NORTHWESTERN UNIVERSITY

Application of Constrained Optimization Models to Recommender Systems

A DISSERTATION

SUBMITTED TO THE GRADUATE SCHOOL

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

for the degree

DOCTOR OF PHILOSOPHY

Field of Industrial Engineering and Management Sciences

By

Sinan Seymen

EVANSTON, ILLINOIS

June 2023

ABSTRACT

Application of Constrained Optimization Models to Recommender Systems

Sinan Seymen

Recommender systems (RSs) have become essential tools that provide personalized recommendations to their users. These systems may consider user, item provider, and system requirements simultaneously. With the inclusion of possibly clashing considerations, there is a growing focus on solving multiple-objective recommender system (MORS) problems as efficiently as possible. The constrained optimization models can be applied to MORS problems and obtain the optimal solution that can beat myopic heuristics approaches. In this dissertation, we investigate the applications of constrained optimization models to tackle MORS problems.

In Chapter 1, we give an introduction and overview of the application of constrained optimization models to RS problems.

In Chapter 2, we unify different considerations into a constrained optimization framework where different sets of metrics can be improved by simply using different sets of constraints. Rather than focusing solely on user needs, we tackle some of the most frequently investigated considerations in RSs, such as novelty, diversity, calibration, and fairness of the recommendations. We offer models that can handle multiple considerations simultaneously. Our scalable

constrained optimization model tackling the calibration problem is the first in the RS literature. Also, our models are simple and easy to generalize with other considerations. Our experimental results show that the optimization models we offer can outperform state-of-the-art heuristics. We illustrate reasons why the heuristics might struggle to find the optimal solution using a small example.

In Chapter 3, we offer a novel constrained optimization model that combines the RS ideas with inventory management. We consider both the preferences of the customers and retailer considerations while direct customer demand by item recommendations. These recommendations will consider perishability and inventory in an online retailing setting, in which we aim to minimize the number of wasted and stockout products. Our model can solve problems with stochastic supply and demand, where the demand, perishability, and inventory are considered not deterministic. We reformulate this model to be able to handle large data and stochasticity. We also note that creating recommendation lists only considering user needs or retailer needs can be counterproductive to the quality of the solution. If the user needs are the exclusive focus, this can lead to stockouts and a large number of perished items. Similarly, if the retailer's needs are the exclusive focus, this can lead to low utility recommendations to the users. Our model tackles this MORS problem by reducing waste by recommending soon-to-perish items, reducing stockouts by considering inventory, and making retailer and user-relevant recommendations simultaneously. We propose heuristic methods to improve the scalability issue the constrained optimization models may face. We compare the performance of these heuristics and note that the optimal solution quality does not decrease significantly.

In Chapter 4, we focus on ways to alleviate feasibility and scalability issues that can arise using constrained optimization models in MORS problems. We propose a Dantzig-Wolfe

(DW) decomposition-inspired optimization model that overcomes these limitations. We define within-list and across-list constraints, and how our model handles scalability considering these constraints. We compare our model with a recently proposed constrained optimization model; and state-of-the-art heuristics that specialize in one objective at a time using the MovieLens 20M dataset. We claim that our model can scale, find near-optimal solutions, and solve MORS problems with the flexibility to incorporate different considerations.

We discuss the future research possibilities of applying constrained optimization models to RS problems in Chapter 5.

Acknowledgments

I would like to express my sincere gratitude to my advisor Prof. Edward C. Malthouse for his guidance, understanding, kindness, and support. I have improved both as a scholar and as a person under his supervision.

I am extremely grateful to my other committee members Prof. David Morton and Dr. Anna-Lena Sachs for their valuable feedback and support throughout my Ph.D. journey.

I would like to extend my thanks to Prof. Bill White and Prof. Marita Poll for their support and dedication. I learned a great deal from our discussions.

I am indebted to my parents, Sevim and Suat Seymen. This work would not be possible without their sacrifices. I am grateful to my spouse for always supporting me and being accommodating. I am incredibly lucky to have them.

I appreciate that the Northwestern IEMS staff and my Ph.D. colleagues were always there with help, advice, and friendship. Similarly, I am glad to have our cats, Alfie and Maggie. They are always friendly and fluffy and made working from home an enjoyable experience.

Thank you, Mariana Escallon-Barrios, Melody Qiming Xuan, and Qimeng Yu for making my time during Ph.D. fun and memorable. I am also grateful to everyone that joined and/or helped me throughout this journey: Ross Gregory, Handan Kilincli, Shima Dezfulian, Stephen Pedersen, and Agnes Kaminski. It has been an amazing journey with everyone.

Table of Contents

List of Tables

List of Figures

CHAPTER 1

Introduction

1.1. Overview

In the most general terms, recommender systems (RSs) are information filtering systems that recommend an item or items to users. Initially, the preference of the user was the main criterion for these recommendations [Melville and Sindhwani, 2017]. In recent years, this idea is challenged by including factors other than the needs of the users, such as the needs of the item providers, the platform itself, or any kind of stakeholders in the recommendation process. With the addition of other considerations, the problems became bigger or more complex, requiring trade-offs between multiple objectives. In the literature, these problems can be tackled by greedy approaches that recommend one item at a time optimizing objective functions myopically [Patro et al., 2020, Steck, 2018, 2011, Herlocker et al., 2004, Abdollahpouri et al., 2019b]. Our work focuses on offering constrained optimization models that aim to find optimal or near-optimal solutions by creating recommendation lists holistically.

RSs are ingrained in our daily lives. We use these systems while deciding which song to listen to, which movie to watch, and which product to purchase from a retailer [Harper and Konstan, 2015, Chen et al., 2020]. In all of these examples, there is a mechanism trying to come up with the best item recommendation for the users. There can be millions or even billions of possible combinations of item-user pairs. The quality of these combinations depends on the needs of the users, the providers of the items, and the platform itself. The fairness of recommendations is important so that each provider will be represented similarly [Patro et al., 2020]. Some works try to alleviate the problem of popular items getting an unfair advantage by the algorithms [Abdollahpouri et al., 2019a], and others try to “diversify" the items recommended to each user so the user will be happier with the wide range of selections

available to them [Herlocker et al., 2004]. We propose constrained optimization models to tackle these issues in RS settings.

Constrained optimization models are not frequently explored in the context of RS. The constrained mixed-integer programs (MIP) are utilized even less for solving RS problems. We specifically focus on MIP and find items that should be recommended to the users in the system considering a variety of metrics. Sürer et al. [2018] work with a constrained MIP model and consider the fairness of stakeholders. Malthouse et al. [2019b] create an optimization model that chooses the relevant advertisements that should be recommended to the users. Some research considers models that use continuous decision variables [Agarwal et al., 2011, 2012, Jambor and Wang, 2010]. These decision variables usually indicate the probability of recommending an item to a user. They usually function similarly to the continuous relaxation of MIP, and they need to be rounded to the closest integer values. Therefore, coming up with k item lists using continuous decision variables requires additional tuning because continuous values need to be converted to integers. By utilizing MIP ideas, we can directly create a recommendation list with k items for each user. There are other works solving optimization problems, such as a graph optimization approach that is suggested to solve the diversity problem [Antikacioglu and Ravi, 2017]. Although not in the RS setting, there are MIP model suggestions in the literature solving problems retailers face [Nguyen and Chen, 2019, 2022]. Similarly, one work proposes an MIP model that is solved by using a heuristic approach [Chen et al., 2020] in a retailer setting. Overall, we investigate the possibilities and advantages of applying optimization modeling ideas to the RS problems.

Next, we define our motivation, discuss the results of our work so far, and outline the future research opportunities we aim to explore.

1.2. Motivation

The motivation behind this dissertation is to investigate the application of constrained optimization models in the field of RSs. Our work discusses the application of optimization models in solving MORS and Multistakeholder problems where the solution needs to tackle multiple objective functions simultaneously.

In Chapter 2, we propose an optimization toolbox that can combine and tackle multiple objective functions simultaneously. We offer distinct constrained optimization models that share the same objective function and solve various MORS sub-problems. These models can be combined or modified by the system designer according to the problem at hand. We tackle problems that are frequently investigated in the literature, such as popularity bias, fairness, and diversity issues. The constrained optimization model proposed can be solved optimally for thousands of items and users. We show that our approach can perform more highly than state-of-the-art heuristics that focus on a single metric using the MovieLens dataset 1M. Feasibility and scalability are identified as possible drawbacks of using constrained optimization models in an RS setting. We introduce heuristics that can be applied to alleviate the scalability issue, without significantly deteriorating the solution quality.

Next, we propose an optimization model to solve the calibration problem. Firstly, ours is the first optimization programming model that solves the calibration problem in RS. Secondly, we demonstrate the advantages of solving a problem holistically rather than myopically. The shortcomings of the state-of-the-art heuristic approach are illustrated with an example. Then, we also prove the success of our optimization model in the results section. Thirdly, this chapter discusses when the optimization programming techniques are the most successful, where the problem is scalable. When the model can be solved for each user or

item separately, the problems can be solved with millions of users and items, which is the case in this chapter. Therefore, we discover a novel topic and apply optimization models to solve the problems common in this area.

Our motivation in Chapter 2 is to offer constrained optimization models that are efficient and easy-to-use to solve RS problems that are frequently observed in the literature. We investigate the RS literature, and find a gap that could be filled by constrained optimization models. We create a generalizable and easy-to-modify toolbox to show that RS problems can be solved using optimization models in a highly efficient way. The MovieLens dataset is used to show that the optimization problem can find solutions that beat state-of-the-art approaches. We are the first to come up with such a toolbox that is capable of solving different metrics, with mixing and matching, under one model.

In Chapter 3, we propose an optimization model for solving perishability issues in an online retail setting. In this section, a subset of the items are deemed soon-to-perish and will be unfit for consumption if not sold. We use RS to change the demands of the users by recommending items that are soon-to-perish to avoid losses for the retailer. Additionally, we alleviate the sustainability issues emerging from items perishing. We focus on the effect of perishability on the environment and consider our work as a realistic solution. We include uncertainty which results in a multitude of options to recommend items to users. It is not known what would be the demand of the users, the number of perishable items, or the amount of inventory. Because of this uncertainty, stockouts become a problem due to over-recommendation of some items. This results in users not getting the items they desire. We create different scenarios to solve the uncertainty using a sample average approximation method. The model incorporating uncertainty requires more memory relative to the deterministic case with only one scenario. A new optimization model and a heuristic approach

are suggested to deal with the scalability problem. We investigate the solution quality of the optimization model, heuristics, and benchmarks.

In Chapter 4, we propose a constrained optimization model that improves the scalability issue. The constrained optimization models can suffer from scalability with larger datasets, and the problem can be exacerbated when tackling multiple objective functions simultaneously, as in MORS problems. Our model uses a large-scale Dantzig-Wolfe algorithm and can scale well. We define within- and cross-list constraints, and note that cross-list constraints are the primary reason scalability is a problem in constrained optimization models. We show how our model tackles cross-list constraints. We compare the solution quality of our model with state-of-the-art heuristics and an optimization model. We offer a heuristic approach that removes a significant number of decision variables, while not reducing the objective function value.

In Chapter 5, we outline three significant future research directions. Some questions are raised during the results we have produced in the earlier chapters, and we want to offer answers to those questions. Briefly, we want to implement our models in an online setting, where we receive feedback from the users in real time. Secondly, we worked with a variety of metrics during our work. It is unclear how changes in one metric affect the other metrics, and in what ways the dataset properties play a role in these changes. Thirdly, we want to further discuss the scalability problem, and propose more specialized large-scale optimization models to overcome MORS problems.

CHAPTER 2

Constrained Optimization Models Applied to Recommender System Problems

2.1. A Unified Optimization Toolbox for Solving Popularity Bias, Fairness, and Diversity in Recommender Systems

.

2.1.1. Introduction

The initial focus of recommender systems (RS) was on estimating users' preferences accurately, where measures including Root Mean Squared Error (RMSE), precision and recall were the primary objectives. Researchers later recognized the importance of other metrics such as diversity and novelty [Herlocker et al., 2004], fairness between multiple stakeholders [Abdollahpouri et al., 2020] and so on. Various types of criteria have been recognized as important considerations for the success of a RS and for each of them numerous algorithms have been proposed. For example, for improving the fairness of the RS from the providers' perspective, algorithms such as FairRec [Patro et al., 2020] (based on fair resource allocation) and FairMatch [Mansoury et al., 2020] (based on a graph-based maximum flow approach), and PFAR [Liu and Burke, 2018] (based on the weighted sum of relevance and fair exposure using the Maximum Marginal Relevance approach) are proposed, each offering a different approach for solving the same problem. Similarly, for mitigating the popularity bias problem Kamishima et al. [2014] uses the concept of neutrality for controlling this bias, Abdollahpouri et al. [2019a] mitigates popularity bias via ensuring a balanced exposure of two groups of popular and less popular items in each recommendation list, Vargas and Castells [2014] swaps the role of items and users and change the recommendation process as if the goal is to recommend users to each item, and many others. For improving the diversity of the recommendations within each list, Zhou et al. [2010] proposes a "heat-spreading" algorithm that

can be coupled in a highly efficient hybrid with a diffusion-based recommendation method [Zhou et al., 2007], Eskandanian et al. [2017] performs collaborative filtering independently on different segments of users based on the degree of diversity in their profiles, and Di Noia et al. [2017] uses a post-processing re-ranking technique to enhance the diversity of an initial recommendation list, and numerous other techniques.

One issue is that all the mentioned approaches for tackling different non-accuracy problems are implemented in isolation and cannot be easily combined to improve two or more aspects at the same time. We address this limitation by developing a constrained optimization toolkit that addresses popularity, fairness, and diversity metrics. Our framework is easy to implement and incorporate other metrics. We believe unifying all different non-accuracy related problems in RS under one umbrella can greatly benefit the research community and therefore we study how to create a list of k items for each user, assuming the preferences of each user for each item have already been estimated using some existing RS. We show how to write various considerations (e.g., diversity, popularity, fairness) as an optimization problem, and show how it can be solved as a post-processing step. We show that our toolkit achieves a comparable performance to the best-in-class algorithms for each specific task, but our toolkit is also able to improve more than one non-accuracy aspect of the recommendations by combining different constraints designed for separate aspects.

2.1.2. The Constrained Optimization Toolbox

Optimization models are applied to RS in different forms. Rodriguez et al. [2012] formulates recommendations as a constrained optimization problem and proposes the TalentMatch algorithm that matches job candidates to job posts. Another early paper using optimization with RS is Ribeiro et al. [2012], which searches a Pareto frontier balancing accuracy, diversity

and novelty. Jugovac et al. [2017] proposes a multi-objective, post-processing model, reviews the literature on multi-objective RS, and tests different heuristic solutions. Sürer et al. [2018] proposes integer programming models to solve RS by recommending items from stakeholders (providers) in the system in a sufficient amount for fairness. Antikacioglu and Ravi [2017] use a graph optimization approach to increase diversity of the recommendation lists. Similarly, in [Adomavicius and Kwon, 2011b], aggregate diversity is increased by graph-theoretic approach. Gogna and Majumdar [2017] use regularization terms in the objective function to increase the diversity and the novelty of the solution. Other multi-objective optimization models [Agarwal et al., 2011, 2012] are implemented to solve content recommendation problems. Works [Agarwal et al., 2011, 2012] consider an objective function that maximizes the probability of recommendations using continuous decision variables. In another line of work, Jambor and Wang [2010] propose a constrained linear optimization model for increasing the long-tail item recommendations. Most of these approaches have been tailored to solve particular RS problems, while we aim at unifying different non-accuracy aspects of the recommendations into a simple and flexible optimization approach.

This section describes our approach to solve different problems such as popularity bias, provider fairness, and diversity in RS using constrained optimization. For all problems, our technique maximizes the same objective function: the average ratings across all user and item pairs in the recommendations (i.e., the relevance of the recommendations). Different problems are addressed by adding different types of constraints. Thus, all problems have a similar structure, making it very easy to use and understand.

In the literature, some works [Gao and Shah, 2020, Díez et al., 2019] have recently investigated problems including more than one non-accuracy metric. However, it is not easy to modify these models to remove some metrics and include others. Most of the time, these

algorithms need significant changes to be able to incorporate different metrics other than ones that are already proposed. We suggest a framework that alleviates this problem, where different metrics can be easily mixed and matched. In other words, our approach is inspired by how one can create a different oatmeal each morning by simply using different toppings to the base oats: the relevance objective is the base and different types of constraints are the toppings. In the following subsections, we discuss the our optimization model in more details.

2.1.2.1. Base Top-k Model. We now formalize the toolkit, beginning with notation. Let U denote the set of users and I be the set of items in the system. Suppose k items are to be recommended to each user. We assume that the ratings have been predicted with some existing algorithm, with $\hat{r}_{iu}$ representing the predicted rating for user u and item i. Decision variables x_{iu} indicate which items are recommended, with $x_{iu} = 1$ if item i is recommended to user u, and 0 otherwise.

Our base model has an objective to maximize the average predicted ratings of all recommended items, subject to the constraint that each user u receives k recommendations. We can write this as an optimization problem as follows:

$$\max_{x} \quad \frac{1}{k|U|} \sum_{i \in I} \sum_{u \in U} \hat{r}_{iu} x_{iu} \tag{2.1}$$

$$\text{subject to:} \quad \sum_{i \in I} x_{iu} = k \quad (\forall u \in U) \tag{2.2}$$

$$x_{iu} \in \{0, 1\} \quad (\forall i \in I, u \in U) \tag{2.3}$$

Note that $\hat{r}_{iu} x_{iu}$ equals $\hat{r}_{iu}$ for recommended items and 0 otherwise, and therefore their sum divided by the number of recommendations made by the system ($k|U|$) gives the average

rating. Constraint (2.2) forces the model to recommend k items to every user u. Constraint (2.3) forces decision variables x_{iu} to be binary (an item is either recommended or not). This problem can be solved efficiently by sorting items for each user in descending order of predicted ratings, and then selecting the top k items for every user. Both the objective function and constraints are used in the upcoming models. Therefore, we can consider this Top-k model as the base, and add constraints according to the needs of the system.

2.1.2.2. Popularity Model. Many RS have a well-known bias to recommend popular items frequently and not give enough exposure to the majority of other, less popular, items [Abdollahpouri et al., 2019a]. This bias can be avoided with our *popularity optimization model* (Pop-Opt), which extends the base by adding a constraint to limit the aggregate popularity of all recommended items to a given user. We implement this idea by putting an upper bound (α) on the total popularity of the recommended items. We have the same objective function in (2.1) subject to constraints (2.2), (2.3), and

$$\sum_{i \in I} \sum_{u \in U} x_{iu} \omega_i \leq \alpha, \tag{2.4}$$

where ω_i measures the popularity of item i as the ratio of the number of ratings item i received to the total number of ratings of all items in the system. Constraint (2.4) sums the popularity values of the recommended items and forces the sum to be at most α, a tuning parameter that can be adjusted based on the needs of the system. At one extreme, if α is very large then the selected items can be popular without exceeding threshold α and the system can focus on maximizing the average ratings. As we decrease α, the system is forced to make trade-offs and recommend some items with equal or lower ratings that are also less popular (more novelty). Choosing values for α is very intuitive. If we somewhat

care about popularity, the average of ω_i times $k|U|$ can be used as a starting α value. Select a smaller value of α to offer less popular items. Constraint (2.4) is called a *knapsack constraint* in the literature [Martello and Toth, 1987], and one big advantage of using this simple structure is that off-the-shelf optimization programs such as Gurobi are very efficient in solving this common structure. We evaluate the model with the *average recommended popularity* over all lists (ARP), and aggregate diversity (Agg. Div.), which is the number of unique recommended items [Adomavicius and Kwon, 2011a]:

$$ARP = \frac{1}{|U|} \sum_{u \in U} \sum_{i \in L_u} \frac{\omega_i}{|L_u|} \tag{2.5}$$

$$\text{Agg. Div.} = \frac{1}{|I|} \left| \bigcup_{u \in U} L_u \right| \tag{2.6}$$

where L_u is the set of all items recommended to user u. Smaller values of ARP are desirable because they indicate lower popularity (more novelty). Higher values of Agg. Div is desirable because it shows the algorithm has covered a larger number of unique items in its recommendations.

2.1.2.3. Provider Fairness Model. In multi-stakeholder contexts such as a retail platform, provider fairness ensures that different *providers* (e.g., vendors) receive some minimum threshold number of recommendations. We assume that items are partitioned into *groups*. For example, items on a retail platform could be grouped by vendor or news articles could be grouped by publisher (e.g., Fox News, MSNBC, CNN, etc.). Let G_s be the set of item indices in group $s \in S$, where S is the set of all groups. Similar to Pop-Opt, provider fairness can be expressed as a constraint. Our *provider fairness optimization* (Fair-Opt) model uses the same objective function (2.1) as the base, subject to constraints (2.2), (2.3), and a new

one that imposes a lower and upper bound (both can be tuned by the system designer) on the number of times items recommended from each group:

$$\psi_s C_s \geq \sum_{i \in G_s} \sum_{u \in U} x_{iu} \geq \gamma_s C_s \quad (\forall s \in S) \tag{2.7}$$

where $C_s = \frac{|G_s|}{|I|} \cdot k|U|$ is the fraction of items in group s times the total number of items recommended. Tuning parameters γ_s and ψ_s control the lower and upper bounds for the number of times items recommended from group s.

The literature on fairness usually only considers the lower bound [Sürer et al., 2018, Patro et al., 2020]. Without upper bounds, however, some items can be offered significantly more frequently than the rest, which creates an unfair distribution of recommendations across items. We choose upper bound parameter $\psi_s C_s$ as $\lfloor 1 + (2 - \gamma_s)C_s \rfloor$, which depends on γ_s. Different upper bounds can be selected according to the needs of the system.

We now discuss the selection of the number of groups. On one extreme, there could be one group with $|G_1| = |I|$ and $C_1 = k|U|$, which is the total items recommended, and the constraint would have no effect (for reasonable values of ϕ_1 and γ_1). The other extreme is where each item is a separate group, i.e., every provider has one item in the system. This case is called the *item fairness problem*, where all $\gamma_s = \gamma$, $\forall s$.

We measure fairness with Z (Inequality in Producer Exposures) [Patro et al., 2020], which is defined as follows:

$$Z = -\sum_{s \in S} \left(\frac{R_s}{|U|k} \right) \log_{|S|} \left(\frac{R_s}{|U|k} \right), \tag{2.8}$$

where $R_s = \sum_{u \in U} 1(s \in L_u)$ is the total number of recommendations from group s. This metric is 1 when every provider gets the same number of recommendations, and decreases as the disparity between providers increases.

2.1.2.4. Diversity Model. Another consideration in creating top-k lists is to have *diversity* in that the recommended items are not highly similar to each other [Ziegler et al., 2005]. Our approach is to put items in distinct groups (based on their topic, genre, etc.), and constrain the number of distinct groups represented in each top-k list to be at least w, which is another tuning parameter under the control of the system designer. Our *diversity optimization model* (Div-Opt) is the same as the base model, i.e., objective (2.1) subject to (2.2) and (2.3), with additional constraints:

$$\sum_{i \in G_s} x_{iu} \geq y_{su} \quad (\forall s \in S, u \in U) \tag{2.9}$$

$$\sum_{s \in S} y_{su} \geq w \quad (\forall u \in U) \tag{2.10}$$

$$y_{su} \in \{0, 1\} \quad (\forall s \in S, u \in U) \tag{2.11}$$

We introduce a new decision variable y_{su} that indicates whether any items from group s are recommended to user u. Constraint (2.11) guarantees that the new decision variable y_{su} only takes values 0 or 1. Constraint (2.9) ensures that at least one recommendation is made from category s for user u, y_{su} can get value of 1. Otherwise, it is always fixed to 0. Constraint (2.10) ensures that the top-k list for each user u includes at least w distinct categories.

Note that in the Div-Opt formulation every item belongs to exactly one group, which we call the binarized version. This problem can be solved in an efficient manner through sorting [Malthouse et al., 2019b, Bradley and Smyth, 2001]. One solution is to sort through every

category separately according to the estimated ratings of the users, and recommend at least w items from separate categories. Next, we recommend items according to highest ratings until every user has exactly k items in their lists. We need Div-Opt when we mix and match different considerations to optimize combinations of metrics at the same time.

Our metric of choice for the diversity is ILS (Intra-list similarity) [Ziegler et al., 2005], defined as follows:

$$\text{ILS} = \frac{1}{|U|} \sum_{u \in U} \sum_{i \in L_u} \sum_{\substack{i' \in L_u \\ i' \neq i}} \frac{d(i, i')}{|L_u|(|L_u| - 1)}, \tag{2.12}$$

where L_u is the recommendation list of user u, and $d(i, i')$ is the distance between two items in the same list. We slightly modify the metric in [Ziegler et al., 2005], to put our ILS values between 0 and 1, where low values are desirable.

2.1.2.5. Combining different objectives. We combine all the different parts of the optimization models from the previous subsections. Unlike existing approaches, the beauty of our toolbox for solving different non-accuracy aspects of RS is that all constraints introduced so far can be included together. Our *combined model* (Comb-Opt) has the same objective function as in (2.1) subject to constraints (2.2), (2.3), (2.4), (2.7), (2.9), (2.10), and (2.11).

Our models use different ideas from the constrained optimization literature, including upper and lower bounding in Fair-Opt, weighted sums in Pop-Opt, and auxiliary variables in Div-Opt. Using these ideas, readers can include their own metrics with small modifications. Our framework can therefore accommodate different metrics if they can be modeled as constraints. Similarly, the metrics we have investigated can be modeled differently. Our main consideration is to use the same objective function for all the models and keep the constraints as simple as possible.

We now discuss some shortcomings of the combined model and provide potential ways of approaching them. First, some values of parameters w, α, γ, ψ may be *infeasible*, which means that there does not exist any solution satisfying all the constraints at the same time. One solution is to grid-search different parameters. Another solution is to penalize these constraints on the objective function whenever they are not satisfied. Second, creating decision variables for large models can require a lot of memory. In that case, some sort of approximation is required to make the models smaller. One way to do this is to fix some of the decision variables to 0 or 1 according to their estimated utilities before starting the optimization. Some item-user pairs with very low predicted ratings can be ignored at the start of the process, so they basically require no memory. To keep the models and the discussion compact, we leave these for future work.

