formula:

$$H^2(p, q(\pi)) = \frac{1}{2} \sum_{g \in \mathcal{G}} (\sqrt{p(g)} - \sqrt{q(\pi)(g)})^2$$
$$= 1 - \sum_{g \in \mathcal{G}} \sqrt{p(g) \cdot q(\pi)(g)}.$$

Intuitively, a larger $H^2(p, q(\pi))$ indicates that the distributions p and $q(\pi)$ are more dissimilar. Correspondingly, we measure the closeness between two distributions p and $q(\pi)$ using

$$h(p, q(\pi)) = 1 - H^2(p, q(\pi))$$
$$= \sum_{g \in \mathcal{G}} \sqrt{p(g) \cdot q(\pi)(g)}. \tag{2}$$

We provide a running example of the model described in appendix. In summary, our study involves two types of distributions. The first is the target distribution, a predefined distribution provided as part of the problem input. The second is the recommendation distribution, which represents the class distribution of items in the generated recommendation list. Our primary goal is to compute a recommendation list whose distribution closely aligns with the target distribution. However, as we will see later, this task is complicated by the requirement to include specific sponsored items in the recommendation list. The inclusion of these items may conflict with achieving optimal alignment with the target distribution, as sponsored items are often dictated by external constraints and may not perfectly match the desired distribution. Addressing this challenge requires developing strategies to balance adherence to the target distribution with the mandatory inclusion of sponsored items.

Problem Formulation

Now we will introduce our optimization problem for calibrated recommendations with sponsored items (CRSI).

In this problem, we are provided with a set of sponsored items, denoted as $\mathcal{I}^s$, which is a subset of the ground set $\mathcal{I}$. In the context of platforms like Amazon, sponsored items refer to products that are promoted through Amazon's pay-per-click (PPC) advertising system. Sellers have the opportunity to bid on specific keywords in an auction, and the winning bids determine the set of sponsored items, $\mathcal{I}^s$. Once $\mathcal{I}^s$ is determined, it is mandatory to include these sponsored items in the recommendation list. The process of determining $\mathcal{I}^s$ is beyond the scope of this paper. For our study, we assume that $\mathcal{I}^s$ is already provided and known.

Define Π as the set of all *feasible* recommendation lists that satisfy two conditions: (1) their size is at most k, and (2) they include all sponsored items. That is,

$$\Pi = \{\pi \mid |\pi| = k \text{ and } \mathcal{I}(\pi) \supseteq \mathcal{I}^s\} \tag{3}$$

where $\mathcal{I}(\pi)$ denotes the set of items contained within a recommendation list π, i.e., $\mathcal{I}(\pi) = \cup_{j \in [|\pi|]} \{\pi_j\}$.

Our objective is to identify a feasible recommendation list $\pi \in \Pi$ that maximizes a utility function $z : \Pi \to \mathbb{R}_{\geq 0}$ defined as follows:

$$z(\pi) = (1 - \lambda) \cdot \sum_{i \in \mathcal{I}(\pi)} s_i + \lambda \cdot h(p, q(\pi)), \tag{4}$$

where the parameter $\lambda \in [0, 1]$ determines the trade-off between two terms: (a) the predicted "quality" scores s_i for items $i \in \mathcal{I}$ by the recommender system, e.g., s_i might represent the predicted relevance or the estimated rating of item i. (b) the closeness between the target distribution p and the recommendation distribution $q(\pi)$ using the calibration metric defined in (2). Hence, the balance between accuracy-focused recommendations and calibration can be adjusted using the parameter λ. Consequently, a higher value of λ is preferable to achieve better calibration.

A formal description of our problem is listed in **P.0**.

P.0 $\max_{\pi \in \Pi} (1 - \lambda) \cdot \sum_{i \in \mathcal{I}(\pi)} s_i + \lambda \cdot h(p, q(\pi))$
where $\Pi = \{\pi \mid |\pi| = k \text{ and } \mathcal{I}(\pi) \supseteq \mathcal{I}^s\}$.

While our primary focus is on finding a deterministic recommendation list that optimizes the objective in problem **P.0**, we later extend our study to consider randomized solutions, where the output is a probability distribution over recommendation lists rather than a single fixed list. This extension offers greater flexibility and potentially improved calibration.

Approximation Algorithms

Recalling our original problem, **P.0**, as a sequencing problem, we note that the order of selected items plays a critical role. Unfortunately, solving a sequencing problem is highly challenging because it requires simultaneously optimizing both the identities of the selected items and their positions in the recommendation list. In contrast, subset selection problems-where the goal is to select a subset of items without considering their order-are well-studied in the literature. One approach to address sequencing problems is to transform them into subset selection problems, allowing the use of existing subset selection algorithms. However, a key challenge in this transformation is preserving positional information, which is crucial for sequencing problems. To address this, we adopt the approach proposed in (Kleinberg, Ryu, and Éva Tardos 2023), introducing a set function defined on the ground set of item-position pairs, $\mathcal{U} = \{(i,j) \mid i \in \mathcal{I}, j \in [k]\}$. In this formulation, the selection of a pair (i,j) can be interpreted as assigning item i to position j in the recommendation list. This transformation allows us to leverage subset selection methods while effectively incorporating positional bias inherent in sequencing problems. Below, we first explain how to transform our problem into a subset selection problem, and then discuss how to convert the resulting solution back into a feasible recommendation list.

Phase 1: Transformation and Approximation

Formally, given a set $U \subseteq \mathcal{U}$, we define $\mathcal{I}(U)$ as the set consists of all items $i \in \mathcal{I}$ for which there exists at least one element (i,j) in U for some $j \in [k]$, represented by $\mathcal{I}(U) = \{i \in \mathcal{I} \mid \exists j \in [k], (i,j) \in U\}$. For each $i \in \mathcal{I}(U)$, inspired by (Kleinberg, Ryu, and Éva Tardos 2023; Tang et al. 2024), we define $w(U,i)$ as the weight of the best position assigned to item i, that is, $w(U,i) = \max_{j \in [k]:(i,j) \in U} w_j$. Then we define the set function $f : 2^{\mathcal{U}} \to \mathbb{R}_{\geq 0}$ as follows

$$f(U) = (1-\lambda) \cdot \sum_{i \in \mathcal{I}(U)} s_i + \lambda \cdot \sum_{g \in \mathcal{G}} \sqrt{p(g) \cdot \sum_{i \in \mathcal{I}(U)} w(U,i) \cdot q_i(g)}. \quad (5)$$

A notable property of f is that it is a monotone and submodular function[2], which facilitates the application of advanced optimization techniques to efficiently solve related problems. This property is formally stated in the following lemma, with its proof provided in the appendix.

Lemma 1 *The set function f defined in (5) is a monotone and submodular function.*

To solve **P.0**, we introduce a subset selection problem over $\mathcal{U}$, referred to as **P.1**. We will demonstrate that a solution to **P.1** can be transformed into a feasible solution for our original problem **P.0**. Let $\mathcal{U}^j$ denote the set of all elements from $\mathcal{U}$ that involve position $j \in [k]$, i.e., $\mathcal{U}^j = \{(i,j) \mid i \in \mathcal{I}\}$, and $\mathcal{U}_i$ denote the set of all elements from $\mathcal{U}$ that involve item $i \in \mathcal{I}$, i.e., $\mathcal{U}_i = \{(i,j) \mid j \in [k]\}$.

