Carnegie Mellon University

MELLON COLLEGE OF SCIENCE

THESIS

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF Doctor of Philosophy
in Algorithms, Combinatorics and Optimization

TITLE Quantifying and Improving Sales Diversity in Recommender Systems

PRESENTED BY Arda Antikacioglu

ACCEPTED BY THE DEPARTMENT OF Mathematical Sciences

Ramamoorthi Ravi	May 2017
MAJOR PROFESSOR	DATE
Thomas Bohman	May 2017
DEPARTMENT HEAD	DATE

APPROVED BY THE COLLEGE COUNCIL

Rebecca W. Doerge	May 2017
DEAN	DATE

CARNEGIE MELLON UNIVERSITY

Quantifying and Improving Sales Diversity in Recommender Systems

by

Arda Antikacioglu

A thesis submitted in partial fulfillment for the degree of Doctor of Philosophy

in the

Department of Mathematical Sciences

May 2017

Declaration of Authorship

I, Arda Antikacioglu, declare that this thesis titled, 'Quantifying and Improving Sales Diversity in Recommender Systems' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

CARNEGIE MELLON UNIVERSITY

Abstract

Department of Mathematical Sciences

Doctor of Philosophy

by Arda Antikacioglu

Collaborative filtering approaches have produced some of the most accurate and personalized recommender systems to date by mining for similarities in large-scale datasets. However, despite their stellar performance in accuracy based metrics, researchers have demonstrated a propensity by such algorithms to exaggerate the biases inherent in the data such as popularity or the affinity of users to certain kinds of content. Meanwhile, recommender systems have only grown in importance and have become an integral part of the internet ecosystem, with many users interacting with many recommender systems daily on e-commerce sites, social networks and apps. Therefore, the biases in recommender systems have come to critically impact a company's bottom line, user satisfaction levels and public image, making it an imperative to develop recommendation diversification methods to explicitly counteract them.

In this thesis we make three key contributions to the growing field of sales diversity, which aims to reduce popularity biases inherent in many collaborative filtering based recommender systems. First, we consider the problem of making item-item recommendations, with the goal of redundantly linking from popular items to less popular items in order to bring them more exposure on the web. Next, we consider to the setting of user-item recommendations, and develop a metric we call "discrepancy" to measure the distance between the recommendation distribution desired by a business and the distribution obtained by the recommender system, and develop algorithms to reduce discrepancy while maintaining high recommendation quality. Lastly, we turn our attention to item catalogs and user bases where items and users are clustered into disjoint or overlapping subgroups, and develop metrics to quantify the recommendation diversity experienced both by the users and the items.

Our approaches to all three of these problems are unified under a framework of subgraph selection, the use of network flow problems for modeling, and a focus on providing either exact polynomial algorithms or efficient approximation algorithms with concrete performance guarantees. This stands in contrast with existing approaches, most of which are reranking based heuristics for which no performance guarantees can be given. In each of these settings, we augment our theoretical findings with an empirical evaluation on real life datasets from online retailers or standard recommender system datasets provided by Netflix and the MovieLens group, and show that our methods provide superior sales diversity value when compared with competing approaches.

Acknowledgements

I would first like to thank Ravi, my advisor, without whom this thesis would not even be a possibility. Ravi has an incredibly sharp technical mind, and an excellent intuition for what makes a problem practically interesting and feasible to solve. His help has guided me throughout my graduate school career, and my gratitude for his mentorship goes beyond what I can express in this short acknowledgements section. Ravi took me on as his student in my second year and never gave up on me, always seeing the best in me and my work. It is my sincerest hope that this work makes him proud.

I also extend my thanks to Dr. Srinath Sridhar and Bloomreach Inc, for providing me with the motivation for the problem addressed in Chapter 2, and providing the datasets we used for the validation of our methods. Bloomreach and Srinath provided me with all the tools and the funding I needed to have a productive research internship, and I could not have asked for a better experience. I would also like to extend my thanks to Dr. Charalampos Tsouriakis and Prof. Alan Frieze for sharing their expertise in random graphs and guiding our research questions in Chapter 2. Finally, I would like to thank Tanvi Bajpai, who provided invaluable help in formulating proofs and collecting the data which went into the experimental validation of Chapter 4. I also want to thank the members of my PhD committee, Professors Alan Frieze and Christos Faloutsos for their suggestions in general.

On a final note, I would like to thank my family and friends. Nobody chooses their family and that is an unfair fact of life. I was lucky, profoundly lucky, to have the parents who gave birth to me. My mother, an excellent programmer and entrepreneur, and my father an academic himself. This PhD thesis lies in the intersection of both of their interests. They are kind, generous and loving people who not only instilled the love of science and math in me, but also left me wanting for nothing when it came to providing me with the opportunities I needed to excel. They are my lifeblood, and I dedicate this work to them.

Contents

List of Figures

List of Tables

Dedicated to my parents

Chapter 1

Introduction

Recommender systems have been an area of active and independent field of research since at least the middle of the 1990s, with the influence and reach of the field expanding ever since. This timeline corresponds very closely with the genesis and popularization of the World Wide Web. The vast amounts of data that the web generated was responsible for an urgent need to summarize and present catalogs of content or merchandise with which human curated recommendations could not keep up.

Initially, the answer to this problem came in the form of specialized recommender systems called content-based recommenders. These recommenders work by reducing each item in the catalog to a vector of explicitly defined features. Treating every item in a homogenous way as a collection of numerical values makes it possible to mine similarities among items. In order for this approach to be successful, the features must be chosen carefully and must be broadly applicable across the catalog, as well as easily computable given the raw data for an item. Therefore, content-based systems merely offload the difficulty of finding similar items to the task of defining and extracting features that adequately summarize the data. If the automated extraction of a feature is not feasible, then this step must be substituted with large-scale crowdsourcing tasks. Due to these difficulties, content-based systems can only be applied in very specific domains, such as music or movies. Since features used for one type of content are not easy to carry over to another type of content, the versatility of these systems are similarly limited.