2.1.3. Results

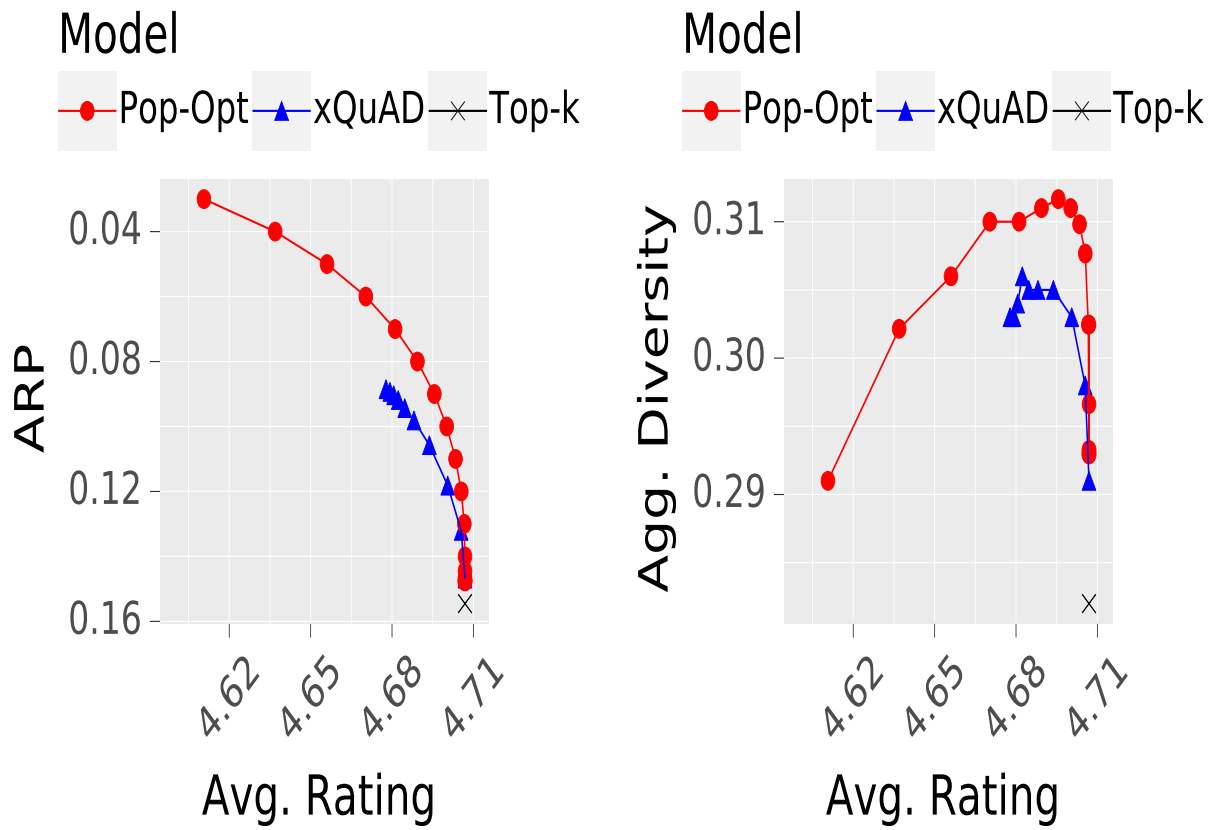
In this section, we compare our optimization model for solving the non-accuracy aspects of RS (popularity bias, diversity, and provider fairness) with various other models proposed specifically to solve each individual problem. We use the MovieLens 1M data set [Harper and Konstan, 2015] since it contains the genre of each movie, which is needed for the diversity problem. If a movie has more than one genre then we randomly assign one out of listed genres and keep the assignments consistent throughout. All the problems are solved using a laptop with Intel(R) Core (TM) i7-8750H CPU @ 2.20GHz 2.21 GHz, processor information, and with 16.0 GB of installed RAM. We use the Gurobi software [Gurobi Optimization, LLC, 2021] with optimization gap 10^{-4} throughout. Gurobi uses the branch & bound (B&B) algorithm, and it can have exponential time complexity [Morrison et al., 2016] in worst-case

scenario. However, Gurobi includes heuristics on top of B&B, and in practice the solution time performance is significantly better than exponential.

We apply the Singular Value Decomposition (SVD) [Koren et al., 2009] method implemented in the Python Surprise package [Hug, 2017] to estimate ratings for every user-item pair, although any RS algorithm could be used. We solve the optimization problems as a post-processing step using the predicted rating matrix. All our optimization models and benchmarks are solved using the same predicted ratings matrix. We set the size of the recommendation list given to each user to $k = 10$. Our code is available at GitHub: https://github.com/sseymen-tech/unified_toolbox.

2.1.3.1. Popularity Bias. We compare our Pop-Opt technique with the approach proposed in [Abdollahpouri et al., 2019a] (we label it $xQuAD$), which is one of the most efficient approaches proposed in recent years for controlling popularity bias. Figure 2.1 shows the average popularity of the recommendations (ARP) and the aggregate diversity for Pop-Opt and xQuAD for different values of α. From left to right, α changes from 0.03 to 0.16 with increments of 0.01. Larger α values give the same result because Constraint (2.4) is satisfied trivially. We can see that, for larger values of α, the Pop-Opt achieves a comparable average popularity to the xQuAD model with even slightly better average rating. Regarding aggregate diversity, we also see that our model outperforms xQuAD for larger values of α which, as we mentioned before, is a tunable parameter. Thus, Pop-Opt can achieve a comparable performance with the state-of-the-art technique to mitigate popularity bias.

2.1.3.2. Provider Fairness. For provider fairness, we compare our Fair-Opt with FairRec [Patro et al., 2020], which is specifically designed for this task. Figures 2.2 and 2.3 show the fairness metric Z and average rating for both techniques. From left to right the γ parameters used in both algorithms are $(0.99, 0.95, 0.9, 0.8, 0.5, 0.3, 0)$. We observe that for both relaxed



ARP metric comparison

Agg. Diversity metric comparison

Figure 2.1. Comparison of xQuAD with Pop-Opt

and strict upper bound choices, Fair-Opt beats FairRec in both average rating and fairness metrics. When we compare Fair-Opt results with and without upper bounds in Figures 2.2 and 2.3, we notice that upper bounds improve fairness value Z without reducing the average rating of the recommendations.

2.1.3.3. Diversity. We compare our diversity model (Div-Opt) with one of the most common ways of improving diversity in recommendation lists that uses a simple weighted sum of relevance and Intra List Similarity (ILS) [Bradley and Smyth, 2001] (It is denoted by WS in the plot). In Figure 2.4, from left to right Div-Opt parameter w takes values

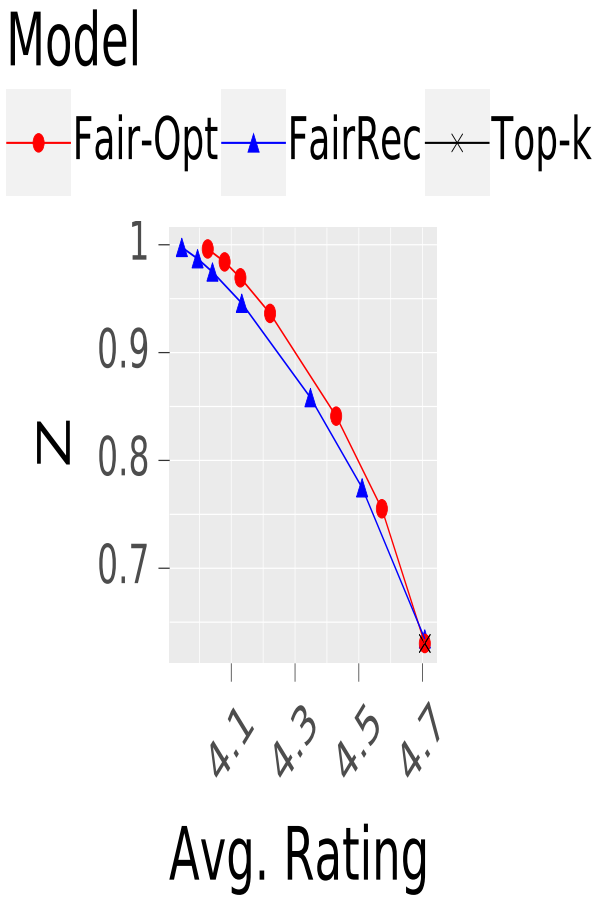


Figure 2.2. Fairness Results without upper bounds

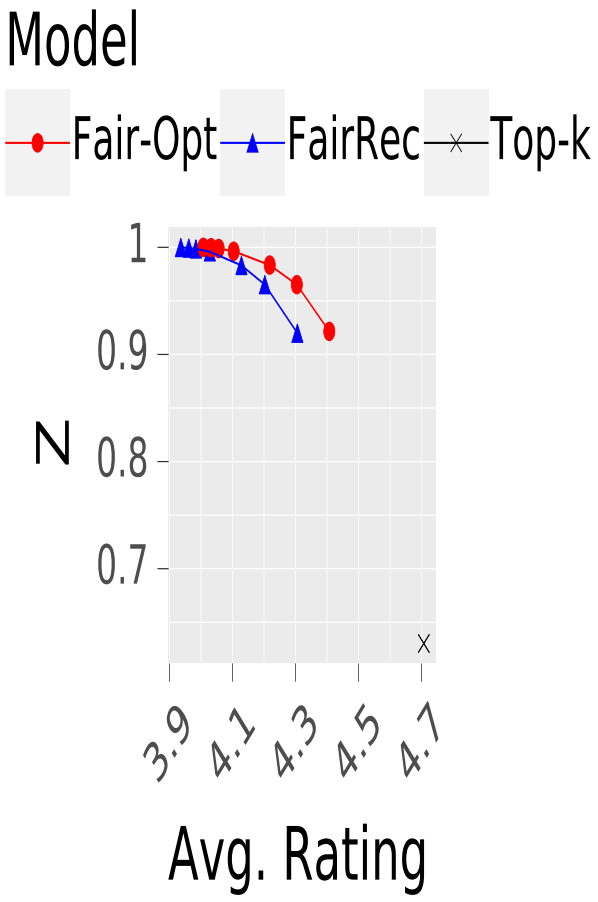


Figure 2.3. Fairness Results with upper bounds

$(10, 9, 8, 7, 6, 5, 0)$. From 5 to 0 solutions are the same. for $w = 10$, we achieve ILS=0, because every user is recommended one item from every distinct category. Thus, Div-Opt can increase the number of distinct recommended genres to every user without a significant decrease in average rating. Div-Opt has achieved a comparable performance to the WS algorithm when $w = (9, 8, 7, 6, 5, 0)$ yet outperforms it for $w = 10$ as its ILS is close to zero but has a higher average rating.

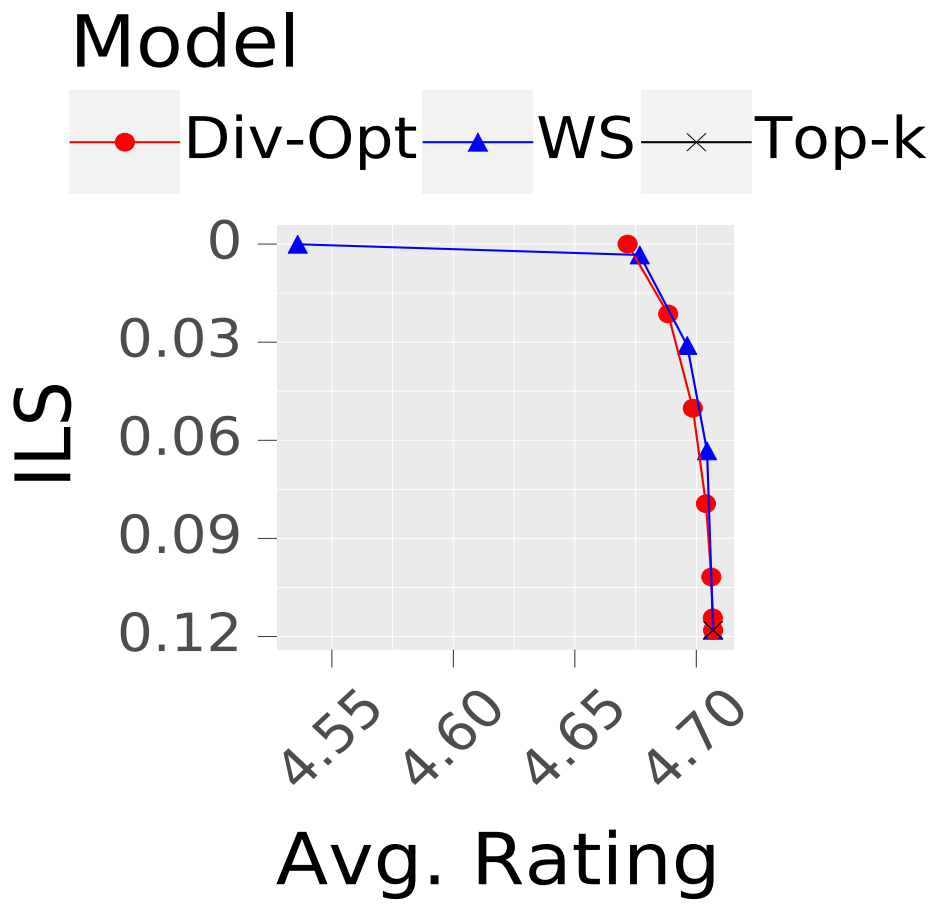


Figure 2.4. Diversity Results

2.1.3.4. Combined Model Results. Table 2.1 exhibits the results of our Comb-Opt model. Algorithms that optimize for a specific metric should perform well on that metric, but Comb-Opt can achieve great performance on ILS, Z and ARP without significantly lowering the average rating value. The results for xQuAD with lowest and highest popularity metric ARP (highest and lowest average rating respectively) are reported. Similarly, the results for WS for lowest and highest ILS are reported. For the item fairness metric, we remove upper bounds from (no ψ parameter) both FairRec and Comb-Opt, except with the solutions with superscript $*$, which represent the solutions with upper bounds.

Model	w	γ	α	ILS	Z	ARP	Avg. Rating	Agg. Div.
Comb-Opt	1	0.9	0.09	0.142	0.969	0.067	4.128	1.0
Comb-Opt*	6	0.9	0.055	0.129	0.999	0.053	4.055	1.0
Comb-Opt	7	0.1	0.085	0.080	0.668	0.085	4.672	1.0
Comb-Opt	7	0.5	0.08	0.089	0.843	0.079	4.421	1.0
Comb-Opt	7	0.1	0.08	0.080	0.668	0.080	4.669	1.0
Comb-Opt	7	0.5	0.05	0.092	0.838	0.050	4.393	1.0
Comb-Opt	7	0.99	0.05	0.104	0.994	0.050	4.017	1.0
Comb-Opt	8	0.8	0.1	0.058	0.934	0.076	4.206	1.0
Comb-Opt	9	0.1	0.08	0.022	0.658	0.080	4.653	1.0
Top-k	-	-	-	0.118	0.630	0.155	4.707	0.282
FairRec	-	0.8	-	0.143	0.946	0.077	4.133	1.0
FairRec	-	0.99	-	0.147	0.998	0.054	3.945	1.0
FairRec*	-	0.99	-	0.147	0.999	0.052	3.936	1.0
xQuAD	-	-	-	0.103	0.629	0.089	4.678	0.303
xQuAD	-	-	-	0.118	0.634	0.147	4.707	0.291
WS	-	-	-	0	0.698	0.118	4.536	0.351
WS	-	-	-	0.118	0.634	0.147	4.707	0.291

Table 2.1. Results of Comb-Opt with different parameter choices

Note that in Table 2.1 Comb-Opt*($w = 6, \gamma = 0.9, \alpha = 0.055$) beats all FairRec results in all metrics, except ARP of FairRec*. Comb-Opt($w = 7, \gamma = 0.1, \alpha = 0.08$) beats both xQuAD solutions in all metrics except average rating. Comb-Opt loses a bit of average rating but improves all the other metrics.

Recall that high Z, Agg. Div., and ARP values, and low ARP and ILS values are desired. In our experiments with Comb-Opt, we see that all metrics can be improved simultaneously. If one does not care about a certain metric, the constraints for that metric can be ignored in the combined model. Parameters can overwrite each other, for example in Comb-Opt with $w = 1, \gamma = 0.9$ and $\alpha = 0.09$, we have a relatively low $ARP = 0.067$ when our bound is 0.09. This happens because when γ goes to 1, every item is recommended proportionally close to each other, which gives non-popular items to be recommended as frequently as popular ones. Therefore, the $\alpha = 0.09$ bound is trivially satisfied by the fairness constraints. However, this

is normal since we are dealing with different problems at the same time, and it is expected that they have some effect on each other.

Parameter choice plays a significant role, and while optimizing more than one metric, grid-search of parameters are suggested to see the behavior of the data. Comb-Opt with parameters $w = 7, \gamma = 0.5$, and $\alpha = 0.08$, for example, seem to find a good balance of all metrics. If some metrics are not important in the given problem, their corresponding constraints can be discarded. For example, if diversity is not important, Comb-Opt with parameters $w = 1, \gamma = 0.9$, and $\alpha = 0.09$ finds a solution with good fairness Z and popularity ARP metrics at the same time. Overall, parameter choice can be made according to the needs of the system.

Overall, Comb-Opt tends to improve all the metrics at the same time. If the RS goal is to alleviate popularity bias, fairness, and diversity metrics, we suggest adding all the constraints proposed in this chapter. However, if one does not care about fairness, then it makes sense to remove the fairness constraints from the Comb-Opt. Then we immediately have a solution that can alleviate popularity bias and diversity metrics. On the other hand, the constraints proposed in this chapter can be removed and new ones can be added easily, according to the needs of the specific RS.

2.1.4. Conclusions and discussion

We propose an optimization toolbox for solving different types of non-accuracy problems. This method focuses on finding the optimal solution of the system given different constraints, which aim to solve various non-accuracy problems such as popularity bias, provider fairness, and diversity. We show that, all these different metrics can be considered at the same time

while generating recommendations, and they can even beat algorithms specialized for the specific problems.

One downside of our model is its memory requirement. Therefore, for larger problems, scaling of the models is an issue. There are possible solutions to these problems, such as focusing on sub-optimal solutions which are close to optimal, fixing values to some decision variables before starting to optimization and so on. We leave the problem of finding good approximations of Comb-Opt for future work.

Our toolbox can be applied to many other problems. For example, retailers concerned with stock-outs can add upper bounds for how often an item is offered. Likewise, a platform with perishable items (e.g., fresh produce or meat, hotel rooms, airline seats) could add a lower bound so they are offered more frequently. In situations with sponsored recommendations, the manufacturer of the item may have a maximum advertising budget that it is willing to spend, which could be handled by adding upper bounds. Seymen et al. [2021a] applies a similar approach to the top-k list calibration problem. All these individual problems and various combinations of them and can easily be solved simply by modifying the constraints or adding new ones. Thus, post-processing optimization models offer a flexible toolkit for managing RS involving multiple objectives and/or stakeholders.

2.2. A Constrained Optimization Approach for Calibrated Recommendations

2.2.1. Introduction

Recommender systems (RS) are important tools to help users find relevant and interesting items from a potentially large pool of candidates. While the initial focus of RS evaluation was on the accuracy of the recommended items, other criteria such as fairness [Wang and Joachims, 2021, Patro et al., 2020], popularity [Abdollahpouri et al., 2017a, 2019b], diversity [Kaminskas and Bridge, 2016, Mansoury et al., 2020] and others [Ge et al., 2010, Kotkov et al., 2018] have attracted attention due to their crucial role in the success of the RS. One topic that recently has gained attention is *calibration*. Assume items are classified into topics. Simply, a calibrated recommendation list for each user is the one that consists of items in the same proportion of the topics the user has previously liked. For example, in a movie recommender, if a user's previous history contains 20% drama, 60% action, and 20% romance movies, a calibrated recommendation list would match the previous proportions of genres. The goal of calibrated recommendations is to reflect the interests of a user in the recommended list in their appropriate proportions while also maximizing the sum of the ratings of the recommended items.

Even though the concept of calibration was discussed more than a decade ago [Bella et al., 2010], its importance was realized recently by Steck in [Steck, 2018], which proposed a greedy approach to iteratively construct a top-k list that is calibrated and also relevant to the user. The greedy algorithm adds one item at a time to the list until k items are included for each user. Although greedy approaches for calibration often provide great improvements in terms of making the recommendations more calibrated, we show that they can result in inefficient solutions in that a better solution is missed because of their myopic nature. This

chapter contributes to the literature by suggesting an optimization model that can calibrate the recommendation lists and maintain a good level of accuracy at the same time. Due to the non-linearity of the Kullback-Leibler divergence (aka KL-divergence) measure used by Steck, we propose using a different distance measure and show it has experimental and intuitive advantages. The main advantage of using an optimization model is the possibility to consider items in a holistic manner. This way, all k items are considered at the same time, so that, intricate combinations of different items are considered before creating the top-k lists. Optimization models are also easy to modify, and we discuss how our model can easily include other considerations such as fairness or diversity.

We apply our approach to solve the calibration problem using the MovieLens dataset and compare our results with Steck's state-of-the-art heuristic algorithm. Experimental results show that our optimization technique outperforms the heuristic both in terms of making the recommendation lists more calibrated and maintaining as good or better level of accuracy. Additionally, we observe that our approach has secondary effects to reduce popularity bias and increase the fairness of the recommendations and results in fairer recommendations with less concentration on popular items. Although there are works applying optimization ideas to RS [Sürer et al., 2018, Malthouse et al., 2019b, Agarwal et al., 2011], to the best of our knowledge, ours is the first to solve calibration using a non-greedy (an example of a greedy approach is the one proposed in [Steck, 2018]) approach.

2.2.2. Calibration

This section discusses the different models used to solve the calibrated RS problem for top-k lists, where every user is recommended exactly k items. We also propose a multi-objective optimization model to solve the calibration problem.

Let I be defined as the set of items, U the set of users and G the set of *types* (e.g., genres) we want to calibrate. Define $\hat{r}_{iu}$ as the estimated rating of each item $i \in I$ offered to user $u \in U$. Note that the total number of recommendations made is $k|U|$. Items are grouped into *types*. For example, in the case of movies the types could be genres, or for news articles the types could be topics. When an item belongs to multiple types (e.g., a movie can have multiple genres) we set equal probabilities to each type for that item. Let I_u denote the set of items rated by the user u in our training set, and let $L(u)$ be defined as the list of items recommended to user u where $|L(u)| = k, \forall u \in U$.

Next, define $p(g|u)$ as the proportion of items that user u rated in the past of type g. It represents the distribution of the user's preference of each type considering user's past history. Then, $q(g|u)$ is the proportion of items of type g we recommend to user u. Assuming equally weighted types, our distributions can be represented as follows:

$$p(g|u) = \frac{\sum_{i \in I_u} p(g|i)}{|I_u|} \quad (\forall g \in G, u \in U) \tag{2.13}$$

$$q(g|u) = \frac{\sum_{i \in L(u)} p(g|i)}{|L(u)|} \quad (\forall g \in G, u \in U) \tag{2.14}$$

Similar to Steck [2018], we define $p(g|i)$ as the fraction of items for each type g. Therefore, if an item has two types, $p(g|i)$ is 0.5 for those two types and 0 for the rest. Calibration is generally an NP-hard problem, because the problem is combinatorial. It is therefore not trivial to find the best set of items that matches the user's past history distribution p, while also offering items with high estimated ratings.

2.2.2.1. Steck's Heuristic Model. For the calibration problem, Steck uses the KL-distance of distributions $p(g|u)$ and $q(g|u)$ [Steck, 2018]:

$$C_{KL}(p,q) = \sum_{g \in G} p(g|u) \log \left(\frac{p(g|u)}{\dot{q}(g|u)} \right), \tag{2.15}$$

where $\dot{q}(g|u) = (1-\alpha)q(g|u) + \alpha p(g|u)$ with small values of α to make the equation well defined when $q(g|u) = 0$. We drop types where $p(g|u) = 0$ from the summation to alleviate the divergence problem when $q(g|u) = 0$. In this chapter, we used $\alpha = 0.01$ as in Steck [2018]. Smaller values of $C_{KL}(p,q)$ are desirable because they indicate the distributions p and q are close to each other, while larger values indicate the distributions are far away.

Steck [2018] proposed the following objective function for each user u:

$$\max_{L(u), |L(u)|=k} \left((1-\lambda) \sum_{i \in I} \hat{r}_{iu} - \lambda C_{KL}\left(p, q(L(u))\right) \right). \tag{2.16}$$

For every user u, equation 2.16 is maximized to find the best overall recommendation list $L(u)$. Therefore, $C_{KL}(p, q(L(u)))$ values are penalized by λ and the scores of the items are promoted by $(1-\lambda)$ where $\lambda \in [0,1]$. We use a grid-search for obtaining λ. In Steck [2018], equation 2.16 is solved in a greedy manner, where every user u starts with an empty list of recommendations $L(u)$. Then, in every iteration the item i with the highest value in equation 2.16 is added to the $L(u)$ until k items are recommended to every user u in the system. Note that as with other greedy approaches, making a decision in every step results in myopic, and sometimes suboptimal solutions [Pearl, 1984, Wilt and Ruml, 2016]. An item that was the best option before can lose its desirability after adding other items to the list.

2.2.2.2. Potential Drawback of Heuristic Calibration. Before discussing our optimization technique for calibration, we demonstrate a problem with greedy heuristics such

as Steck [2018], Oh et al. [2011] to show the value that our approach can bring. Assume that we are solving the top-k calibration problem with $k = 3$ for a single user. There are six items and three genres in the system: A, B and C. This user watched every genre with equal probability in the past, so $p(g|u) = [1/3, 1/3, 1/3]$. We give the estimated rating and genre information of the six items in the system for this user in Table 2.2. For $\alpha = 0.01$ and $\lambda = 0.9$, the best solution is to offer items $[1, 2, 3]$, with total estimated ratings 13.5 and KL-distance 0. Steck's heuristic offers the list $[6, 4, 3]$ with total estimated ratings 11.7 and KL-distance 0.026. This happens because in first step of the greedy approach, Item-6 maximizes equation (2.16), although it has a low estimated rating for that user.

Item	Estimated Rating	Genre
Item-1	4.5	[A]
Item-2	4.5	[B]
Item-3	4.5	[C]
Item-4	4.2	[A, B]
Item-5	4.2	[A, C]
Item-6	3	[A, B, C]

Table 2.2. Toy example for a single user

2.2.2.3. Optimization Model. Our model must decide which k items to recommend to each user. For this purpose, let decision variable $x_{iu} = 1$ if item i is recommended to user u, and 0 otherwise (Constraint 2.19). For every user u, exactly k of the x_{iu} values should equal 1 (Constraint 2.18). Ignoring calibration for now, the only criterion is to maximize the

estimated ratings $\hat{r}_{iu}$. This model can be represented as follows:

$$\max_{x} \quad \frac{1}{k|U|} \sum_{i \in I} \sum_{u \in U} \hat{r}_{iu} x_{iu} \tag{2.17}$$

$$\text{subject to:} \quad \sum_{i \in I} x_{iu} = k \quad (\forall u \in U) \tag{2.18}$$

$$x_{iu} \in \{0, 1\} \quad (\forall i \in I, u \in U) \tag{2.19}$$

Note that this problem can be solved by sorting the values of $\hat{r}_{iu}$ for every user u and recommending the first k items. Next, we add calibration to this base model and solve a multi-objective optimization model.

As mentioned earlier, due to the non-linearity of KL Divergence measure, we use the *total variation* [Levin and Peres, 2017, Chambolle, 2004] between two probability measures P and Q, which is defined as follows when both measures are defined on a finite set:

$$\text{Total Variation}(P, Q) = \frac{1}{2} \|P - Q\|_1 = \frac{1}{2} \sum_{g \in G} |P(g) - Q(g)| \tag{2.20}$$

We use the total variation distance because it is intuitive and efficient to solve in a commercial optimization software. Notice that the p and q distributions come from a finite set since there is a finite set of items in the system. Total variation is the ℓ_1 norm between the distributions [Levin and Peres, 2017], and we try to minimize this distance for every user. Total variation gives a linear objective function, which commercial optimization software can efficiently solve. Next, we add an intuitive adjustment to the total variation formulation:

$$\text{Weighted Total Variation}(P, Q) = \frac{1}{2} \sum_{g \in G} \zeta(P(g)) |P(g) - Q(g)| \tag{2.21}$$

Equation 2.21 includes a general function $\zeta(P(g))$ of users' proportion of genres in the past history P. Genres g with large values of $p(g|u)$ indicate that user u is more inclined to prefer items from genre g. Therefore, we want to penalize differences with large p values more, so that we offer the types which the user prefers the most. Therefore, ζ should be a non-negative (because calibration problem aims to minimize the distance) and non-decreasing function of P. In our experiments we use $\zeta(x) = x + 1$, which is an increasing, non-negative function that takes values between 1 to 2. This function can be tuned differently for other data sets, e.g., an exponential function can be used if one wants to increase the penalty for higher p values, or different scale depending on the distance metric and the range of ratings.

The optimization model we solve that considers calibration, which we name *Calib-Opt*, includes Constraints (2.18) and (2.19) with the following objective function:

$$\max_{x} \quad \frac{1}{k|U|} \sum_{u \in U} \left[(1-\lambda) \sum_{i \in I} \hat{r}_{iu} x_{iu} - \lambda \sum_{g \in G} \frac{\zeta(p(g|u))}{2} \Big| p(g|u) - \sum_{i \in I} x_{iu} \frac{p(g|i)}{k} \Big| \right], \tag{2.22}$$

where $\sum_{i \in I} x_{iu} \frac{p(g|i)}{k} = q(g|u)$ is the proportion of items of type g recommended to user u. As with Steck's algorithm, parameter λ is the weight given to calibration and $(1 - \lambda)$ is the weight given to estimated ratings. Larger values of λ imply more emphasis on calibration. In our illustrative example in Table 2.2, optimization works holistically and considers all combinations at the same time to obtain the best solution $[1, 2, 3]$.

Our optimization model is scalable. We solve objective function (2.22) for every user separately, so the number of users do not hinder our model. We can solve our model with very large item sets by only considering items that have the highest estimated utility (e.g., top 1000 items) which in practice works efficiently [Bradley and Smyth, 2001].