P.1 $\max_{U \subseteq \mathcal{U}} f(U)$
subject to:
$|U \cap \mathcal{U}^j| \leq 1, \forall j \in [k]$.
$|U \cap (\cup_{i \in \mathcal{I} \setminus \mathcal{I}^s} \mathcal{U}_i)| \leq k - |\mathcal{I}^s|$.

The objective of **P.1** is to find a set $U \subseteq \mathcal{U}$ that maximizes $f(U)$ while satisfying two constraints: 1. U contains at most one element from each position. 2. The total number of elements corresponding to organic items contained in U is at most $k - |\mathcal{I}^s|$. Informally, the first constraint ensures that we assign at most one item to each position. The second constraint sets an upper limit on the total number of organic items that can be selected. By imposing this constraint, we ensure that there is sufficient space remaining for placing the sponsored items alongside the organic ones. To solve **P.1**, it is important to observe that the first set of constraints constitutes a partition matroid constraint[3]; the second set of constraint is also a partition matroid constraint, this can be easily verified by introducing an additional dummy constraint: $|U \cap (\cup_{i \in \mathcal{I}^s} \mathcal{U}_i)| \leq |\cup_{i \in \mathcal{I}^s} \mathcal{U}_i|$. Because f is monotone and submodular (Lemma 1), **P.1** is a submodular maximization problem subject to two matroid constraints. It has been shown that a simple greedy algorithm achieves a $1/3$ approximation ratio (Calinescu et al. 2007). The state-of-the-art algorithm (Lee, Sviridenko, and Vondrák 2010) achieves an approximation ratio of $1/(2+\epsilon)$.

Remark (Kleinberg, Ryu, and Éva Tardos 2023) investigated a special case of our problem, assuming no sponsored items are presented. Therefore, the second set of constraints is not required in their setting. They propose a laminar matroid reduction to ensure the feasibility of a list, contrasting with our partition matroid reduction adopted in the first set of constraints. While their formulation allows for a constant approximation, it requires the set function f to satisfy an additional property called *strongly monotone diminishing return*, in addition to monotonicity and submodularity. Our results, however, do not rely on this property.

The next lemma states that the optimal solution of **P.1** is no less than the optimal solution of the original problem. The proof of this lemma is moved to the appendix.

Lemma 2 *Let U^* denote the optimal solution of **P.1** and π^* denote the optimal solution of the original problem **P.0**, we have*

$$f(U^*) \geq z(\pi^*). \quad (6)$$

[2] A function $h : 2^V \to \mathbb{R}$ is said to be submodular if, for any subsets X and Y of V such that $X \subseteq Y$, and for any item $e \in V \setminus Y$, the following inequality holds: $h(X \cup \{e\}) - h(X) \geq h(Y \cup \{e\}) - h(Y)$. Additionally, a function $h : 2^V \to \mathbb{R}$ is considered monotone if, for any set $X \subseteq V$ and any item $e \in V \setminus X$, it holds that $h(X \cup \{e\}) - h(X) \geq 0$.

[3] A matroid constraint is defined by a finite ground set V and a collection $\mathcal{M}$ of subsets of V. It satisfies the following properties:

- Non-empty: $\emptyset \notin \mathcal{M}$.
- Hereditary: If $A \in \mathcal{M}$ and $B \subseteq A$, then $B \in \mathcal{M}$.
- Exchange property: For any $A, B \in \mathcal{M}$ such that $|A| < |B|$, there exists an element $x \in B \setminus A$ such that $A \cup \{x\} \in \mathcal{M}$.

Algorithm 1: Approximation Algorithm for CRSI

Input: Problem instance **P.0**
Output: Recommendation list π^{P1}
Phase 1: Transformation and Approximation
transform **P.0** to a subset selection problem **P.1**
apply (Lee, Sviridenko, and Vondrák 2010)'s algorithm to solve **P.1** and obtain a set of item-position pairs U^{P1}.
Phase 2: Conversion to a Feasible Recommendation List
$\mathcal{I}(U^{P1}) = \{i \in \mathcal{I} \mid \exists j \in [k], (i,j) \in U^{P1}\}$
for $i \in \mathcal{I}(U^{P1})$ **do**
    $j' \leftarrow \arg\max_{j \in [k]:(i,j) \in U^{P1}} w_j$
    $\pi^{P1}_{j'} \leftarrow i$, that is, assign i to position j' of π^{P1}.
end for
add $\mathcal{I}^s \setminus \pi^{P1}$ to π^{P1} in an arbitrary manner, that is, add all "unlisted" sponsored items, if any, to the list in an arbitrary manner.
fill up all remaining empty slots, if any, of π^{P1} in an arbitrary manner.
return π^{P1}

Phase 2: Conversion to a Feasible Recommendation List

After solving **P.1** to obtain an approximate solution, say U^{P1}, we then transform this solution into a feasible solution for **P.0**. Based on the analysis in this section, we demonstrate that this algorithm achieves a constant factor approximation for **P.0**. Formally, in the second phase, we convert a set of item-position pairs U^{P1} into a feasible list of recommendations π^{P1} for **P.0**. We note that the same item might appear multiple times in U^{P1}. To ensure that each item appears at most once in π^{P1}, for each $i \in \mathcal{I}(U^{P1})$, we assign i to its best position j' according to U^{P1}, i.e., i is assigned to position $j' = \arg\max_{j \in [k]:(i,j) \in U^{P1}} w_j$. However, the current solution might still not be a feasible solution for **P.0**, as there might be some sponsored items, denoted as $S \subseteq \mathcal{I}^s$, that are not included in this solution. In this case, observe that there are at least $|S|$ empty slots in the current list, as U^{P1} satisfies the second constraint of **P.1**, we insert S to the current list in an arbitrary manner. Finally, we fill up any remaining empty slots, if any, in an arbitrary manner and return the final list as π^{P1}.

A detailed implementation of our algorithm is listed in Algorithm 1. It is easy to verify that π^{P1} is a feasible solution of our original problem **P.0**. Firstly, π^{P1} ensures that each position is assigned exactly one item, resulting in a total of k items. Secondly, during the second phase of our algorithm, all sponsored items are explicitly included in π^{P1}. Hence, the following lemma holds.

Lemma 3 *π^{P1} is a feasible solution of **P.0**.*

We next provide an example to demonstrate the step-by-step execution of Algorithm 1.

Example: Consider a scenario where we have a set of items $\mathcal{I} = \{1, 2, 3, 4, 5\}$, out of which items $1, 2$ are sponsored items. The length of the recommendation list is $k = 3$, and the weights are assumed to be ordered such that $w_1 > w_2 > w_3$. In the first phase of the algorithm, we solve **P.1** and obtain the solution $U^{P1} = \{(1,1), (2,2), (2,3)\}$. Moving to the second phase, we convert U^{P1} into a feasible recommendation list. Since $(1,1) \in U^{P1}$, we place item 1 in position 1 of the recommendation list. Next, considering $(2,2)$ and $(2,3)$ present in U^{P1}, we place item 2 in position 2 of the recommendation list. This decision is based on the fact that $w_2 > w_3$, indicating that position 2 is preferable to position 3. Finally, we fill the remaining position in an arbitrary manner using an item, for instance, item 3. As a result, the final recommendation list $\pi^{P1} = \{1, 2, 3\}$.

