Calibrated Recommendations as a Minimum-Cost Flow Problem

Himan Abdollahpouri
Spotify
New York, USA
himana@spotify.com

Zahra Nazari
Spotify
New York, USA
zahran@spotify.com

Alex Gain
Spotify
New York, USA
again@spotify.com

Clay Gibson
Airbnb
San Francisco, USA
clay.gibson@airbnb.com

Maria Dimakopoulou
Spotify
New York, USA
mdimakopoulou@spotify.com

Jesse Anderton
Spotify
New York, USA
janderton@spotify.com

Benjamin Carterette
Spotify
New York, USA
benjaminc@spotify.com

Mounia Lalmas
Spotify
New York, USA
mounial@spotify.com

Tony Jebara
Spotify
New York, USA
tonyj@spotify.com

ABSTRACT

Calibration in recommender systems has recently gained significant attention. In the recommended list of items, calibration ensures that the various (past) areas of interest of a user are reflected with their corresponding proportions. For instance, if a user has watched, say, 80 romance movies and 20 action movies, then it is reasonable to expect the recommended list of movies to be comprised of about 80% romance and 20% action movies as well. Calibration is particularly important given that optimizing towards accuracy often leads to the user's minority interests being dominated by their main interests, or by a few overall popular items, in the recommendations they receive. In this paper, we propose a novel approach based on the max flow problem for generating calibrated recommendations. In a series of experiments using two publicly available datasets, we demonstrate the superior performance of our proposed approach compared to the state-of-the-art in generating relevant and calibrated recommendation lists.

CCS CONCEPTS

• **Information systems** → **Personalization**.

KEYWORDS

recommender systems, calibration, max flow networks

ACM Reference Format:
Himan Abdollahpouri, Zahra Nazari, Alex Gain, Clay Gibson, Maria Dimakopoulou, Jesse Anderton, Benjamin Carterette, Mounia Lalmas, and Tony Jebara. 2023. Calibrated Recommendations as a Minimum-Cost Flow Problem. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining (WSDM '23), February 27-March 3, 2023, Singapore, Singapore.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3539597.3570402

WSDM '23, February 27-March 3, 2023, Singapore, Singapore

ACM ISBN 978-1-4503-9407-9/23/02…$15.00
https://doi.org/10.1145/3539597.3570402

1 INTRODUCTION

Recommender systems are applied to a wide range of applications to help users navigate a large number of options by showing them items that are likely to be of interest [25]. Early recommendation algorithms focused on maximizing the relevance of the recommendations for the users. Since then, the focus has expanded to include other important characteristics of recommendations such as diversity [5, 30], serendipity [13, 15], novelty [3], fairness [18], and business value [10]. All these requirements can significantly impact the success of a recommender system.

Having other objectives on top of the accuracy of the recommendations requires special algorithms that can take multiple objectives into account in generating the recommendations [26]. One of the recent beyond-accuracy metrics initially described in [21] and made widely aware by [29] is the idea of *calibrated recommendations*. Simply put, calibration in recommender systems measures whether the recommendations delivered to a user are consistent with the diversity of items the user has previously liked and/or consumed. For example, if a user has watched 80% romance movies and 20% action movies, the user might expect to see a similar distribution in their recommendations. If this distribution differs from the one in the user's history, then the user's recommendations are referred to as *miscalibrated*.

One existing approach for calibration proposed by Steck [29] uses a greedy optimization of a surrogate submodular function to generate calibrated recommendations. The algorithm starts with an empty recommendation set and iteratively adds items to it if adding that item makes the list more relevant and calibrated. Another, more recent, approach by Seymen et al. [28] formulated the calibration optimization problem as a mixed integer program with an L1-norm-based miscalibration penalty and used the Gurobi software [6] to solve the mixed integer program. Gurobi implements the branch & bound algorithm to solve mixed integer programs

which in worst-case scenarios can have exponential time complexity [20]. In general, mixed integer programming has *no guarantees* to find the optimal solution efficiently (e.g. in polynomial time).

In this paper, we establish that calibrated recommendations can be cast to the *maximum flow* optimization problem [9]. We propose a novel approach for calibrated recommendations by modeling this problem via a graph where costs associated with connecting adjacent nodes are derived from different parts of an objective function that we use for calibration as you will see in section 3. In other words, the solution for calibrated recommendations is equivalent to the solution where we can have maximum flow in the mentioned graph with minimum cost. We test our proposed algorithm on the public MovieLens 20 Million dataset [8] and the Last.fm music dataset and compare its performance to existing calibration techniques of Seymen et al. [28] and Steck [29]. We demonstrate that our algorithm consistently outperforms state-of-the-art approaches in terms of relevance and miscalibration trade-off. In addition, we also evaluate the performance of our proposed algorithm in comparison to the existing approaches on different groups of users with a varying degree of interest in popular items and with varying degree of genre diversity in their consumption history. We show that our approach consistently achieves better performance across different user groups. This is particularly important since the purpose of calibrated recommendations is to ensure users receive recommendations that matches the diversity of their interest, especially for users who have little interest in popular items and those with a more focused and niche taste.

Our paper makes the following contributions:

- We propose a novel approach based on maximum flow to solve the calibration problem.
- On two publicly available datasets, we demonstrate that our approach consistently outperforms the existing approaches for calibrated recommendations.
- We also show that, in addition to the superior performance across all users on average, our algorithm gives better performance for different user groups with varying degree of interest in popular items or with different levels of genre diversity.

2 CALIBRATION IN RECOMMENDER SYSTEMS

Calibration in recommendation refers to the scenario where the recommendations a user receives are representative and in the proportion to their historical interactions in terms of different item categories they have interacted with. However, since many recommender algorithms optimize for accuracy metrics, recommended lists are often dominated by either popular content (even if the user has shown no interest in them), or by the user's main interests, ignoring their minority tastes [29]. Therefore, a mechanism is needed to make the recommended lists more calibrated according to the user's range of interests.