2.2.3. Experimental Setting

Our experiments use the MovieLens 1M data set. We remove movies rated fewer than 10 times, leaving 3260 movies, 6040 users and 18 item types (genres). We divide the remaining ratings data to 80% training and 20% test set and use the training set to estimate the rating matrix with the Singular Value Decomposition (SVD) implemented in the Python surprise package [Hug, 2020b, 2017]. The ratings of the items are explicit, and take values between 1 and 5.

Our goal is to compare Calib-Opt with Steck's algorithm. For KL-distance Equation (2.15), we use $\alpha = 0.01$, consistent with Steck. For optimization, we use the Gurobi software [Gurobi Optimization, LLC, 2021] and 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 [Morrison et al., 2016]. However, because of the algorithms and heuristics included with branch&bound, Gurobi performs significantly better than exponential complexity in practice.

We compare the algorithms on Precision, Recall, KL-distance (Equation 2.15), and Total Variance (Equation 2.20). Additionally, we now define the evaluation metrics for fairness Z (Inequality in Producer Exposures) [Patro et al., 2020], popularity ARP (Average Recommendation Popularity) [Abdollahpouri et al., 2019a] and Aggregate Diversity (Agg. Div.) [Adomavicius and Kwon, 2011a]:

$$\text{Z} = -\sum_{i \in I} \left(\frac{R_i}{|U|k} \right) \log_{|I|} \left(\frac{R_i}{|U|k} \right) \tag{2.23}$$

$$\text{ARP} = \frac{1}{|U|} \sum_{u \in U} \sum_{i \in L(u)} \frac{\omega_i}{|L(u)|} \tag{2.24}$$

$$\text{Agg. Div.} = \frac{1}{|I|} \left| \bigcup_{u \in U} L(u) \right| \tag{2.25}$$

In Equation (2.23), $R_i = \sum_{u \in U} 1(i \in L(u))$ gives the total number of times item i is recommended. If every item is recommended exactly the same number of times then $Z = 1$, but as the inequality between item recommendations increases Z decreases. Therefore, larger Z values are desirable. In Equation (2.24), ω_i is the number of times item i is rated by the users. Lower popularity (higher novelty) solutions have lower ARP values, which are more desirable for alleviating the popularity problem. Agg. Div. in Equation (2.25) calculates the proportion of items recommended at least once, and a value 1 indicates that every item is recommended at least once.

2.2.4. Results

Figures 2.5–2.11 compare Calib-Opt and Steck's heuristic on different metrics as a function of λ values, where Calib-Opt is labeled "Opt" and the heuristic is labeled "H." We begin with accuracy. Figures 2.5 and 2.6 show that precision and recall have inverted-U shaped curves that increase up to a certain value of λ and decrease thereafter. We believe the reason for the inverted-U shape is that making recommendations more calibrated can help with recommending items that match user's interests, which in turn improves the accuracy of the recommendations as well. However, when λ is large there will be too little emphasis on the predicted ratings, resulting in a loss in accuracy. Comparing the approaches, we can see that

Calib-Opt has better precision and recall for most λ values, which shows the superiority of our approach in giving more accurate recommendations.

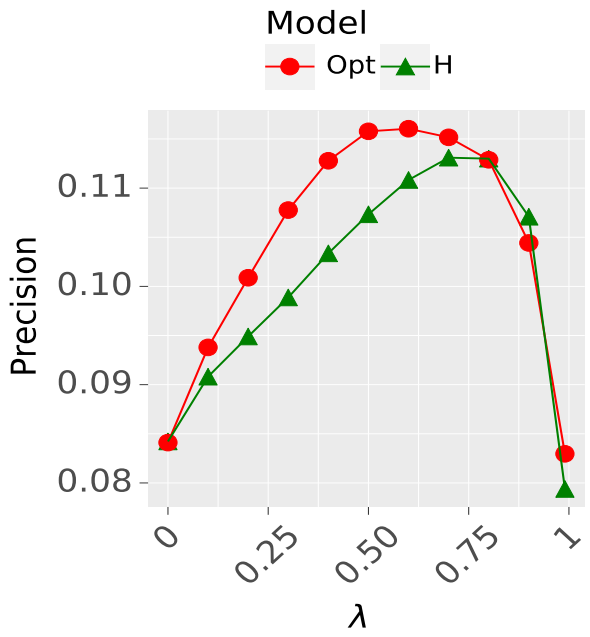


Figure 2.5. Precision value

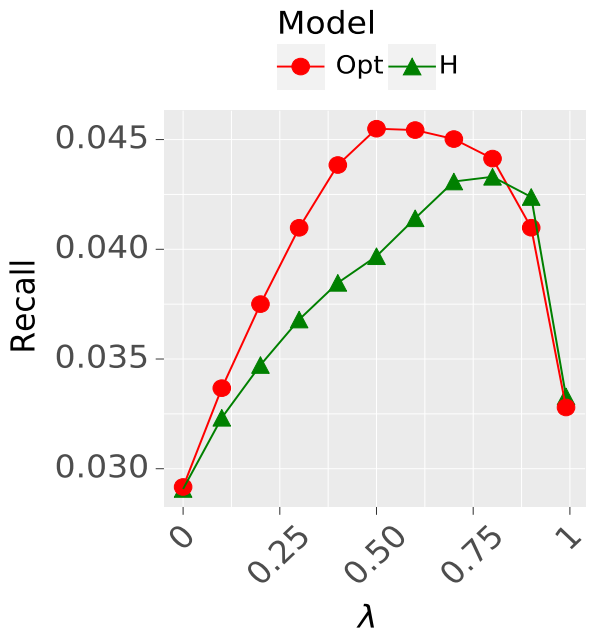


Figure 2.6. Recall value

Next, we discuss calibration metrics. Even though Calib-Opt uses the Weighted Total Variation distance, Figure 2.7 shows that it also outperforms the heuristic on KL-divergence. This can be explained by the fact that two distances implemented are highly correlated [Pinsker, 1964]. Figure 2.8 shows that Calib-Opt always performs better on Total Variation. Taken together, these results indicate that Calib-Opt better matches the distribution of genre preferences.

We now study whether Calib-Opt offers secondary benefits on fairness, popularity and aggregate diversity. Figure 2.9 shows that Calib-Opt dominates the heuristic on fairness across λ values. Likewise, Figure 2.10 shows that Calib-Opt has lower values of ARP across λ. Larger λ values should create lists with a larger variety of personalized items for the user.

We observe, however, that the greedy heuristic perpetuates these biases we would like to avoid. Moreover, Figure 2.11 shows that Calib-Opt gives better value of Agg. Div. than the heuristic. For $\lambda \in [0, 0.8]$ Calib-Opt has substantially better Agg. Div. than the heuristic indicating that Calib-Opt recommends a higher proportion of items at least once from the item set.

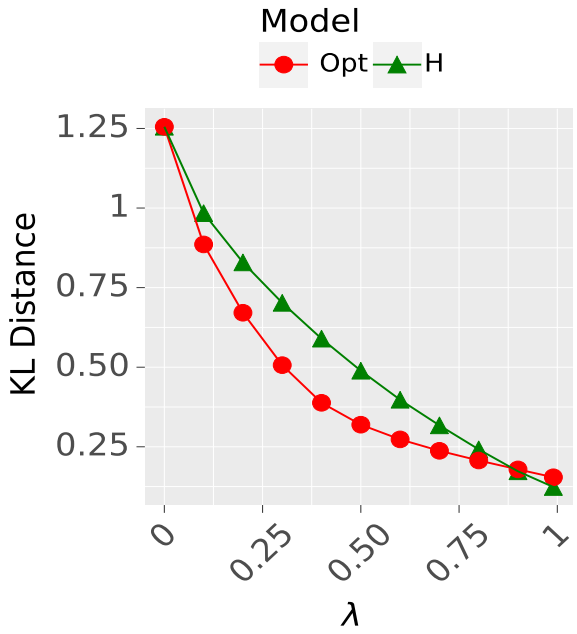


Figure 2.7. KL-divergence value

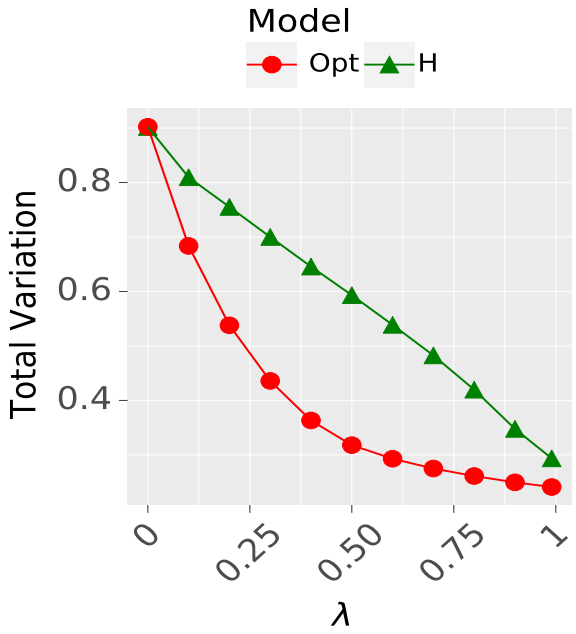


Figure 2.8. Total Variation value

We now attempt to explain why greedy approaches exacerbate the popularity and fairness biases by investigating how the heuristic selects items. Figure 2.11 shows that with increasing λ values, aggregate diversity increases. The heuristic approach seems to struggle with recommending different items, at least compared to the Calib-Opt. Most notably, for $\lambda \in [0, 0.8]$ the heuristic approach does not improve aggregate diversity significantly. A low aggregate diversity metric means that the heuristic approach recommends only a small subset of the items in the system. This might explain why popularity metric is not improving.

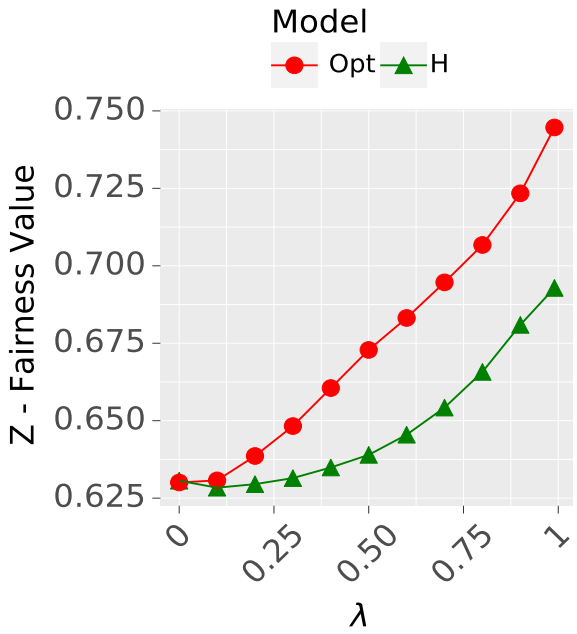


Figure 2.9. Fairness Z - value

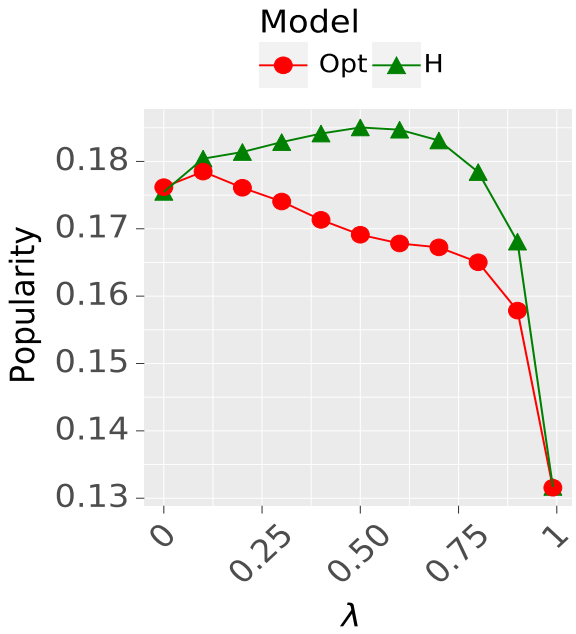


Figure 2.10. Popularity value

The same "popular" items are recommended, and there is little change in aggregate diversity values in heuristic approach. Calib-Opt, however, has better popularity and aggregate diversity metrics. We think this is the result of considering the lists holistically.

Figure 2.12 shows the total number of times an item is recommended across all users, plotted against the number of genres the item has. "Fair" shows the number of recommendations if every item is recommended in almost equal amounts (fairest), "First" and "Last" show the total number of times items are recommended in first and last (item k) iterations respectively, considering the number of genre tags. We investigate the quality of the solution for $\lambda = 0.8$ because it has the best accuracy metrics. For the first item selected by the heuristic, items with three to six genres are over-recommended while one and two genres are under-recommended. This happens because when the list is empty, the decrease in KL-distance is greater when we include items with more genres as we discussed in Section 2.2.

This bias creates unfairness, because, items with more genres are represented more than the items with fewer genres. This problem is alleviated as we add more items to the recommendation lists, because calibration of the list only marginally changes when we add the last item compared to the first one.

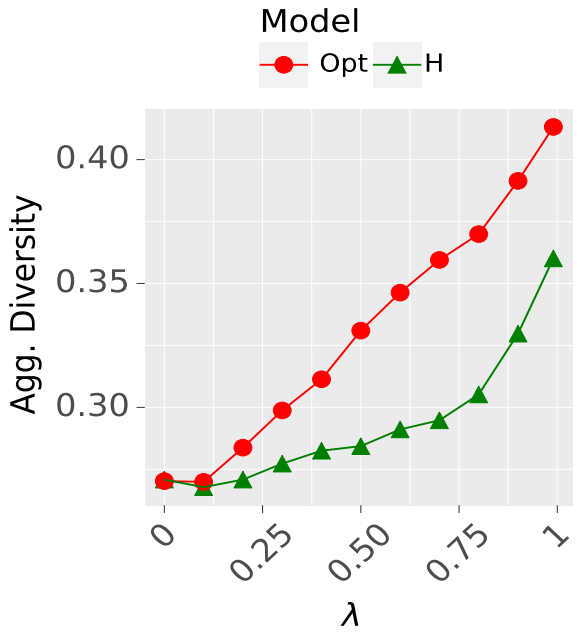


Figure 2.11. Aggregate Diversity value

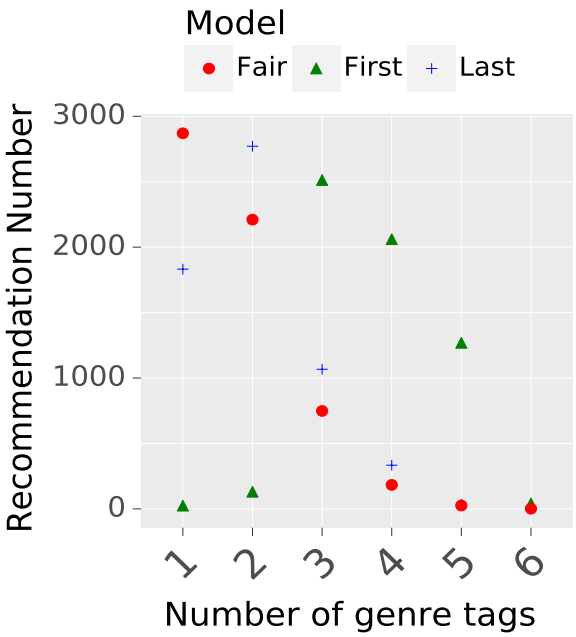


Figure 2.12. Heuristic solution for $\lambda = 0.8$

2.2.5. Conclusion

In this chapter, we proposed and solved a multi-objective constrained optimization model to solve top-k RS problem with calibration. We showed that our optimization model was easy to scale and beneficial to use, making significant improvements to the state-of-the-art heuristic approaches in the literature. We propose a small example that illustrates why the heuristic fails to find the optimal solution. Additionally, our model improved popularity and fairness metrics as a by-product of considering the recommended list holistically.

Solving an optimization model for calibration offers another benefit. Our optimization model only makes changes in the objective function, therefore, it is easy to modify and adapt for other metrics frequently discussed in RSs. Popularity, fairness can be optimized directly, instead of improved as a by-product. Similarly, other metrics such as diversity and inventory considerations can be added to our model as different constraints. We leave the implementation of these additional ideas on top of calibration metric as a future research direction.

CHAPTER 3

Making Smart Recommendations for Perishable and Stockout Products

3.1. Introduction

With about one-third of food lost or wasted, avoiding food waste has become a global challenge due to its environmental and economic impact [Blakeney, 2019]. One out of four calories of food intended for consumption was wasted in 2009 [Lipinski et al., 2013]. Additionally, 28% of the items sold by retailers are wasted because they expire without other flaws [Lebersorger and Schneider, 2014]. Reasons for waste include spillage, spoilage, and a significant decrease in the quality of the product. Wasted food emits significant amounts of greenhouse gas to the air [Stavi and Lal, 2013], deforests the Amazon [Marchand, 2012], and causes significant negative effects on the climate with very little in return.

Retail food waste occurs when supply exceeds demand. A simple solution is for retailers to order less and thus decrease supply. This, however, increases the number of stockouts and consequently reduces customer goodwill [Anderson et al., 2006]. While the stockout problem has been studied for many years [Jing and Lewis, 2011], it has become more critical recently because of supply chain disruptions due to extreme weather events, port congestion, labor relations [Goodman, 2022], COVID-19 lockdowns, manufacturing delays and human errors (e.g., Suez Canal being blocked) [Zimmerman, 2021]. Increasing inventory can cause higher holding costs and lower clearance sale pricing, which results in lower overall revenue for the retailer [Smith and Achabal, 1998].

Although the decision maker has a notion of what to expect in the future, customer demand and whether they would buy a recommended item is not deterministic. Consequently, inventory levels are also stochastic since the decision maker does not know in advance how much demand will occur on a specific day and how much stock will be left at the end of the day. Additionally, inventory levels might not be recorded correctly, or a set of items might

arrive sooner or later than expected, causing the inventory to fluctuate unexpectedly. Some soon-to-perish items might go bad sooner or later than expected according to the storage conditions and the type of item [Hertog et al., 2014]. We propose a novel solution using recommender systems (RS) that uses a personalized top-k list to direct demand toward items that will perish soon and away from items that are nearing stockout while accounting for the above-mentioned uncertainties. Therefore, we make recommendations that not only consider consumer preferences, but also the status of the available stock. Thus, we show how to use RS to manage inventory, integrating inventory management, RS, and promotions literatures.

In the RS literature, earlier works mostly focused solely on user preferences. More recent works started incorporating the needs of the item providers, the RS platform/system itself, and other stakeholders. Multistakeholder recommender systems (MRS) are introduced as systems that include the objectives of parties other than the users. These objectives may conflict with each other, and solutions recommended by MRS algorithms take all these objectives into account while creating the recommendation lists that consist of items offered to the users [Abdollahpouri et al., 2020]. Abdollahpouri et al. [2017b, 2020] discuss multiple real-world problems that can be solved using MRS algorithms. For example, in problems that are encountered in an e-commerce retailer setting, the business and retailer considerations are intuitive to consider as well as the user considerations. A traditional RS algorithm could recommend the highest estimated utility items to maximize the user utility/rating. However, these solutions can cause problems in the system by increasing the stockouts and perishability significantly. In our work, the system is considered a stakeholder and aims to lower the number of perished items and stockouts. Similarly, the retailer is another stakeholder that wants to increase the sales revenue as much as possible from the possible revenue obtained from the item recommendations. Consequently, we offer a model that

solves an e-commerce retailer MRS problem [Burke and Abdollahpouri, 2017] by including system (sustainability, stockouts) and the provider (revenue) considerations on top of the users' considerations.

Given the proliferation of digital shopping environments such as websites and mobile shopping apps where many user behaviors can be recorded, more data is available to tailor promotions to the needs of individual users [Cheung et al., 2003]. Relevant interactions can be facilitated with an RS so that users receive personalized recommendations that match their preferences. Creating recommendations, however, is more complicated than recommending items of interest to a user because multiple objectives and stakeholders are involved [Abdollahpouri et al., 2020, 2017b]. While we are unaware of any research that uses multi-objective RS to avoid perishables and stockouts at the same time, there has been research that studied other retail objectives and stakeholders. Sürer et al. [2018] consider vendors on a retail platform as well as the preference of users when creating top-k lists. Seymen et al. [2021a,b] suggest MIP models that create top-k lists utilizing interactions of the users with vendors on a retail platform.

We contribute to the RS literature by proposing and solving a mixed-integer-programming (MIP) model that combines RS and inventory management. Our proposed model creates recommendation lists that consider both user and retailer perspectives, as well as the system-enforced perishability and stockout concerns. Additionally, our model accounts for the stochastic nature of perishables, demands, and inventory levels. By including stochasticity and considerations of the users, the retailer, and the system, we discuss the complex economical impacts of RS in an online retailer setting. As far as we know, our approach to this online retailing problem and our optimization model are novel additions to the RS literature. Furthermore, using ideas from our optimization model, we propose heuristics that improve

scalability efficiently. Finally, we discuss the results of our optimization model and heuristics and show that our approaches offer solutions that improve user, retailer, and system objectives simultaneously.

3.2. Related Works

This section surveys relevant literature from sales promotion, RS, and inventory management. There is extensive research in the sales promotion literature showing the effectiveness of email promotions, and RS articles have already investigated the effects of personalization in driving retail sales. Likewise, there is research in the inventory management literature showing the importance of alleviating the stockout and perishability issues, mostly focusing on the retailer. Our contribution is to combine these approaches to create personalized email promotions that consider both the user's preferences as well as retailer and supply chain objectives to avoid perishables and stockouts.

Sales promotion is a widely used strategy defined as an "action-oriented marketing event whose purpose is to have a direct impact on the behavior of the firm's customers" [Neslin, 2002, p. xvii]. Promotions include price discounts, feature advertising, and other touch points such as targeted emails, special displays, and coupons. Promotions stimulate market demand, enable retailers to sell excess items, and increase their revenue. There is a long literature studying sales promotions and their effects [Neslin, 2002], including some with a specific focus on e-commerce and email promotions. Most of the e-commerce promotions literature shows a positive link between sales promotions and increases in demand [Lewis and Reiley, 2014, Agrawal et al., 2020]. Sahni et al. [2017] conclude that personalized emails increase the total expenditure of customers by 37.2%.

Others have used RS to implement personalized shopping lists. Lin et al. [2005] show that timely recommendations resulted in sales growth. Malthouse et al. [2019a] solve the problem of sponsored recommendations and content in retailing including ad revenue and user utility. Kaminskas et al. [2017] implement recommendations by using text and co-occurrence based approaches. Wan et al. [2018] solve recommendations using natural language processing ideas. Chen et al. [2020] solve linear optimization-based algorithms, which offer items to users in a deterministic setting using RS considering user ratings with perishable items. Dadouchi and Agard [2018] propose an RS method that lowers stockouts. Specifically, RS can be used to increase the revenue of the firm by engaging customers to buy more in the immediate time period [Dias et al., 2008]. Our MIP model offers an exact solution to creating personalized shopping lists by incorporating both the user and retailer perspectives and also actively modifying the demands before the customer arrivals.

Perishable items and limited inventory appear in multiple works in the dynamic assortment and inventory control literature, but without considering how promotions can be used to manage them. Bernstein et al. [2015] solve which items should be offered to which users with limited inventory, not considering perishable items. Talebian et al. [2014] create dynamic assortments considering perishable items with infinite inventory. Amiri et al. [2020] maximize the number of perishable items sold considering one vendor and multiple buyers. Fan et al. [2020] consider both replenishment strategies and the pricing of the perishables. Chua et al. [2017] decide whether and how much to discount items using dynamic programming. Others solve problems with perishable inventory by incorporating partial information [Duan and Liao, 2013], uncertain demands [Gutierrez-Alcoba et al., 2017, Kırcı et al., 2019], or time delays happening in the supply chain [Dolgui et al., 2018]. Nguyen and Chen [2019, 2022] build a MIP model and consider perishability and stockouts in an inventory model

setting. Chen et al. [2020] propose a RS model with perishability in a deterministic setting without stockouts, Dadouchi and Agard [2018] propose an RS method that lowers stockouts, Nguyen and Chen [2019, 2022] propose a model with both perishability and stockouts, but not including user ratings/utility and item recommendation aspects of RS. We close this gap by developing a model that considers stochasticity, perishability, stockouts, user ratings and retailer revenue/sales simultaneously. Furthermore, we offer heuristics to improve the scalability of the suggested optimization model.

3.3. Problem Definition and Formulation

This section proposes a MIP model and a heuristic to solve an item recommendation problem considering both retailer and user perspectives. We maximize both the quality of the recommendations for the users and the impact on the retailer's profitability (perishability, sales). For this purpose, we first discuss the problem definition and assumptions. We then illustrate and explain our optimization model and heuristic.

3.3.1. Problem Definition and Optimal Solution

We consider an online retailer selling physical items to their users over the internet. The retailer periodically creates a personalized top-k promotion list such as a weekly email recommending items (I) to a known set of users (U); hereafter we use the RS term *user* instead of retailing terms *customer* or *consumer*. Therefore, we solve the problem of selecting which k items should be recommended to the users in the form of personalized email promotions. While creating top-k lists is a common RS task, we allow lists with fewer than k items to accommodate situations with very low inventory levels; hereafter we use the term "top-k list" with the understanding that some lists may have fewer than k items. We assume that

recommending item i to user u might result in a demand increase γ_{iu} for the said item. Note that the RS can increase demand without reducing the price and thus profit margin. We assume that the *retailer's purchase cost* of item i is c_i, and that the retailer earns profit $\rho \times c_i$ when it sells the item, where ρ is the *markup* and $p_i = (1 + \rho) \times c_i$ is the *revenue* (selling price). We allow for items to have different prices p_i but assume a constant markup ρ (this assumption can be easily relaxed).

When deciding which items should be recommended, the following four aspects are considered. Firstly, since the available in-stock inventory ($\boldsymbol{L}$) is limited, there is a possibility of stockouts if users are unable to buy items due to unavailability. When users attempt to purchase a recommended item i that is out of stock, we assume that the retailer incurs a penalty of q_i, which measures a loss of goodwill and revenue from the sale. Secondly, some items are perishable with quantity $\boldsymbol{E}$ considered *soon-to-perish* ($0 \leq E_i \leq L_i, \ \forall i \in I$). Soon-to-perish items must be sold as soon as possible; otherwise, they will be discarded since the retailer cannot sell expired items. For any discarded item, the retailer incurs a penalty c_i (i.e., there is no cost to dispose of the perished items nor salvage value). We also assume that items are shipped following the FIFO (first-in-first-out) method, where items that expire the soonest are shipped first. Retailers enforce FIFO by keeping longer-lasting items in storage. FIFO is often used in the literature with perishable items [Karaesmen et al., 2011]. Thirdly, different users may have different *ratings* ($\hat{r}_{iu}$) for different items; the RS should recommend items that are of interest to the user, and the user would not have purchased without the recommendation. Lastly, different items have different prices p_i, so the RS should recommend items that result in higher revenue for the retailer.

We consider these four aspects from either the user's or retailer's *perspective*. We create these perspectives by including system-enforced sustainability and stockout considerations to

the traditional user consideration of utility/rating and retailer consideration of revenue. From the user's perspective, the recommended list should have items with a high average rating, while avoiding stockouts. The retailer wants to stimulate demand for items that generate high profit, while also avoiding items to perish. Balancing between these perspectives is crucial. The retailer may, for example, need to recommend items with a lower user rating to avoid items perishing. The retailer thus determines the recommendation lists considering the number of perishable items, demand, inventory, and possible demand increase after recommendations. If these values are known with certainty, the problem is deterministic. In this case, after finalizing all recommendation lists, the decision maker knows exactly how many items will be sold, what the profits will be, the average user rating, the numbers of stockout and perished items, and their costs. However, in practice, the decision maker rarely has access to complete information.