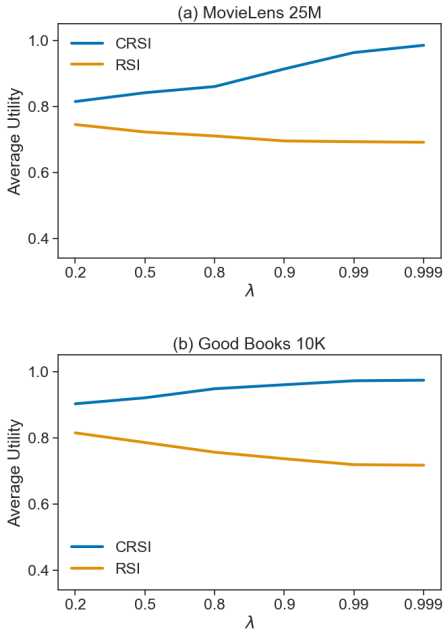

Figure 2: Average utility yielded by CRSI (proposed algorithm) and RSI (benchmark) with respect to the degree of calibration on two large scale datasets: (a) MovieLens 25M, and (b) Good Books 10K.

Performance Analysis

Next, we demonstrate that π^{P1} achieves an approximation ratio of $1/(2+\epsilon)$ for **P.0**. The proof of the following theorem is moved to the appendix.

Theorem 1 *π^{P1} achieves an approximation ratio of $1/(2+\epsilon)$ for **P.0**.*

Performance Evaluation

We evaluate our theoretical results on two large scale datasets, MovieLens 25M and Good Books 10K, that have been widely used for recommender systems evaluation. Various calibration metrics proposed in the literature capture different aspects, and the corresponding calibration algorithms are designed to optimize their specific metric, not necessarily performing well on the other metrics. Therefore,

λ	Algorithm	Util. Mean	Util. Median	Util. Std. Dev.
0.2	CRSI	0.8148	0.8151	0.0020
	RSI	0.7352	0.7357	0.0062
0.5	CRSI	0.8415	0.8422	0.0049
	RSI	0.7225	0.7236	0.0155
0.9	CRSI	0.9136	0.9157	0.0133
	RSI	0.6954	0.6932	0.0311

Table 2: Statistics of the Utility (Util.) Yielded by CRSI vs. RSI on MovieLens 25M Dataset.

λ	Algorithm	Util. Mean	Util. Median	Util. Std. Dev.
0.2	CRSI	0.9027	0.9030	0.0064
	RSI	0.8149	0.8163	0.0210
0.5	CRSI	0.9208	0.9200	0.0075
	RSI	0.7857	0.7892	0.0525
0.9	CRSI	0.9603	0.9598	0.0060
	RSI	0.7367	0.7431	0.0946

Table 3: Statistics of the Utility (Util.) Yielded by CRSI vs. RSI on Good Books 10K Dataset.

we restrict the scope of our experiments to validate the performance of the proposed algorithm with respect to the metric proposed in this work, compared with a benchmark. The proposed calibration algorithm for CRSI and the benchmark are implemented in Python and all experiments are run on a Linux server with Intel Xeon 2.40GHz CPU and 128GB memory.

Experimental Setting

Datasets. We evaluate our algorithm on two large scale datasets, MovieLens 25M and Good Books 10K. MovieLens 25M (Harper and Konstan 2019) is the latest MovieLens dataset released in December 2019 as an extension of the original MovieLens dataset (Harper and Konstan 2015). It records 25 million ratings across $62,423$ movies from $162,541$ users. Among these movies, $54,479$ movies have both listed genres and user ratings. The quality score of each movie can be measured by the average rating across all users or the relevance predicted by the recommender system. Due to space limitation and the results being similar for these approaches, we present the results using the average rating (normalized between 0 and 1) as the quality score. Good Books 10K extended (Simard-Hanley 2021), released in 2021, contains six million ratings for ten thousand most popular (with most ratings) books on Goodreads. As an extension of the original Good Books dataset (Zajac 2019), it records $5,976,479$ ratings across $10,000$ books from $53,424$ users.

Benchmark. We evaluate the proposed algorithm against a benchmark algorithm. Denote the set of the sponsored items as $\mathcal{I}^s$, the number of the sponsored items is denoted as $\eta = |\mathcal{I}^s|$. The benchmark follows a common industry practise by assigning the sponsored items to top η positions in the recommendation list π^{P1}. Then the benchmark fills up the remainder slots by iteratively selecting the items with the largest quality score s_i for $i \in \mathcal{I}$. In our experimental results, the proposed algorithm is marked as CRSI (calibrated recommendation with sponsored items) and the benchmark is marked as RSI (recommendation with sponsored items).

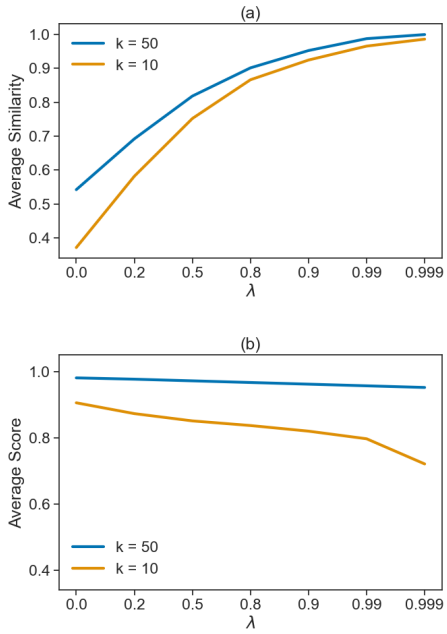

Figure 3: (a) Average similarity in genres distribution between recommended list and the user's play history with respect to the degree of calibration. (b) Average quality score of the recommended list with respect to the degree of calibration. Results are shown for CRSI on MovieLens 25M dataset.

Parameter Settings. For each user u, we obtain the target distribution, $p_u(g)$, that captures the user's preference over genres $g \in \mathcal{G}$, from the user's rating data. Here $\mathcal{G}$ denotes the set of all genres in the dataset. Denote by H_u the set of movies rated by a user u, we define $p_u(g) = \sum_{i \in H_u} p_i(g)/|H_u|$, where $p_i(g)$ captures the distribution of genres $g \in \mathcal{G}$ for each movie i. For movies listed with multiple genres, we assign equal weights to each genre listed for the movie. This is used for calculating the user target distribution and the genres distribution of a recommended list of movies. In specific, the distribution of genres $g \in \mathcal{G}$ for a movie i is calculated as $p_i(g) = 1/l$ where l is the number of genres listed for movie i.

In our experiments, to compute the distribution of genres for a given recommended list $\pi = \{\pi_1, \pi_2, \cdots, \pi_t\}$, we assign the weight w_j associated with position j in the list as $1/j$ and then normalize w_j so that $\sum_{j=1}^{t} w_j = 1$.