Figure 1 shows the listening history of a hypothetical user on a streaming platform that streams both music and podcast to the listeners: 80% of the user's listening history is podcast and 20% is music. We also see three different recommendation sets generated for that user. The first two are non-calibrated since one is highly

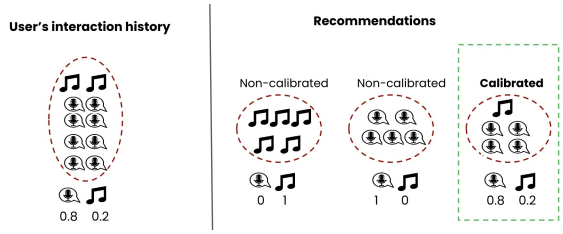


Figure 1: The listening history a user and three different recommendation sets generated for that user.

dominated by music (often music is a more popular content across all users) and the other one is dominated by podcasts which is the main interest of the user. In other words, they do not match the history of user listening in terms of two item categories (music vs podcast). The last recommendation set is calibrated as it fully matches users' listening history.

We can now formally define calibrated recommendations. Suppose we have c categories to which items can belong. Given the distribution over item category k of the set of items H liked by the user u, $p(k)$, and the distribution over item categories k of the set of items I recommended to user u, $q(k)$, we are interested in measuring how close $q(k)$ is to $p(k)$. If we have:

$$p(k) = \frac{\sum_{i \in H} p(k|i)}{\sum_{i \in H}} \quad , q(k) = \frac{\sum_{i \in I} p(k|i)}{\sum_{i \in I}} \tag{1}$$

Next, we describe how we can cast the calibration to a minimum cost flow problem.

3 MINIMUM COST FLOW FOR CALIBRATION

The minimum-cost flow (MCF) is an optimization problem to find the cheapest possible way of sending a certain amount of flow through a flow network [9]. An example is finding the best delivery route from a factory to a warehouse where the road network has some capacity and cost associated. The minimum cost flow problem is one of the most classic problems in the computer science and algorithms literature as many other optimization problems can be cast as an *MCF* problem and hence be solved efficiently [23]. In recommender systems, there are some existing work leveraging max flow problem. For instance, Mansoury et al. [19] and Antikacioglu and Ravi [2] modeled recommendations as bipartite graphs (where one side are item nodes and one side are user nodes) to solve the problem of long-tail recommendation and improving aggregate diversity [1]. Our goal is to find a way to cast the calibration problem as a minimum cost flow problem.

Generally, the goal of a recommender system is to put n out of m possible items into n slots on each user's list where $m >> n$ since there are many available items to be recommended yet the size of a recommendation list is limited. Assume that an oracle (e.g., a standard recommender algorithm) predicts the (expected) value A_{ij} of putting item i into slot j of the user u's recommendation list. This information is represented as a matrix A of size $m \times n$ (giving each user a personalized matrix). The best list is found via a matching which can be written as a binary matrix M of size $m \times n$, where if $M_{ij} = 1$ then item i goes into slot j. A matching must pack

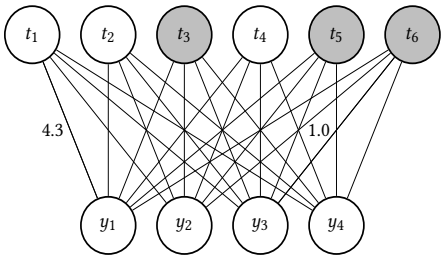


Figure 2: In this example, there are 4 slots on the list ($n = 4$), 6 items in the candidate set ($m = 6$) and 2 allowable content categories ($c = 2$). Items 1, 2, and 4 belong to category 1, and items 3, 5, and 6 belong to category 2. Placing each t_i in each y_j has a certain value $A_{i,j}$. For example, the value of placing item t_1 into slot y_1 is 4.3 ($A_{1,1} = 4.3$), and the value of placing item t_6 into slot y_3 is 1.0 ($A_{6,3} = 1.0$)

the list fully and never double-pack a slot. This gives the following constraints on the matrix M:

$$\sum_{j=1}^{n} M_{ij} \leq 1 \quad \text{(each item should be put in the user's list at most once)} \tag{2}$$

$$\sum_{i=1}^{m} M_{ij} = 1 \quad \text{(each slot should be filled by exactly one item)} \tag{3}$$

The optimal packing M^* can be found by solving the following maximum weight assignment problem:

$$M^* = \arg\max_{M} \sum_{i=1}^{m} \sum_{j=1}^{n} M_{ij} A_{ij} \tag{4}$$

We can model the recommendation list generation in a graph structure where a list of candidate item nodes $\{t_1, \ldots, t_m\}$ and a list of slot nodes $\{y_1, \ldots, y_n\}$ are provided. Each A_{ij} represent the value of connecting node i to j. Figure 2 shows an example of a recommended list with 4 slots and 6 potential candidate items we can show to the user.

To get a calibrated list, we must make sure that the items on the list are distributed across c possible content categories in a way that is close to that of the user's historical liked content. We can represent which content category an item belongs to through another matrix of size $m \times c$ called G, where G_{ik} indicates whether item i belongs to category k. In other words, looking at each column in G representing a given category, we can see which items contain that category (the entries that are 1). Note that if an item i belongs to multiple categories (e.g., a movie is both drama and romance), all corresponding G_{ik} for that item are set to 1 meaning that item contains all of those categories.