With content-based systems facing the significant limitations mentioned above, researchers directed their attention to more general-purpose algorithms for building recommendation systems. Interestingly, the scale of the problem also proved to be the solution to the recommendation task: with enough data, feedback from users could be mined for patterns that could not be uncovered in smaller datasets. Explicit features describing

the items are not needed to establish similarity if we can find an abundance of users who have a simultaneous preference for two different items. This is the main idea behind the field of collaborative filtering, that has achieved tremendous success in producing recommender systems with high predictive accuracy [1–4].

Unfortunately, the collaborative filtering algorithms have their own drawbacks. The foremost among these is the tendency to exaggerate biases in the data. Recommender systems cannot make an unlimited number of recommendations due to space and time constraints. When a large interaction history is condensed into a small representative set of recommendations, some information is necessarily lost. This becomes an undesirable feature when we consider the fact that the output of recommender systems are consumed by users or other algorithms, and their feedback is often fed back into recommender as training data. This creates an undesirable feedback loop which can diminish the usefulness of a recommender, and skew its results. Filter bubbles pigeonhole users into narrow interest bands which can hurt user satisfaction, while echo chambers can hamper a business' ability to collect meaningful feedback from its users [5, 6]. These problems have only compounded in importance as more and more businesses have come to rely on recommender systems for driving both revenue and user satisfaction [7].

While significant effort has been expended by the recommendation diversification field to combat this problem and diversify the kinds of recommendation any one user sees [8–11], significantly less effort has been directed towards sales diversity measures, which measure diversity as a function of the distribution of recommendations among the items in the catalog. Sales diversity measures address the needs of a business to surface more of the items in their catalog and direct users towards particular sets of items. Improvements in sales diversity have been linked to favorable outcomes for businesses such as increased consumption [12], more efficient use of inventories [13, 14] and even increased user satisfaction levels [15, 16]. Therefore, the main purpose of this thesis is the development of optimization frameworks which can be used to maximize a variety of sales diversity measures.

1.1 The Structure of This Thesis

In Chapter 2, we consider the problem of making item-to-item recommendations. We partition the item set into two parts, one consisting of items which are adequately surfaced to search engines and another which consists of items we would like to bring into the first half. Since redundant linking is a necessary part of getting a product page into search engine indices, we seek to maximize the number of items which have

a minimum number of recommendations from items in the first half. We model this problem as a subgraph selection problem, and develop 3 different algorithms to solve it: two streaming algorithms (the greedy and sampling algorithms) and one based on perfect matchings (the partition algorithm). For sampling algorithm we provide constant factor approximation guarantees which hold in expectation. For the partition and greedy algorithms we provide approximation guarantees which obtain only an additive sublinear error term. Finally, for the greedy algorithm we prove a constant factor approximation guarantee which holds even in the worst case. We validate the effectiveness of our algorithms on both synthetic data which match our random graph models, as well as on data provided by Bloomreach Inc. from large internet retailers. This work has appeared in WWW '15 [17].

In Chapter 3, we consider the problem of making user-to-item recommendations and consider the need of a business to shape the distribution of recommendations provided by their recommender systems. We introduce a new measure of sales diversity we call "discrepancy" in order to measure the difference between the distribution obtained by a recommender system and the distribution desired by the business. By formulating the discrepancy minimization problem as a minimum cost-flow problem, we obtain polynomial time algorithms which can maximize recommendation relevance while simultaneously minimizing the discrepancy from the desired target. Our framework is stated in a very general way and contains as a special case the problem of maximizing aggregate diversity (or coverage), which is the sales diversity measure which has previously attracted the most attention by researchers. We validate the effectiveness of our discrepancy minimization framework on MovieLens and Netflix Prize datasets which are standard in recommender systems research, and show that our algorithms are better suited for optimizing not only our discrepancy metric, but other sales diversity metrics such as entropy and the Gini index when compared with other reranking based approaches.

In Chapter 4, we once again consider the problem of making user-to-item recommendations, this time in a setting where category information is available on the item set and type information is available for the user set. For this setting, we define two new diversity measures called $TUDiv$ and $TIDiv$. The former is a measure of the diversity of categories among the items presented to a user, and the latter is a measure of the diversity of user types to whom an item is shown. $TUDiv$ mimics the structure and the goals of many previous approaches which have been used to provide diversified links to users, and is similar to existing intent-aware metrics. $TIDiv$ is a novel refinement of aggregate diversity, whose goal is to promote not only coverage, but coverage across many different types of users, whose feedback would otherwise be ignored due to filter bubbles formed by unmodified collaborative filtering algorithms. We show that both of

these metrics can be optimized by minimum cost flow models when the categories and types are disjoint. We provide NP-hardness and approximation results for the case when types and categories are allowed to overlap. Once again, we validate our results using the MovieLens dataset and show that none of the existing category-aware approaches we tested perform well on improving $TIDiv$ or other sales diversity measures.

Taken together, our methods contribute new methods to a growing of field increasing sales diversity in recommender systems based on collaborative filtering. Our contributions are unified under a framework of subgraph selection. In each chapter of this work, we begin with a bipartite graph G_0 of known interactions. In Chapter 2, these known interactions are co-purchasing information between different items in the catalog. In Chapters 3 and 4, the known interactions are ratings assigned to items by users. We use G_0 as an input to standard collaborative filtering algorithms to generate a new graph G of candidate recommendations we can make. We then use explicit optimization algorithms, often based on the machinery of minimum cost flows, in order to produce a subset H of G which has higher sales diversity values than the standard recommendation approaches. In this way, we present a unified optimization based framework for sales diversity maximization, and provide a new perspective and new tools to the field of sales diversity in recommender systems.