Experimental Results

Recall that utility function $z(\pi)$, which is defined in Eq. (4), serves as the evaluation metric to assess the overall performance of a recommendation list. Intuitively, a higher value of $z(\pi)$ indicates a better recommendation list, as it balances both quality and calibration. In our experiment, to better un-

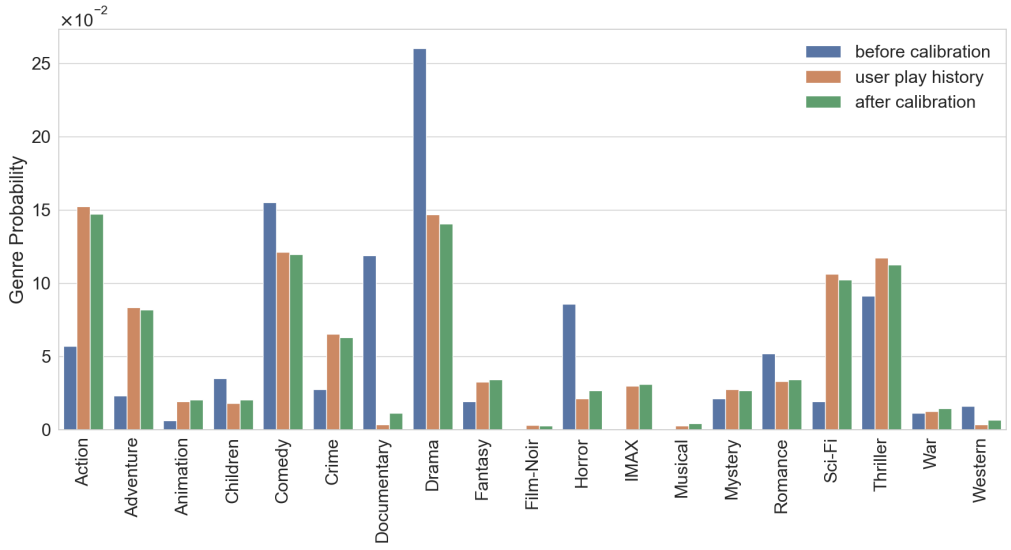

Figure 4: Genres distribution of recommended list before and after calibration compared with the user's play history.

derstand the performance of the recommendation algorithm, we provide fine-grained results by reporting not only the overall performance $z(\pi)$ but also the values of each individual component (e.g., the quality score and the similarity). Specifically, we evaluate the performance of the calibration algorithm under various settings of the degree of calibration, size of the recommended list, as well as the number of sponsored items. We also report the average difference in genre's proportion between the recommended list and the user's play history, while various numbers of sponsored items are considered. Finally we present the running time of the algorithm under various sizes of the recommended list.

Comparison of CRSI and RSI Across Different λ. In this set of experiments, we evaluate the average utility $z(\pi)$ of the proposed algorithm (CRSI) and that of the benchmark (RSI), with respect to changes in the value of λ. As shown in Figure 2, we observe that CRSI outperforms the benchmark by more than 10% across all tested cases on both datasets. We summarize the statistics of the utility yielded by both algorithms over 100 test users in Table 2 for MovieLens 25M and in Table 3 for Good Books 10K.

Impact of λ on Similarity-Quality Tradeoff. In the second set of experiments, we evaluate the performance of the calibration algorithm under varying degrees of calibration. We randomly select 100 users as our test users, and measure the performance of the algorithm averaged over all test users in terms of 1) the similarity in genres distribution between the recommended list and the user's play history (*Average Similarity*), and 2) the average quality score of the movies recommended to the user (*Average Score*). Figure 3(a) illustrates the average similarity with respect to changes in the value of parameter λ. We consider the size of the recommended list to be $k = 10$ and $k = 50$, respectively, with 3 sponsored movies. As shown in the figure, the average similarity increases as the value of λ increases. This verifies that a better calibration can be achieved by increasing the value of λ, compared to the baseline when $\lambda = 0$. We also observe that the genres distribution is better calibrated based on 50 rather than 10 recommended movies, but both achieve an average similarity of over 0.9 when $\lambda = 0.999$. To illustrate the effect of calibration with sponsored items, we plot in Figure 4 the genres distribution of 50 recommended movies including 3 sponsored movies when $\lambda = 0.999$. As shown in the figure, even with the presence of sponsored movies, the genres distribution of the recommended list aligns well with the user's play history after calibration.

Figure 3(b) shows the average quality score of the movies recommended to the test users. We scale the quality score of each movie in the range of $[0, 1]$. For instance, if a movie has an average rating of 4.5 out of 5, its quality score is recorded as $4.5/5 = 0.9$. We observe that a better calibration comes with the price of a decreased quality score, but for small values of λ, calibration can be improved significantly with a slight change in the quality score.

- For $k = 10$, when $\lambda = 0.8$, the algorithm yields an average similarity of 0.866 with an average quality score of 0.837.
- For $k = 50$, when $\lambda = 0.99$, the algorithm yields an average similarity of 0.987 with an average quality score of 0.957.

In practice, the following guidelines can be used to select an appropriate λ.

1. When k is large (i.e., when a large number of items can be displayed), we can select a larger λ to maintain both a high similarity and quality score. As seen in Fig. 3, when $k = 50$, the quality score of the recommendation remains relatively stable, maintaining a value of at least 0.95 across all λ. This suggests that when we can display a sufficient number of items, we have enough flexibility

to achieve a high quality score regardless of the value of λ. In this case, the focus can shift to maximizing similarity by choosing a larger λ.

2. When k is small (i.e., when only a few items can be displayed), careful selection of λ is necessary. In Fig. 3, when $k = 10$, a clear tradeoff between similarity and quality score emerges as λ varies. One potential strategy is to adopt a threshold-based approach. For example, the platform owner could specify a minimum quality score requirement that the recommendation system must meet. In this case, λ can be chosen to maximize similarity while ensuring the quality score meets the specified threshold. For instance, with $k = 10$ and a minimum average quality score of 0.85, Fig. 3 (b) suggests that $\lambda = 0.5$ would be the optimal choice.

Impact of Sponsored Items. In this set of experiments, we vary the size of $\mathcal{I}^s$, the set of sponsored items, to test its impact on the performance of the calibration algorithm. Given a particular size m, we randomly select m items from the whole collection as a sponsored set. Figure 6(a) (now moved to appendix) shows the average similarity yielded by the algorithm CRSI as m increases from 0 to 5 for MovieLens 25M. We consider $k = 10$ and $k = 50$ with $\lambda = 0$ (before calibration) and $\lambda = 0.999$ (after calibration). We observe that with strong calibration, the introduction of sponsored items leads to a slight change in the average similarity. It is interesting to observe that before calibration, as more sponsored movies are recommended, the resulting genres distribution aligns better with the user's play history. The underlying reason is that without calibration, the recommended list consists of movies with highest average ratings that typically belong to a limited set of genres. As sponsored movies are introduced, the recommended list shows an increased diversity in terms of genres, leading to an improved alignment with the user's play history on average. Figure 6(b) (now moved to appendix) shows the similarity yielded by the algorithm CRSI as m increases from 0 to 5 for Good Books 10K. As expected, with strong calibration, the introduction of sponsored items leads to a slight change in the average similarity. This implies that our proposed algorithm is robust to various sizes of sponsored items in terms of the achieved similarity between the recommended list and the users read/play history. We summarize the statistics of the similarity obtained by CRSI with $\lambda = 0.999$ (marked as CRSI) and CRSI with $\lambda = 0$ (marked as score-only) over 100 test users in Table 8 for MovieLens 25M and in Table 9 for Good Books 10K (now moved to the appendix).

Table 5, 6 and 7 (now moved to the appendix) present the calibration effectiveness across all genres for recommended lists of size 50 with 5, 10 and 15 sponsored movies, respectively. The calibration parameter is set to $\lambda = 0.99$. We measure the effectiveness by the difference in each genre's distribution between the recommended list and the user's play history, compared to the baseline of $\lambda = 0$ (*before calibration*). We report positive differences (over-represented cases) and negative differences (under-represented cases), respectively, averaged over all test users. We observe that, with the presence of sponsored movies, for each and every genre, the calibrated recommended list aligns well with the user's play history.

# Recommendations	# Sponsored Items				
	0	5	10	15	20
50	5.968	4.749	3.992	3.345	2.707
80	16.529	14.885	13.414	12.136	10.680
100	24.354	22.524	20.798	19.676	17.482

Table 4: Running Time for Calibration Algorithm (in mins).