Given a potential solution matching M, the distribution of content category k on the list is simply given by $q(k)$ as follows:

$$q(k) = \frac{1}{n} \sum_{i=1}^{m} \sum_{j=1}^{n} G_{ik} M_{ij} \tag{5}$$

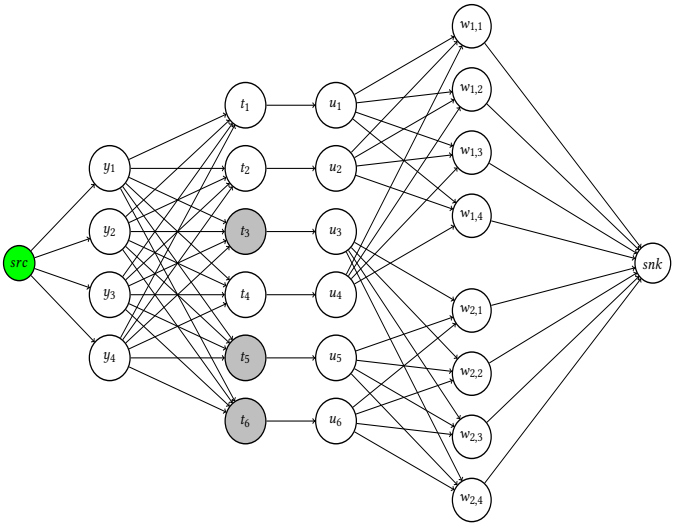


Figure 3: In this example, there are 4 slots on the list(n=4), 6 items in the candidate set (m=6) and 2 allowable content categories (c=2). Items 1, 2, and 4 belong to category 1, and items 3, 5, and 6 belong to category 2. The direction of the edges indicate the direction of the flow from the source node to the sink node

To penalize how much q deviates from p, we must modify the objective function as is done in [29] by introducing a penalty that grows with the divergence $D_{KL}(q\|p)$ (e.g., Kullback-Liebler divergence) in a weighted way as follows:

$$M^* = \arg\max_{M} (1-\lambda) \times \sum_{i=1}^{m} \sum_{j=1}^{n} M_{ij} A_{ij} - \lambda \sum_{k=1}^{c} q(k) \log \frac{q(k)}{p(k)} \tag{6}$$

If $q(k) = j/n$, that is, if the solution picks j items of category k, the penalty term corresponding to category k in the above objective can be written a as:

$$E_{k,j} = \frac{j}{n} \log\left(\frac{j}{n}\right) - \frac{j}{n} \log\left(p(k)\right) \tag{7}$$

Using this notation, we can re-write the objective as:

$$M^* = \arg\max_{M} (1-\lambda) \times \sum_{i=1}^{m} \sum_{j=1}^{n} M_{ij} A_{ij} - \lambda \times \sum_{k=1}^{c} E_{k,nq(k)} \tag{8}$$

We are now ready to cast the calibration problem to the min-cost flow problem. For network flow we need to define what the cost of sending a certain amount of flow from one node to another is. In the min-cost flow problem, the costs are allowed to be positive or negative. As we saw in the Equation 8 the objective function has two components: 1) the sum of the relevance scores of the n items that are placed into n slots, and 2) the sum of the miscalibrations we get if we have those n items in the list. Note that our goal is to maximize Equation 8 and so in order to cast our problem to min cost flow we negate both terms and use them as costs in our network on the appropriate edges as we will describe below. We also need

to assign a capacity for each edge connecting two nodes. For the calibration problem, since there is no real meaning of having a capacity for an edge, we assign capacity 1 to every edge. In section 6 we will discuss the optimality of the recommendations generated by this approach.

We construct an instance of the min-cost flow problem consisting of the following sets of nodes and edges, as illustrated in Figure 3.

(1) A source node src.
(2) n nodes $y_1, \ldots, y_n$, each corresponding to a slot, with zero-cost edges from src.
(3) m nodes $t_1, \ldots, t_m$, each corresponding to an item. For every $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, m\}$, we connect y_i to t_j with an edge of cost $-A_{ij}$.
(4) m nodes $u_1, \ldots, u_m$, each corresponding to an item. For every $j \in \{1, \ldots, m\}$, we connect t_j to u_j with a zero-cost edge. This ensures each item to be picked for no more than one slot, as only a maximum flow of one is possible from t_j to u_j.
(5) For each category $k \in \{1, \ldots, c\}$, n nodes $w_{k,1}, w_{k,2}, \ldots, w_{k,n}$. For each item $j \in \{1, \ldots, m\}$, if this item belongs to category k, we add zero-cost edges from u_j to all the n category nodes $w_{k,i}$ corresponding to category k.
(6) A sink node snk with edges from all category nodes $w_{k,i}$. The cost of the edge from node $w_{k,i}$ to snk is set to $E_{k,i} - E_{k,i-1}$

4 EXPERIMENTAL SETTING

We now describe the details for evaluating our proposed algorithm, the data we used, the metrics we measured, and the comparison to the existing approaches.

4.1 Data

We used two publicly available datasets that contain item categories (genres) in addition to the user-item ratings. Although, similar to [29], we used genres in this paper to represent item categories, any kind of other categorization (e.g., popular vs non-popular, movies vs TV series vs Live shows, etc.) can be used if that information is available and the recommendation platforms care about calibrating the recommendations towards those categories for each user. The first dataset is Movielens 20M dataset [8] which contains 20 million ratings applied to 27,000 movies by 138,000 users. Following Steck [29], we retain only ratings of 4 stars and higher, converting them into a binary "liked" signal. Examples of some popular genres in this dataset are *Drama*, *Romance*, *Action*, and *Thriller*.

The second dataset is the Last.fm 1b music dataset [27]. We created a sample of the dataset with 10,000 users and 10,000 tracks resulting in 2,765,176 interactions. Some popular genres in this dataset are *Alternative Rock*, *Indie*, *Pop*, and *Electronic*.

4.2 Baselines and candidate generation

A calibration algorithm needs a candidate set, larger than the final recommendation set, generated by a recommender algorithm to extract a calibrated and relevant list. Any algorithm can be used to generate this candidate set as calibration is done on top of this set so the mechanism of the calibration is independent from the algorithm that generated the candidate set. We used Bayesian Personalized Ranking (*BPR*) [24] for this purpose as it can be used on binary ratings. We set the size of the candidate set to 100 and use calibration algorithms to extract the final list of 10 items to be recommended (i.e., $m = 100$, $n = 10$). We compare our proposed approach to two existing techniques for calibrating recommendations:

- **Greedy:** the greedy approach proposed by Steck [29]. The process is as follows: starting out with the empty set, the algorithm iteratively appends one item i at a time. At step n, when the algorithm has already constructed the set I_{n-1} comprised of $n-1$ items, the item i that maximizes an objective function which is a combination of relevance of the recommendations and the miscalibration degree of the constructed list as described in [29]. The process is repeated to generate the full I^* of size n.
- **Mixed Integer Programming (MIP):** this algorithm proposed by Seyman et al in [28] outperformed the greedy approach proposed by Steck as described in [28]. As authors stated in their paper, they used the Gurobi software [7] and they limit the time to 10 minutes with optimization gap 10^{-4} for every user separately. Gurobi implements the branch&bound algorithm to solve mixed integer programs. In worst-case scenarios, this algorithm can have exponential time complexity [20]. We use the same settings for this algorithm as suggested by the authors.