Chapter 2

Increasing Discoverability on the Web

TABLE 2.1: Notation for Chapter 2

L	Set of popular items.
R	Set of less popular items.
$G(L, R, E)$	Bipartite graph of candidate recommendations.
H	A subset of G denoting the recommendations made by the system.
S	Number of items with at least a incoming recommendations.
d	The number of candidate recommendations per user.
c	The display constraint.
a	The degree requirement for an item in R .
k	The ratio of L to R.

2.1 Introduction

2.1.1 Web Relevance Engines

The digital discovery divide [18] refers to the problem of companies not being able to present users with what they seek in the short time they spend looking for this information. The problem is prevalent not only in e-commerce websites but also in social networks and micro-blogging sites where surfacing relevant content quickly is important for user engagement.

BloomReach is a big-data marketing company that uses the client's content as well as web-wide data to optimize both customer acquisition and satisfaction for e-retailers. BloomReach's clients include top internet retail companies from around the country. In this chapter, we describe the structure optimizer component of BloomReach's Web Relevance Engine. This component works on top of the recommendation engine so as to carefully add a set of links across pages that ensures that users can efficiently navigate the entire website.

2.1.2 Structural Optimization of Websites

An important concern of retail website owners is whether a significant fraction of the site is not recommended at all (or 'hardly' recommended) from other more popular pages within their site. One way to address this problem is to try to ensure that every page will obtain at least a baseline number of links from popular pages so that great content does not remain undiscovered, and thus bridge the discovery divide mentioned above. If the website remains connected, this also ensures a simple conductance for the underlying link graph.

We use this criterion of discoverability as the objective for the choice of the links to recommend. We start with a small set of already discovered or popular nodes available at a site, and want to use this set to make as many new nodes discoverable as possible. This objective leads to a new structural formulation of the recommendation selection problem. In particular, we think of commonly visited pages in a site as the already discovered pages, from which there are a large number of possible recommendations available (using more traditional information retrieval methods) to related but less visited peripheral pages. The problem of choosing a limited number of pages to recommend at each discovered page can be cast with the objective of maximizing the number of peripheral non-visited pages that are redundantly linked. We formulate this as a recommendation subgraph problem, and study practical algorithms for solving these problems at scale with real-life data.

2.1.3 Recommendation Systems as a Subgraph Selection Problem

Formally, we divide all pages in a site into two groups: the discovered pages and the undiscovered ones. Furthermore, we assume that traditional recommendation systems [9, 19, 20] provide us with a large set of related candidate undiscovered page recommendations for each discovered page using relevance metrics. In this work, we assume d such related candidates are available per page creating a candidate recommendation

bipartite graph (with degree d at each discovered page node). Our goal is to analyze how to prune this set to $c < d$ recommendations such that globally we ensure that the number of undiscovered pages that have at least $a \geq 1$ recommendations to them in the chosen subgraph is maximized. This gives the (c, a)-recommendation subgraph introduced in Section 2.3.1. Even though the case of $a = 1$ reduces to a polynomially solvable version of a matching problem, the more usual cases of $a > 1$ are most likely NP-hard prohibiting exact solution methods at scale. Even the simple versions that reduce to matching are too computational expensive on memory and processing to run on real-life instances

2.1.4 Our Contributions

We introduce three simple heuristic methods that can be implemented in linear or near-linear time and thoroughly investigate their theoretical performance. In particular, we delineate when each method will work effectively on popular random graph models, and when a practitioner will need to employ a more sophisticated algorithm. We then evaluate how these simple methods perform on simulated data, both in terms of solution quality and running time. Finally, we show the deployment of these methods on BloomReach's real-world client link graph and measure their actual performance in terms of running-times, memory usage and accuracy. It is worthwhile to note that the simplest of the three methods that we propose (sampling) can be easily adapted to the incremental dynamic setting when the set of pages and candidate recommendations is changing rapidly.

To summarize, our contributions are as follows.

1. The development of a new structural model for recommendation systems as a subgraph selection problem for maximizing discoverability (Section 2.3).

2. The proposal of three methods (sampling, greedy and partition) with increasing sophistication to solve the problem at scale along with associated theoretical performance guarantee analyses (Section 2.4). In particular, we show very strong theoretical bounds on the size of the discoverable set for the sampling algorithm in the fixed degree random graph model (Theorem 2.1); in the Erdös-Renyi model for the greedy algorithm (Theorem 2.7) and for a partition-based algorithm (Theorem 2.10).

3. An empirical validation of our conclusions with simulated and real-life data (Section 2.6). Our simulations show that sampling is the least resource intensive and

performs satisfactorily, while partition is the most resource intensive but performs better for small values of discoverability threshold a; Greedy is the overall best-performer using a single pass over the data and producing good results over a variety of parameters. In the tests with real retailer data, we see these trends broadly reflected in the results: Greedy performs well when c gets moderately large giving almost optimal starting from $a = 2$. The partition method is promising when the targeted a value is low. Sampling is typically worse than greedy, but unlike the partition algorithm, its performance improves dramatically as c becomes larger, and does not worsen as quickly when a gets larger.

2.2 Related Work

Recommendation systems have been studied extensively in the literature, and can be broadly separated into two different streams: collaborative filtering systems and content-based recommender systems [21]. Much attention has been focused on the former approach, where either users are clustered by considering the items they have consumed or items are clustered by considering the users that have bought them. Both item-to-item and user-to-user recommendation systems based on collaborative filtering have been adopted by many industry giants such as Twitter [22], Amazon [23] and Google [4] and Netflix [24].