- In particular, before calibration, several genres are under-represented (eg. Adventure, by 69.1%) or over-represented (eg. Horror, by 190.7%).
- After calibration, even for $m = 15$, the difference in the genre's probability across all genres is less than 0.007. This again verifies the effectiveness of our algorithm for calibrated recommendation with sponsored items.

Running Time. We test the running time of the proposed algorithm for various lengths of the recommended list. Table 4 presents the running time of the proposed algorithm for $k = 50, 80, 100$, respectively, all finished within 30 minutes.

Extension: Randomized Recommendation

Recall that the utility of a deterministic recommendation π is

$$(1-\lambda)\cdot\sum_{i\in\mathcal{I}(\pi)} s_i + \lambda\cdot\sum_{g\in\mathcal{G}}\sqrt{p(g)\cdot\sum_{j\in[k]} w_j\cdot q_{\pi_j}(g)}. \quad (7)$$

Let $x \in [0,1]^{\Pi}$ denote a randomized solution such that $x_\pi \in [0,1]$ represents the selection probability of a recommendation list $\pi \in \Pi$. Then the recommendation distribution over classes, denoted as $q(x)$, for a random solution x is defined as follows:

$$q(x)(g) = \sum_{\pi\in\Pi} x_\pi\Big(\sum_{j=1}^{k} w_j\cdot q_{\pi_j}(g)\Big) \quad (8)$$

where w_j is the weight of position j, and we assume that $w_j \geq 1$.

Given a target distribution p, by abusing of notation, we measure the utility of a randomized solution using a utility function $z : [0,1]^{\Pi} \to \mathbb{R}_{\geq 0}$ defined as follows:

$$z(x) = (1-\lambda)\cdot\sum_{\pi\in\Pi} x_\pi\Big(\sum_{i\in\mathcal{I}(\pi)} s_i\Big) + \lambda\cdot\sum_{g\in\mathcal{G}}\sqrt{p(g)\cdot q(x)(g)}.$$

Our objective is to find a randomized solution, denoted as $x \in [0,1]^{\Pi}$, satisfying the constraint $\sum_{\pi\in\Pi} x_\pi \leq 1$, with the goal of maximizing the objective function $z(x)$. A formal description of our problem is listed in **P.A**.

P.A $\max_{x\in[0,1]^{\Pi}} z(x)$ **subject to:** $\sum_{\pi\in\Pi} x_\pi \leq 1$.

We first tackle a simplified version of problem **P.A**, where we make the assumption that the value of $\sqrt{q(x^*)(g)}$ is already known. Here, x^* represents the optimal solution of **P.A**. Later, we remove this assumption by employing an iterative approach to estimate the value of $\sqrt{q(x^*)(g)}$. All omitted proofs in this section have been relocated to the appendix.

Optimal Algorithm with Known $\sqrt{q(x^*)(g)}$

Let $\alpha_g = \sqrt{q(x^*)(g)}$ for all $g \in \mathcal{G}$. Assuming that the values of α_g are given, we introduce an optimization problem denoted as **P.B**.

P.B $\max_{x\in[0,1]^\Pi} \sum_{\pi\in\Pi} x_\pi(\sum_{i\in\mathcal{I}(\pi)} s_i)$
subject to: $\sqrt{q(x)(g)} \geq \alpha_g, \forall g \in \mathcal{G}$ and $\sum_{\pi\in\Pi} x_\pi \leq 1$.

The next lemma states that the optimal solution of **P.B** is also an optimal solution of **P.A**.

Lemma 4 *Let x^B be an optimal solution to problem **P.B**, we have $z(x^B) \geq z(x^*)$ where x^* is the optimal solution of **P.A**.*

Ellipsoid-based Method By the definition of $\sqrt{q(x)(g)}$ (as listed in (8)), **P.B** can be rewritten as follows:

P.B $\max_{x\in[0,1]^\Pi} \sum_{\pi\in\Pi} x_\pi(\sum_{i\in\mathcal{I}(\pi)} s_i)$
subject to:
$\sum_{\pi\in\Pi} x_\pi(\sum_{j=1}^{k} w_j \cdot q_{\pi_j}(g)) \geq \alpha_g^2, \forall g \in \mathcal{G}$.
$\sum_{\pi\in\Pi} x_\pi \leq 1$.

It is important to emphasize that the number of elements in Π, and consequently the number of variables in **P.B**, can increase exponentially with respect to the number of items n. As a result, standard linear programming (LP) solvers can not solve this LP efficiently. To overcome this challenge, we can consider the dual problem of **P.B**.

The dual problem of **P.B**, denoted as **Dual of P.B**, introduces a weight $y_g \in \mathbb{R}\geq 0$ for each class $g \in \mathcal{G}$. Additionally, an extra variable $d \in \mathbb{R}_{\geq 0}$ is introduced in this dual formulation.

Dual of P.B $\min_{y\in[0,1]^\mathcal{G}, d\in\mathbb{R}_{\geq 0}} \sum_{g\in\mathcal{G}} -\alpha_g^2 \cdot y_g + d$
subject to:
$-\sum_{g\in\mathcal{G}}(\sum_{j=1}^{k} w_j \cdot q_{\pi_j}(g)) \cdot y_g + d \geq \sum_{i\in\mathcal{I}(\pi)} s_i, \forall \pi \in \Pi$.

We should note that the number of constraints in **Dual of P.B** may grow exponentially with respect to n. To address this, we employ the ellipsoid algorithm (Grötschel, Lovász, and Schrijver 1981) to simplify the problem by reducing the number of constraints to a polynomial amount while preserving the optimal solution.

Prior to presenting the algorithm for **Dual of P.B**, we introduce an important combinatorial optimization problem called MAX-WEIGHT-RECSYS. The solution to this problem will play a crucial role as a subroutine within our final algorithm.

Definition 1 (Max-Weight-RecSys) *Given a vector $y \in [0,1]^\mathcal{G}$, the problem* MAX-WEIGHT-RECSYS(y) *is listed as follows:*

$$\max_{\pi\in\Pi} \sum_{g\in\mathcal{G}}(\sum_{j=1}^{k} w_j \cdot q_{\pi_j}(g)) \cdot y_g + \sum_{i\in\mathcal{I}(\pi)} s_i. \tag{9}$$

I.e., the problem MAX-WEIGHT-RECSYS(y) *aims to find a feasible $\pi \in \Pi$ that maximizes the objective function $\sum_{g\in\mathcal{G}} \left(\sum_{j=1}^{k} w_j \cdot q_{\pi_j}(g)\right) \cdot y_g + \sum_{i\in\mathcal{I}(\pi)} s_i$.*

In Section , we present an optimal solution for the above problem. Now assuming the problem MAX-WEIGHT-RECSYS(y) can be solved optimally for every $y \in [0,1]^\mathcal{G}$, we will present our algorithm for **Dual of P.B**, which uses the solution to the MAX-WEIGHT-RECSYS problem as a separation oracle to determine whether a given solution lies inside or outside the feasible region.