Throughout the rest of the paper, we denote the candidate generation algorithm as *Base*, the greedy approach proposed in [29] as *Greedy*, the mixed integer programming technique proposed in [28] as *MIP*, and our proposed Min Cost Flow approach as *MCF*.

4.3 Evaluation Metrics

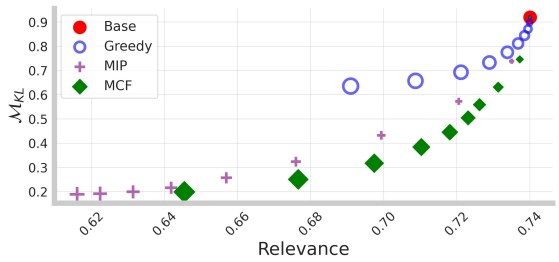
To evaluate the performance of each of the calibration approaches, we consider several metrics to compare how each algorithm performs from different angels: 1) the average expected relevance of the recommendations, 2) how calibrated the recommendation lists are for different users, and for different user groups, 3) what is the performance of each algorithm on ranking metrics, and 4) how they perform in terms of processing time.

- **Relevance Score:** This is the average predicted score for items in the recommended list. It models the expected relevance of the recommended items to an end-user.
- **Miscalibration ($\mathcal{M}_{KL}$):** the KL-divergence between the distribution of different genres in the recommended list (Q) for each user and the distribution of genres in their profile (P). Lower values for miscalibration indicate better performance in terms of calibration. c is the set of genres.

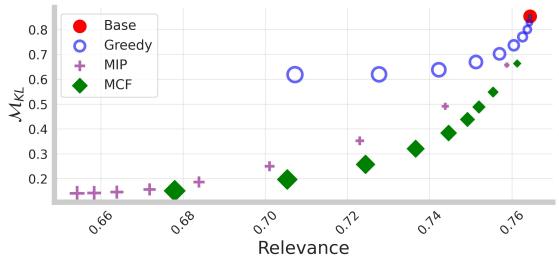
$$\mathcal{M}_{KL} = D_{KL}(P||Q) = -\sum_{k=1}^{c} q(k) \log \frac{q(k)}{p(k)} \tag{9}$$

In addition, we measure the performance of the algorithms on common ranking metrics such as **precision**, **recall**, and **Normalized Discounted Cumulative Gain (NDCG)**.

- **Precision:** Number of items in the recommended list that were previously liked by the user divided by size of the list.
- **Recall:** number of liked items in user's historical interaction that appear in the recommendation list.
- **Normalized Discounted Cumulative Gain (NDCG):** similar to precision but a liked item in higher positions on the list



(i) MovieLens



(ii) Last.fm

Figure 4: Miscalibration vs relevance. Larger marker size indicates larger λ values.

has higher weight and the weight decreases logarithmically for lower positions.

5 RESULTS

In this section, we present the results of the experiments we conducted to compare our proposed calibration algorithm with existing approaches in terms of relevance of the recommendations, the miscalibration degree, and also some other ranking metrics. We do that both across all users and also for different user groups as you will see in section 5.2.

5.1 Overall performance

We first compare our proposed algorithm with existing approaches in terms of the trade-off between the relevance of the recommendations and miscalibration, and also several ranking metrics across all users.

5.1.1 Trade-off between relevance and miscalibration. Besides the Base algorithm, all the other algorithms in this section aim at building a recommendation list that has a good trade-off between the relevance of the recommendations in the list and how calibrated the list is in comparison to the user's historical interactions. Figure 4 shows the trade-off between the average relevance score of the recommended items and miscalibration for *Greedy*, *MIP*, and *MCF* approaches compared to the Base algorithm on both datasets. As expected, all algorithms have reduced expected relevance (as modeled by the average predicted score) to a certain degree in order to achieve more calibrated recommended lists for larger values of λ. What can be seen in this plot is that the green points for *MCF* solution is on the Pareto frontier and there is no λ value for the Greedy approach nor for the *MIP* approach that yields simultaneously better relevance estimates and better miscalibration which indicates the superior performance of our proposed approach in achieving the best trade-off between expected relevance and miscalibration.

5.1.2 Recommendation ranking accuracy and calibration quality. The *Precision*@10, *Recall*@10, *NDCG*@10, and Miscalibration values for all algorithms are shown in Figures 5(i) and 5(ii).

On Movielens, we can see that, unlike other calibration techniques, *MCF* has maintained ranking metrics such as precision, recall, and NDCG for λ values up to 0.7 while lowering miscalibration. *Greedy* has lost the highest degree of *NDCG* meaning the algorithm does mess up the ranking quality compared to the other two calibration techniques due to its iterative approach. Overall, *MCF* has greatly lowered miscalibration compared to the *Base*. For the same level of miscalibration, other calibration techniques have performed poorer compared to *MCF* in ranking metrics indicating a better performance of *MCF* achieving a better trade-off between miscalibration and ranking quality. For example, up to $\lambda = 0.6$, *MCF* has not lost any degree of Precision, Recall or NDCG but has substantially lowered the miscalibration. On the music data, the ranking metrics are actually improved for *MCF* for λ values up to 0.7. Both *MCF* and *MIP* has done well in lowering miscalibration but *MCF* has performed better in relevance and ranking metrics. Also, remind that the *MIP* is based on Mixed Integer Programming and there is no guaranty of finding an optimal solution if the size of the problem is large.