By considering stochasticity in our model we obtain top-k lists that account for unexpected situations. Therefore, we incorporate uncertainties in inventory ($\boldsymbol{L}$) and soon-to-perish items ($\boldsymbol{E}$) (both vectors of length $|I|$), and demands with $|I| \times |U|$ matrix $\boldsymbol{D}$. For example, if each value in $\boldsymbol{L}, \boldsymbol{E}, \boldsymbol{D}$ were binary then there would be a total of $2^{2|I|+|I||U|}$ scenarios to investigate. Even with only two realizations, an exhaustive enumeration would be intractable. One solution to this problem is to use Monte Carlo sampling to obtain a finite number of scenarios and the Sample Average Approximation (SAA) method to solve them [Kim et al., 2015]. We define a function $\max_{x \in X} [F(x) \coloneqq \mathbb{E}_{\xi} f(x, \xi)]$. We assume f is real-valued and cannot be computed directly, x is a point in solution space $X \subseteq R^d (d < \infty)$, and ξ is some random vector independent of x. We first generate S scenarios $\xi^1, \xi^2, \ldots, \xi^S$ from random vector ξ that are independent and identically distributed unless noted otherwise. Assume that the probability of each scenario equals $1/S$.

Applying the SAA method, we solve $\max_{x \in X} \left[f_S(x) \coloneqq \frac{1}{S} \sum_{j=1}^{S} f(x, \xi^j) \right]$, with the maximizer of $f_S(x)$ converging (as $S \to \infty$) to the maximizer of $f(x)$ under mild conditions [Ahmed and Shapiro, 2002]. In our problem, assume that we know the distributions of $\boldsymbol{L}, \boldsymbol{E}, \boldsymbol{D}$. Then, by Monte Carlo sampling, assume we obtain S scenarios by sampling, such that $\xi^1 = (\boldsymbol{L}^1, \boldsymbol{E}^1, \boldsymbol{D}^1), \ldots, \xi^j = (\boldsymbol{L}^j, \boldsymbol{E}^j, \boldsymbol{D}^j), \ldots, \xi^S = (\boldsymbol{L}^S, \boldsymbol{E}^S, \boldsymbol{D}^S)$. The superscript j is the scenario index, and each possible scenario ξ^j contains the information of the triple $(\boldsymbol{L}^j, \boldsymbol{E}^j, \boldsymbol{D}^j)$. The number S is decided by the decision maker. We use the shorthand notation $[S] = \{1, 2, \ldots, S\}$.

3.3.1.1. Preliminary Optimization Model. Decision variable a_{iu} equals 1 if we recommend item i to user u and 0 otherwise. The estimated rating of item i for user u is denoted as $\hat{r}_{iu}$. For scenario j and item i, the quantity sold is x_i^j, the quantity of unmet demand (number of stockouts) is y_i^j, and the number of perished items is z_i^j. Every user u has a demand value D_{iu}^j for a given item i and scenario j, and demand increases by γ_{iu} if i is recommended to u. Increasing the demand of an item recommended to a user assumption is used in simulations in the literature [Fleder and Hosanagar, 2009, Hazrati and Ricci, 2022].

$$(3.1)\quad \max_{a,x,y,z} \quad \frac{1}{S}\sum_{j=1}^{S}\left[\lambda\left(\sum_{i\in I}\sum_{u\in U}\hat{r}_{iu}a_{iu}/k - \sum_{i\in I}q_i y_i^j\right) + (1-\lambda)\left(\sum_{i\in I}\rho c_i x_i^j - \sum_{i\in I}c_i z_i^j\right)\right]$$

subject to:

$$(3.2)\quad \sum_{i\in I} a_{iu} \leq k \quad (\forall u \in U)$$

$$(3.3)\quad x_i^j \leq L_i^j \quad (\forall i \in I, j \in [S])$$

$$(3.4)\quad z_i^j \geq E_i^j - x_i^j \quad (\forall i \in I, j \in [S])$$

$$(3.5)\quad y_i^j + x_i^j = \sum_{u\in U}\left(D_{iu}^j + \gamma_{iu}a_{iu}\right) \quad (\forall i \in I, j \in [S])$$

$$(3.6)\quad x_i^j, y_i^j, z_i^j \geq 0, a_{iu} \in \{0,1\} \quad (\forall i \in I, u \in U, j \in [S])$$

Equation (3.1) specifies our multi-objective function that considers both the user and retailer perspectives. Inside the left parenthesis, we maximize the average estimated ratings and minimize the cost incurred when a demanded item is out of stock (stockout). Inside the right parenthesis, we maximize the revenue of the retailer from the sales and minimize the cost incurred from the perished items. We divide the estimated ratings by k to lower the effect of choice of k on the solution quality. Trade-off parameter $\lambda \in [0,1]$ is specified by the analyst to control the weight given to the user versus retailer perspective. If $\lambda = 1$ then we only consider the user perspective, and if $\lambda = 0$ then we only consider the retailer perspective. Constraint (3.2) enforces that we recommend at most k items to each user. We cannot sell more items than the available inventory, enforced by constraint (3.3). If the number of items sold for i is less than the soon-to-perish count E_i, then the difference is assumed to perish. Constraint (3.5) controls the flow of demand with the right-hand side equaling the modified demand after recommendations. Therefore, total modified demand is

either sold or is considered stockout. Constraint (3.6) specifies the values that the decision variables can take.

3.3.1.2. Reformulated Optimization Model. In a deterministic setting with $S = 1$, this model can be easily solved for thousands of users and items. When the number of scenarios increases, however, the number of constraints and decision variables increases rapidly. Next, we propose an optimization model that is more efficient with memory and time as the number of scenarios grows.

Since we assume FIFO we can preprocess some of the uncertainties. For each scenario j and item i, we update $E_i^j = \max\left(0, E_i^j - \sum_{u \in U} D_{iu}^j\right)$ and $L_i^j = \max\left(0, L_i^j - \sum_{u \in U} D_{iu}^j\right)$. These are the respective counts of inventory (L_i^j) and soon-to-perish items (E_i^j) when the RS is not applied (we only consider initial demands). Next, suppose that we sort the number of perishable items ($\boldsymbol{E}$) and inventory ($\boldsymbol{L}$) as follows for each i, $E_i^1 \leq E_i^2 \leq \cdots \leq E_i^{S-1} \leq E_i^S$, and $L_i^1 \leq L_i^2 \leq \cdots \leq L_i^{S-1} \leq L_i^S$. Then, we create two incremental increase arrays for each i: $\tilde{E}_i^j = E_i^j - E_i^{j-1}$, $\forall j \in \{2, \ldots, S\}$ with $\tilde{E}_i^1 = E_i^1$, and, $\tilde{L}_i^j = L_i^{j+1} - L_i^j$, $\forall j \in \{1, \ldots, S-1\}$ with $\tilde{L}_i^S = L_i^S$. Define $X_i = \sum_{u \in U} (\gamma_{iu} \times a_{iu})$, which represents the expected demand increase for item i after implementing the RS. Next, we apply the remodeling idea suggested by Ferguson and Dantzig [1956] as follows:

$$\max_{a,\tilde{y},\tilde{z}} \quad \frac{1}{S}\sum_{j=1}^{S}\lambda\left(\sum_{i\in I}\sum_{u\in U}\hat{r}_{iu}a_{iu}/k - j\sum_{i\in I}q_i\tilde{y}_i^j\right) + \tag{3.7}$$

$$(1-\lambda)\left(\sum_{i\in I}\rho c_i X_i^j - (S+1-j)\sum_{i\in I}c_i(\tilde{E}_i^j - \tilde{z}_i^j) - j\sum_{i\in I}\rho c_i\tilde{y}_i^j\right)$$

subject to:

$$\sum_{i\in I}a_{iu} \leq k \quad (\forall u \in U) \tag{3.8}$$

$$\tilde{y}_i^j \leq \tilde{L}_i^j \quad (\forall i \in I, j \in [S]) \tag{3.9}$$

$$\tilde{z}_i^j \leq \tilde{E}_i^j \quad (\forall i \in I, j \in [S]) \tag{3.10}$$

$$\sum_{j=1}^{S}\tilde{z}_i^j \leq X_i \quad (\forall i \in I) \tag{3.11}$$

$$\sum_{j=1}^{S}\tilde{y}_i^j \geq X_i - L_i^1 \quad (\forall i \in I) \tag{3.12}$$

$$\tilde{y}_i^j, \tilde{z}_i^j \geq 0, a_{iu} \in \{0,1\} \quad (\forall i \in I, u \in U, j \in [S]) \tag{3.13}$$

Equation (3.7) is a reformulation of equation (3.1) with new decision variables. For item i, $\tilde{z}_i^j$ is the number of soon-to-perish items sold and $\tilde{y}_i^j$ is the number of stockouts in j scenarios. Then, for calculating the perished items we use the coefficient $(S+1-j)$ and penalize $c_i(\tilde{E}_i^j - \tilde{z}_i^j)$. or calculating the stockouts we use the coefficient j and penalize $q_i\tilde{y}_i^j$. Note that we use incremental increase arrays $(\tilde{E}, \tilde{L})$ in this step. Because of how X_i^j is defined, every recommendation increases the equation (3.7) by ρc_i. The additional term $j\sum_{i\in I}\rho c_i\tilde{y}_i^j$ removes this increase if item i stockouts. Constraint (3.8) remains the same as constraint (3.2). Constraints (3.9) and (3.10) are simple bounds on decision variables $\tilde{y}$ and $\tilde{z}$, which are generally easily handled by commercial optimization software. Constraint (3.11) enforces

that sum of $\tilde{z}_i^j$ cannot exceed X_i. Constraint (3.12) enforces that sum of $\tilde{y}_i^j$ should be greater than or equal to $X_i - L_i^1$. Constraint (3.13) remains the same as constraint (3.6), except in this model, decision variable x is removed. The value of $\tilde{E}_i^j$ is 0 if $E_i^j = E_i^{j-1}$ (same with L_i^j). Then, the corresponding decision variable $\tilde{z}_i^j$ ($\tilde{y}_i^j$) is removed since it is fixed to 0. These changes improve the previous optimization model's computational time and memory requirement.

3.3.2. Heuristic Algorithm

For very large datasets, optimization models can struggle with large memory requirements and slow solution times. We offer a heuristic that scales better for larger datasets. This heuristic is both easy to implement and fast to run. We first randomize the order of users. In each step, we recommend a single item i to user u, and then move on to the next user. This procedure continues until either all users have k items in their lists or adding an item to a user's list decreases the objective function value, which only happens if all the inventory is depleted. For a given user u, we recommend the item i^* that solves the objective function:

(3.14)
$$i_u^* = \underset{i \in I}{\operatorname{argmax}} \frac{1}{S} \sum_{j=1}^{S} \lambda(\hat{r}_{iu}/k - q_i \min(0, L_i^j - \gamma_{iu})) + (1-\lambda)(\rho c_i \min(L_i^j, \gamma_{iu}) + c_i \min(E_i^j, \gamma_{iu}))$$

After each recommendation, the $\boldsymbol{L}, \boldsymbol{E}$ and recommendation lists of users are updated: $E_{i^*}^j = \max(0, E_{i^*}^j - \gamma_{i^*u})$, and $L_{i^*}^j = \max(0, L_{i^*}^j - \gamma_{i^*u})$. Each item i^* recommended to user u has rating $\hat{r}_{i^*u}/k$. The value of γ_{i^*u} is the demand increase of item i^* for user u. If the term $L_{i^*} - \gamma_{i^*u}$ is negative, recommending item i^* to user u causes a stockout having penalty q_{i^*}. When a soon-to-perish item i^* is recommended to user u, we incentivize that recommendation by the cost of the item (c_{i^*}) times the number of soon-to-perish items sold ($\min\{E_{i^*}, \gamma_{i^*u}\}$).

Lastly, when the retailer sells an item, the objective value increases by $\rho \times c_i$. The heuristic requires solving objective function (3.14) at most $k|U|$ times.

3.3.3. Solving the Model with Massive Datasets

Our optimization model can be solved optimally for hundreds of scenarios and thousands of users and items. However, in cases with thousands of scenarios and millions of users and items, we need approximation methods to obtain a solution in a reasonable length of time. This subsection discusses two approximation approaches. First, co-clustering [George and Merugu, 2005] can be applied to users, items, or both. Rather than focusing on each item and user individually, we can cluster them. This can be done by aggregating items similar to each other as only one item, or by aggregating users in clusters if they have similar preferences for similar items. Consequently, even billions of users and items can be manageable in smaller clusters. We apply this idea by reducing the item space using the taxonomy of the categories in our dataset.

Second, we apply a simple idea that we call the "best-N approach," in which we recommend item i to user u only if either item i is one of the top N rated items by that user u, or user u is one of the top N users that rated that item i the highest. None of the other user-item pairs will be considered for recommendation. The decision maker can choose N by using a grid-search algorithm. A similar approach was implemented in the RS literature [Bradley and Smyth, 2001] with good results by solely focusing on the user top N lists. Normally, the number of decision variables created for recommending an item i to user u (a_{iu}) is $|I| \times |U|$. Implementing the best-N approach decreases this number to at most $N \times (|I| + |U|)$. Thus, we reduce the number of decision variables by orders of magnitude, which alleviates both the solution time and memory requirements of the model. The heuristic model's solution

time improves similarly. With the best-N approach, some users and items might appear significantly more than others and result in over recommending some items and under recommending others. We alleviate this issue by subtracting the mean rating of each item and user by itself, i.e., de-biasing them. Consequently, items or users will be included in the best-N list only if the rating gain observed is higher relative to the item-user pair.

3.4. Computational Study

This section discusses the data used in this chapter and the computational results of the models proposed. All results are obtained by using a laptop with Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz 2.21 GHz processor information with 16.0 GB of installed RAM specifications. The optimization model is solved using Gurobi, a commercially available mathematical programming solver, Gurobi Optimization, LLC [2021] 9.5.0 with the optimization gap set to 10^{-4} and a one-hour time limit. Gurobi obtains the global optimal solution (within the optimization gap) using a linear-programming based branch&bound algorithm [Morrison et al., 2016]. In the worst case, the algorithm implemented in Gurobi can have exponential time complexity [Morrison et al., 2016], but in practice, the time complexity is much better than exponential. The heuristic approach is solved with Python 3.7.

3.4.1. Dataset Description

We consider an online grocery retailer located in the United States and learn user preferences from their purchase history. We examine regular users who have shopped at least 15 times within the last 6 months, giving 3731 users ($|U|$) with 78,195 orders. These orders include 27,158 unique item stock-keeping units (SKUs). Each SKU has a price, brand, and category from a retailer-provided taxonomy. We aggregate the SKUs following Malthouse

et al. [2019a]: we match the retailer-provided taxonomy of sub-categories with brand names to come up with 4106 ($|I|$) unique item names. For example, "HAIR PRODUCTS - BRAND NAME", is considered one item. We manually tag delivery items, fresh market items, and some dairy items as soon-to-perish items using sub-categories. The number of perishable items is 651 out of 4106.

Average demand values are calculated by averaging the quantity bought for each user-item pair for a week, denoted as $\overline{D}_{iu}$. We create a $|U| \times |I|$ rating matrix by using the function $\log(x+1)$, x as the number of orders including item $i \in I$ for user $u \in U$. Next, we compare prediction algorithms SVD, k-NN, and Co-clustering using the Surprise package [Hug, 2020a] to estimate the values of unknown item-user pairs. We use the SVD algorithm, which performed the best using 5-fold cross-validation, with RMSE and MAE values of 0.24 and 0.20, which is better than for k-NN (RMSE=0.26, MAE=0.21) and co-clustering (RMSE=0.60, MAE=0.55). Finally, the values are min-max normalized between 0 and 1. Denote the value calculated for item-user pair as η_{iu}, which we take as the probability of buying an item i in case it is recommended to the user u. We assume that the probability of buying and money spent on an item correlates with the utility of that item to the user. Therefore, the ratings are calculated as $\hat{r}_{iu} = (1+\rho) \times c_i \times \eta_{iu}$, which is the price times purchase probability. With this assumption, all values in the objective function are in dollars. More complex utility/rating choices can be considered by the decision maker. The γ_{iu} values are calculated as $\left\lceil \overline{D}_{ui} + 0.1 \right\rceil \times \eta_{iu}$. This value is continuous and can be interpreted as the average expected increase in demand when item i is recommended to user u. We assume high purchase probability and high previous demand are both important in recommender quality. We consider that recommendations can play two different roles: either the user will be reminded to order their regular needs, or they will be recommended novel items that they

might want to buy. Because γ is not related to c_i, items with low purchase probability and low previous demand will rarely get recommended for that user, even if the item has a high price. Stockout costs q_i are set equal to costs of the items c_i, $\forall i \in I$. We use a constant markup value $\rho = 0.26$ [Richards and Liaukonytė, 2023] for simplicity, but the decision maker can choose different markups for different products.

3.4.2. Evaluation Procedure

This subsection discusses the settings and metrics used to evaluate our models, and benchmarks implemented to compare our approaches. These benchmarks are denoted B_p, B_r, B_s, and B_u, and they solely optimize perishability, retailer sales, stockouts, and average user rating objective functions, respectively. In other words, each benchmark obtains a solution considering only one criterion. Therefore, we compare our solutions solving multiple objectives with those focusing on a singular objective. The letter H indicates the heuristic solution and O indicates optimization. This letter is followed by a number indicating the weight (λ) value. For example, the optimization solution with weight $\lambda = 0.5$ is denoted as $O5$. We obtain solutions for $\lambda \in \{0.1, 0.5, 0.9\}$, corresponding to a higher focus on retailer perspective, equal focus, and a higher focus on user perspective, respectively.

We investigate four settings: high perishables and low stockout risk (HL), low perishables and high stockout risk (LH), both high risk (HH), and finally, both low risk (LL) settings. Settings with a high risk of perishables have a larger number of soon-to-perish items $\boldsymbol{E}$, and those with a high risk of stockout have lower levels of inventories $\boldsymbol{L}$. We choose $\psi = \{0.2, 0.6\}$ and $\epsilon_L = \{30, 100\}$, and their combinations create our four settings. We generate inventory levels using $L_i \sim \text{Poisson}(\epsilon_L + \sum_{u \in U} \overline{D}_{iu})$, where $\overline{D}_{iu}$ is the expected demand for item i and user u. Average demands vary greatly from as low as 0.4 to as much as thousands. We choose

lower ϵ_L values for high-risk stockouts and higher values otherwise. The demand values D_{iu} are distributed as Poisson($\overline{D}_{iu}$). In this way, we create upper bounds that are tight relative to the total demand of each item. We create a number of soon-to-perish items as a percentage of inventories such as $E_i = L_i\psi_i$, where ψ_i is distributed as a truncated normal $\mathcal{N}(\psi, 0.1)$ with bounds $[0, 1]$, $\forall i \in I$. Some items perish sooner than others, and the decision maker can incorporate this by choosing higher values of ψ_i for items that perish more rapidly and lower values otherwise. We base the number of soon-to-perish items as a percentage of the inventory, so higher inventory will result in a larger number of soon-to-perish items. For high-risk perishable cases ψ_i will be closer to 1, and 0 otherwise. As a shorthand notation, HH is $\psi = 0.6, \epsilon_L = 30$, HL is $\psi = 0.6, \epsilon_L = 100$, LH is $\psi = 0.2, \epsilon_L = 30$, and finally, LL is $\psi = 0.2, \epsilon_L = 100$.

Unless stated otherwise, we generate 500 scenarios (S) for $\boldsymbol{L}, \boldsymbol{E}$, and $\boldsymbol{D}$ considering each of the four settings. We obtain solutions for each approach by solving the problem with these 500 scenarios. Then, we generate 2500 new out-of-sample scenarios to test the quality of the obtained solutions. Overall, a solution is better if it has higher user ratings, retailer sales, or lower perishability and stockout objective values. These objective values (metrics) are calculated as follows, where x_{iu} is 1 if item i is recommended to user u and 0 otherwise, s_i is the number of item i sold, z_i is the number of perished item i, y_i is the number of stockouts of item i:

$$\text{Ratings} = \sum_{i \in I} \sum_{u \in U} \frac{\hat{r}_{iu}}{k} \mathrm{x}_{iu}, \qquad \text{Sales} = \sum_{i \in I} \rho c_i \mathrm{s}_i, \tag{3.15}$$

$$\text{Perishability} = \sum_{i \in I} c_i \mathrm{z}_i, \qquad \text{Stockouts} = \sum_{i \in I} q_i \mathrm{y}_i,$$

$$\text{User Perspective} = \text{Ratings} - \text{Stockouts},$$

$$\text{Retailer Perspective} = \text{Sales} - \text{Perishability}$$

3.4.3. Results

This subsection presents and discusses results obtained using our optimization model and heuristics, and compares them with solutions of benchmark models. Furthermore, we investigate the effect of including stochasticity and heuristics on the solution quality.

3.4.3.1. Solution Quality of Optimization Approach. We compare our model's solution with the benchmarks in Figures 3.1 to 3.6. In the Figures 3.5,3.6 lower values and for the rest higher values are better. We use the best-N method with $N = 100$ and offer at most $k = 10$ items to every user. Figure 3.1 shows that the overall objective values for the **user perspective**, which includes user ratings and stockouts, are similar for all approaches, except for B_u, which is the benchmark where only the average user rating is optimized. Despite B_u performing well in terms of user rating (Figure 3.3), users would experience many stockouts (Figure 3.5) that reduce customer goodwill, which results in significantly worse user perspective objective value. As expected, in the high stockout risk settings (HH and LH) the number of stockouts observed is greater, causing slightly lower user perspective objective values. The optimal approach achieves the highest user perspective objective values for all settings.

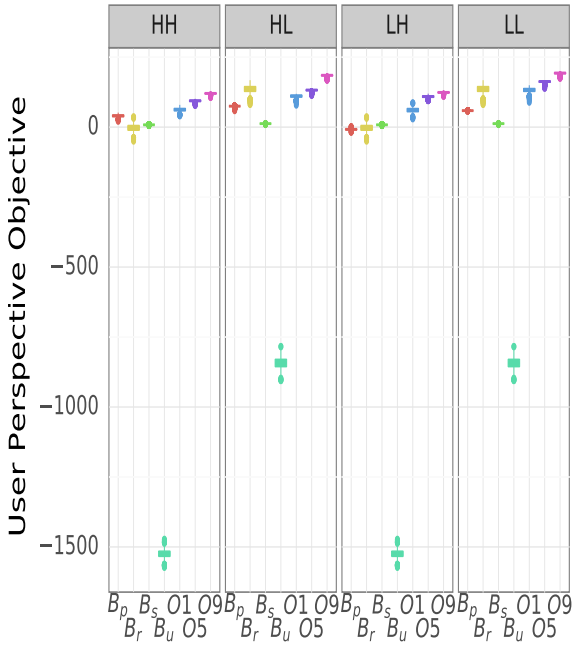


Figure 3.1. Benchmark comparison of user perspective objective values

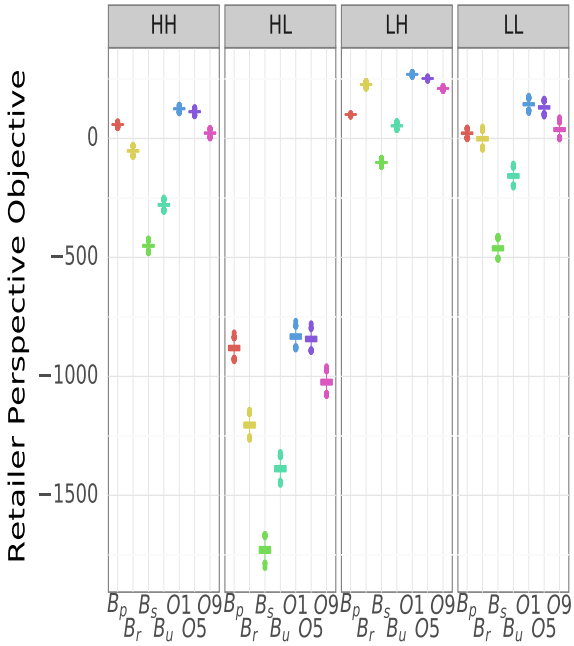


Figure 3.2. Benchmark comparison of retailer perspective objective values

Considering the objective values from the **retailer perspective**, which include perished items and sales, we observe more varied results than those from the user perspective. The values for the high perishables and low stockouts (HL) are the lowest, indicating that many items perish, which generates high waste costs, as illustrated by Figure 3.6. The exact opposite setting with low perish and high stockout (LH) achieves the highest retailer perspective values because of the low waste costs. The retailer perspective objective values are similar for HH and LL, which is surprising at first because they consider exact opposite situations. Looking into these results in more detail, we see that for LL, the sales are higher than for HH (Figure 3.4), but so are the number of perished items. In general, the solutions tend to slightly decrease the sales objective values to improve the perishability objective, resulting

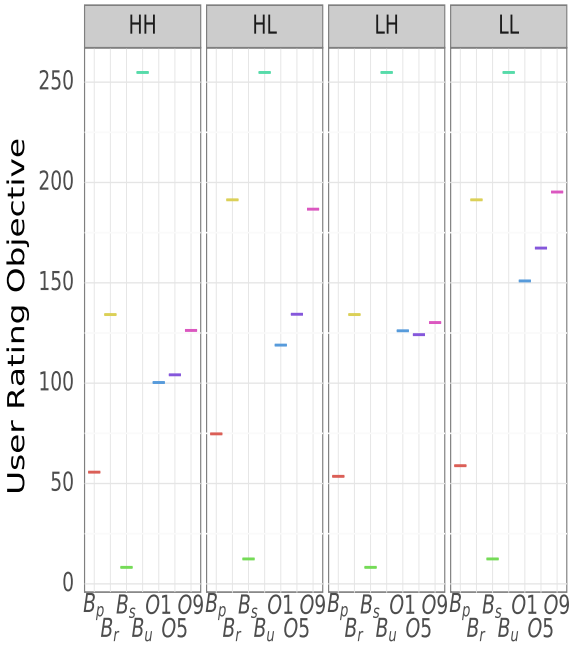


Figure 3.3. Benchmark comparison of user rating objective values

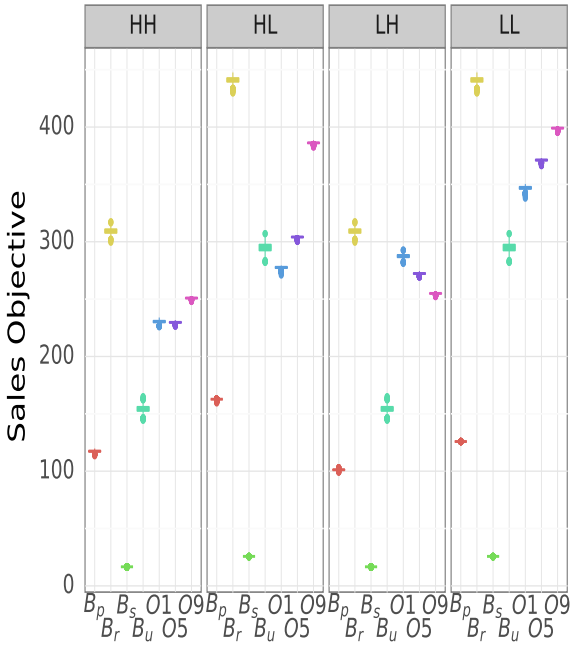


Figure 3.4. Benchmark comparison of retailer sales objective values

in a better retailer perspective objective value. We next discuss each individual objective in more detail.