Let $C(L)$ denote the set of $(y \in [0,1]^\mathcal{G}, d \in \mathbb{R}_{\geq 0})$ satisfying the following conditions:

$$\sum_{g\in\mathcal{G}} -\alpha_g^2 \cdot y_g + d \leq L,$$

$$d \geq \sum_{g\in\mathcal{G}}(\sum_{j=1}^{k} w_j \cdot q_{\pi_j}(g)) \cdot y_g + \sum_{i\in\mathcal{I}(\pi)} s_i, \quad \forall \pi \in \Pi.$$

One can verify that L is attainable with respect to **Dual of P.B** if and only if $C(L)$ is non-empty. Next, following a common practice as outlined by Tang et al. (2023), we employ a binary search method to determine the minimum value of L for which $C(L)$ is non-empty. Given a value of L and the tuple (z, w), we first evaluate the inequality $\sum_{g\in\mathcal{G}} -\alpha_g^2 \cdot y_g + d \leq L$. If this inequality is satisfied, the algorithm proceeds to a subroutine $\mathcal{A}$ to solve the MAX-WEIGHT-RECSYS(y) problem. Recall that $\mathcal{A}$ identifies the feasible subset $\pi \in \Pi$ that maximizes $\sum_{g\in\mathcal{G}}(\sum_{j=1}^{k} w_j \cdot q_{\pi_j}(g)) \cdot y_g + \sum_{i\in\mathcal{I}(\pi)} s_i$. Let $\pi^\mathcal{A}$ denote the list returned by $\mathcal{A}$.

1. If $\sum_{g\in\mathcal{G}}(\sum_{j=1}^{k} w_j \cdot q_{\pi_j^\mathcal{A}}(g)) \cdot y_g + \sum_{i\in\mathcal{I}(\pi^\mathcal{A})} s_i \leq d$, it implies that $d \geq \sum_{g\in\mathcal{G}}(\sum_{j=1}^{k} w_j \cdot q_{\pi_j}(g)) \cdot y_g + \sum_{i\in\mathcal{I}(\pi)} s_i$ for all $\pi \in \Pi$, as $\pi^\mathcal{A}$ represents the optimal solution of MAX-WEIGHT-RECSYS(y). In this case, we mark $C(L)$ as a non-empty set and proceed to explore a smaller value of L.

2. Otherwise, if $(y, d) \notin C(L)$, the list $\pi^\mathcal{A}$ acts as a separating hyperplane. To advance the optimization process, we identify a smaller ellipsoid whose center adheres to this constraint. This procedure is repeated iteratively: either we find a feasible solution within $C(L)$, allowing us to attempt a smaller L, or the volume of the bounding ellipsoid diminishes to such an extent that it can be considered empty in relation to $C(L)$. In this latter case, we determine that the current objective is unattainable and subsequently attempt a larger L.

We will now examine the performance of this algorithm. Let $L^* - \epsilon$ be the maximum value of L for which the algorithm determines that $C(L)$ is empty, where ϵ represents the precision of the binary search. Our goal is to find a solution to the original problem **P.A**. Following the standard approach in the ellipsoid method (Tang and Yuan 2023), we will focus solely on the feasible lists from Π that correspond to the separating hyperplanes identified by the separation oracle. To achieve this, we define a subset of Π, denoted as Π', which consists of all feasible lists where the dual constraint is violated during the implementation of the ellipsoid algorithm on $C(L^* - \epsilon)$. The size of Π' is polynomial since the dual constraints are violated for only a polynomial number of feasible lists. Using Π', we can construct a polynomial-sized linear program for **P.A**, referred to as **Poly-sized P.A**.

Poly-sized P.A $\max_{x\in[0,1]^{\Pi'}} z(x)$ **subject to:** $\sum_{\pi\in\Pi'} x_\pi \leq 1$.

Given that the size of **Poly-sized P.A** is polynomial, we can efficiently solve it and obtain an optimal solution for the original problem **P.A**.

Note: For a comprehensive understanding of the step-by-step process involved in running the ellipsoid algorithm with separation oracles and achieving approximate guarantees (both multiplicative and additive), we suggest referring to Chapter 2 of (Bubeck et al. 2015).

Optimal Solution for MAX-WEIGHT-RECSYS It is easy to verify that MAX-WEIGHT-RECSYS(y) can be rewritten as

$$\max_{\pi\in\Pi} \sum_{j=1}^{k} (\sum_{g\in\mathcal{G}} w_j \cdot q_{\pi_j}(g) \cdot y_g + s_{\pi_j}). \tag{10}$$

We next demonstrate that MAX-WEIGHT-RECSYS(y), for any $y \in [0,1]^{\mathcal{G}}$, can be transformed into a maximum weighted perfect matching problem on a bipartite graph, which can be solved efficiently in polynomial time (Schrijver et al. 2003).

Given a MAX-WEIGHT-RECSYS(y) problem, we construct a bipartite graph G as follows: G consists of two sets of nodes, denoted as $\mathcal{I}$ and $\mathcal{N}$, with each set containing $|\mathcal{I}|$ nodes. Intuitively, mapping a node $i \in \mathcal{I}$ to a node $j \in \mathcal{N}$ corresponds to assigning item i to position j in the world of recommendation. Notably, there are at most k positions in the recommendation list. Therefore, only the first k nodes in $\mathcal{N}$ correspond to actual positions, while the remaining $|\mathcal{I}| - k$ nodes are considered dummy positions. Assigning an item to a node in $\mathcal{N}$ beyond the first k nodes indicates that the item is not included in the recommendation list at any of the actual positions.

We next explain how to add edges and determine their weights in graph G. Let $\mathcal{N}'$ represent the set of the first k nodes in $\mathcal{N}$. For each organic item $i \in \mathcal{I} \setminus \mathcal{I}^s$ and every node $j \in \mathcal{N}$, we introduce an edge between i and j. The weight of the edge (i,j) is defined as follows:

$$\text{weight}(i,j) = \begin{cases} \sum_{g\in\mathcal{G}} w_j q_i(g) y_g + s_i & \text{if } i \in \mathcal{N}' \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

That is, when i is a organic item and j represents an actual position, the weight of the edge (i,j) is determined by the utility of assigning item i to position j, as specified in the objective function of (10). However, if j corresponds to a dummy position, the weight of the edge (i,j) is set to zero.

For each sponsored item $i \in \mathcal{I}^s$ and every node $j \in \mathcal{N}'$, we introduce an edge between i and j. The weight of the edge (i,j) is defined as follows:

$$\text{weight}(i,j) = \sum_{g\in\mathcal{G}} w_j \cdot q_i(g) \cdot y_g + s_i \tag{12}$$

That is, when i is a sponsored item, we only introduce an edge between i and the nodes in $\mathcal{N}'$ (the actual positions in a list of recommendations). The weight of the edge (i,j) is determined by the utility of assigning item i to position j, as specified in the objective function of (10). In other words, there are no edges between items $i \in \mathcal{I}^s$ (sponsored items) and the nodes corresponding to dummy positions in G.

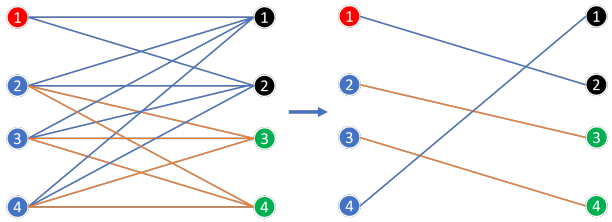

Figure 5: An example illustrating the Maximum Weight Perfect Matching problem. In our original problem, we have a set of four items $\mathcal{I} = \{1,2,3,4\}$, where item 1 is the only sponsored item ($\mathcal{I}^s = \{1\}$), and there are $k = 2$ available slots. The left figure depicts the corresponding perfect matching problem, where the red node represents the sponsored item, blue nodes represent organic items, black nodes represent the $k = 2$ actual positions, and green nodes represent two dummy positions. The brown lines indicate edges with zero weight. The right figure shows a potential outcome after solving the Maximum Weight Perfect Matching problem. Using this result, we can construct a recommendation list by assigning item 4 to the first slot and item 1 (the sponsored item) to the second slot.