Overall, the proposed *MCF* algorithm has achieved a better trade-off between the miscalibration, relevance, and other ranking metrics. When the order of ranking matters (*NDCG* metric), *Greedy* has performed the poorest due to its iterative construction of the lists.

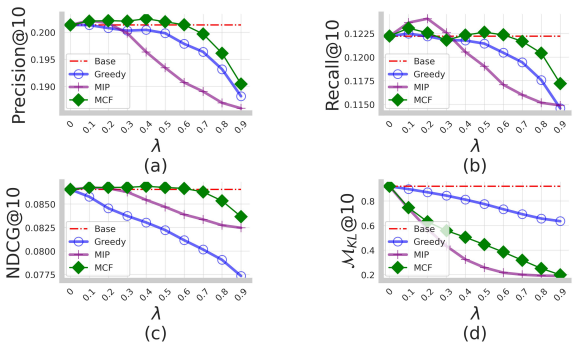
Next we examine the performance of different calibration approaches on different user groups with varying degree of interest in popular items and varying degree of genre diversity.

5.2 User Groups Analysis

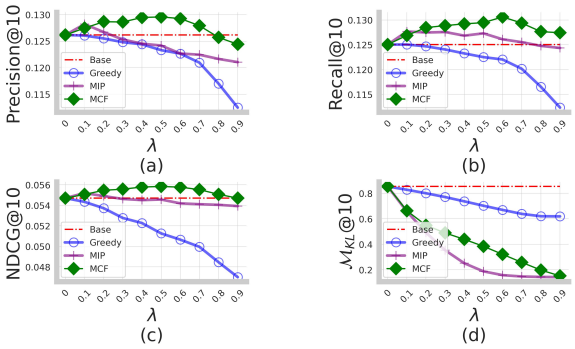
Often it is possible for a certain algorithm to perform well on average but not for different groups of users with different characteristics. For example, an algorithm may give highly calibrated recommendations to users who have mainstream interests but perform poorly for those with a more niche-focused taste. A successful recommender platform needs to ensure that it delivers a consistent performance for different groups of users. We investigate this from the perspective of 1) users with varying degree of interest in popular items and 2) users with different levels of genre diversity.

Grouping based on users' popularity tendency. Users have different tendency towards popular items and it is important for an algorithm to give consistent performance across different user groups with varying degrees of interest towards popular items. Popularity of an item is measured as the ratio of users who have consumed (or interacted with) that item [11]. For this purpose, similar to [16, 17], we bin the users into three groups based on their tendency towards popular items.

- **L-Pop**:The bottom 20% of users in terms of popularity tendency(i.e. the users who have low interest in popular items.
- **H-Pop**: The top 20% of users in terms of popularity tendency. These are the users who have high interest in popular items.



(i) MovieLens



(ii) Last.fm

Figure 5: Miscalibration vs relevance, precision, recall, and NDCG. Larger marker size indicates larger λ values.

- **M-Pop**: These are the users who have medium interest in popular items and fall between L-Pop and H-Pop users.

Grouping based on users' genre Diversity.

It is also important to note that users have different levels of diversity in terms of the number of different genres they like: some are concentrated on few genres and some are more open to try different types of genres. For instance, some users may only listen to Rock and Indie music but some others may listen to a wide variety of different music genres. The question is how each of the calibration approaches performs for these different groups of users. To answer these questions, we group the users as follows:

- **L-Diverse**: The bottom 20% of users in terms of genre count. These are the most focused users with the lowest number of genres in their consumption history.
- **H-Diverse**: The top 20% of users in terms of genre count. These are the users who have a diverse taste and have the highest number of genres in their consumption history.
- **M-Diverse**: These are the users who fall between L-Diverse and H-Diverse users.

Figure 6 shows the average miscalibration for all the algorithms for different user groups on the MovieLens dataset. Firstly, we can see that all algorithms have lowered miscalibration for all user groups. *MCF* consistently has the lowest miscalibration for all user

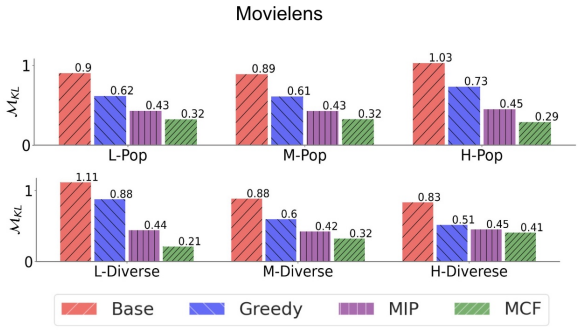


Figure 6: Average Miscalibration of MovieLens recommendations for different users groups L-Pop (users with low popularity tendency), M-Pop (users with medium popularity tendency), H-Pop (users with high popularity tendency), Low-Diverse (users with low genre count), Med-Diverse (users with medium genre count), H-Diverse (users with high genre count).

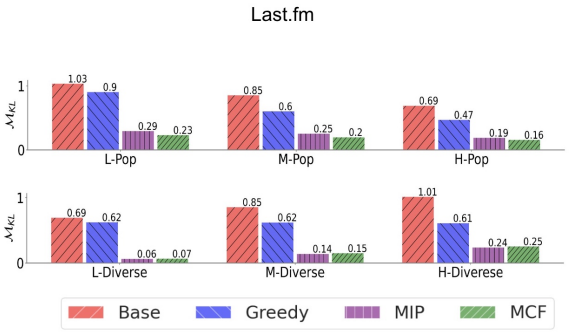


Figure 7: Average Miscalibration of music recommendations for different users groups L-Pop (users with low popularity tendency), M-Pop (users with medium popularity tendency), H-Pop (users with high popularity tendency), Low-Diverse (users with low genre count), Med-Diverse (users with medium genre count), H-Diverse (users with high genre count).

groups regardless of their popularity tendency or genre diversity. Interestingly, for groups based on genre diversity, all algorithms except *MCF* have their best performance for users who are interested in a wider range of genres (*H-Div*). In other words, it is harder for these algorithms to deliver more calibrated recommendations for users who are focused on few genres. *MCF*, on the other hand, has performed superior for this focused users indicating that the algorithm can consistently produce calibrated recommendations for users regardless of the diversity of their taste. Overall, we can see that *MCF* approach consistently has lowered miscalibration for all user groups.