Content based systems instead look at each item and its intrinsic properties. For example, Pandora has categorical information such as Artist, Genre, Year, Singer, Tempo etc. on each song it indexes. Similarly, Netflix has a lot of categorical data on movies and TV such as Cast, Director, Producers, Release Date, Budget, etc. This categorical data can then be used to recommend new songs that are similar to the songs that a user has liked before. Depending on user feedback, a recommender system can learn which of the categories are more or less important to a user and adjust its recommendations.

A drawback of the first type of system is that is that they require multiple visits by many users so that a taste profile for each user, or a user profile for each item can be built. Similarly, content-based systems also require significant user participation to train the underlying system. These conditions are possible to meet for large commerce or entertainment hubs, but not very likely for most online retailers that specialize in a just a few areas, but have a long-tail [25] of product offerings.

Because of this constraint, in this chapter we focus on a recommender system that typically uses many different algorithms that extract categorical data from item descriptions

and uses this data to establish weak links between items (candidate recommendations). In the absence of other data that would enable us to choose among these many links, we consider every potential recommendation to be of equal value and focus on the objective of discovery, which has not been studied before. Using heuristics for building this graph is not only practical, but is theoretically sound as well[26]. In this way, our work differs from all the previous work on recommendation systems that emphasize on finding recommendations of high relevance and quality rather than on structural navigability of the realized link structure. However, some of our approaches can be extended to the more general case where different recommendations have different weights (See Theorem 2.5).

On the graph algorithms side, our problem is related to the bipartite matching and more generally, the maximum b-matching problems. There has been considerable work done in this area. In particular, both the weighted matching and b-matching problems have exact polynomial time solutions [27]. Furthermore the matching problem admits a near linear time $(1-\epsilon)$-approximation algorithm [28], while the weighted b-matching problem admits a $1/2$-approximation algorithm [29]. However, all such algorithms are based on combinatorial properties of matchings and b-matchings, and do not carry over to the more important version of our problem when $a > 1$.

Finally, our problem bears resemblance to some covering problems. For example, the maximum coverage problem asks for the maximum number of elements that can be covered by a fixed number of sets and has a greedy $(1 - 1/e)$-approximation [30]. However, as mentioned earlier, our formulation requires multiple coverage of elements. Furthermore note that the collection of sets that can be used in the redundant coverage are all possible subsets of c out of the d candidate links, and is expressed implicitly in our problem. The currently known theoretical methods for maximum coverage heavily rely on the submodularity of the objective function, which our objective doesn't satisfy. Hence the line of recent work on approximation algorithms for submodular maximization does not apply to our problems.

2.3 Our Model

We model the structure optimization of recommendations by using a bipartite digraph, where one partition L represents the set of discovered (i.e., often visited) items for which we are required to suggest recommendations and the other partition R representing the set of undiscovered (not visited) items that can be potentially recommended. If needed, the same item can be represented in both L and R.

Input: A bipartite graph $G(L, R, E)$, a display constraint c, a target degree a.
Output: A subgraph $H \subseteq G$, with maximum degree c at any node in L, which maximizes the number of vertices in R which have degree at least a.

FIGURE 2.1: The definition of the (c, a)-recommendation subgraph problem

2.3.1 The Recommendation Subgraph Problem

We introduce and study this as the **the (c, a)-recommendation subgraph problem** in this chapter :

Note that if $a = c = 1$ this is simply the maximum bipartite matching problem [31]. If $a = 1$ and $c > 1$, we obtain a b-matching problem, that can be converted to a bipartite matching problem [27]. The typical and interesting cases when $a > 1$ is most likely NP-hard, ruling out the possibility of efficient exact algorithms.

We now describe typical web graph characteristics by discussing the sizes of L, R, c and a in practice. As noted before, in most websites, a small number of 'head' pages contribute to a significant amount of the traffic while a long tail of the remaining pages contribute to the rest [32–34]. This is supported by our own experience with the 80/20 rule, i.e. 80% of a site's traffic is captured by 20% of the pages. Therefore, the ratio $k = |L|/|R|$ is typically between 1/3 to 1/5, but may be even lower.

From our own work at BloomReach (and by observing recommendations of Quora, Amazon, and YouTube), typical values for c range from 3 to 20 recommendations per page. Values of a are harder to nail down but it typically ranges from 1 to 5.

2.3.2 Practical Requirements

There are two key requirements in making graph algorithms practical. The first is that the method used must be very simple to implement, debug, deploy and most importantly maintain long-term. The second is that the method must scale gracefully with larger sizes.

Graph matching algorithms require linear memory and super-linear run-time which does not scale well. For example, an e-commerce website of a client of BloomReach with 1M product pages and 100 recommendation candidates per product would require easily over 160GB in main memory to store the graph and run exact matching algorithms; this can be reduced by using graph compression techniques but that adds more technical difficulties in development and maintenance. Algorithms that are time intensive can sometimes be sped-up by using distributed computing techniques such as map-reduce [35].

However, efficient map-reduce algorithms for graph problems are notoriously difficult. Finally, all of these methods apply only to the special case of our problem when $a = 1$, leaving open the question of solving the more interesting and typical cases of redundant coverage when $a > 1$.

2.3.3 Simple Approximation Algorithms

To satisfy these practical requirements, we propose the study of three simple approximate solutions strategies that not only can be shown to scale well in practice but also have good theoretical properties that we demonstrate using approximation ratios.

- **Sampling:** The first solution is a simple random sampling solution that selects a random subset of c links out of the available d from every page. Note that this solution requires no memory overhead to store these results a-priori and the recommendations can be generated using a random number generator on the fly. While this might seem trivial at first, for sufficient (and often real-world) values of c and a we show that this can be optimal. Also, this method is very easy to adapt to the case when the underlying graph is dynamic with both nodes and edges changing over time. Furthermore, our approach can be extended to the case where the recommendation edges have weights representing varying strengths of association as is typically provided by the traditional methods that generate candidate recommendation links.