When considering the **user ratings** in Figure 3.3, B_u performs the best, but as discussed before, suffers from a high number of stockouts (Figure 3.5), which negatively affects users. Interestingly, the solution B_r results in high user ratings as well but incurs higher stockouts too. The B_r solution offers items with the greatest monetary gain considering the users' probability of buying the item recommended to them. This can also be observed from Figure 3.4, where B_r achieves the highest sales objective values. This benchmark is different from only offering items with high price margins, and additionally considers the probability of users buying the recommended item, thus achieving recommendations with high user ratings. Therefore, this competitive benchmark considers both retailer and user information,

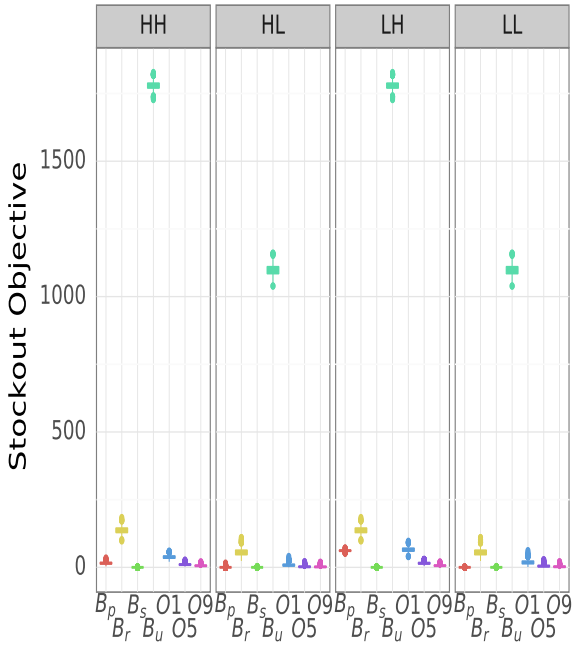


Figure 3.5. Benchmark comparison of stockout objective values

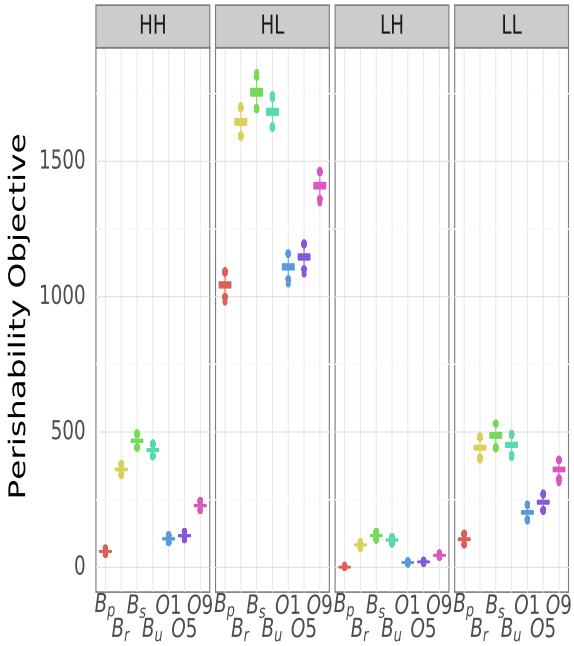


Figure 3.6. Benchmark comparison of perishability objective values

and is most similar to approaches that maximize the retailer's expected profit [Das et al., 2009, Akoglu and Faloutsos, 2010]. Since the monetary return is the only consideration, B_r solutions result in large numbers of stockouts and perished items. Since the recommendations are not scenario dependent, the user rating of each solution is the same for all scenarios.

Figure 3.4 shows the **sales** objective function values. The LL setting results in the greatest sales due to the retailer being able to sell the highest profit items in a low-risk stockout and perishability environment. Counterintuitively, most of the time sales increase when the optimization solution focuses less on the retailer perspective. This happens because an increased focus on the retailer perspective usually lowers the current sales in favor of selling lower-margin and lower-rating items that will perish soon, causing long-term profits instead of short-term. Generally, less focus on retailer perspective results in less focus on recommending

soon-to-perish items, thus increasing the retailer sales but decreasing the retailer perspective objective value due to perished items. This is again an important distinction that traditional RS might miss while trying to maximize the revenue of the retailer. However, the LH setting is an exception to this rule. The optimization solutions in setting LH already have a low number of perished items (Figure 3.6) and thus a weight increase results in recommending high-rating items that are not necessarily high profit for the retailer while lowering the stockouts. Benchmarks B_s and B_p perform the worst because sales are negatively affected when the only concern is selling soon-to-perish items or keeping the stockouts to a minimum.

The number of **stockouts** (Figure 3.5) is in a similar range for all approaches except for benchmark B_u. Interestingly, the B_u approach might seem to be user-centric, however, it results in a poor user experience because of the large number of stockouts, where users cannot purchase items that were recommended to them. This is an important and general shortcoming of the top-k recommendation rule and is a disadvantage of applying RS without considering demand changes that follow. Our optimization solution considers user experience overall and results in users obtaining the items that are recommended to them. Next, benchmark B_s solely focuses on minimizing the stockout objective function. In this benchmark, not recommending anything is one solution with the optimal value of 0. This benchmark is useful for analyzing the solution quality when little to no recommendations are made. The objective values of perishability, user ratings, and sales are significantly underperforming. Interestingly, due to the existence of multiple-optima solutions, the sales objective value is not exactly zero in some cases. For the optimal solutions, larger weights result in better stockout objective values because the focus shifts to the user perspective.

Figure 3.6 shows the **perishability** objective function values. HL has the highest perishability values due to high inventory (low stockout risk) and high-risk perishability, which

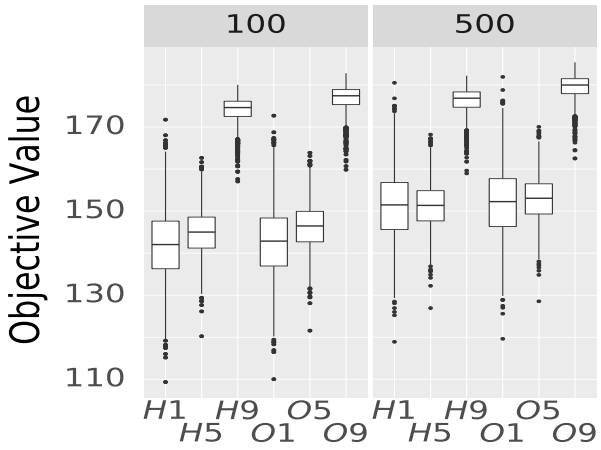
results in the greatest number of soon-to-perish items. We observe that low stockout with a low perishability ratio (LL) can result in a larger number of perished items than low inventory with a high perishability ratio (HH). Thus, we note that low risk stockout is not always desirable and the retailer should order less to reduce the number of perished items due to high inventory levels. The benchmark B_p minimizes the perishability objective function by recommending soon-to-perish items first, and thus achieves the best perishability results. The optimization solution with weight 0.1 is a close second to B_p in the perishability objective. However, benchmark B_p struggles when it comes to sales and user ratings. The proposed recommendations are neither what users prefer nor the items with high-profit margins. If the RS focuses solely on soon-to-perish items, both retailer and user perspectives suffer. The perishability objective, in general, decreases when the weight decreases due to more focus on the retailer perspective.

Overall, considering the user and retailer perspectives in Figures 3.1 and 3.2, the benchmark B_u performs badly on the user metrics due to stockouts, and B_s performs badly on retailer metrics due to low sales. Benchmark B_r performs worse in user perspective when stockout is high risk, and in retailer perspective when perishability is high risk. Benchmark B_r might seem like a good option for the retailer at first, however, the stockouts and perished items that are ignored make it undesirable. Benchmark B_p performs worse than our models on user ratings and sales, due to recommendations being made towards selling the soon-to-perish items without considering user ratings or revenue. We have investigated two additional benchmarks: user-perspective (consider both user rating and stockouts) and retailer-perspective (consider both retailer sales and perishability). The user-perspective benchmark was only slightly better in user perspective objective value while significantly worse in retailer perspective objective value relative to the optimization solution due to a

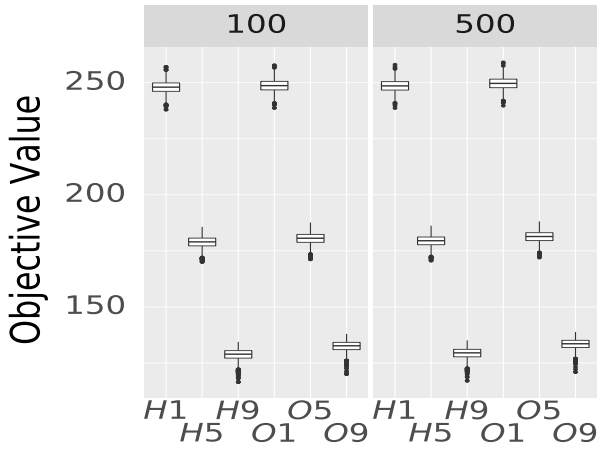
high number of perished items. The retailer-perspective objective value was almost identical to the optimization solution but significantly worse in user perspective solution due to a high number of stockouts. Overall, the optimization solutions perform the best when considering both the user and retailer perspectives.

3.4.3.2. Solution Quality of Heuristic Approach. Figure 3.7 compares the objective function values for heuristic approaches. Each subfigure considers one of the four settings with best-N approach applied, where $N \in \{100, 500\}$. The difference in the objective values between $N = 100$ and 500 is minimal. The objective function value difference is even smaller for $N > 500$. This result is very useful for a retailer offering a large number of products since only considering a smaller subset of the most preferred products for each user and most preferred users for each product decreases the problem size significantly. Therefore, using the best-N approach improves the memory and solution time for both models while maintaining the solution quality. We note that our heuristic approach also achieves objective function values close to the optimal solution (within 1%). The heuristic performs similarly to the optimization model that considers all the metrics in Equation 3.15.

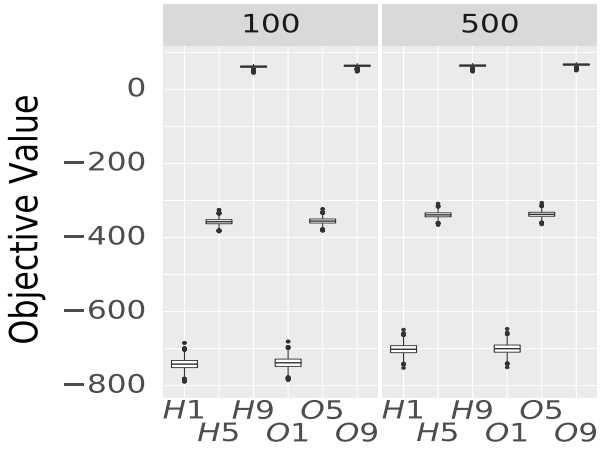
3.4.3.3. Solution Quality Comparison of Stochastic and Deterministic Cases. We analyze the solutions using expected values of $\boldsymbol{L}, \boldsymbol{E}, \boldsymbol{D}$ instead of creating 500 scenarios for demand, inventory, and perishability. We denote these solutions with A and use the same weights as before, i.e., $\{0.1, 0.5, 0.9\}$. Figure 3.8 shows that it is advantageous to consider stochasticity because using only the expected values ignores the variance of the data. We conclude that it is better to account for uncertainty compared to using a solution estimated with averages.



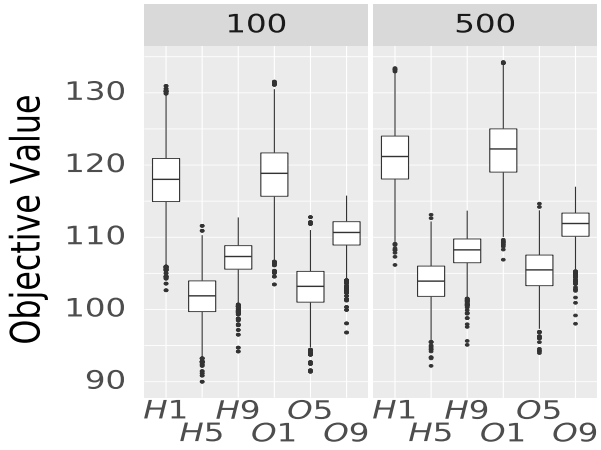
(a) $\psi = 0.2$ and $\epsilon_L = 100$ (Setting LL)



(b) $\psi = 0.2$ and $\epsilon_L = 30$ (Setting LH)



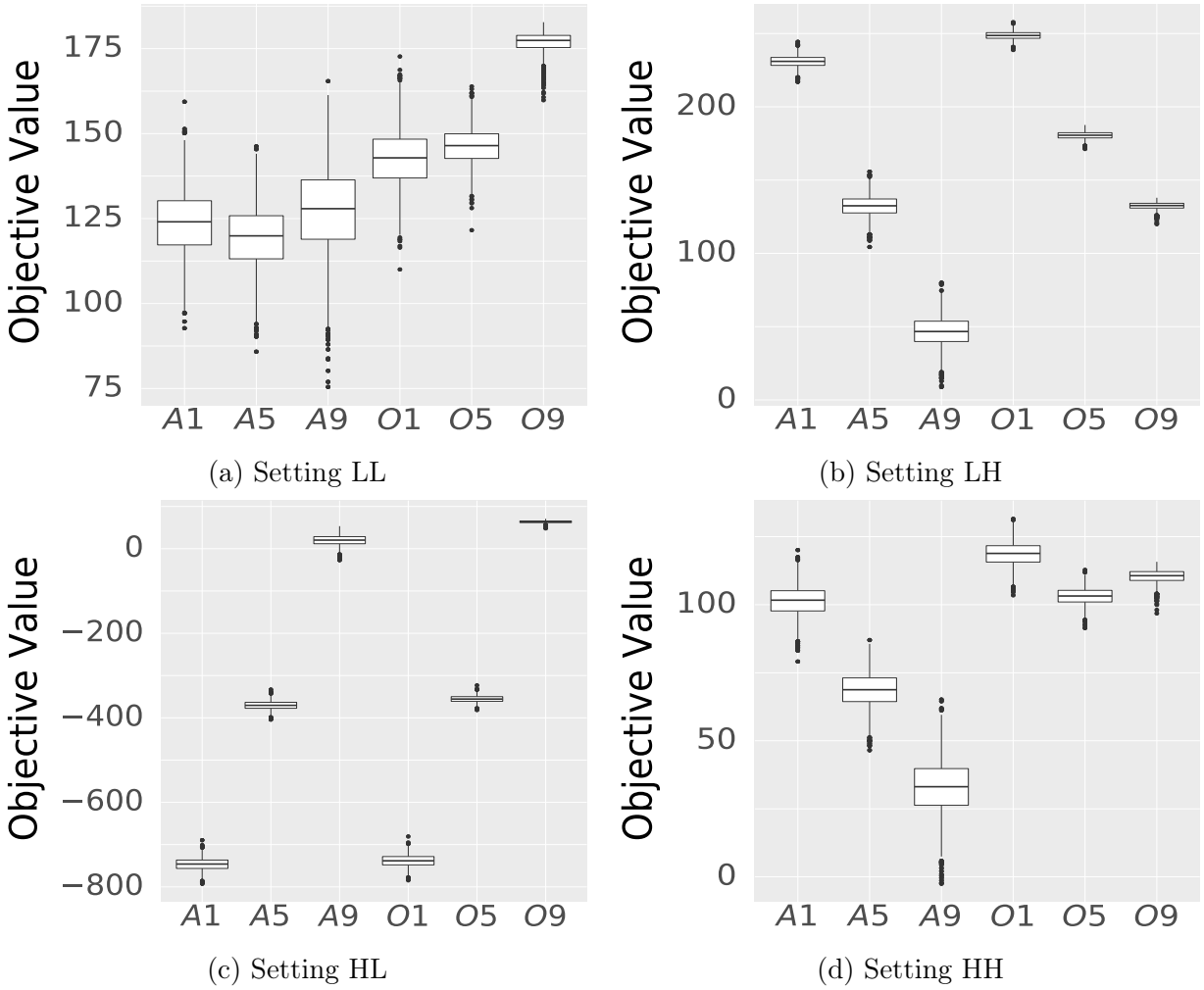
(c) $\psi = 0.6$ and $\epsilon_L = 100$ (Setting HL)



(d) $\psi = 0.6$ and $\epsilon_L = 30$ (Setting HH)

Figure 3.7. Objective value comparison for $N = \{100, 500\}$

3.4.3.4. Solution Quality Changes with Weights and Number of Scenarios. Figure 3.9 provides additional insights by focusing on HH (high in both perishability and stock-outs). The discussions are similar in other settings. Subfigure (a) shows that creating 500 scenarios is more than adequate, and after 100 scenarios the improvement in the objective function value is negligible. We also observe that fewer scenarios result in a significant drop in performance for weight $\lambda \in \{0.1, 0.5, 0.9\}$. Therefore, solutions obtained by considering a variety of scenarios perform better. For example, a decision maker who considers only the



(a) Setting LL

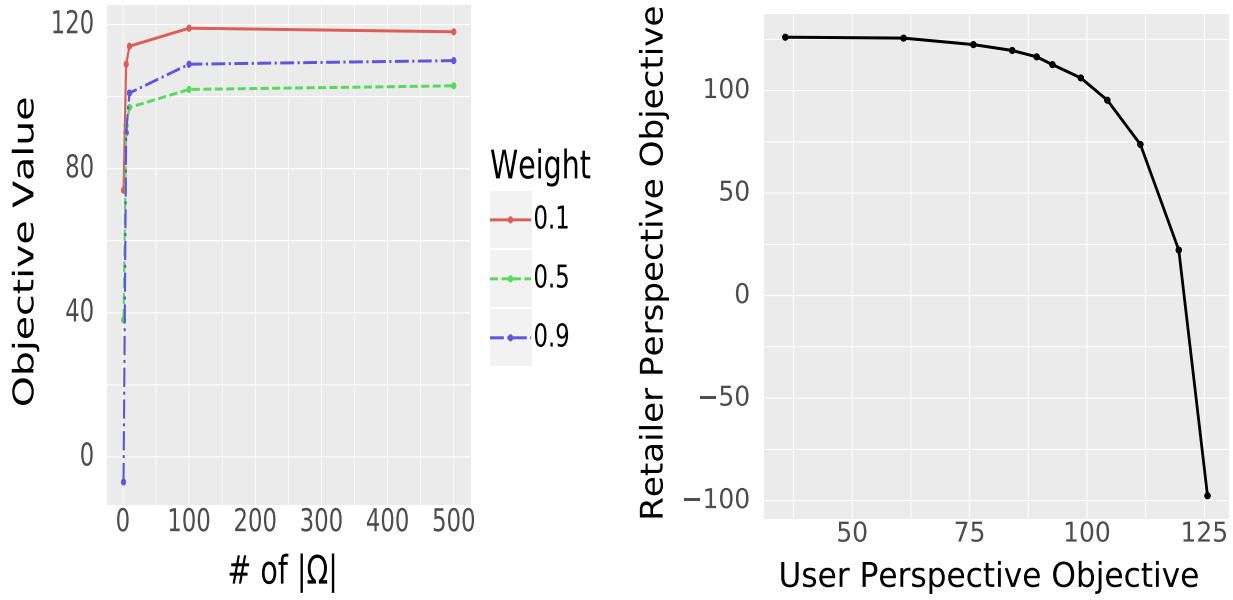
(b) Setting LH

(c) Setting HL

(d) Setting HH

Figure 3.8. Objective value comparison of using scenarios

latest 3 weeks' worth of data ($S = 3$) would be at a significant disadvantage. Subfigure (b) shows the trade-off curve between the user and retailer perspectives with weights ranging from 0 to 1. We observe that increasing the weight from 0 to 0.5 almost triples the user perspective objective while reducing only 0.1 of retailer perspective objective. Therefore, considering both perspectives rather than only one improves the overall quality of the solution. The decision maker can select the best weight according to the needs of the user, retailer, or both.



(a) (b)

Figure 3.9. Objective value changes with the number of scenarios in (a), Objective values for different perspectives with different weights in (b), in setting HH

3.5. Future Research & Conclusion

This article proposes a MIP model and heuristics that consider RS objectives from the user and retailer perspectives. The user perspective aims to obtain highly rated item recommendations while minimizing stockouts. The retailer perspective aims to maximize profit while minimizing the losses incurred by the perished items. Our models find solutions that are high in quality for both criteria. We offer approximation methods that scale better than the optimization model. We propose a heuristic model and show that its solution quality is nearly as good as the optimal one. Therefore, the reader can decide to aim for the optimal solution and use the optimization model, or use the heuristic, which is faster to solve and more scalable. Finally, we study the improvements made to the objective function values

using our models, and compared the solutions with each of the benchmarks in different settings.

Our work can be extended in multiple directions. Firstly, more emphasis on inventory could be incorporated into the retailer perspective. In this way, excess inventory could be considered and items with higher storage spaces might need to be recommended more often. In our work, we create recommendations considering inventory as constant (although not deterministic), however, inventory levels of items could be added as decision variables as well. Note that, different user segments could behave differently to stockouts (e.g., purchasing a substitute), and incurring different penalties to different user segments can be a possible direction.

The stochasticity can be applied to different parts with more knowledge of the uncertainties. Scenarios can be generated using the background knowledge of the system. For example, the inventory generation process can represent a problem with supply chain disruptions. Our work assumes the distribution of the parameters, but more work can be done solely focusing on the solution quality changes tied to the distributions. If more complex distributions are considered, the solution quality of the models may change, and it would be worthwhile to investigate the changes in solution quality with changes in distributions.

While creating our parameters such as the ratings of the items for the users and the effect of recommendation on the increased demand for a given item, we had to make certain assumptions. We use an online retailer dataset in an offline setting. If our work is extended to an online setting then it would be possible to understand and update our parameters accordingly. Next, by obtaining data from the users continuously, we could improve the quality of the parameters for each user to reflect their needs better. Even in an offline

setting, future research can include different parameter creation ideas and investigate the changes in solution qualities in different settings.

CHAPTER 4

A Large-scale Constrained Optimization Model for Multi-Objective Recommender Systems

4.1. Introduction

Recommender systems (RSs) filter information to make personalized recommendations based on the user's needs. One of their main objectives is recommending each user "relevant" items. However, "relevant" can convey different ideas for different users and problems. For example, businesses can employ RSs to recommend travel destinations, clothing, movies, songs, and many other types of products or services to their user base. While issues such as diversity and novelty have been discussed for over a decade in RS setting [Ziegler et al., 2005, Vargas and Castells, 2011], the focus has largely been on accuracy and decision-support metrics of the user such as root mean square error (RMSE), precision or recall. However, focusing only on the user results in item provider and system needs and objectives being ignored. In recent years, the RS literature has increasingly considered the objectives of item providers and the RS platform in addition to the user needs, leading to the emergence of RS problems with multiple objectives [Abdollahpouri et al., 2020].

Multi-objective recommender systems (MORSs) are designed to optimize or balance multiple objectives simultaneously [Jannach, 2022]. The methods solving MORS problems find a solution that considers all the objectives. For example, multistakeholder recommender systems [Abdollahpouri et al., 2020] consider the needs of multiple stakeholders and can be studied as a special type of MORS [Jannach, 2022]. Given that most businesses need to tackle multiple objectives [Jannach and Jugovac, 2019], it is crucial to solve MORS problems as efficiently as possible. For example, some job matching RSs such as LinkedIn recommend jobs matching both business-side requirements with user qualifications and job-seeking intent [Rodriguez et al., 2012]. However, because of the inclusion of multiple objective functions, MORS problems may require more specialized approaches [Jannach, 2022].

Appropriately implemented constrained optimization models provide an effective means to solve multi-objective problems, and therefore they have been recently used to solve MORS problems [Seymen et al., 2021b, Sürer et al., 2018, Seymen et al., 2022]. They can find a solution optimizing an objective function satisfying a given set of constraints that are expressed as mathematical expressions. For example, if we want to offer items to users matching their previous history as in the calibration RS problem [Seymen et al., 2021a, Steck, 2018], we can employ a constrained optimization model that finds a list that optimizes user utilities and a calibration metric simultaneously. These models use constraints to address considerations encountered in RS problems.

Several problems can arise while implementing constrained optimization models. When the models include integer decision variables, such as in mixed-integer programs (MIP), scalability and feasibility can prove hard to achieve. Seymen et al. [2021b] discuss the feasibility and scalability issues of the MIP model they offer to solve MORS problems with diversity, popularity, and fairness considerations. Solving a MIP model can require significant memory to handle a large branch-and-bound tree [Morrison et al., 2016] used in the solution method. As the number of users and items increases, scalability can become a significant drawback. There are a couple of ways to alleviate this issue.

One solution to the scalability problem is to use heuristics [Bradley and Smyth, 2001, Seymen et al., 2022, Agarwal et al., 2011], which make the problem smaller or offer items one step at a time. However, the heuristic solutions obtained this way may be of inferior quality according to the objective function due to their myopic nature [Seymen et al., 2022, 2021a]. Some considerations are harder to scale than others, which is discussed further in Section 4.2, and heuristic approaches can relax these considerations to improve scalability. Another solution to the scalability problem is to reformulate the models [Seymen et al.,

2022] to alleviate memory issues. Furthermore, finding feasible solutions to optimization models and the parameter selection can pose challenges when there are multiple objectives to consider. It is not always trivial to choose parameter settings that result in high-quality solutions. In our work, we implement a large-scale optimization algorithm that tackles both the feasibility and scalability issues.

In the literature, solving constrained optimization models in RSs using commercially available software is usually deemed disadvantageous due to their memory requirements. In this work, we offer a large-scale optimization model that solves a variety of MORS problems. It recommends k items to users in post-processing. We estimate the utilities of item-user pairs using some RS algorithm and then decide which k items to recommend users in the system considering multiple objectives. Our main goal is to alleviate the scalability problem of constrained optimization models while dealing with multiple objectives. We also define and discuss the types of constraints that can affect scalability for constrained optimization models. We illustrate how our large-scale optimization model exploits the structure of these constraints to improve scalability. We discuss the performance of our model using the MovieLens 20M dataset tackling fairness, popularity, and diversity considerations. Additionally, we compare the results of our large-scale model with solutions of a state-of-the-art constrained optimization model and heuristics.

4.2. Literature Review

In recent years, constrained optimization models have been applied to a variety of problems in MORSs. Several mixed-integer programming (MIP) models are offered, where the goal is to recommend k items to each user considering a variety of constraints. These models efficiently tackle multiple objectives simultaneously [Sürer et al., 2018, Seymen et al., 2021b,

2022]. Sürer et al. [2018] propose a MIP model to solve a multistakeholder RS problem. However, their Lagrangian relaxation procedure results in infeasible solutions that should be made feasible, which essentially is a heuristic approach. Seymen et al. [2021b] offer a MIP model addresses diversity, popularity, and fairness metrics simultaneously and beats state-of-the-art heuristics, but their model can have issues with feasibility and scalability. Malthouse et al. [2019a] employ integer decision variables to trade off user utility with ad revenue, but their model does not address what we will call across-list constraints.

Other works [Agarwal et al., 2011, 2012] employ linear programming models to find recommendations as probabilities. Chen et al. [2020] use incentives to sell soon-to-perish items rather than including explicit constraints for perishability. Jambor and Wang [2010] use linear decision variables to decrease the popularity bias of recommendations. These linear decision variables might represent the probability or importance factor of an item for a user, so higher values would make the item more likely to be recommended. These decision variables are later binarized to specify whether an item is recommended to a user or not, similar to MIP models.

Popularity bias in RSs refers to the inclination to recommend popular items beyond their justified popularity while under-recommending relevant but less popular items, despite their potential value to certain users [Abdollahpouri et al., 2019b, Abdollahpouri, 2020]. This bias is extensively investigated in the RS literature due to its prevalence and importance [Abdollahpouri et al., 2019a, Jannach et al., 2015, Zhang et al., 2021, Deldjoo et al., 2022]. One way to tackle this issue with constrained optimization is to restrict the overall popularity of the recommendations with an upper bound (UB) [Seymen et al., 2021b]. The popularity value of each item can be calculated by the number of users who interacted with the item before. In this way, the system designer can control the popularity of the overall recommendations.

There are a variety of ways [Kunaver and Požrl, 2017, Castells et al., 2021] researchers define and measure diversity. Castells et al. [2021] assess diversity by the difference of items from each other in a user list in RS context. This is one way to evaluate diversity at the user-level. There are works considering the diversity of items for all users across their recommendation lists as well [Adomavicius and Kwon, 2011a, Zhou et al., 2010]. Seymen et al. [2021b], Sürer et al. [2018] aim to improve the diversity of each user's top-k list by forcing each list to have at least some specified number of unique items. In this way, the system designer can set the acceptable lower limit of diversity for each user's list. For example, a music recommendation system designer can force the optimization model to recommend at least 8 different artists in a top-10 list, assuming the artist information is available for those songs.

Fairness is extensively researched in RSs. Deldjoo et al. [2023] discuss fairness as a subjective and social construct. Wang et al. [2023] provide definitions for fairness in certain contexts, such as individual fairness, outcome fairness, process fairness and more. They note that alleviating RS unfairness is a crucial goal that results in increased user interaction, greater motivation for the item providers, increased long-term efficiency of the system and many other benefits. We focus on RSs that create unfair distributions of recommended items among the item providers. This unfairness can lead to relevant providers not getting enough recommendations for their items [Patro et al., 2020]. Additionally, item provider might choose to leave the system if their items are not recommended to the users. The system designer can specify a lower bound on the number of items recommended for each provider to ensure that every item provider is recommended at some level [Patro et al., 2020, Sürer et al., 2018]. Similarly, upper bounds can be included as constraints so that no provider gets

over-recommended [Seymen et al., 2021b]. In this way, the system designer can specify that the number of allowed recommendations for each provider is in a set interval.