Figure 7 shows the same information for the music data. The same pattern also can be seen on this dataset.

6 OPTIMALITY

We presented our empirical results in section 5 and we saw that the proposed algorithm consistently outperformed existing approaches in having better calibrated recommendations while maintaining good accuracy and ranking metrics. In this section, we prove that our proposed approach can theoretically guaranty an *optimal* solution.

Theorem 6.1. *A minimum cost flow that sends n units of flow from src to snk in the graph defined in section 3 corresponds to the optimal solution of the calibration problem.*

Proof. Note that since $m > n$, it is possible to send n units of flow from *src* to *snk* [4].

If we assume, the total cost from *src* to *snk* is the sum of the negative of the relevance scores ($-A$) and the miscalibration score ($\sum_{k=1}^{c} E_{k,nq(k)}$), we aim to minimize this cost and hence maximizing the objective function. More formally:

$$\begin{aligned} &\arg\max_{M}(1-\lambda)\times\sum_{i=1}^{m}\sum_{j=1}^{n}M_{ij}A_{ij}-\lambda\times\sum_{k=1}^{c}E_{k,nq(k)}= \\ &\arg\min_{M}(\lambda-1)\times\sum_{i=1}^{m}\sum_{j=1}^{n}M_{ij}A_{ij}+\lambda\times\sum_{k=1}^{c}E_{k,nq(k)} \end{aligned} \tag{10}$$

Our goal is to prove that a solution to the calibration problem with objective value $V = (1-\lambda)\times\sum_{i=1}^{m}\sum_{j=1}^{n}M_{ij}A_{ij} - \lambda\times\sum_{k=1}^{c}E_{k,nq(k)}$ can be turned into an n-unit flow in the graph shown in Figure 3 of cost $-V$, and conversely, a min-cost way to send n-units of flow from *src* to *snk* at cost C can be turned into a solution to the calibration problem with objective value $-C$. We will use the following proposition in our proof:

Proposition 6.2. *For every k, $E_{k,j}$ is a convex function of j. Therefore, $E_{k,j} - E_{k,j-1}$ is non-decreasing in j.*

We start with the more straightforward direction: assume we are given a solution M to the calibration problem with objective value V. A flow in the above graph can be constructed as follows: for each $i \in \{1,\dots,n\}$, we send one unit of flow from *src* to y_i, and then from y_i to t_j, where j is the item placed in slot i. This flow will then be passed to u_j. Note that this is possible, since each item appears in at most one slot. Next, this flow should be sent to one of the nodes $w_{k,i}$, where k is the category of j. We know that the solution M contains precisely $nq(k)$ items of category k. For each of these $nq(k)$ items, say item j, we pick one of the nodes $w_{k,1},\dots,w_{k,nq(k)}$, and send a unit of flow from u_j to this node. This ensures that among all the n nodes $w_{k,i}$ for category k, the first $nq(k)$ of them receive one unit of flow, and the rest receive no flow. The nodes that receive a unit of flow pass this to *snk*. We now calculate the cost of this solution. The cost of the flow on the edges between y_i's and t_j's is precisely $-\sum_{i=1}^{m}\sum_{j=1}^{n}M_{ij}A_{ij}$. Also, for every category k, the cost of the edges from $w_{k,i}$'s to *snk* is $\sum_{i=1}^{nq(k)}(E_{k,i}-E_{k,i-1}) = E_{k,nq(k)} - E_{k,0} = E_{k,nq(k)}$. All the other edges have zero cost. Therefore, the total cost of this flow is $-\sum_{i=1}^{m}\sum_{j=1}^{n}M_{ij}A_{ij} + \sum_{k=1}^{c}E_{k,nq(k)} = -V$.

For the other direction, assume the min-cost n-unit flow in the above graph has cost C. By the integrality theorem of min-cost flows [14], there is a min-cost flow that is also integral, i.e., it sends either 0 or 1 units of flow on each edge. For each category k, if there are two nodes $w_{k,r}$ and $w_{k,r'}$ with $r < r'$ such that there is no flow on the edge from $w_{k,r}$ to *snk* but there is flow on the edge from $w_{k,r'}$ to *snk*, we can change the flow as follows: the node u_j that is sending one unit of flow to $w_{k,r'}$ will now send it to $w_{k,r}$ instead, which will send it to *snk*. By Proposition 6.2, the cost of the edge from $w_{k,r}$ to *snk* is not larger than the one from $w_{k,r'}$ to *snk*, so this change will not increase the cost of the flow. By iteratively doing such changes, we get a min-cost flow such that for each category k, there is one unit of flow on edges from $w_{k,1},\dots,w_{k,r(k)}$ to *snk* and zero flow on edges from $w_{k,r(k)+1},\dots,w_{k,n}$ to *snk*. This means that the total cost incurred on the edges from $w_{k,1},\dots,w_{k,n}$ to *snk* is $\sum_{i=1}^{r(k)}(E_{k,i}-E_{k,i-1}) = E_{k,r(k)} - E_{k,0} = E_{k,r(k)}$. We can define a solution to the calibration problem based on this flow as follows: for each slot i, we place in this slot the item j such that there is a unit flow on the edge from y_i to t_j. It is easy to see that this is a valid solution to the calibration problem, and its value is $-C$ □