Now we are ready to formally introduce the maximum weight perfect matching problem. Please refer to Figure 5 for an example of this problem.

Definition 2 (Max-Matching) *Given an instance of* MAX-WEIGHT-RECSYS(y) *and the corresponding bipartite graph G as described above, the objective of the maximum weight perfect matching problem (labeled as* MAX-MATCHING(G)*) is to find a maximum weight perfect matching M in G. Here we say M is a perfect matching if every node in G is adjacent to exactly one edge in M. In other words,* MAX-MATCHING(G) *seeks to find a perfect matching in G that maximizes the sum of the weights of the matched edges, where the weight of an edge corresponds to the utility of assigning an item to a position in the recommendation list.*

Lemma 5 *To find an optimal solution for* MAX-WEIGHT-RECSYS(y) *for any $y \in [0,1]^{\mathcal{G}}$, it suffices to solve its corresponding maximum weight perfect matching problem* MAX-MATCHING(G) *optimally.*

Proof: Given MAX-WEIGHT-RECSYS(y) and its corresponding bipartite graph G, let M^* denote an optimal solution of MAX-MATCHING(G). Then we build a list of recommendations π' such that we place item $i \in \mathcal{I}$ in position $j \in [k]$ of π' if and only if $(i,j) \in M^*$. We next show that π' must be an optimal solution of MAX-WEIGHT-RECSYS(y), that is, $\pi' \in \arg\max_{\pi\in\Pi} \sum_{j=1}^{k} (\sum_{g\in\mathcal{G}} w_j \cdot q_{\pi_j}(g) \cdot y_g + s_{\pi_j})$.

First, we demonstrate that π' is a feasible solution, i.e., $\pi' \in \Pi$. As M^* is a perfect matching in G, each item $i \in \mathcal{I}$ is assigned to a unique location, and no item appears in multiple locations. Additionally, since there are no edges between

sponsored items and dummy positions, all sponsored items must be assigned to positions in $\mathcal{N}'$ (the actual k positions).

Second, we prove that if there exists a different list π'' such that

$$\sum_{j=1}^{k}(\sum_{g\in\mathcal{G}} w_j \cdot q_{\pi''_j}(g)\cdot y_g + s_{\pi''_j}) > \sum_{j=1}^{k}(\sum_{g\in\mathcal{G}} w_j \cdot q_{\pi'_j}(g)\cdot y_g + s_{\pi'_j}), \quad (13)$$

then there must exist a perfect matching M with a weight larger than M^*, contradicting the assumption that M^* is a maximum weight perfect matching. This, in turn, shows that π' is an optimal solution for the MAX-WEIGHT-RECSYS(y) problem.

Assuming such π'' exists, we construct a perfect matching M as follows: we map item $i \in \mathcal{I}$ to position $j \in [k]$ only if item i is placed in the j-th position of π''; then M assigns the remaining $|\mathcal{I}| - k$ items in $\mathcal{I} \setminus \mathcal{I}(\pi'')$ to $|\mathcal{I}| - k$ dummy positions in an arbitrary manner. It can be easily verified that M is a perfect matching in G because, given that $\pi'' \in \Pi$, every sponsored item is mapped to a unique node in $\mathcal{N}'$, and all organic items can find a perfect matching in $\mathcal{N}$. We will now demonstrate that the weight of M is greater than that of M^*. Note that both M and M^* assign $|\mathcal{I}| - k$ organic items to dummy positions, which contribute zero weight to the matching. Thus, we only need to compute the weight of edges adjacent to $\mathcal{N}'$. According to the definition of $\text{weight}(i, j)$ (listed in (11) and (12)), the total weight of all edges from M adjacent to $\mathcal{N}'$ is $\sum_{j=1}^{k}(\sum_{g\in\mathcal{G}} w_j \cdot q_{\pi''_j}(g)\cdot y_g + s_{\pi''_j})$, while the total weight of all edges from M^* adjacent to $\mathcal{N}'$ is $\sum_{j=1}^{k}(\sum_{g\in\mathcal{G}} w_j \cdot q_{\pi'_j}(g)\cdot y_g + s_{\pi'_j})$. Based on inequality (13), the weight of M is greater than the weight of M^*. □

Approximation Algorithm for P.A

The previous algorithm assumes prior knowledge of the values of $\alpha_g = \sqrt{q(x^*(g))}$ for all $g \in \mathcal{G}$. However, since these values are unknown, we can approximate them by sampling a few values, allowing us to approach arbitrarily close to the optimal solution.

Our algorithm proceeds as follows. It assumes that the value of α_g is unknown and aims to approximate it.

1. Initialize a set V of threshold values
$$V = \{l \cdot \epsilon \mid l \in \{0, 1, 2, \cdots, 1/\epsilon\}\}. \quad (14)$$
2. For each value of $o \in V^{\mathcal{G}}$: Solve **P.B** using $o_g \in V$ as the estimated value of α_g, to obtain a solution x^o.
3. Return the solution with the highest objective function value among all estimates $\{x^o \mid o \in V^{\mathcal{G}}\}$.

Note that the above algorithms require solving **P.B** for every value in $V^{\mathcal{G}}$, which results in invoking the ellipsoid method $\left(\frac{1}{\epsilon}\right)^m$ times. By iterating over different threshold values in $V^{\mathcal{G}}$, the algorithm is guaranteed to find a threshold that provides a reasonably accurate estimate of α_g for all $g \in \mathcal{G}$. More formally, there should exist at least one threshold $\tilde{\alpha} \in V^{\mathcal{G}}$ that satisfies

$$\forall g \in \mathcal{G} : \alpha_g - \epsilon \leq \tilde{\alpha}_g \leq \alpha_g. \quad (15)$$

By substituting the estimated value $\tilde{\alpha}$ for α in **P.B**, we derive a modified optimization problem denoted as **P.B.1**.

P.B.1 $\max_{x\in[0,1]^{\Pi}} \sum_{\pi\in\Pi} x_\pi(\sum_{i\in\mathcal{I}(\pi)} s_i)$
subject to: $\sqrt{q(x)(g)} \geq \tilde{\alpha}_g, \forall g \in \mathcal{G}$ and $\sum_{\pi\in\Pi} x_\pi \leq 1$.

We will now demonstrate that solving the problem **P.B.1** provides an approximate solution to the original problem **P.A**. This performance bound also applies to our algorithm, as it returns the best solution with the highest objective function value among all estimates, including the estimated threshold $\tilde{\alpha}$.

Lemma 6 *Let $\tilde{x}^B$ denote the optimal solution of **P.B.1**, we have $z(\tilde{x}^B) \geq z(x^*) - \epsilon\lambda\sqrt{m}$ where x^* is the optimal solution of **P.A** and m is the size of $\mathcal{G}$.*

Recall that our algorithm invokes the ellipsoid method $\left(\frac{1}{\epsilon}\right)^m$ times. By selecting an appropriate value of ϵ, we can balance the tradeoff between performance and complexity.