The issues mentioned above can be modeled as constraints in different ways. It is useful to group constraints as *within-list* and *across-list*. The scope of within-list constraints is a single user list independent from other users. For example, a constraint to recommend 60% horror movies to every user individually is within-list since the recommendations of each user are independent of each other. Formally, within-list constraints only include decision variables with one specific user. Models that exclusively incorporate within-list constraints [Seymen et al., 2021a, Malthouse et al., 2019a] have fewer scalability issues because the problem separates, and each user's problem can be solved independently. In such settings, the memory requirements become trivial and the computing time can be improved greatly by solving different subproblems on different processors.

Across-list constraints apply across more than one users' recommendation lists concurrently. For example, limiting the number of recommendations of an item or group of items due to stockout possibility can be formulated as an across-list constraint [Seymen et al., 2022], since the total number of recommendations depend on all the users. Sürer et al. [2018] increase the recommended retailer/provider diversity using both within-list and across-list constraints. Therefore, when across-list constraints are considered, the recommendation decision for one user changes the decision for another, and then the problem should consider every user recommendation list simultaneously. This is why the across-list constraints in RS problems can be considered *complicating constraints*, since their inclusion makes separating the problem impossible. It is worthwhile to remove or relax across-list constraints and capture them in different ways. Depending on the problem, across-list constraints can be represented as within-list constraints, or even pre-processing steps, without resulting in

significant accuracy loss. Table 4.1 summarizes which papers have addressed the different types of constraints.

Consideration	Within-List	Across-List
Diversity	[Seymen et al., 2021b] [Sürer et al., 2018]	[Sürer et al., 2018]
Popularity		[Seymen et al., 2021b] [Jambor and Wang, 2010]
Fairness	[Malthouse et al., 2019a]	[Seymen et al., 2022, 2021b] [Agarwal et al., 2011, 2012]

Table 4.1. Literature review of RS considerations categorized into within- and across-list constraints.

4.3. Methodology

This section defines the post-processing top-k RS problem and then describes our Dantzig-Wolfe decomposition algorithm to solve large-scale RS problems.

4.3.1. Problem definition

We formulate a post-processing top-k RS problem as a constrained optimization model. Let U denote the set of users and I be the set of items in the system. We assume that items are partitioned into *groups*. For example, items on a retail platform can be grouped by vendor, movies in terms of popularity, songs by the artist on a music platform, or news articles by the publisher (e.g., FNC, CNN, etc.). Let I_s be the partitions of set I for $s \in S$, with S as the set of all groups. Therefore, $\cup_{s \in S} I_s = I$ and $I_s \cap I_{s'} = \emptyset$ for $s \neq s'$. Index s stands for the supplier, although groups could be entities other than suppliers, such as a manual grouping of retail items based on similarity as in dairy, or expiration day, as in perishable or non-perishable [Seymen et al., 2022].

We assume that the predicted ratings or utilities $\hat{r}_{iu}$ have already been estimated with an existing RS algorithm for item i and user u. The decision variables are denoted as x_{iu}, which take the value 1 if the system recommends item i to user u, and 0 otherwise. Our generic optimization model is as follows:

$$\max_{x} \quad \frac{1}{k|U|} \sum_{u \in U} \sum_{s \in S} \sum_{i \in I_s} \hat{r}_{iu} x_{iu}$$

subject to

$$\beta_s \geq \sum_{u \in U} \sum_{i \in I_s} x_{iu} \geq \alpha_s \quad (\forall s \in S) \tag{4.1}$$

$$\sum_{s \in S} \sum_{i \in I_s} x_{iu} = k \quad (\forall u \in U) \tag{4.2}$$

$$x_{iu} \in \{0, 1\} \quad (\forall s \in S, i \in I_s, u \in U) \tag{4.3}$$

The objective function maximizes the average utility of all recommended items in the top-k lists. Constraint (4.1) forces the model to recommend at least α_s and at most β_s many items from the group s. Because the set I_s denotes the set of items provided by each group s, if recommendations for each group are determined in proportion to the number of items in their corresponding set, as expressed by the ratio $\frac{k|U||I_s|}{|I|}$, it is possible to establish upper and lower bounds using the set I_s. For example, setting $\alpha_s = 0.9 \times |I_s|, \beta_s = 1.1 \times |I_s|$ (rounded to the nearest integer) would lead to recommendations falling within a narrow interval for each group s. Constraint (4.2) ensures that the model recommends exactly k items to every user $u \in U$. Constraint (4.3) requires the decision variables x_{iu} to be binary.

The across-list constraint (4.1) is common in the MORS literature. By maximizing user utility in the objective function and enforcing constraints on the number of items recommended from each group, the model aims to satisfy the preferences of individual users, while also ensuring that groups are adequately represented in the recommendations. Furthermore, popularity and diversity considerations can be represented as within- or across-list constraints as discussed in Section 4.2.

Solving this model can be cumbersome when the user and item sets are large. The scalability is a significant disadvantage of this model because it has $|U| \cdot |I|$ decision variables and $2|S| + |U|$ constraints. Without the across-list constraints of type (4.1), this problem could be solved for every user through sorting. We propose using the well-known DW decomposition algorithm with delayed column generation to solve these large-scale problems by handling the across-list Constraint (4.1) in an efficient way.

4.3.2. Dantzig-Wolfe Decomposition Approach

The Dantzig-Wolfe (DW) decomposition is an algorithm for solving large-scale optimization models, originally developed by Dantzig and Wolfe [1960]. The DW decomposition breaks down the original optimization problem into a set of smaller subproblems by exploiting special structures. Then, each of these sub-models are solved separately. In this way, the DW decomposition can solve problems with many decision variables or constraints that are difficult or impossible to solve using other methods. Assume a model with the following simple representation:

$$\max_{x} \quad cx,$$

$$\text{subject to}$$

$$Ax \leq b$$

$$x \geq 0.$$

Now assume that matrix A in constraint $Ax \leq b$ has the following block-angular structure:

$$A = \begin{bmatrix} A_{11} & A_{12} & \dots & A_{1n} \\ A_{21} & 0 & \dots & 0 \\ 0 & A_{32} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & A_{mn} \end{bmatrix}.$$

If the first row of the matrix A is removed, the structure of the remaining rows allows the associated optimization problem to separate into smaller, independent subproblems that can be solved concurrently. Note that, the first block of rows, A_{11} to A_{1n} corresponds to the across-list Constraint (4.1) and the block diagonal part of A maps to users.

Let P be the set of all solutions that satisfy the Constraints (4.2) and (4.3). Let B_p denote a specific solution, where $B_{iup} = x_{iu}, \quad p \in P$. Define $\hat{R}_p$ as the utility of solution p i.e., its objective function value, and W_p as the weight of given solution p, whose meaning will become clear shortly. We take the convex combination of all the solutions B_p, which gives us the convex hull of the problem. The aim is to start with a subset of known solutions B_p, and find the maximizing solution in the convex hull. Consequently, the DW master problem is given below:

$$\max_{W} \quad \sum_{p \in P} \hat{R}_p W_p$$

subject to

$$\sum_{i \in I_s} \sum_{u \in U} \sum_{p \in P} B_{iup} W_p \geq \alpha_s \quad (\forall s \in S) \tag{4.4}$$

$$\sum_{i \in I_s} \sum_{u \in U} \sum_{p \in P} B_{iup} W_p \leq \beta_s \quad (\forall s \in S) \tag{4.5}$$

$$\sum_{p \in P} W_p = 1 \tag{4.6}$$

$$W_p \geq 0 \quad (\forall p \in P) \tag{4.7}$$

In the DW master problem, we maximize a linear program, using convex combination of solutions represented by B_{iup}. Constraints (4.4) and (4.5) represent our across-list constraints. Constraint (4.6) requires weights W_p to sum to 1. Constraint (4.7) forces weights of the solutions to be non-negative.

Next, using the dual variable information obtained from the solution of the master problem, we separate and solve the following problem for each user u independently:

$$\min_{x} \quad \sum_{u \in U} \sum_{s \in S} \sum_{i \in I_s} (q_s - p_s - \hat{r}_{iu}) x_{iu} - v$$

subject to

Constraints (4.2) and (4.3)

The objective function is the reduced cost of W decision variables, where q_s, p_s, and, v are the dual variables of Constraints (4.4), (4.5), and (4.6), respectively. Reduced costs represent the improvement that can be made by including solutions to the master problem that were not considered before. If we have any non-positive reduced cost for a user u, we include that

solution p to the set of solutions P in the master problem, and solve the master problem again. We continue to solve these subproblems until all of them have positive reduced cost values.

The DW approach is more desirable when the subproblems are easy to solve. In our work, the subproblems for user u can be solved by simple sorting of predicted ratings modified by dual variables, $q_s - p_s - \hat{r}_{iu}$, and recommending items until every user is recommended exactly k items. Note that the DW approach stops when the optimal solution to the relaxed model is found. We name our model using the DW approach the DW model. In our results section, we show that the solution is in very close proximity to the global optimum for the MovieLens 20M data.

Next, we discuss the application of penalties when considering popularity bias and diversity metrics. Here, we shift our focus to MORSs by solving multiple different subproblems simultaneously. We address popularity bias by penalizing each item by its popularity. With diversity, we follow [Seymen et al., 2021b] and put items into distinct groups based on their genre. We define list diversity as the number of distinct groups recommended to a user in their top-k lists. In contrast to [Seymen et al., 2021b], we do not include diversity as a constraint, but as a separate objective function. As a result, the diversity is handled differently between the two models.

We incentivize recommending items that belong to distinct genres to increase recommendation list diversity for the users. We define a trade-off parameter between diversity and utility. A high incentive parameter might cause the DW model to recommend lower utility items in order to increase the diversity of a recommendation list, while a low incentive parameter might result in higher utility and lower diversity recommendation lists.

In the DW model, we address the popularity bias problem by penalizing the recommendation of popular items using the following pre-processing step: $\hat{r}_i = \hat{r}_i - \gamma\omega_i, \forall i \in I, \gamma \in [0, \infty)$, where ω_i denotes the popularity value of item i and γ is the popularity penalty value. Our approach is better suited for addressing feasibility and scalability issues, compared to models that utilize across-list constraints [Seymen et al., 2021b].

Considering the DW model, we create a new set I^*_{su} that includes all the items that are candidates to be offered to each user u from supplier s if we solve the DW model to optimality. First, we define V_{su} as the set of k items with highest utility (after applying popularity penalty) for supplier s and user u. We also define a function $g(\cdot)$ that determines the set of distinct genres of a list of items. Therefore, $|g(V_{su})|$ results in the number of distinct genres belonging to the top-k solution of each $u-s$ pair. Next, we take the item with the maximum utility for each supplier s, user u, and genre g. Then, the value $\eta = min(k, |g(I_s)|) - |g(V_{su})|$ is the number of distinct genres that can be added to the set V_{su} to include at least k or $|g(I_s)|$ many distinct items. We add η many items with distinct genres to the set V_{su} which results in $|I^*_{su}| = |V_{su}| + \eta$.

Theorem 1: Consider DW model that tackles fairness by Constraint (4.1), popularity with popularity penalties, and diversity with diversity incentives. We recommend k items in set I for users in set U while maximizing the overall predicted utility. Every item belongs to a genre group $g \in T$ and supplier group $s \in S$. The optimal recommendation list of the DW model can be obtained recommending items exclusively from set $I^*_{su}, \forall s \in S, u \in U$.

Proof 1: Assume a feasible solution to the DW model $\tilde{x}$ with objective function value $f(\tilde{x})$. Define $L_{s'u'}(x)$ as the list of items recommended to user u' from supplier s'. Assume $L_{s'u'}(x)$ has at least one item $i' \notin I^*_{s'u'}$ with $x_{i'u'} = 1$ for user u' supplier s'. Since, $\tilde{x}$ is feasible, constraints of type 4.1 are satisfied.

Define a set $I^{-}_{s'u'} := \{i | i \in I^{*}_{s'u'} \text{ and } x_{iu'} = 0\}$. Firstly, assume $g(L_{s'u'}(x) \setminus i') = g(L_{s'u'}(x))$. Note, set $I^{-}_{s'u'} \cap V_{s'u'} \neq \emptyset$ because $x_{i'u'} = 1$, $i' \notin I^{*}_{s'u'}$ and $|I^{*}_{s'u'}| \geq k$. Then, for any $i'' \in I^{-}_{s'u'} \cap V_{s'u'}$, we can obtain a new solution with $x_{i''u'} = 1, x_{i'u'} = 0$ which is feasible (fairness stays the same) with $f(\tilde{x}^2) \geq f(\tilde{x}^1)$ because $\hat{r}_{i''u'} \geq \hat{r}_{i'u'}$ (conclusion 1). Secondly, assume $|g(L_{s'u'}(x) \setminus i')| < |g(L_{s'u'}(x))|$ (cond. 2). If $\exists i'' \in I^{-}_{s'u'}$ s.t. $g(i'') = g(i')$ conclusion 1 applies. Then, $\nexists i'' \in I^{-}_{s'u'}$ s.t. $g(i'') = g(i')$. This means $g(i') \notin I^{*}_{s'u'}$ due to cond. 2. Therefore, $|g(I_{s'})| > k$, and $\exists i''$ s.t. $|g(i'' \cup (L_{s'u'}(x) \setminus i'))| = |g(L_{s'u'}(x))|$, because $|g(L_{s'u'}(x) \setminus i')| < k$, while $|g(I^{*}_{s'u'})| = k$. Due to construction of set $I^{*}_{s'u'}$, all items $i'' \in I^{-}_{s'u'}$ result in better objective function values when switched with i'.

Using Theorem 1, the number of decision variables can be reduced from $|I| \times |U|$ to at most $2|S| \times k \times |U|$ in the preprocessing step. This reduction allows us to focus on the "best" items each supplier provides to each user. When the number of suppliers is low and the number of items they provide is high, we can eliminate millions of decision variables without a decrease in the objective function value.

4.4. Benchmarks

In this section we compare state-of-the-art benchmarks with DW. We present a constrained optimization model as a benchmark that can solve MORS problems optimally, but may face scalability issues in larger datasets. We also present single-constraint models that scale well, but struggle with tackling multiple objective functions simultaneously.

4.4.1. Unified Optimization Toolbox

Seymen et al. [2021b] propose an integer optimization model to alleviate the problems of popularity bias, diversity, and, fairness (discussed in Chapter 2). We call their model Uni, short for *unified*, and briefly discuss the details of it here:

$$\max_{x} \quad \frac{1}{k|U|} \sum_{i \in I} \sum_{u \in U} \hat{r}_{iu} x_{iu} \tag{4.8}$$

$$\text{subject to:} \quad \text{Constraints (4.1,4.2,4.3)} \tag{4.9}$$

$$\sum_{i \in I} \sum_{u \in U} x_{iu} \omega_i \leq \psi, \tag{4.10}$$

$$\sum_{i \in F_j} x_{iu} \geq y_{ju} \quad (\forall j \in T, u \in U) \tag{4.11}$$

$$\sum_{j \in T} y_{ju} \geq w \quad (\forall u \in U) \tag{4.12}$$

$$y_{ju} \in \{0, 1\} \quad (\forall j \in T, u \in U) \tag{4.13}$$

Constraints (4.1,4.2,4.3) are shared between our generic suggested model and Uni so that the two models produce the same top-k lists for their users with fairness implemented as an across-list type constraint. The Uni model forces that each user receives at least w many distinct genre recommendations by implementing within-list type constraints (4.11,4.12,4.13), where T is the set of genres, and F_j is the set of items with genre $j \in T$. The Uni model also implements across-list constraints (4.10) to make sure that overall popularity of the suggested items are below parameter ψ, which is defined by the system designer.

For given parameters ψ and w, the advantage of Uni is the ease of explanation. We know exactly the value of total popularity and number of distinct items recommended to each user. However, this approach does not scale to large instances since popularity and fairness

are defined as across-list constraints. Additionally, although diversity is controlled with a set of within-list constraints, the constraints are imposed on users without considering how they react to them. For example, one user might only want recommendations from a small set of groups, and by forcing the diversity constraint we might decrease the utility of this user significantly. In that case, it could be beneficial to use a w_u value that is different for each user, but then this value is not easy to come up with. In our DW approach, we solve these problems by implementing incentives to increase diversity, and penalties to decrease the popularity.

4.4.2. Single-constraint models

This subsection discusses single-constraint models in more detail. A single-constraint model refers to a model that solves only one of the subproblems of fairness, popularity bias, and diversity in addition to maximizing user utility. Throughout, $L_u = \{i \mid x_{iu} = 1\}$ represents the set of items that are recommended to user u.

4.4.2.1. Diversity. Diversity in top-k lists has received considerable attention in the literature [Kunaver and Požrl, 2017, Castells et al., 2021, Ziegler et al., 2005]. The objective is to increase the diversity of the recommended items, where each item is assumed to belong to some group and two items belonging to two different groups are dissimilar. In this case, the problem can be solved independently for each user. Consequently, simple sorting ideas are useful to maximize recommendation list diversity. We use an efficient heuristic model [Smyth and McClave, 2001] aimed at maximizing both the diversity and utility of the user recommendation lists. We call this heuristic the "Diversity" model. The heuristic uses a weight parameter to create a trade-off between utility and Intra-List Diversity (ILD) [Smyth

and McClave, 2001], where ILD is defined as:

$$\text{ILD} = \frac{1}{|U|} \sum_{u \in U} \sum_{i \in L_u} \sum_{\substack{i' \in L_u \\ i' \neq i}} \frac{1 - \text{sim}(i, i')}{|L_u|(|L_u| - 1)}. \tag{4.14}$$

ILD metric measures the average diversity of all items in a given top-k list, across all users. The sim$(\cdot)$ is a function that measures similarity between two items. In this work, this measure takes value 1 if two items share the same genre, and 0 otherwise. Higher diversity lists have higher ILD values.

4.4.2.2. Popularity. Popularity is another important consideration. Like diversity, popularity can be solved without employing across-list constraints. Therefore, we can use an algorithm that is similar to the one we use for diversity [Kunaver and Požrl, 2017]. The heuristic uses a weight parameter to create a trade-off between utility and total popularity of the recommendation lists. We call this heuristic the "Popularity" model. Consistent with DW, we use a weight parameter to penalize the popularity of the item by minimizing the ARP (Average Recommendation Popularity [Abdollahpouri et al., 2019a]). ARP is defined as:

$$\text{ARP} = \frac{1}{|U|} \sum_{u \in U} \sum_{i \in L_u} \frac{\omega_i}{|L_u|}. \tag{4.15}$$

ARP value is the average popularity value of items recommended to users. We define ω_i as the number of users who have rated an item i in the dataset. Lower popularity recommendations have lower ARP values.

4.4.2.3. Fairness. Furthermore, we consider fairness in our computational study. The provider fairness problem requires across-list constraints, which may result in issues with

scalability. We compare our model with a state-of-the-art heuristic algorithm, FairRec [Patro et al., 2020]. FairRec tackles fairness in a similar manner to ours, where the aim is to allocate items fairly. We modify FairRec to accommodate multiple item recommendations by the same provider, and call it the "Fairness" model. We employ one of the fairness metrics suggested in this work:

$$Z = -\sum_{s \in S} \left(\frac{R_s}{|U|k} \right) \log_{|S|} \left(\frac{R_s}{|U|k} \right), \tag{4.16}$$

where $R_s = \sum_{u \in U} \sum_{i \in L(u)} 1(i \in I_s)$ gives the total number of times items of item provider s is recommended. If all providers' items are recommended in equal amounts, then Z value becomes 1. When the inequality between item provider recommendations increases, the Z value decreases.

4.5. Computational Study

We conduct a computational study using the MovieLens 20M dataset. In the first subsection, we describe the dataset and evaluation procedure in more detail. Then, we compare the DW model with Uni and the single-constraint models. All the results were obtained by running the computations on a pc with 12th Gen Intel(R) Core(TM) i7-12700H, 2.70 GHz processor and 16GB of RAM.

4.5.1. Dataset Description

The MovieLens 20M dataset has 20 million recorded ratings between items (movies) and users. Users rate items with a value between 0.5 and 5 as an explicit rating. We have a total of 138,493 users and 18,345 items. We remove items rated less than five times because we do not deal with the cold start problem. We use the ALS algorithm in PySpark to estimate

utilities of the user-item pairs without rating information in the dataset. We employ the ALS algorithm with 5-fold cross-validation to estimate the utilities due to its efficiency with handling large-scale data.

First, we create 15 groups ($S = 15$) and distribute items randomly. In practice these groups would be likely created using known item features, e.g., provider information. After estimating the utilities, we apply Theorem 1 to remove items that will not be in the optimal recommendation lists. Some items may have more than one genre in the dataset. Only one genre is assigned to those items randomly, as in [Seymen et al., 2021b]. We use the resulting dataset in the following section.

4.5.2. Results

We divide our results into two subsections. First, we discuss the results of DW and Uni models using a subset of the MovieLens 20M dataset. Second, we solve the whole dataset with DW Model, and compare our methods with the benchmarks discussed in 4.4.2.

We evaluate sixteen configurations of different diversity and popularity parameters. We provide the parameters used for each sample run in Table 4.2 below. If not noted otherwise, we chose $1.4|I_s|$ as upper bounds and $0.6|I_s|$ as lower bounds for item providers throughout this chapter. This selection improves the fairness metric Z while not being too strict on the utilities.

An increase in the popularity penalty value should result in lower utility with a lower popularity bias. An increase in the diversity incentive should result in lower utility with a higher diversity value. Utility decreases with the increase in both parameters because popularity and diversity have a trade-off relationship with utility. If we only maximize the predicted utilities of the recommendation lists, we offer the highest utility items without

Sample No	Pop penalty	Div. incentive	Sample No	Pop penalty	Div. incentive
1	0	0	9	1.0	0
2	0	0.1	10	1.0	0.1
3	0	0.2	11	1.0	0.2
4	0	0.5	12	1.0	0.5
5	0.5	0	13	2.0	0
6	0.5	0.1	14	2.0	0.1
7	0.5	0.2	15	2.0	0.2
8	0.5	0.5	16	2.0	0.5

Table 4.2. Sample run configurations

considering popularity bias or diversity. But if we put more weight on diversity and popularity, the models might select lower utility items to create less popular and more diverse recommendation lists. A solution is better if it has higher utility, ILD and Z values, or lower ARP values. For better readability, we divide utilities with 10^4.

4.5.3. Small data: DW vs. Uni

For comparing DW and Uni constrained optimization models, we first solve the DW model with parameters given in Table 4.2 using four randomly selected subsets of 2500 users. We use a smaller part of MovieLens 20M because Uni faces scalability issues. We calculate the values of popularity, diversity, and the number of provider recommendations. Then, we solve the Uni model using these values to match the metrics as closely as possible. Note that we round the diversity value for Uni because selecting non-integer w value is same as selecting $\lceil w \rceil$, since the left-hand-side will always have integer values in Constraint (4.12). Therefore, we round the value to match the diversity value of DW solution.

4.5.3.1. Without Diversity. Since the DW and Uni models handle diversity differently, we start by comparing the solutions of the models without the diversity consideration. Therefore, in the DW model, the diversity incentive is selected as 0, and $w = 0$ in Constraint 4.12

in the Uni model. We compare the performance regarding popularity, utility and fairness in the boxplots shown in Figures 4.2 to 4.4 for the parameters presented in Table 4.2.

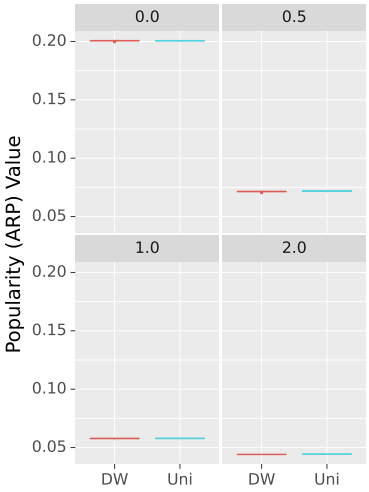


Figure 4.2. Popularity values for different popularity penalty values

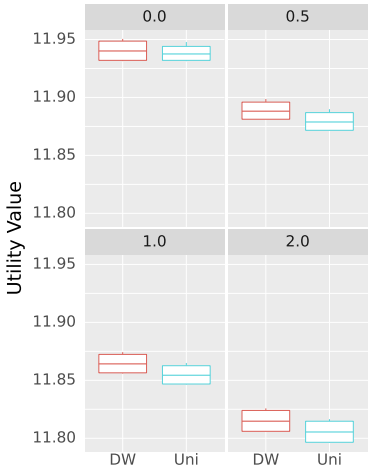


Figure 4.3. Utility values for different popularity penalty values

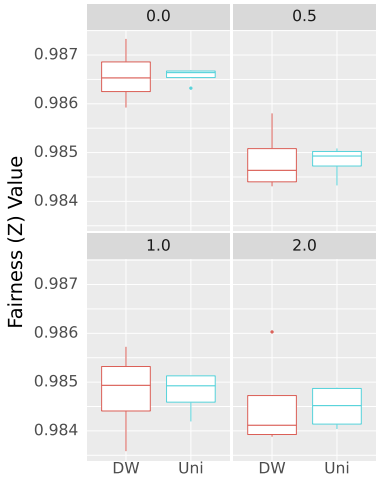


Figure 4.4. Fairness values for different popularity penalty values

We note that the popularity values are almost the same, as can be observed in Figure 4.2. As expected, the popularity value approaches 0 with an increase in the popularity penalty. If we set the diversity incentive to zero, the overall utilities decrease with the increase in popularity penalty (Figure 4.3) because the models offer lower utility items to lower the overall popularity of the recommendations. Note that both models still perform very close to each other, with DW having only a slight advantage over Uni. This shows that DW produces solutions very close to optimal without the scalability problem of Uni model. The DW model has a slightly lower fairness value as observed in Figure 4.4. The very small difference in fairness can be explained by the DW model finding a slightly higher utility

solution than Uni, showcasing the trade-off between them. The Z values are very close to 1, meaning we achieve fair recommendations for providers using either model. The upper and lower bounds can be relaxed or tightened for looser or stricter fairness.

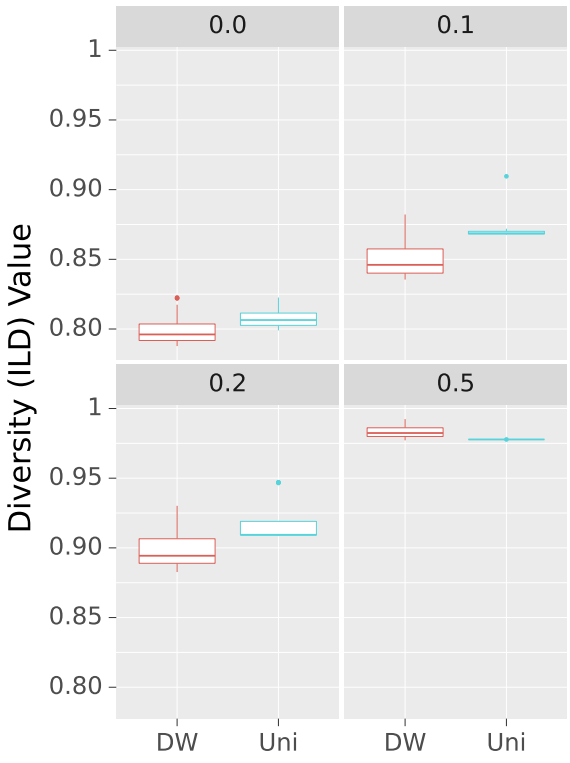


Figure 4.5. Diversity for different diversity incentive values

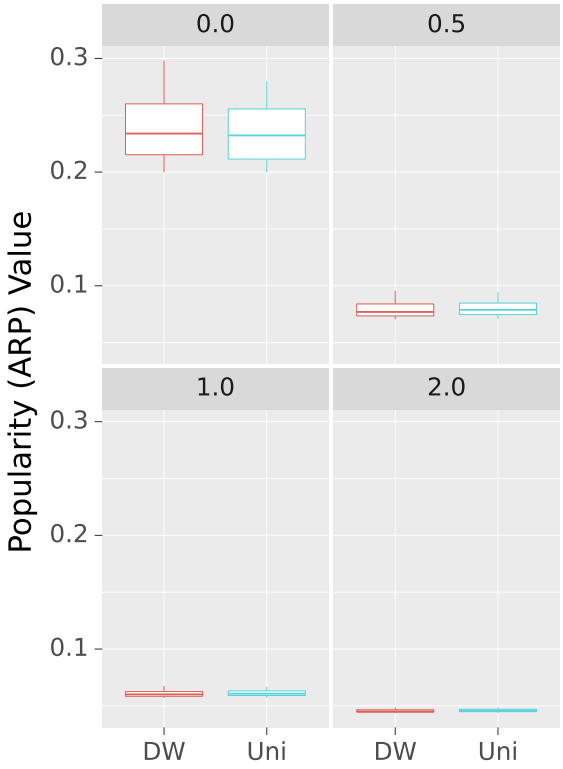


Figure 4.6. Popularity for different popularity penalty values

4.5.3.2. With Diversity. In addition to fairness, utility, and, popularity, we now take diversity into consideration. Figures 4.5 to 4.10 show the results of comparing the DW and Uni models. We observe that the DW model can find solutions within approximately 0.1% of the optimal solution. In all graphs, the x-axes show the models, and the y-axes show the metric value.