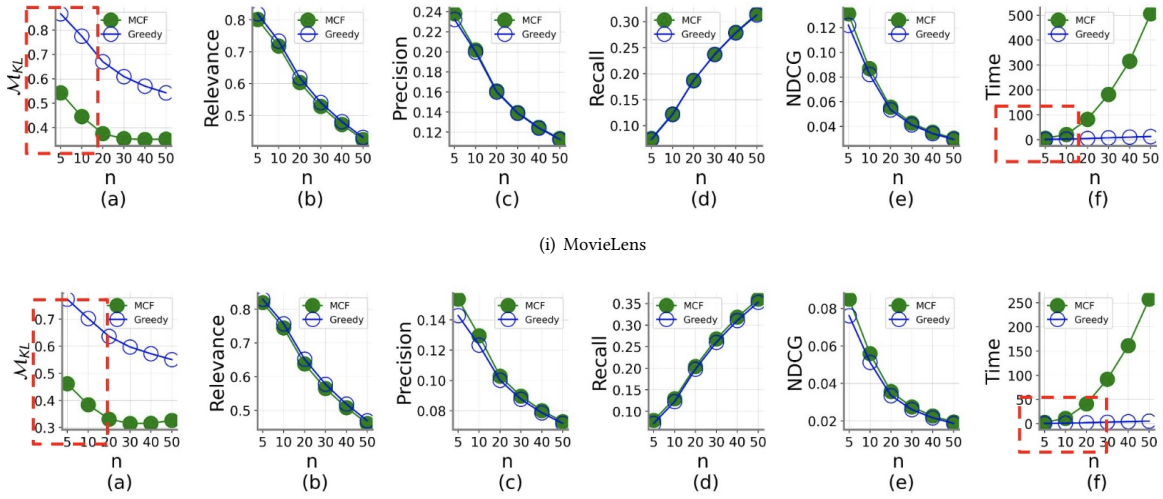
To summarize, since we cast the calibration problem to the minimum cost flow, and the graph we constructed according to the instructions we described in section 3 contains no negative cost directed *cycle*, it can be proved that there is a feasible flow from the source node to the sink node and, by definition, the flow is optimal [12]. Moreover, the process for finding a feasible flow has a Polynomial time complexity [22, 23].

7 LIMITATIONS

In this section we describe a limitation of our proposed approach which is processing time especially when we generate larger recommendation lists.

On the one hand, when the number of available slots (n) in the recommended list is too small compared to the number of categories (c), it would be challenging to achieve a calibrated list that covers the diversity of users' interest. On the other hand, if n is too large, it would impact the time and memory complexity of the calibration algorithms. In this section, we evaluate the performance of our proposed algorithm in comparison with the Greedy algorithm in generating recommendation lists of different sizes to see the trade-off between recommendation size, calibration, relevance, ranking metrics and, time. We did not include *MIP* here since, as we mentioned in section 4.2, the algorithm needs a time limit for each iteration (authors recommended 10 minutes) in order for the algorithm to stop even for smaller recommendation sizes as otherwise the algorithm could take days on the dataset we used. Therefore, the processing time does not have much meaning for that approach in this case. Figure 8(i) and 8(ii) show the effect of recommendation size (n) on miscalibration, accuracy metrics, and also processing time in minutes.

We can see that, on MovieLens, miscalibration initially drops for lists with larger sizes as there is more room for incorporating different types of content in the list but once it reaches the size 30 it stops decreasing. Moreover— as expected— the average relevance scores of the recommendations consistently drops for larger recommendation sizes since items with lower relevance scores make their way into the list. What can be seen is that *MCF* has the best miscalibration for all recommendation sizes. Regarding



(i) MovieLens

(ii) Last.fm

Figure 8: The effect of recommendation size on miscalibration, relevance, precision, recall, NDCG, and processing time

the ranking metrics (precision, recall, and NDCG), also as expected, recall improves since there is more room for the recommender to correctly recall the previously *liked* items by the user (generally, $\forall i, j \in \mathbb{N} : recall@i > recall@j$ if $i > j$) but precision and *NDCG* drops. However, what is also clear is that the Greedy approach is much faster than our approach especially for larger recommendation sizes. However, as we can see in the figure, the processing time for both algorithms for smaller recommendation sizes are relatively close which makes the time complexity a bit less important since many recommendation sizes in real-world applications are around 10-20 items. In those cases, *MCF* has lowered miscalibration substantially more than *Greedy* as can be seen by the dotted red rectangle around the recommendation with smaller sizes. With that being said, if large recommendation sizes are needed and processing time is more important than having better calibrated recommendations then the Greedy approach can be a preferred approach.

The same pattern exists for Last.fm. What is important to take way from this section is the fact that 1) the size of the recommendation list given to each user is a key factor for controlling the degree of calibration. However, also one should note that larger recommendation size is associated with larger drop in average relevance of the recommendations and also larger processing time especially for *MCF* approach.

8 CONCLUSION AND FUTURE WORK

Calibrating the recommendation list for each user in order to cover a wider range of the user's interest and in the right proportion is an important list-wise property. We proposed a solution based on minimum cost flow problem for calibration, which achieves the best possible trade-off between calibration and relevance. We evaluated our algorithm on two datasets, showing that the proposed solution is superior to existing ones in several ways: 1) it generates recommendation lists with the optimal trade-off between relevance and miscalibration, 2) unlike existing calibration techniques, our approach rarely hurts ranking metrics and in some cases it improves them, and 3) it performs consistently well for different user groups with varying degree of interest in popular items and different levels of genre diversity. One limitation of our approach compared to the Greedy technique is the processing time especially for larger recommendation sizes— Greedy is much faster for larger lists. However, we saw that the two have relatively similar processing time for shorter recommendation sizes (up to 20 items in the list which is a typical size for many real-world recommenders) yet our proposed approach can lower miscalibration to a much greater extent.

Several interesting research questions remain open and we leave them for future research. Whether fully calibrating a list according to the user's consumption history is ideal needs further research, and the short answer is most likely no. The reason is that many users also appreciate some degree of exploration and surprise, so deviating from historical consumption every once in a while might be helpful. In addition, the calculation of user's historical distribution across content categories can be done in a more sophisticated way by taking into account the recency of consumed content or by assigning confidence levels to each user's content distribution based on how mature the user's profile is (e.g., a user who has joined the platform a week ago may still need time to explore enough so their consumption history could better represent their taste).

ACKNOWLEDGMENTS

We would like to thank Samuel Way, Azin Ghazimatin, Claudia Hauff, Praveen Chandar, and Jussy Karlgren for their feedback and comments. Also special thanks to the reviewers of WSDM 2023 for their constructive feedback and suggestions.