- **Greedy:** The second solution we propose is a greedy algorithm that chooses the recommendation links so as to maximize the number of nodes in R that can accumulate a in-links. In particular, we keep track of the number of in-links required for each node in R to reach the target of a and choose the links from each node in L giving preference to adding links to nodes in R that are closer to the target in-degree a. This method bears close resemblance in strategy with greedy methods used for maximum coverage and its more general submodular maximization variants.

- **Partition:** The third solution is inspired by a theoretically rigorous method to find optimal subgraphs in sufficiently dense graphs: it partitions the edges into a subsets by random sub-sampling, such that there is a good chance of finding a perfect matching from L to R in each of the subsets. The union of the matchings so found will thus result in most nodes in R achieving the target degree a. We require the number of edges in the underlying graph to be significantly large for this method to work very well; moreover, we need to run a (near-)perfect matching

	Sampling	Greedy	Partition
Time	$O(\|E\|)$	$O(\|E\|)$	$O(\|E\|\sqrt{\|V\|})$
Working Space	$O(1)$	$O(V)$	$O(\|E\|)$

FIGURE 2.2: Complexities of the different algorithms (assuming constant a and c)

algorithm in each of the edge-subsets which is also a computationally expensive subroutine. Hence, even though this method works very well in dense graphs, its resource requirements may not scale well in terms of running time and space.

As a summary, the Figure 2.1 below shows the time and space complexity of our different algorithms.

In the next section, we elaborate on these methods, their running times, implementation details, and theoretical performance guarantees. In the section after that, we present our comprehensive empirical evaluations of all three methods, first the results on simulated data and then the results on real data from some clients of BloomReach.

2.4 Algorithms for Recommendation Subgraphs

2.4.1 The Sampling Algorithm

We present the sampling algorithm for the (c, a)-recommendation subgraph formally below.

Data: A bipartite graph $G = (L, R, E)$
Result: A (c, a)-recommendation subgraph H
for u *in* L **do**
  $S \leftarrow$ a random sample of c vertices without replacement in $N(u)$;
  for v *in* S **do**
    $H \leftarrow H \cup \{(u, v)\}$;
  end
end
return H;

Algorithm 1: The sampling algorithm

Given a bipartite graph G, the algorithm has runtime complexity of $O(|E|)$ since every edge is considered at most once. The space complexity can be taken to be $O(1)$, since the adjacency representation of G can be assumed to be pre-sorted by the endpoint of each edge in L.

We next introduce a simple random graph model for the supergraph from which we are allowed to choose recommendations and present a bound on its expected performance when the underlying supergraph $G = (L, R, E)$ is chosen probabilistically according to this model.

Fixed Degree Model: In this model for generating the candidate recommendation graph, each vertex $u \in L$ uniformly and independently samples d neighbors from R with replacement. While this allows each vertex in L to have the same vertex as a neighbor multiple times, in reality $r \gg d$ is so edge repetition is very unlikely. This model is similar to, but is distinct from the more commonly known Erdös-Renyi model of random graphs [36]. In particular, while the degree of each vertex in L is fixed under our model, concentration bounds can show that the degrees of the vertices in L would have similarly been concentrated around d for $p = d/r$ in the Erdös-Renyi model. We prove the following theorem about the performance of the Sampling Algorithm. We denote the ratio of the size of L and R by k, i.e., we define $k = \frac{l}{r}$.

Theorem 2.1. *Let S be the random variable denoting the number of vertices $v \in R$ such that $\deg_H(v) \geq a$ in the fixed-degree model. Then*

$$\mathrm{E}[S] \geq r\left(1 - e^{-ck + \frac{a-1}{r}} \frac{(ck)^a - 1}{ck - 1}\right)$$

To get a quick sense of the very good performance bounds reflected in this theorem, please see Figure 2.3 that plots the approximation ratio as a function of ck for the $(c, 1)$-recommendation subgraph problem, as well as Figure 2.4 that shows how large c needs to be (in terms of k) for the solution to be 95% optimal for different values of a, both in the fixed degree model.

Proof. We will analyze the sampling algorithm as if it picks the neighbors of each $u \in L$ with replacement, the same way the fixed-degree model generates G. This variant would obviously waste some edges, and perform worse than the variant which samples neighbors without replacement. This means that any performance guarantee we prove for this variant holds for our original statement of the algorithm as well.

To prove the claim let X_v be the random variable that represents the degree of the vertex $v \in R$ in our chosen subgraph H. Because our algorithm uniformly subsamples a uniformly random selection of edges, we can assume that H was generated the same way as G but sampled c instead of d edges for each vertex $u \in L$. Since there are cl edges in H that can be incident on v, and each of these edges has a $1/r$ probability of being incident on a given vertex in L, we can now calculate that

$$\begin{aligned}\Pr[X_v = i] &= \binom{cl}{i}(1-\frac{1}{r})^{cl-i}\left(\frac{1}{r}\right)^i \\ &\leq (cl)^i(1-\frac{1}{r})^{cl-i}\left(\frac{1}{r}\right)^i\end{aligned}$$

Using a union bound, we can combine these inequalities to upper bound the probability that $\deg_H(v) < a$.