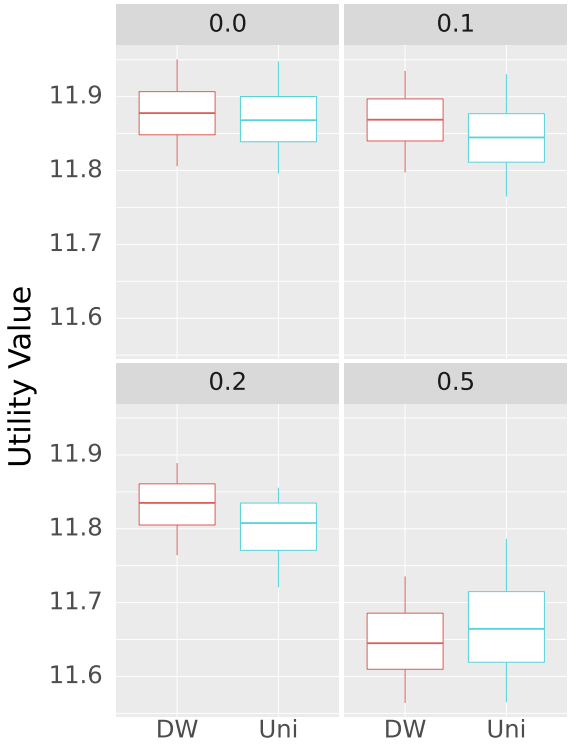


Figure 4.7. Utility values for different diversity incentive values

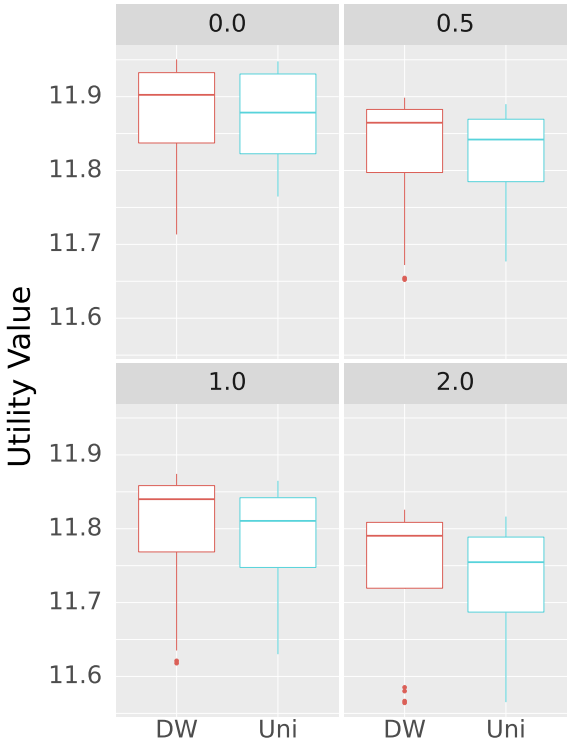


Figure 4.8. Utility values for different popularity penalty values

Figure 4.5 shows that the DW and Uni models obtain slightly different values of the diversity metric ILD, with DW having higher variance. The results for diversity are closely linked to utility and fairness, and if one solution is better in diversity then it is worse in either utility or fairness (Figures 4.7,4.9). For example, DW solutions with diversity incentive 0.5 results in higher diversity than Uni solutions. In those samples, we observe that DW obtains lower utility (Figure 4.7). This result shows that we do not observe Pareto dominance between the two models.

Figure 4.6 compares the average popularity of the recommendations. Both models perform similarly regarding popularity. The popularity value decreases (values closer to zero are better) with the increase in popularity penalty, but the rate of decrease slows down. The

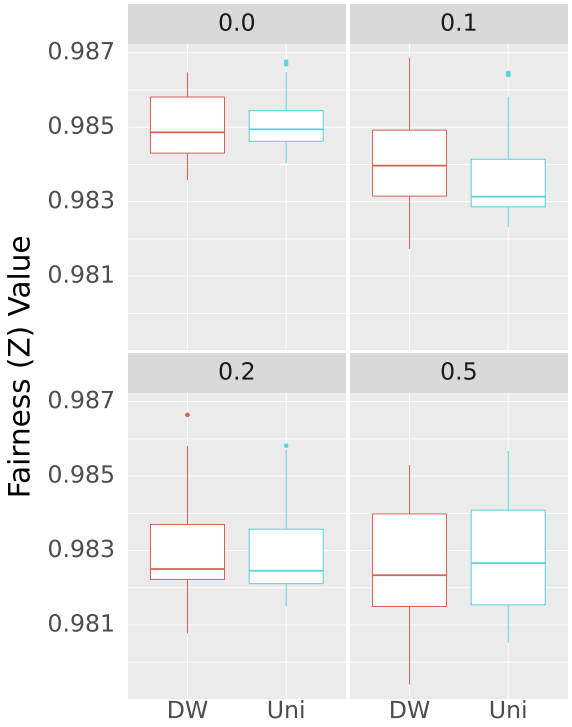


Figure 4.9. Fairness values for different diversity incentive values

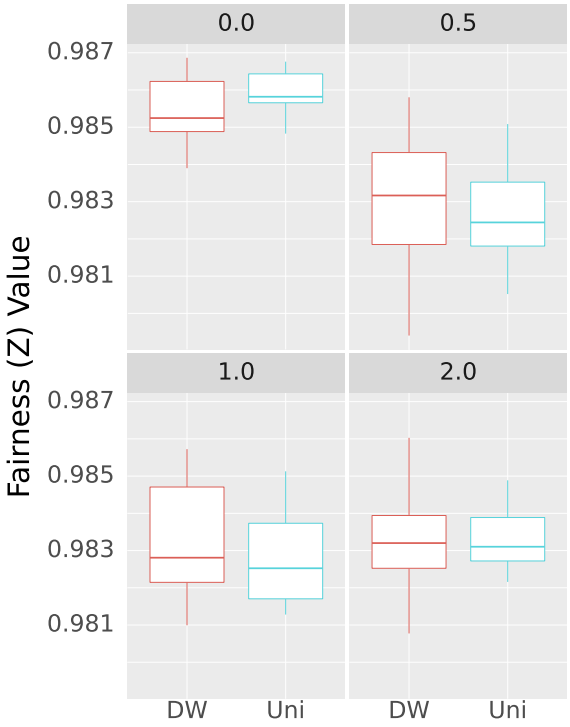


Figure 4.10. Fairness values for different popularity penalty values

system designer can determine whether utility or popularity is more important, and set the parameters accordingly.

Overall, utility decreases with an increase in diversity and popularity (Figures 4.7,4.8). However, the decrease in utility is not very large between samples, which means our solutions can improve on diversity and popularity while not resulting in a strong decrease in utility. Increasing diversity and lowering popularity bias results in, at most, a 3% predicted utility decrease, while the diversity of each users' recommendation lists can be improved by up to approximately 20% and popularity values by up to approximately 80%.

In Figures 4.9 and 4.10, we observe fairness value Z. We set the upper and lower bounds to the same values throughout this results section. The system designer can change this

interval, and tighter intervals will result in higher Z values. Our interval selection is quite strict, therefore, all the Z values observed are close to 1. Since fairness is not included in the objective function, as long as the constraints of upper and lower bounds are satisfied, the models do not aim to improve fairness further. Therefore, small changes between values are expected and do not necessarily mean one solution is better. If one model beats the other one in fairness, they do worse in either utility, diversity, or popularity.

Overall, the DW method finds solutions very close to the Uni model optimal solution. In comparison, DW is much better in scalability. We posit that the scalability and efficiency of DW make it a better model for large datasets considering the insignificant decrease in the solution quality compared to the Uni model optimal solution.

4.5.4. Big data: DW vs. single-constraint heuristics

We compare DW model with single-constraint heuristics that focus exclusively on one of the considerations of fairness, popularity bias, or diversity. The models are compared using the whole MovieLens 20M dataset. The notations Div(X) and Pop(Y) are used to describe solutions with diversity incentive value X, and popularity penalty value Y.

4.5.4.1. Diversity. In this subsection, we conduct a comparison between the results of the DW model and the "Diversity" model (see Section 4.4.2.1). The solutions of the heuristic is presented in Figure 4.11 under the label "Diversity." The rest of the labels illustrate the solutions of the DW Model with respect to different popularity penalty values. For each model, we select various diversity incentive values (Table 4.2) to compare the solution qualities. We show the popularity and fairness values of these models in Figure 4.12 and Figure 4.13, respectively.

The DW solution with zero popularity penalty increases the fairness metric (Figure 4.13) while performing very similarly to Diversity model in utility and diversity (Figure 4.11) considerations. We can observe that the utility decreases around 1% when we increase the popularity penalty, but the popularity value improves as well (Figure 4.12). Overall, DW model with popularity penalty zero finds as high diversity solutions as the Diversity model, while improving the fairness.

4.5.4.2. Popularity. In this subsection, we conduct a comparison between the results of the DW model and the "Popularity" model (see Section 4.4.2.2). The solutions of the heuristic is presented in Figure 4.15 under the label "Popularity." The rest of the labels illustrate the solutions of the DW Model with respect to different diversity incentive values. For each model, we select various popularity penalty values (Table 4.2) to compare the solution qualities. We show the diversity and fairness values of these models in the Figure 4.16 and Figure 4.17, respectively.

We note that the DW model with diversity incentive zero (model does not consider diversity) results in higher fairness (Figure 4.17) while decreasing the utility by less than 0.5% compared to Popularity heuristic solution. For example, DW solution with diversity incentive zero with worst fairness has Z value of 0.984, with standard deviation 26566. The number of recommendations take values between 53 to 130 thousands. Comparatively, Popularity model with worst fairness has Z value of 0.963, with standard deviation 40786. The number of recommendations take values between 29 to 183 thousands. In other terms, inclusion of fairness consideration increases the recommendations from less represented providers, while decreases the recommendations from more represented ones. Lastly, we observe that as the diversity incentive increases, utility decreases (Figure 4.15) as expected. However, the DW

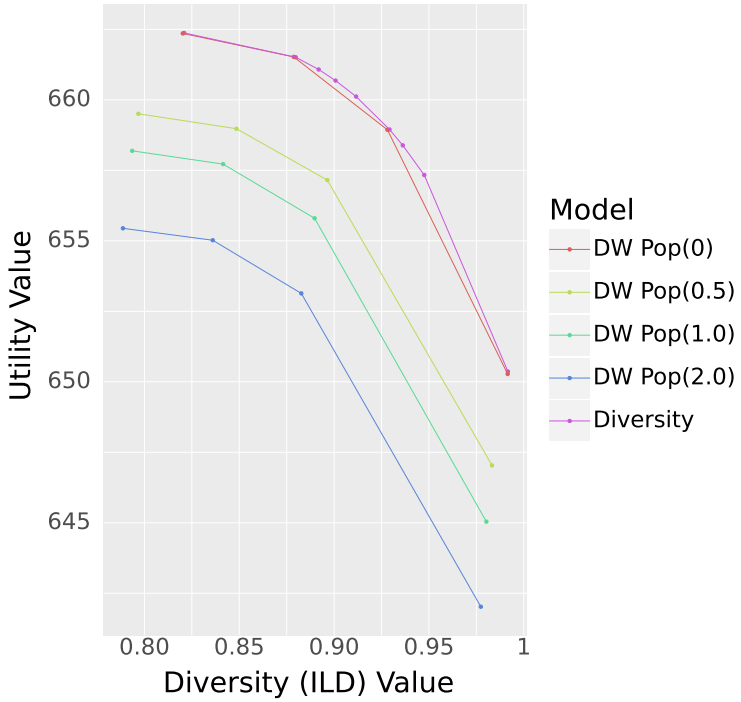


Figure 4.11. Diversity-Utility Graph

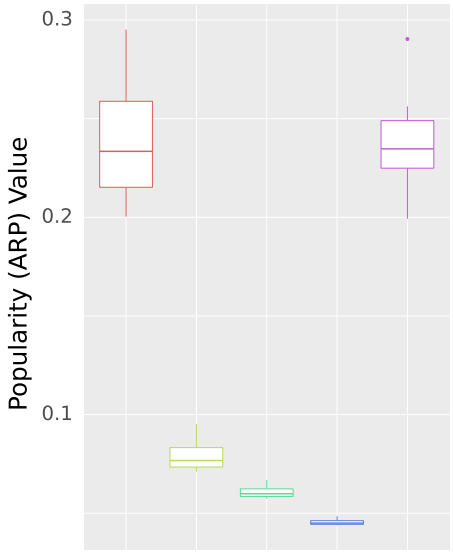


Figure 4.12. Popularity metric evaluation of DW and Diversity models

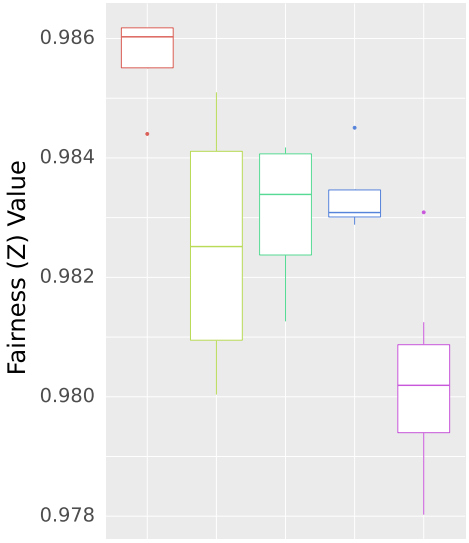


Figure 4.13. Z metric evaluation of DW and Diversity models

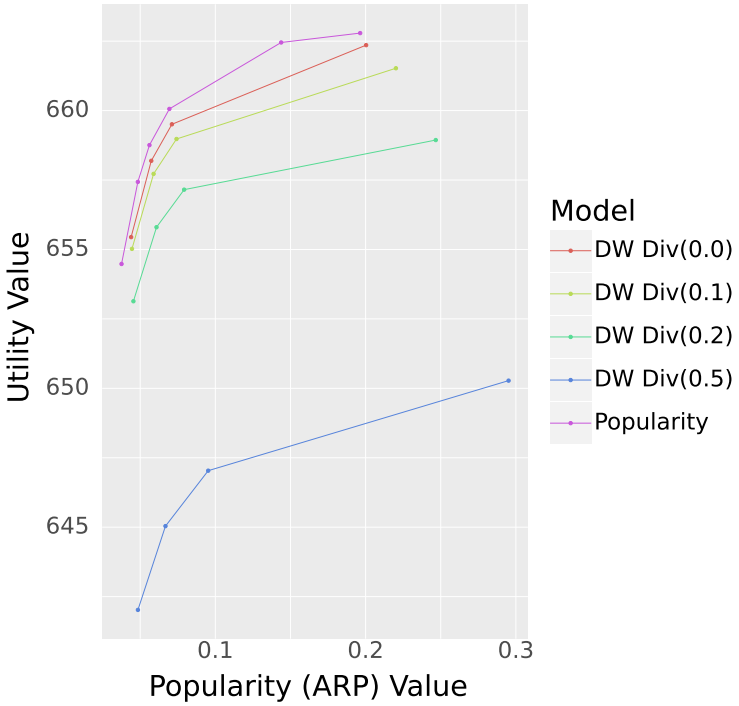


Figure 4.15. Popularity-Utility Graph

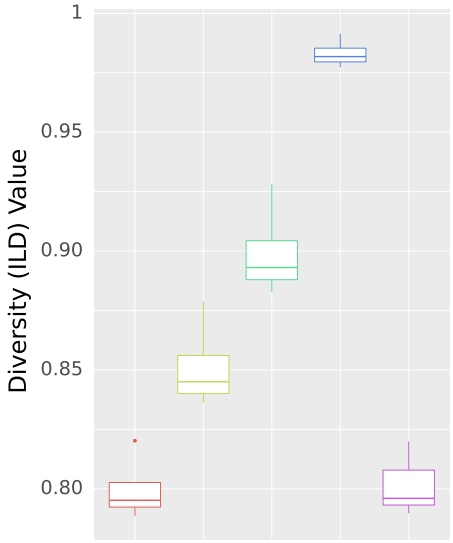


Figure 4.16. Diversity metric evaluation of DW and Popularity models

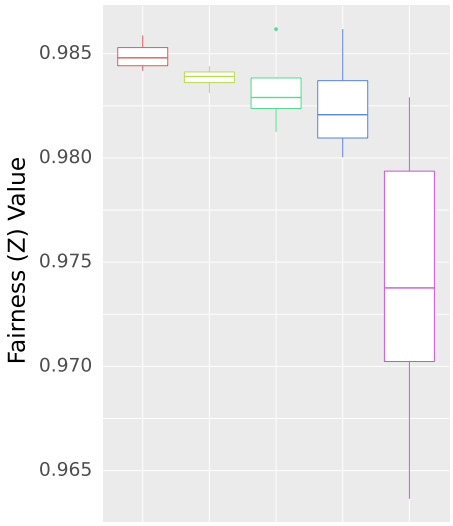


Figure 4.17. *Z* metric evaluation of DW and Popularity models

solution with diversity incentive 0.1 improves all the considerations simultaneously with a utility reduction of less than 0.5%.

4.5.4.3. Fairness. In this subsection, we compare DW results with a state-of-the-art algorithm Patro et al. [2020] that tackles fairness consideration (see Section 4.4.2.3). We modify the available code to solve fairness at the provider level, and name it "Fairness" model. The labels starting with DW in Figure 4.19 illustrate the solution quality of the DW Model with respect to different diversity incentive and popularity penalty values. For each model, we select variety of different weights to control the fairness. We show the popularity and diversity values of these models in the Figure 4.20 and Figure 4.21, respectively.

The DW solution with zero popularity penalty and diversity incentive performs better than the Fairness model in all considerations. DW model uses optimization ideas that offer lists holistically [Seymen et al., 2021a] rather than one step at a time, which might explain the higher utility and fairness solutions observed in Figure 4.19. Interestingly, the DW solutions with zero popularity penalty performs better than the Fairness solutions (Figure 4.20) in the popularity consideration. This might be due to incorporating upper bounds to item provider recommendation numbers, resulting in a decreased number of "popular" items recommended [Seymen et al., 2021b]. Lastly, we note that DW solution with diversity incentive 0.1 and popularity penalty 0.5 can improve diversity 3% and popularity bias 70% with a decrease less than 1% in utility.

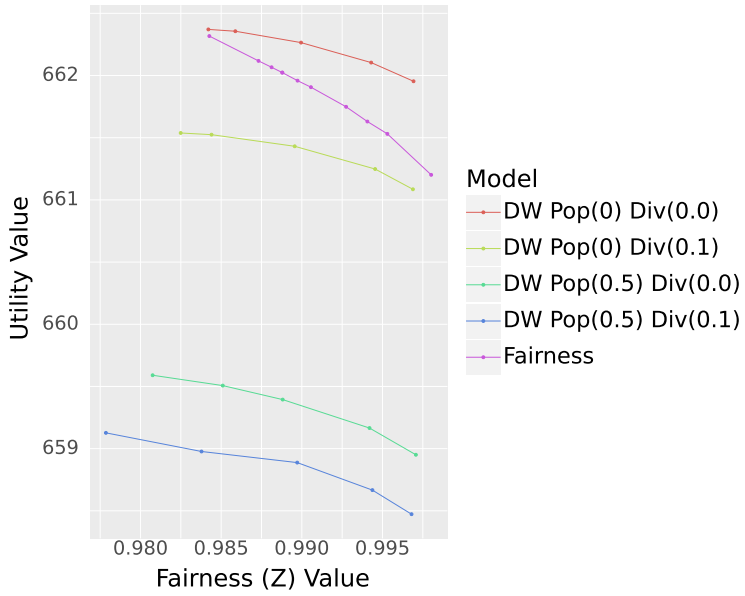


Figure 4.19. Fairness-Utility Graph

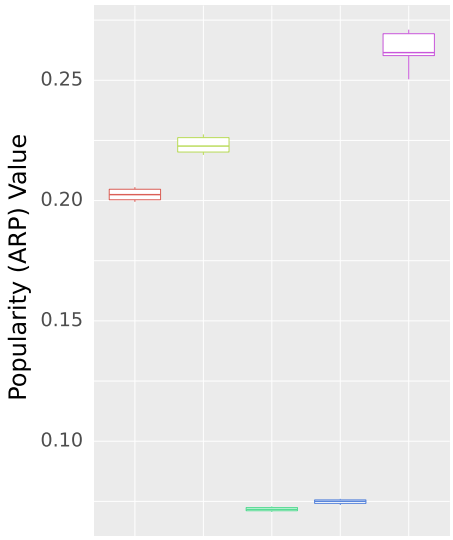


Figure 4.20. Popularity metric evaluation of DW and Fairness models

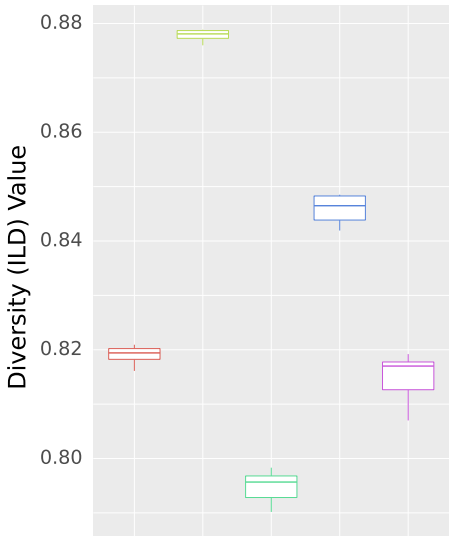


Figure 4.21. Diversity metric evaluation of DW and Fairness models

4.6. Discussion & Future Direction

In this work, we offer a post-processing large-scale optimization model inspired by Dantzig-Wolfe (DW) decomposition. The DW model improves the scalability of constrained optimization models by leveraging the structure of across-list constraints in RS, and can obtain near-optimal solutions.

First, we compare DW model with a constrained optimization model which finds the exact solution using a small fraction of MovieLens 20M dataset. We show that solution quality of DW is near-optimal. Next, we compare the DW model with benchmarks focusing on only one consideration at a time using the entire MovieLens 20M dataset. We note that our model can improve multiple objectives simultaneously and perform as good as the benchmarks focusing only on one objective. Therefore, we argue that our model is scalable, efficient in finding near-optimal solutions, and capable of handling multiple objective functions simultaneously in RS.

We offer a heuristic to increase the memory-efficiency of our model, which can find the sets of most relevant items considering popularity, diversity, and fairness. This heuristic removes a significant number of items that would not be recommended to users. Therefore, the optimal solution is not removed from the feasible region. This approach is not model dependent, and can be used by other RS approaches in the literature.

Feasible region reduction ideas are important for optimization models that may face memory-related problems. We note that our model and the heuristic work most efficiently when the number of across-list constraints in the optimization model is relatively small. Therefore, one future research area is coming up with ideas that work well in case of high number of across-list constraints. Similarly, investigation of across-list constraints and ways

to deal with them in an online setting is a highly relevant future research area, especially considering businesses that offer real-time recommendations to customers.

The objective of this work is to address the issue of scalability in constrained optimization models by leveraging the structure of across-list constraints. We showcase the high research potential of using large-scale optimization ideas in MORS problems. We note that using a variety of different metrics and measures can lead to markedly different contributions in tackling a multitude of MORS problems.

CHAPTER 5

Future Work

In-depth discussion on scalability problem: In Chapter 2, we solve the scalability issue completely by solving calibrated recommender system (RS) problems for each user independently. In Chapter 3, we propose heuristics and reformulations to alleviate the scalability problem, without causing a significant objective function value decrease. In Chapter 4, we propose a large-scale optimization model that uses the Dantzig-Wolfe (DW) decomposition method idea to solve massive datasets. However, the application of large-scale optimization ideas to RS problems is not frequently researched. Therefore, there is still a significant amount of research potential in investigating optimization models to tackle massive RS problems. Combining heuristics with large-scale models, or improving the base DW algorithm to improve the computational time of our model can be investigated further. We defined complicating constraints as across-list constraints, and there could be more specialized ways to tackle these constraints by exploiting the special structures arising in different problems. Using the DW approach might not be the best choice to apply to some problems, and future research can define and propose applications of other algorithms.

Optimization in online setting: We conducted our experiments in the offline setting, however, the constrained optimization models we offer can be extended to the online setting. One direction that can be taken is testing the validity of our models in an online setting, where the utilities and preferences of the users can be updated after the recommendations. This is relevant for a multitude of reasons. Firstly, the recommended items can get feedback from the user after the creation of recommendation lists. The calibration values in Chapter 2 can be continuously updated, which might result in a better understanding of user preferences. The users' preferences and interests might change over time, and this change can be captured better in an online setting. Secondly, in Chapter 3, an online setting can

capture the stockout and perishability changes better, giving more insight into the solution quality of the optimization model. In an offline setting, it is more difficult to measure the benefits of recommending a novel item to a user. Because the recommendation is not in the test set, the recommendation of a novel item would be a miss. However, in reality, the user might be positively surprised about the new recommendation our models suggest, and rate that recommendation highly, which can be observed more easily in an online setting.

Investigating interactions of metrics: In Chapter 2, we notice that improving fairness might result in recommendations with less popularity bias. The increase in the calibration of the recommendations can increase the accuracy first, then decrease it. In Chapter 3, recommending soon-to-perish items results in lower stockouts. One future research focus is to better our understanding of the interactions of these metrics and considerations. To that end, we might want to apply different constrained optimization models to tackle multiple considerations using a variety of datasets. Currently, the impact of each consideration on the quality of the solution is not well understood. A comprehensive work delving into further details of metric interactions would be one future research area.

5.1. Publications

The works discussed in this dissertation resulted in three publications:

- First section of Chapter 2 published as [Seymen et al., 2021b]: Sinan Seymen, Himan Abdollahpouri, and Edward Carl Malthouse. A unified optimization toolbox for solving popularity bias, fairness, and diversity in recommender systems. In Workshop of Multi-Objective Recommender Systems (MORS'21), in conjunction with the 15th ACM Conference on Recommender Systems, RecSys, 2021b.

- Second section of Chapter 2 published as [Seymen et al., 2021a]: Sinan Seymen, Himan Abdollahpouri, and Edward C. Malthouse. A constrained optimization approach for calibrated recommendations. In Proceedings of the 15th ACM Conference on Recommender Systems, RecSys '21, page 607–612, New York, NY, USA, 2021a. Association for Computing Machinery. ISBN 9781450384582. doi: 10.1145/3460231.3478857. URL https://doi.org/10.1145/3460231.3478857.
- Chapter 3 published as [Seymen et al., 2022]: Sinan Seymen, Anna-Lena Sachs, and Edward C Malthouse. Making smart recommenda- tions for perishable and stockout products. In Workshop of Multi-Objective Recommender Systems (MORS'22), in conjunction with the 16th ACM Conference on Recommender Systems, RecSys, 2022.

References

Himan Abdollahpouri. *Popularity Bias in Recommendation: A Multi-stakeholder Perspective*. PhD thesis, University of Colorado Boulder, https://arxiv.org/pdf/2008.08551.pdf, 8 2020.

Himan Abdollahpouri, Robin Burke, and Bamshad Mobasher. Controlling popularity bias in learning-to-rank recommendation. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*, RecSys '17, page 42–46, New York, NY, USA, 2017a. Association for Computing Machinery. ISBN 9781450346528. doi: 10.1145/3109859.3109912. URL https://doi.org/10.1145/3109859.3109912.