REFERENCES

[1] Gediminas Adomavicius and YoungOk Kwon. 2011. Improving aggregate recommendation diversity using ranking-based techniques. *IEEE Transactions on*

Knowledge and Data Engineering 24, 5 (2011), 896–911.
[2] Arda Antikacioglu and R Ravi. 2017. Post processing recommender systems for diversity. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 707–716.
[3] Pablo Castells, Neil Hurley, and Saul Vargas. 2022. Novelty and diversity in recommender systems. In *Recommender systems handbook*. Springer, 603–646.
[4] Jack Edmonds and Richard M Karp. 1972. Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the ACM (JACM)* 19, 2 (1972), 248–264.
[5] Mouzhi Ge, Carla Delgado-Battenfeld, and Dietmar Jannach. 2010. Beyond accuracy: evaluating recommender systems by coverage and serendipity. In *Proceedings of the fourth ACM conference on Recommender systems*. ACM, 257–260.
[6] LLC Gurobi Optimization. 2018. Gurobi optimizer reference manual.
[7] Gurobi Optimization, LLC. 2021. Gurobi Optimizer Reference Manual. https://www.gurobi.com
[8] F Maxwell Harper and Joseph A Konstan. 2015. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)* 5, 4 (2015), 1–19.
[9] Masao Iri. 1969. *Network flow, transportation, and scheduling; theory and algorithms*. Vol. 57. Academic Press.
[10] Dietmar Jannach and Michael Jugovac. 2019. Measuring the business value of recommender systems. *ACM Transactions on Management Information Systems (TMIS)* 10, 4 (2019), 1–23.
[11] Dietmar Jannach, Lukas Lerche, Iman Kamehkhosh, and Michael Jugovac. 2015. What recommenders recommend: an analysis of recommendation biases and possible countermeasures. *User Modeling and User-Adapted Interaction* 25, 5 (2015), 427–491.
[12] William S Jewell. 1958. Optimal flow through networks. In *Operations Research*, Vol. 6. INST OPERATIONS RESEARCH MANAGEMENT SCIENCES 901 ELKRIDGE LANDING RD, STE ..., 633–633.
[13] Marius Kaminskas and Derek Bridge. 2016. Diversity, serendipity, novelty, and coverage: a survey and empirical analysis of beyond-accuracy objectives in recommender systems. *ACM Transactions on Interactive Intelligent Systems (TiiS)* 7, 1 (2016), 1–42.
[14] Jon Kleinberg and Éva Tardos. 2006. *Algorithm Design*. Addison Wesley.
[15] Denis Kotkov, Shuaiqiang Wang, and Jari Veijalainen. 2016. A survey of serendipity in recommender systems. *Knowledge-Based Systems* 111 (2016), 180–192.
[16] Dominik Kowald, Peter Muellner, Eva Zangerle, Christine Bauer, Markus Schedl, and Elisabeth Lex. 2021. Support the underground: characteristics of beyond-mainstream music listeners. *EPJ Data Science* 10, 1 (2021), 14.
[17] Dominik Kowald, Markus Schedl, and Elisabeth Lex. 2020. The unfairness of popularity bias in music recommendation: A reproducibility study. In *European conference on information retrieval*. Springer, 35–42.
[18] Yunqi Li, Hanxiong Chen, Zuohui Fu, Yingqiang Ge, and Yongfeng Zhang. 2021. User-oriented fairness in recommendation. In *Proceedings of the Web Conference 2021*. 624–632.
[19] Masoud Mansoury, Himan Abdollahpouri, Mykola Pechenizkiy, Bamshad Mobasher, and Robin Burke. 2020. Fairmatch: A graph-based approach for improving aggregate diversity in recommender systems. In *Proceedings of the 28th ACM conference on user modeling, adaptation and personalization*. 154–162.
[20] David R Morrison, Sheldon H Jacobson, Jason J Sauppe, and Edward C Sewell. 2016. Branch-and-bound algorithms: A survey of recent advances in searching, branching, and pruning. *Discrete Optimization* 19 (2016), 79–102.
[21] Jinoh Oh, Sun Park, Hwanjo Yu, Min Song, and Seung-Taek Park. 2011. Novel recommendation based on personal popularity tendency. In *2011 IEEE 11th International Conference on Data Mining*. IEEE, 507–516.
[22] James Orlin. 1988. A faster strongly polynomial minimum cost flow algorithm. In *Proceedings of the Twentieth annual ACM symposium on Theory of Computing*. 377–387.
[23] James B Orlin. 1997. A polynomial time primal network simplex algorithm for minimum cost flows. *Mathematical Programming* 78, 2 (1997), 109–129.
[24] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*. 452–461.
[25] Francesco Ricci, Lior Rokach, and Bracha Shapira. 2011. Introduction to recommender systems handbook. In *Recommender systems handbook*. Springer, 1–35.
[26] Mario Rodriguez, Christian Posse, and Ethan Zhang. 2012. Multiple objective optimization in recommender systems. In *Proceedings of the sixth ACM conference on Recommender systems*. 11–18.
[27] Markus Schedl. 2016. The lfm-1b dataset for music retrieval and recommendation. In *Proceedings of the 2016 ACM on International Conference on Multimedia Retrieval*. 103–110.
[28] Sinan Seymen, Himan Abdollahpouri, and Edward C Malthouse. 2021. A Constrained Optimization Approach for Calibrated Recommendations. In *Fifteenth ACM Conference on Recommender Systems*. 607–612.
[29] Harald Steck. 2018. Calibrated recommendations. In *Proceedings of the 12th ACM Conference on Recommender Systems*. ACM, 154–162.
[30] Saúl Vargas and Pablo Castells. 2011. Rank and Relevance in Novelty and Diversity Metrics for Recommender Systems. In *Proceedings of the Fifth ACM Conference on Recommender Systems* (Chicago, Illinois, USA) *(RecSys '11)*. ACM, New York, NY, USA, 109–116. https://doi.org/10.1145/2043932.2043955