$$\begin{aligned}\Pr[X_v < a] &= \sum_{i=0}^{a-1}\binom{cl}{i}\left(1-\frac{1}{r}\right)^{cl-i}\left(\frac{1}{r}\right)^i \\ &\leq \sum_{i=0}^{a-1}\left(\frac{cl}{r}\right)^i\left(1-\frac{1}{r}\right)^{cl-i} \\ &\leq \left(1-\frac{1}{r}\right)^{cl-(a-1)}\sum_{i=0}^{a-1}(ck)^i \\ &\leq \left(1-\frac{1}{r}\right)^{cl-(a-1)}\frac{(ck)^a-1}{ck-1} \\ &\leq e^{-ck+\frac{a-1}{r}}\frac{(ck)^a-1}{ck-1}\end{aligned}$$

Letting $Y_v = [X_v \geq a]$, we now see that

$$\mathrm{E}[S] = \mathrm{E}\left[\sum_{v\in R} Y_v\right] \geq r\left(1 - e^{-ck+\frac{a-1}{r}}\frac{(ck)^a-1}{ck-1}\right)$$

□

We can combine this lower bound with a trivial upper bound to obtain an approximation ratio that holds in expectation.

Theorem 2.2. *The above sampling algorithm gives a* $\left(1-\frac{1}{e}\right)$*-factor approximation to the* $(c,1)$*-graph recommendation problem in expectation.*

Proof. The size of the optimal solution is bounded above by both the number of edges in the graph and the number of vertices in R. The former of these is $cl = ckr$ and the latter is r, which shows that the optimal solution size $OPT \leq r\min(ck,1)$. Therefore, by simple case analysis the approximation ratio in expectation is at least $(1-\exp(-ck))/\min(ck,1) \geq 1-\frac{1}{e}$ □

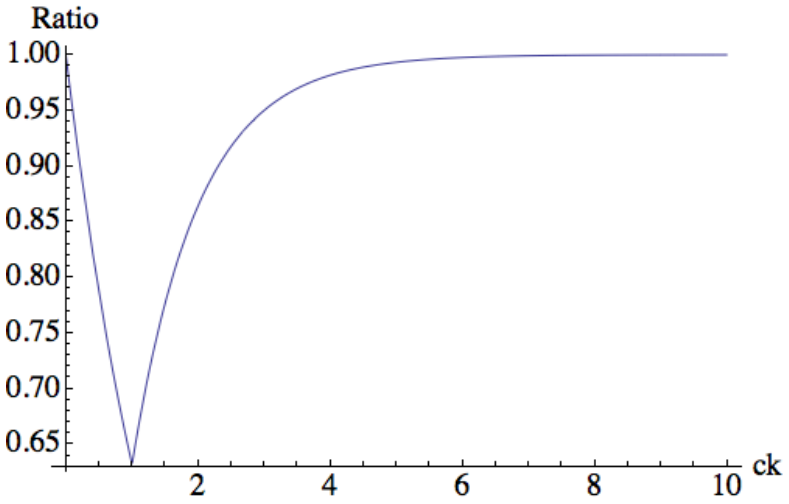


FIGURE 2.3: Approx ratio as a function of ck

a	1	2	3	4	5
c	$3.00k^{-1}$	$4.74k^{-1}$	$7.05k^{-1}$	$10.01k^{-1}$	$13.48k^{-1}$

FIGURE 2.4: The required ck to obtain 95% optimality for (c, a)-recommendation subgraph

For the $(c, 1)$-recommendation subgraph problem the approximation obtained by this sampling approach can be much better for certain values of ck. In particular, if $ck > 1$, then the approximation ratio is $1 - \exp(-ck)$, which approaches 1 as $ck \to \infty$. When $ck = 3$, then the solution will be at least 95% as good as the optimal solution even with our trivial bounds. Similarly, when $ck < 1$, the approximation ratio is $(1-\exp(-ck))/ck$ which also approaches 1 as $ck \to 0$. In particular, if $ck = 0.1$ then the solution will be at 95% as good as the optimal solution. The case when $ck = 1$ represents the worst case outcome for this model where we only guarantee 63% optimality. Figure 2.3 shows the approximation ratio as a function of ck for the $(c, 1)$-recommendation subgraph problem in the fixed degree model.

For the general (c, a)-recommendation subgraph problem, if $ck > a$, then the problem is easy on average. This is in comparison to the trivial estimate of cl. For a fixed a, a random solution gets better as ck increases because the decrease in e^{-ck} more than compensates for the polynomial in ck next to it. However, in the more realistic case, the undiscovered pages in R too numerous to be all covered even if we used the full set of budgeted links allowed out of L, i.e. $cl < ra$ or rearranging, $ck < a$; in this case, we need to use the trivial estimate of ckr/a, and the analysis for $a = 1$ does not extend here. For practical purposes, the table in Figure 2.4 shows how large c needs to be (in terms of k) for the solution to be 95% optimal for different values of a, again in the fixed degree model.

We close out this section by showing that the main result that holds in expectation also hold with high probability for $a = 1$, using the following variant of Chernoff bounds.

Theorem 2.3. *[37] Let $X_1, \ldots, X_n$ be non-positively correlated variables. If $X = \sum_{i=1}^{n} X_i$, then for any $\delta \geq 0$*

$$\Pr[X \geq (1+\delta)\mathrm{E}[X]] \leq \left(\frac{e^{\delta}}{(1+\delta)^{1+\delta}}\right)^{E[X]}$$

Using this we can convert our expectation result to one that holds with high probability.