Himan Abdollahpouri, Robin Burke, and Bamshad Mobasher. Recommender systems as multistakeholder environments. In *Proceedings of the 25th Conference on User Modeling, Adaptation and Personalization*, UMAP '17, page 347–348, New York, NY, USA, 2017b. Association for Computing Machinery. ISBN 9781450346351. doi: 10.1145/3079628.3079657. URL https://doi.org/10.1145/3079628.3079657.

Himan Abdollahpouri, Robin Burke, and Bamshad Mobasher. Managing popularity bias in recommender systems with personalized re-ranking. In *The Thirty-Second International Flairs Conference*, 2019a.

Himan Abdollahpouri, Masoud Mansoury, Robin Burke, and Bamshad Mobasher. The unfairness of popularity bias in recommendation. *arXiv preprint arXiv:1907.13286*, 2019b.

Himan Abdollahpouri, Gediminas Adomavicius, Robin Burke, Ido Guy, Dietmar Jannach, Toshihiro Kamishima, Jan Krasnodebski, and Luiz Pizzato. Multistakeholder recommendation: Survey and research directions. *User Modeling and User-Adapted Interaction*, 30: 127–158, 2020. URL https://doi.org/10.1007/s11257-019-09256-1.

Gediminas Adomavicius and YoungOk Kwon. Improving aggregate recommendation diversity using ranking-based techniques. *IEEE Transactions on Knowledge and Data Engineering*, 24(5):896–911, 2011a.

Gediminas Adomavicius and YoungOk Kwon. Maximizing aggregate recommendation diversity: A graph-theoretic approach. In *Proc. of the 1st International Workshop on Novelty and Diversity in Recommender Systems (DiveRS 2011)*, pages 3–10. Citeseer, 2011b.

Deepak Agarwal, Bee-Chung Chen, Pradheep Elango, and Xuanhui Wang. Click shaping to optimize multiple objectives. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '11, page 132–140, New York, NY, USA, 2011. Association for Computing Machinery. ISBN 9781450308137. doi: 10.1145/2020408.2020435. URL https://doi.org/10.1145/2020408.2020435.

Deepak Agarwal, Bee-Chung Chen, Pradheep Elango, and Xuanhui Wang. Personalized click shaping through lagrangian duality for online recommendation. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '12, page 485–494, New York, NY, USA, 2012. Association for Computing Machinery. ISBN 9781450314725. doi: 10.1145/2348283.2348350. URL https://doi.org/10.1145/2348283.2348350.

Narendra Agrawal, Sami Najafi-Asadolahi, and Stephen A Smith. Optimization of operational decisions in digital advertising: A literature review. *Channel Strategies and Marketing Mix in a Connected World*, pages 99–146, 2020. doi: 10.1007/978-3-030-31733-1_5.

URL https://doi.org/10.1007/978-3-030-31733-1_5.

Shabbir Ahmed and Alexander Shapiro. The sample average approximation method for stochastic programs with integer recourse. *Science*, 12, 01 2002.

Leman Akoglu and Christos Faloutsos. Valuepick: Towards a value-oriented dual-goal recommender system. In *2010 IEEE International Conference on Data Mining Workshops*, pages 1151–1158. IEEE, 2010.

Seyyed Amir Hossein Salehi Amiri, Ali Zahedi, Morteza Kazemi, Javad Soroor, and Mostafa Hajiaghaei-Keshteli. Determination of the optimal sales level of perishable goods in a two-echelon supply chain network. *Computers & Industrial Engineering*, 139:106156, 2020.

Eric T Anderson, Gavan J Fitzsimons, and Duncan Simester. Measuring and mitigating the costs of stockouts. *Management Science*, 52(11):1751–1763, 2006.

Arda Antikacioglu and R. Ravi. Post processing recommender systems for diversity. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, page 707–716, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450348874. doi: 10.1145/3097983.3098173. URL https://doi.org/10.1145/3097983.3098173.

Antonio Bella, Cèsar Ferri, José Hernández-Orallo, and María José Ramírez-Quintana. Calibration of machine learning models. In *Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques*, pages 128–146. IGI Global, 2010.

Fernando Bernstein, A Gürhan Kök, and Lei Xie. Dynamic assortment customization with limited inventories. *Manufacturing & Service Operations Management*, 17(4):538–553, 2015.

Michael Blakeney. *Food loss and food waste: Causes and solutions*. Edward Elgar Publishing, 2019.

K. Bradley and B. Smyth. Improving recommendation diversity. In *Proceedings of the 12th National Conference in Artificial Intelligence and Cognitive Science*, pages 75–84, Maynooth, Ireland, 2001.

Robin Burke and Himan Abdollahpouri. Patterns of multistakeholder recommendation. *arXiv preprint arXiv:1707.09258*, 2017.

Pablo Castells, Neil Hurley, and Saul Vargas. Novelty and diversity in recommender systems. In *Recommender Systems Handbook*, pages 603–646. Springer, 2021. URL `https://doi.org/10.1007/978-1-0716-2197-4_16`.

Antonin Chambolle. An algorithm for total variation minimization and applications. *Journal of Mathematical Imaging and Cision*, 20(1):89–97, 2004.

Du Chen, Yuming Deng, Guangrui Ma, Hao Ge, Yunwei Qi, Ying Rong, Xun Zhang, and Huan Zheng. Inventory based recommendation algorithms. In *2020 IEEE International Conference on Big Data (Big Data)*, pages 617–622. IEEE, 2020.

Kwok-Wai Cheung, James T Kwok, Martin H Law, and Kwok-Ching Tsui. Mining customer product ratings for personalized marketing. *Decision Support Systems*, 35(2):231–243, 2003.

Geoffrey A Chua, Reza Mokhlesi, and Arvind Sainathan. Optimal discounting and replenishment policies for perishable products. *International Journal of Production Economics*, 186:8–20, 2017.

C. Dadouchi and B. Agard. Lowering penalties related to stock-outs by shifting demand in product recommendation systems. *Decision Support Systems*, 114:61–69, 2018. ISSN 0167-9236. doi: https://doi.org/10.1016/j.dss.2018.08.004. URL `https://www.`

sciencedirect.com/science/article/pii/S0167923618301258.

George B Dantzig and Philip Wolfe. Decomposition principle for linear programs. *Operations research*, 8(1):101–111, 1960.

Aparna Das, Claire Mathieu, and Daniel Ricketts. Maximizing profit using recommender systems. *arXiv preprint arXiv:0908.3633*, 2009.

Yashar Deldjoo, Dietmar Jannach, Alejandro Bellogin, Alessandro Difonzo, and Dario Zanzonelli. A survey of research on fair recommender systems. *arXiv preprint arXiv:2205.11127*, 2022.

Yashar Deldjoo, Dietmar Jannach, Alejandro Bellogin, Alessandro Difonzo, and Dario Zanzonelli. Fairness in recommender systems: research landscape and future directions. *User Modeling and User-Adapted Interaction*, pages 1–50, 2023. URL https://doi.org/10.1007/s11257-023-09364-z.

Tommaso Di Noia, Jessica Rosati, Paolo Tomeo, and Eugenio Di Sciascio. Adaptive multi-attribute diversity for recommender systems. *Information Sciences*, 382:234–253, 2017.

M. Benjamin Dias, Dominique Locher, Ming Li, Wael El-Deredy, and Paulo J.G. Lisboa. The value of personalised recommender systems to e-business: A case study. In *Proceedings of the 2008 ACM Conference on Recommender Systems*, RecSys '08, page 291–294, New York, NY, USA, 2008. Association for Computing Machinery. ISBN 9781605580937. doi: 10.1145/1454008.1454054. URL https://doi.org/10.1145/1454008.1454054.

Jorge Díez, David Martínez-Rego, Amparo Alonso-Betanzos, Oscar Luaces, and Antonio Bahamonde. Optimizing novelty and diversity in recommendations. *Progress in Artificial Intelligence*, 8(1):101–109, 2019. ISSN 2192-6360. doi: 10.1007/s13748-018-0158-4. URL https://doi.org/10.1007/s13748-018-0158-4.

Alexandre Dolgui, Manoj Kumar Tiwari, Yerasani Sinjana, Sri Krishna Kumar, and Young-Jun Son. Optimising integrated inventory policy for perishable items in a multi-stage supply chain. *International Journal of Production Research*, 56(1-2):902–925, 2018.

Qinglin Duan and T Warren Liao. A new age-based replenishment policy for supply chain inventory optimization of highly perishable products. *International Journal of Production Economics*, 145(2):658–671, 2013.

Farzad Eskandanian, Bamshad Mobasher, and Robin Burke. A clustering approach for personalizing diversity in collaborative recommender systems. In *Proceedings of the 25th Conference on User Modeling, Adaptation and Personalization*, pages 280–284, 2017.

Tijun Fan, Chang Xu, and Feng Tao. Dynamic pricing and replenishment policy for fresh produce. *Computers & Industrial Engineering*, 139:106127, 2020.

Allen R Ferguson and George B Dantzig. The allocation of aircraft to routes—an example of linear programming under uncertain demand. *Management Science*, 3(1):45–73, 1956.

Daniel Fleder and Kartik Hosanagar. Blockbuster culture's next rise or fall: The impact of recommender systems on sales diversity. *Management Science*, 55(5):697–712, 2009. URL `https://doi.org/10.1287/mnsc.1080.0974`.

Ruoyuan Gao and Chirag Shah. Toward creating a fairer ranking in search engine results. *Information Processing & Management*, 57(1):102138, 2020. ISSN 0306-4573. doi: https://doi.org/10.1016/j.ipm.2019.102138. URL `https://www.sciencedirect.com/science/article/pii/S0306457319304121`.

Mouzhi Ge, Carla Delgado-Battenfeld, and Dietmar Jannach. Beyond accuracy: Evaluating recommender systems by coverage and serendipity. In *Proceedings of the Fourth ACM Conference on Recommender Systems*, RecSys '10, page 257–260, New York, NY, USA, 2010. Association for Computing Machinery. ISBN 9781605589060. doi: 10.1145/1864708.

1864761. URL https://doi.org/10.1145/1864708.1864761.

Thomas George and Srujana Merugu. A scalable collaborative filtering framework based on co-clustering. In *Fifth IEEE International Conference on Data Mining (ICDM'05)*, pages 4–pp. IEEE, 2005.

Anupriya Gogna and Angshul Majumdar. Diablo: Optimization based design for improving diversity in recommender system. *Information Sciences*, 378:59–74, 2017.

Peter S. Goodman. New supply chain risk: 22,000 dockworkers who may soon strike. *New York Times*, 03 2022. URL https://www.nytimes.com/2022/03/28/business/dockworkers-strike-supply-chain.html.

Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2021. URL https://www.gurobi.com.

Alejandro Gutierrez-Alcoba, Roberto Rossi, Belen Martin-Barragan, and Eligius MT Hendrix. A simple heuristic for perishable item inventory control under non-stationary stochastic demand. *International Journal of Production Research*, 55(7):1885–1897, 2017.

F. Maxwell Harper and Joseph A. Konstan. The movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst.*, 5(4), dec 2015. ISSN 2160-6455. doi: 10.1145/2827872. URL https://doi.org/10.1145/2827872.

Naieme Hazrati and Francesco Ricci. Recommender systems effect on the evolution of users' choices distribution. *Information Processing & Management*, 59(1):102766, 2022. ISSN 0306-4573. doi: https://doi.org/10.1016/j.ipm.2021.102766. URL https://www.sciencedirect.com/science/article/pii/S0306457321002466.

Jonathan L Herlocker, Joseph A Konstan, Loren G Terveen, and John T Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)*, 22(1):5–53, 2004.

Maarten LATM Hertog, Ismail Uysal, Ultan McCarthy, Bert M Verlinden, and Bart M Nicolaï. Shelf life modelling for first-expired-first-out warehouse management. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 372(2017):20130306, 2014.

Nicolas Hug. Surprise, a Python library for recommender systems. `http://surpriselib.com`, 2017.

Nicolas Hug. Surprise: A python library for recommender systems. *Journal of Open Source Software*, 5(52):2174, 2020a. doi: 10.21105/joss.02174. URL `https://doi.org/10.21105/joss.02174`.

Nicolas Hug. Surprise: A python library for recommender systems. *Journal of Open Source Software*, 5(52):2174, 2020b.

Tamas Jambor and Jun Wang. Optimizing multiple objectives in collaborative filtering. In *Proceedings of the Fourth ACM Conference on Recommender Systems*, RecSys '10, page 55–62, New York, NY, USA, 2010. Association for Computing Machinery. ISBN 9781605589060. doi: 10.1145/1864708.1864723. URL `https://doi.org/10.1145/1864708.1864723`.

Dietmar Jannach. Multi-objective recommender systems: Survey and challenges. *arXiv preprint arXiv:2210.10309*, 2022.

Dietmar Jannach and Michael Jugovac. Measuring the business value of recommender systems. *ACM Trans. Manage. Inf. Syst.*, 10(4), dec 2019. ISSN 2158-656X. doi: 10.1145/3370082. URL `https://doi.org/10.1145/3370082`.

Dietmar Jannach, Lukas Lerche, Iman Kamehkhosh, and Michael Jugovac. What recommenders recommend: an analysis of recommendation biases and possible countermeasures. *User Modeling and User-Adapted Interaction*, 25:427–491, 2015. URL `https:`

//doi.org/10.1007/s11257-015-9165-3.

Xiaoqing Jing and Michael Lewis. Stockouts in online retailing. *Journal of Marketing Research*, 48(2):342–354, 2011. doi: 10.1509/jmkr.48.2.342. URL https://doi.org/10.1509/jmkr.48.2.342.

Michael Jugovac, Dietmar Jannach, and Lukas Lerche. Efficient optimization of multiple recommendation quality factors according to individual user tendencies. *Expert Systems with Applications*, 81:321–331, 2017.

Marius Kaminskas and Derek Bridge. Diversity, serendipity, novelty, and coverage: A survey and empirical analysis of beyond-accuracy objectives in recommender systems. *ACM Trans. Interact. Intell. Syst.*, 7(1), dec 2016. ISSN 2160-6455. doi: 10.1145/2926720. URL https://doi.org/10.1145/2926720.

Marius Kaminskas, Derek Bridge, Franclin Foping, and Donogh Roche. Product-seeded and basket-seeded recommendations for small-scale retailers. *Journal on Data Semantics*, 6 (1):3–14, 2017.

Toshihiro Kamishima, Shotaro Akaho, Hideki Asoh, and Jun Sakuma. Correcting popularity bias by enhancing recommendation neutrality. In *RecSys Posters*, 10 2014.

Itir Z Karaesmen, Alan Scheller-Wolf, and Borga Deniz. Managing perishable and aging inventories: review and future research directions. *Planning Production and Inventories in the Extended Enterprise*, pages 393–436, 2011. doi: 10.1007/978-1-4419-6485-4_15. URL https://doi.org/10.1007/978-1-4419-6485-4_15.

Sujin Kim, Raghu Pasupathy, and Shane G. Henderson. *A guide to sample average approximation*, pages 207–243. Springer, New York, NY, 2015. ISBN 978-1-4939-1384-8. doi: 10.1007/978-1-4939-1384-8_8. URL https://doi.org/10.1007/978-1-4939-1384-8_8.

Mervegül Kırcı, Işık Biçer, and Ralf W Seifert. Optimal replenishment cycle for perishable items facing demand uncertainty in a two-echelon inventory system. *International Journal of Production Research*, 57(4):1250–1264, 2019.

Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009. doi: 10.1109/MC.2009.263.

Denis Kotkov, Joseph A. Konstan, Qian Zhao, and Jari Veijalainen. Investigating serendipity in recommender systems based on real user feedback. In *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*, SAC '18, page 1341–1350, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450351911. doi: 10.1145/3167132.3167276. URL https://doi.org/10.1145/3167132.3167276.

Matevž Kunaver and Tomaž Požrl. Diversity in recommender systems – a survey. *Knowledge-Based Systems*, 123:154–162, 2017. ISSN 0950-7051. doi: https://doi.org/10.1016/j.knosys.2017.02.009. URL https://www.sciencedirect.com/science/article/pii/S0950705117300680.

Sandra Lebersorger and Felicitas Schneider. Food loss rates at the food retail, influencing factors and reasons as a basis for waste prevention measures. *Waste Management*, 34(11):1911–1919, 2014.

David A Levin and Yuval Peres. *Markov chains and mixing times*, volume 107. American Mathematical Soc., 2017.

Randall A Lewis and David H Reiley. Online ads and offline sales: measuring the effect of retail advertising via a controlled experiment on yahoo! *Quantitative Marketing and Economics*, 12(3):235–266, 2014.

Koung-Lung Lin, JY-J Hsu, Han-Shen Huang, and Chun-Nan Hsu. A recommender for targeted advertisement of unsought products in e-commerce. In *Seventh IEEE International*

Conference on E-Commerce Technology (CEC'05), pages 101–108. IEEE, 2005.

Brian Lipinski, Craig Hanson, Richard Waite, Tim Searchinger, James Lomax, and Lisa Kitinoja. Installment 2 of "creating a sustainable food future": Reducing food loss and waste. *World Resources Institute Working Paper*, pages 1–40, 2013. URL `https://search.issuelab.org/resource/installment-2-of-creating-a-sustainable-food-future-reducing-food-loss-and-waste.html`.

Weiwen Liu and Robin Burke. Personalizing fairness-aware re-ranking. In *International Workshop on Fairness Accountability and Transpatency in Recommender Systems (FAC-TREC)*, arXiv preprint arXiv:1809.02921, 2018.

Edward C Malthouse, Yasaman Kamyab Hessary, Khadija Ali Vakeel, Robin Burke, and Morana Fudurić. An algorithm for allocating sponsored recommendations and content: Unifying programmatic advertising and recommender systems. *Journal of Advertising*, 48 (4):366–379, 2019a.

Edward C Malthouse, Khadija Ali Vakeel, Yasaman Kamyab Hessary, Robin Burke, and Morana Fuduric. A multistakeholder recommender systems algorithm for allocating sponsored recommendations. In *In Workshop on Recommendation in Multi-stakeholder Environments with ACM RecSys19*, 2019b.

Masoud Mansoury, Himan Abdollahpouri, Mykola Pechenizkiy, Bamshad Mobasher, and Robin Burke. 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*, UMAP '20, page 154–162, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450368612. doi: 10.1145/3340631.3394860. URL `https://doi.org/10.1145/3340631.3394860`.

Sébastien Marchand. The relationship between technical efficiency in agriculture and deforestation in the brazilian amazon. *Ecological Economics*, 77:166–175, 2012.

Silvano Martello and Paolo Toth. Algorithms for knapsack problems. In Silvano Martello, Gilbert Laporte, Michel Minoux, and Celso Ribeiro, editors, *Surveys in Combinatorial Optimization*, volume 132 of *North-Holland Mathematics Studies*, pages 213–257. North-Holland, 1987. doi: https://doi.org/10.1016/S0304-0208(08)73237-7. URL https://www.sciencedirect.com/science/article/pii/S0304020808732377.

Prem Melville and Vikas Sindhwani. *Recommender Systems*, pages 1056–1066. Springer, Boston, MA, 2017. ISBN 978-1-4899-7687-1. doi: 10.1007/978-1-4899-7687-1_964. URL https://doi.org/10.1007/978-1-4899-7687-1_964.

David R. Morrison, Sheldon H. Jacobson, Jason J. Sauppe, and Edward C. Sewell. Branch-and-bound algorithms: A survey of recent advances in searching, branching, and pruning. *Discrete Optimization*, 19:79–102, 2016. ISSN 1572-5286. doi: 10.1016/j.disopt.2016.01.005. URL https://www.sciencedirect.com/science/article/pii/S1572528616000062.

Scott A Neslin. Sales promotion. *Handbook of Marketing*, 13:311–338, 2002. URL https://sk.sagepub.com/reference/hdbk_marketing.

Duc Huy Nguyen and Haoxun Chen. Optimization of a perishable inventory system with both stochastic demand and supply: comparison of two scenario approaches. *Croatian Operational Research Review*, 10(1):175–185, 2019.

Duc Huy Nguyen and Haoxun Chen. An effective approach for optimization of a perishable inventory system with uncertainty in both demand and supply. *International Transactions in Operational Research*, 29(4):2682–2704, 2022.

Jinoh Oh, Sun Park, Hwanjo Yu, Min Song, and Seung-Taek Park. Novel recommendation based on personal popularity tendency. In *2011 IEEE 11th International Conference on Data Mining*, pages 507–516. IEEE, 2011. doi: 10.1109/ICDM.2011.110.

Gourab K Patro, Arpita Biswas, Niloy Ganguly, Krishna P. Gummadi, and Abhijnan Chakraborty. Fairrec: Two-sided fairness for personalized recommendations in two-sided platforms. In *Proceedings of The Web Conference 2020*, WWW '20, page 1194–1204, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450370233. doi: 10.1145/3366423.3380196. URL https://doi.org/10.1145/3366423.3380196.

Judea Pearl. *Heuristics: Intelligent search strategies for computer problem solving*. Addison-Wesley Longman Publishing Co., Inc., USA, 1984. ISBN 0201055945.

Mark S Pinsker. *Information and information stability of random variables and processes*. Holden-Day, 1964.

Marco Tulio Ribeiro, Anisio Lacerda, Adriano Veloso, and Nivio Ziviani. Pareto-efficient hybridization for multi-objective recommender systems. In *Proceedings of the Sixth ACM Conference on Recommender Systems*, RecSys '12, page 19–26, New York, NY, USA, 2012. Association for Computing Machinery. ISBN 9781450312707. doi: 10.1145/2365952.2365962. URL https://doi.org/10.1145/2365952.2365962.

Timothy J Richards and Jura Liaukonytė. Switching cost and store choice. *American Journal of Agricultural Economics*, 105(1):195–218, 2023.

Mario Rodriguez, Christian Posse, and Ethan Zhang. Multiple objective optimization in recommender systems. In *Proceedings of the Sixth ACM Conference on Recommender Systems*, RecSys '12, page 11–18, New York, NY, USA, 2012. Association for Computing Machinery. ISBN 9781450312707. doi: 10.1145/2365952.2365961. URL https://doi.org/10.1145/2365952.2365961.

Navdeep S Sahni, Dan Zou, and Pradeep K Chintagunta. Do targeted discount offers serve as advertising? evidence from 70 field experiments. *Management Science*, 63(8):2688–2705, 2017.

Sinan Seymen, Himan Abdollahpouri, and Edward C. Malthouse. A constrained optimization approach for calibrated recommendations. In *Proceedings of the 15th ACM Conference on Recommender Systems*, RecSys '21, page 607–612, New York, NY, USA, 2021a. Association for Computing Machinery. ISBN 9781450384582. doi: 10.1145/3460231.3478857. URL https://doi.org/10.1145/3460231.3478857.

Sinan Seymen, Himan Abdollahpouri, and Edward Carl Malthouse. A unified optimization toolbox for solving popularity bias, fairness, and diversity in recommender systems. In *Workshop of Multi-Objective Recommender Systems (MORS'21), in conjunction with the 15th ACM Conference on Recommender Systems, RecSys*, 2021b.

Sinan Seymen, Anna-Lena Sachs, and Edward C Malthouse. Making smart recommendations for perishable and stockout products. In *Workshop of Multi-Objective Recommender Systems (MORS'22), in conjunction with the 16th ACM Conference on Recommender Systems, RecSys*, 2022.

Stephen A Smith and Dale D Achabal. Clearance pricing and inventory policies for retail chains. *Management Science*, 44(3):285–300, 1998.

Barry Smyth and Paul McClave. Similarity vs. diversity. In *Case-Based Reasoning Research and Development: 4th International Conference on Case-Based Reasoning, ICCBR 2001 Vancouver, BC, Canada, July 30–August 2, 2001 Proceedings 4*, pages 347–361. Springer, 2001. URL https://doi.org/10.1007/3-540-44593-5.

Ilan Stavi and Rattan Lal. Agriculture and greenhouse gases, a common tragedy. a review. *Agronomy for Sustainable Development*, 33(2):275–289, 2013.

Harald Steck. Item popularity and recommendation accuracy. In *Proceedings of the Fifth ACM Conference on Recommender Systems*, RecSys '11, page 125–132, New York, NY, USA, 2011. Association for Computing Machinery. ISBN 9781450306836. doi: 10.1145/2043932.2043957. URL https://doi.org/10.1145/2043932.2043957.

Harald Steck. Calibrated recommendations. In *Proceedings of the 12th ACM Conference on Recommender Systems*, RecSys '18, page 154–162, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450359016. doi: 10.1145/3240323.3240372. URL https://doi.org/10.1145/3240323.3240372.

Özge Sürer, Robin Burke, and Edward C. Malthouse. Multistakeholder recommendation with provider constraints. In *Proceedings of the 12th ACM Conference on Recommender Systems*, RecSys '18, page 54–62, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450359016. doi: 10.1145/3240323.3240350. URL https://doi.org/10.1145/3240323.3240350.

Masoud Talebian, Natashia Boland, and Martin Savelsbergh. Pricing to accelerate demand learning in dynamic assortment planning for perishable products. *European Journal of Operational Research*, 237(2):555–565, 2014.

Saúl Vargas and Pablo Castells. Rank and relevance in novelty and diversity metrics for recommender systems. In *Proceedings of the Fifth ACM Conference on Recommender Systems*, RecSys '11, page 109–116, New York, NY, USA, 2011. Association for Computing Machinery. ISBN 9781450306836. doi: 10.1145/2043932.2043955. URL https://doi.org/10.1145/2043932.2043955.

Saúl Vargas and Pablo Castells. Improving sales diversity by recommending users to items. In *Proceedings of the 8th ACM Conference on Recommender systems*, pages 145–152, 2014.

Mengting Wan, Di Wang, Jie Liu, Paul Bennett, and Julian McAuley. Representing and recommending shopping baskets with complementarity, compatibility and loyalty. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, CIKM '18, page 1133–1142, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450360142. doi: 10.1145/3269206.3271786. URL https://doi.org/10.1145/3269206.3271786.

Lequn Wang and Thorsten Joachims. User fairness, item fairness, and diversity for rankings in two-sided markets. In *Proceedings of the 2021 ACM SIGIR International Conference on Theory of Information Retrieval*. ACM, jul 2021. doi: 10.1145/3471158.3472260.

Yifan Wang, Weizhi Ma, Min Zhang, Yiqun Liu, and Shaoping Ma. A survey on the fairness of recommender systems. *ACM Trans. Inf. Syst.*, 41(3), feb 2023. ISSN 1046-8188. doi: 10.1145/3547333. URL https://doi.org/10.1145/3547333.

Christopher Wilt and Wheeler Ruml. Effective heuristics for suboptimal best-first search. *J. Artif. Int. Res.*, 57(1):273–306, sep 2016. ISSN 1076-9757.

Yang Zhang, Fuli Feng, Xiangnan He, Tianxin Wei, Chonggang Song, Guohui Ling, and Yongdong Zhang. Causal intervention for leveraging popularity bias in recommendation. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '21, page 11–20, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450380379. doi: 10.1145/3404835.3462875. URL https://doi.org/10.1145/3404835.3462875.

Tao Zhou, Jie Ren, Matús Medo, and Yi-Cheng Zhang. Bipartite network projection and personal recommendation. *Phys. Rev. E*, 76:046115, Oct 2007. doi: 10.1103/PhysRevE.76.046115. URL https://link.aps.org/doi/10.1103/PhysRevE.76.046115.

Tao Zhou, Zoltán Kuscsik, Jian-Guo Liu, Matúš Medo, Joseph Rushton Wakeling, and Yi-Cheng Zhang. Solving the apparent diversity-accuracy dilemma of recommender systems. *Proceedings of the National Academy of Sciences*, 107(10):4511–4515, 2010.

Cai-Nicolas Ziegler, Sean M. McNee, Joseph A. Konstan, and Georg Lausen. Improving recommendation lists through topic diversification. In *Proceedings of the 14th International Conference on World Wide Web*, WWW '05, page 22–32, New York, NY, USA, 2005. Association for Computing Machinery. ISBN 1595930469. doi: 10.1145/1060745.1060754. URL https://doi.org/10.1145/1060745.1060754.

Sarah Zimmerman. 10 disruptions that rocked supply chains in 2021. *Supply Chain Dive*, 2021. URL https://www.supplychaindive.com/news/top-supply-chain-disruptions-2021/611513/.