Conclusion

In this paper, we extend existing studies on CRS by incorporating the presence of sponsored items. Our goal is to compute a calibrated recommendation list that adheres to the mandatory inclusion of sponsored items. We develop effective algorithms to address this problem and provide theoretical performance bounds for the proposed solutions. There are two directions we aim to explore in the future: 1. We plan to incorporate more realistic constraints on the placement of sponsored items. In this paper, we assume that sponsored items can be placed in any position without restrictions. However, in practice, advertisers often impose specific placement requirements, such as ensuring their items appear within the top positions or avoiding certain slots altogether. 2. Currently, we assume that all sponsored items are provided as input. It would be interesting to explore a setting where we jointly optimize the selection of sponsored items and the allocation of organic items to further enhance performance. 3. In this paper, we assume that the target distribution is pre-specified. However, such information may not be readily available for certain users, such as new users. To address this, one could leverage active learning to actively infer the target distribution from real-time user interactions. Balancing exploration and exploitation in this process could naturally be formulated as a reinforcement learning problem.

Acknowledgements

This work was supported in part by CAHSI-Google institutional Research Program.

References

Agrawal, R.; Gollapudi, S.; Halverson, A.; and Ieong, S. 2009. Diversifying search results. In *Proceedings of the second ACM international conference on web search and data mining*, 5–14.

Ashkan, A.; Kveton, B.; Berkovsky, S.; and Wen, Z. 2015. Optimal Greedy Diversity for Recommendation. In *IJCAI*, volume 15, 1742–1748.

Bubeck, S.; et al. 2015. Convex optimization: Algorithms and complexity. *Foundations and Trends® in Machine Learning*, 8(3-4): 231–357.

Calinescu, G.; Chekuri, C.; Pál, M.; and Vondrák, J. 2007. Maximizing a submodular set function subject to a matroid constraint. In *International Conference on Integer Programming and Combinatorial Optimization*, 182–196. Springer.

da Silva, D. C.; and Durão, F. A. 2022. Introducing a Framework and a Decision Protocol to Calibrate Recommender Systems. *arXiv preprint arXiv:2204.03706*.

Grötschel, M.; Lovász, L.; and Schrijver, A. 1981. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1(2): 169–197.

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

Harper, F. M.; and Konstan, J. A. 2019. https://grouplens.org/datasets/movielens/25m.

Kleinberg, J.; Ryu, E.; and Éva Tardos. 2023. Calibrated Recommendations for Users with Decaying Attention. arXiv:2302.03239.

Lee, J.; Sviridenko, M.; and Vondrák, J. 2010. Submodular maximization over multiple matroids via generalized exchange properties. *Mathematics of Operations Research*, 35(4): 795–806.

Liao, G.; Wang, Z.; Wu, X.; Shi, X.; Zhang, C.; Wang, Y.; Wang, X.; and Wang, D. 2022. Cross dqn: Cross deep q network for ads allocation in feed. In *Proceedings of the ACM Web Conference 2022*, 401–409.

Malthouse, E. C.; Hessary, Y. K.; Vakeel, K. A.; Burke, R.; and Fudurić, M. 2019. An algorithm for allocating sponsored recommendations and content: Unifying programmatic advertising and recommender systems. *Journal of Advertising*, 48(4): 366–379.

Naghiaei, M.; Rahmani, H. A.; Aliannejadi, M.; and Sonboli, N. 2022. Towards Confidence-aware Calibrated Recommendation. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, 4344–4348.

Schrijver, A.; et al. 2003. *Combinatorial optimization: polyhedra and efficiency*, volume 24. Springer.

Seymen, S.; Abdollahpouri, H.; and Malthouse, E. C. 2021. A constrained optimization approach for calibrated recommendations. In *Proceedings of the 15th ACM Conference on Recommender Systems*, 607–612.

Simard-Hanley, O. 2021. https://github.com/malcolmosh/goodbooks-10k-extended.

Steck, H. 2018. Calibrated recommendations. In *Proceedings of the 12th ACM conference on recommender systems*, 154–162.

Tang, S.; Cai, S.; Yuan, J.; and Han, K. 2024. Assortment Planning with Sponsored Products. In *The 30th International Computing and Combinatorics Conference*.

Tang, S.; and Yuan, J. 2023. Beyond submodularity: a unified framework of randomized set selection with group fairness constraints. *Journal of Combinatorial Optimization*, 45(4): 102.

Wang, H.; Tu, S.; and Gionis, A. 2025. Sequential Diversification with Provable Guarantees. In *Proceedings of the Eighteenth ACM International Conference on Web Search and Data Mining*, 345–353.

Yan, J.; Xu, Z.; Tiwana, B.; and Chatterjee, S. 2020. Ads allocation in feed via constrained optimization. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 3386–3394.

Zajac, Z. 2019. https://github.com/zygmuntz/goodbooks-10k.

Paper Checklist

1. For most authors...
   (a) Would answering this research question advance science without violating social contracts, such as violating privacy norms, perpetuating unfair profiling, exacerbating the socio-economic divide, or implying disrespect to societies or cultures? Yes
   (b) Do your main claims in the abstract and introduction accurately reflect the paper's contributions and scope? Yes
   (c) Do you clarify how the proposed methodological approach is appropriate for the claims made? Yes
   (d) Do you clarify what are possible artifacts in the data used, given population-specific distributions? N/A
   (e) Did you describe the limitations of your work? Yes
   (f) Did you discuss any potential negative societal impacts of your work? N/A
   (g) Did you discuss any potential misuse of your work? N/A
   (h) Did you describe steps taken to prevent or mitigate potential negative outcomes of the research, such as data and model documentation, data anonymization, responsible release, access control, and the reproducibility of findings? N/A
   (i) Have you read the ethics review guidelines and ensured that your paper conforms to them? Yes
2. Additionally, if your study involves hypotheses testing...
   (a) Did you clearly state the assumptions underlying all theoretical results? N/A
   (b) Have you provided justifications for all theoretical results? N/A
   (c) Did you discuss competing hypotheses or theories that might challenge or complement your theoretical results? N/A
   (d) Have you considered alternative mechanisms or explanations that might account for the same outcomes observed in your study? N/A
   (e) Did you address potential biases or limitations in your theoretical framework? N/A
   (f) Have you related your theoretical results to the existing literature in social science? N/A
   (g) Did you discuss the implications of your theoretical results for policy, practice, or further research in the social science domain? N/A
3. Additionally, if you are including theoretical proofs...
   (a) Did you state the full set of assumptions of all theoretical results? Yes
   (b) Did you include complete proofs of all theoretical results? Yes
4. Additionally, if you ran machine learning experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? N/A
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? N/A
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? N/A
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? N/A
   (e) Do you justify how the proposed evaluation is sufficient and appropriate to the claims made? N/A
   (f) Do you discuss what is "the cost" of misclassification and fault (in)tolerance? N/A
5. Additionally, if you are using existing assets (e.g., code, data, models) or curating/releasing new assets, **without compromising anonymity**...
   (a) If your work uses existing assets, did you cite the creators? N/A
   (b) Did you mention the license of the assets? N/A
   (c) Did you include any new assets in the supplemental material or as a URL? N/A
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? N/A
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? N/A
   (f) If you are curating or releasing new datasets, did you discuss how you intend to make your datasets FAIR? N/A
   (g) If you are curating or releasing new datasets, did you create a Datasheet for the Dataset? N/A
6. Additionally, if you used crowdsourcing or conducted research with human subjects, **without compromising anonymity**...
   (a) Did you include the full text of instructions given to participants and screenshots? N/A
   (b) Did you describe any potential participant risks, with mentions of Institutional Review Board (IRB) approvals? N/A
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? N/A
   (d) Did you discuss how data is stored, shared, and deidentified? N/A