Theorem 2.4. *Let S be the random variable denoting the number of vertices $v \in R$ such that $\deg_H(v) \geq 1$. Then $S \leq r(1-2\exp(-ck))$ with probability at most $(e/4)^{r(1-\exp(-ck))}$.*

Proof. We can write S as $\sum_{v \in R}(1 - X_v)$ where X_v is the indicator variable that denotes that X_v is matched. Note that the variables $1 - X_v$ for each $v \in R$ are non-positively correlated. In particular, if $N(v)$ and $N(v')$ are disjoint, then $1 - X_v$ and $1 - X_{v'}$ are independent. Otherwise, v not claiming any edges can only increase the probability that v' has an edge from any vertex $u \in N(v) \cap N(v')$. Also note that the expected size of S is $r(1 - \exp(-ck))$ by Theorem 2.1. Therefore, we can apply Theorem 2.3 with $\delta = 1$ to obtain the result. □

For realistic scenarios where r is very large, the above theorem gives very tight bounds on the size of the solution, also explaining the effectiveness of the simple sampling algorithm in such instances.

The results presented in this section can be naturally extended to weighted models as shown by the theorem below. The proof is left out due to space constraints.

Theorem 2.5. *Let $G = K_{l,r}$ be a complete bipartite graph where the edges have i.i.d. weights and come from a distribution with mean μ that is supported on $[0, b]$; Assume that $ck\mu \geq 1 + \epsilon$ for some $\epsilon > 0$. If the algorithm from Section 2.4.1 is used to sample a subgraph H from G and S is the set of vertices in R of incident weight at least one, then*

$$\mathrm{E}[S] = \sum_{v \in R} \mathrm{E}[X_v] = r\left(1 - \exp\left(-\frac{2l\epsilon^2}{b^2}\right)\right)$$

2.4.2 The Greedy Algorithm

The results in the previous section concentrated on producing nearly optimal solutions in expectation. In this section, we will show that it is possible to obtain good solutions regardless of the model that generated the recommendation subgraph.

We next analyze the natural greedy algorithm for constructing a (c, a)-recommendation subgraph H iteratively. In the following algorithm, we use $N(u)$ to refer to the neighbors of a vertex u.

```
Data: A bipartite graph G = (L, R, E)
Result: A (c, a)-recommendation subgraph H
for u in L do
    d[u] ← 0
end
for v in R do
    F ← {u ∈ N(v) | d[u] < c};
    if |F| ≥ a then
        restrict F to a elements;
        for u in F do
            H ← H ∪ {(u, v)};
            d[u] ← d[u] + 1;
        end
    end
end
return H;
```

Algorithm 2: The greedy Algorithm

The algorithm loops through each vertex in R, and considers each edge once. Therefore, the runtime is $\Theta(|E|)$. Furthermore, the only data structure we use is an array which keeps track of $\deg_H(u)$ for each $u \in L$, so the memory consumption is $\Theta(|L|)$. Finally, we prove the following tight approximation property of this algorithm.

Theorem 2.6. *The greedy algorithm gives a* $1/(a+1)$*-approximation to the* (c, a)*-graph recommendation problem.*

Proof. Let $R_{GREEDY}, R_{OPT} \subseteq R$ be the set of vertices that have degree $\geq a$ in the greedy and optimal solutions respectively. Note that any $v \in R_{OPT}$ along with neighbors $\{u_1, \ldots u_a\}$ forms a set of candidate edges that can be used by the greedy algorithm. So we can consider R_{OPT} as a candidate pool for R_{GREEDY}. Each selection of the greedy algorithm might result in some candidates becoming infeasible, but it can continue as long as the candidate pool is not depleted. Each time the greedy algorithm selects some vertex $v \in R$ with edges to $\{u_1, \ldots, u_a\}$, we remove v from the candidate pool. Furthermore each u_i could have degree c in the optimal solution and used each of its edges to make a neighbor attain degree a. The greedy choice of an edge to u_i requires us to remove such an edge to an arbitrary vertex $v_i \in R$ adjacent to u_i in the optimal solution, and thus remove v_i from further consideration in the candidate pool. In other words, by using an edge of u_i, we force it to not use an edge it used to some other v_i, which might cause the degree of v_i to go below a. Therefore, at each step of the greedy

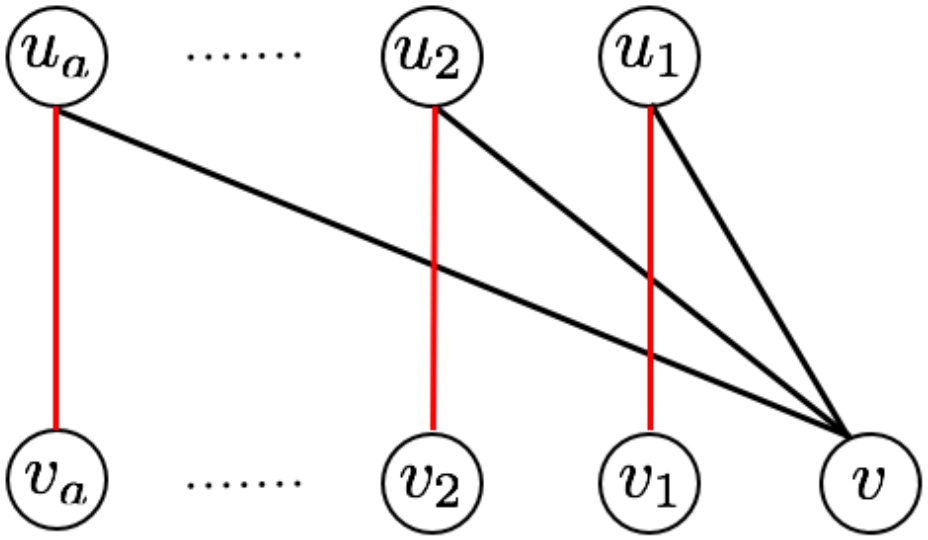


FIGURE 2.5: One step of the greedy algorithm. When v selects edges to $u_1, \ldots, u_a$, it can remove $v_1, \ldots, v_a$ from the pool of candidates that are available. The potentially invalidated edges are shown in red.

algorithm, we may remove at most $a+1$ vertices from the candidate pool as illustrated in Figure 4. Since our candidate pool has size OPT, the greedy algorithm can not stop before it has added $OPT/(a+1)$ vertices to the solution. □

This approximation guarantee is as good as we can expect, since for $a = 1$ we recover the familiar 1/2-approximation of the greedy algorithm for matchings. Furthermore, even in the case of matchings ($a = 1$), randomizing the order in which the vertices are processed is still known to leave a constant factor gap in the quality of the solution [38]. Despite this result, the greedy algorithm fares much better when we analyze its expected performance. Switching to the **Erdös-Renyi model** [39] instead of the fixed degree model used in the previous section, we now prove the near optimality of the greedy algorithm for the (c, a)-recommendation subgraph problem. Recall that in this model (sometimes referred to as $G_{n,p}$), each possible edge is inserted with probability p independent of other edges. In our version $G_{l,r,p}$, we only add edges from L to R each with probability p independent of other edges in this complete bipartite candidate graph. For technical reasons, we need to assume that $lp \geq 1$ in the following theorem. However, this is a very weak assumption since lp is simply the expected degree of a vertex $v \in R$. Typical values for p for our applications will be $\Omega(\log(l)/l)$ making the expected degree $lp = \Omega(\log l)$.

Theorem 2.7. *Let $G = (L, R, E)$ be a graph drawn from the $G_{l,r,p}$ where $lp \geq 1$. If S is the size of the (c, a)-recommendation subgraph produced by the greedy algorithm, then:*

$$\mathrm{E}[S] \geq r - \frac{a(lp)^{a-1}}{(1-p)^a} \sum_{i=0}^{r-1} (1-p)^{l-\frac{ia}{c}}$$

When the underlying random graph is sufficiently dense, Theorem 2.8 shows that the above guarantee is asymptotically optimal.

Proof. Note that if edges are generated uniformly, we can consider the graph as being revealed to us one vertex at a time as the greedy algorithm runs. In particular, consider the event X_{i+1} that the greedy algorithm matches the $(i+1)^{st}$ vertex it inspects. While, X_{i+1} is dependent on $X_1, \ldots, X_i$, the worst condition for X_{i+1} is when all the previous i vertices were from the same vertices in L, which are now not available for matching the $(i+1)^{st}$ vertex. The maximum number of such invalidated vertices is at most $\lceil ia/c \rceil$. Therefore, the bad event is that we have fewer than a of the at least $l - \lceil ia/c \rceil$ available vertices having an edge to this vertex. The probability of this bad event is at most $\Pr[Y \sim Bin(l - \frac{ia}{c}, p) : Y < a]$, the probability that a Binomial random variable with $l - \frac{ia}{c}$ trials of probability p of success for each trial has less than a successes. We can bound this probability by using a union bound and upper-bounding $\Pr[Y \sim Bin(l - \frac{ia}{c}, p) : Y = t]$ for each $0 \leq t \leq a-1$. By using the trivial estimate that $\binom{n}{i} \leq n^i$ for all n and i, we obtain:

$$\begin{aligned}
\Pr[Y \sim Bin(l - \frac{ia}{c}, p) : Y = t] &= \binom{l - \frac{ia}{c}}{t} (1-p)^{l-\frac{ia}{c}-t} p^t \\
&\leq \left(l - \frac{ia}{c}\right)^t (1-p)^{l-\frac{ia}{c}-t} p^t \\
&\leq (lp)^t (1-p)^{l-\frac{ia}{c}-t}
\end{aligned}$$

Notice that the largest exponent lp can take within the bounds of our sum is $a-1$. Similarly, the smallest exponent $(1-p)$ can take within the bounds of our sum is $l - \frac{ia}{c} - a + 1$. Now applying the union bound gives:

$$\begin{aligned}
&\Pr[Y \sim Bin(l - \frac{ia}{c}, p) : Y < a] \\
&\leq \sum_{t=0}^{a-1} \Pr[Y \sim Bin(l - \frac{ia}{c}, p) : Y = t] \\
&\leq \sum_{t=0}^{a-1} (lp)^t (1-p)^{l - \frac{ia}{c} - t} \\
&= a(lp)^{a-1}(1-p)^{l - \frac{ia}{c} - a + 1}
\end{aligned}$$

Finally, summing over all the X_i using the linearity of expectation and this upper bound, we obtain

$$\begin{aligned}
\mathrm{E}[S] &\geq r - \sum_{i=0}^{r-1} \mathrm{E}[\neg X_i] \\
&\geq r - \sum_{i=0}^{r-1} \Pr[Y \sim Bin(l - \frac{ia}{c}, p) : Y < a] \\
&\geq r - a(lp)^{a-1} \sum_{i=0}^{r-1} (1-p)^{l - \frac{ia}{c} - a + 1}
\end{aligned}$$

□

Asymptotically, this result explains why the greedy algorithm does much better in expectation than $1/(a+1)$ guarantee we can prove in the worst case. In particular, for a reasonable setting of the right parameters, we can prove that the error term of our greedy approximation will be sublinear.

Theorem 2.8. *Let $G = (L, R, E)$ be a graph drawn from the $G_{l,r,p}$ where $p = \frac{\gamma \log l}{l}$ for some $\gamma \geq 1$. Suppose that c, a and $\epsilon > 0$ are such that $lc = (1+\epsilon)ra$ and that l and r go to infinity while satisfying this relation. If S is the size of the (c, a)-recommendation subgraph produced by the greedy algorithm, then*

$$E[S] \geq r - o(r)$$

Proof. We will prove this claim by applying Theorem 2.7. Note that it suffices to prove that $(lp)^{a-1} \sum_{i=0}^{r-1} (1-p)^{l - \frac{ia}{c}} = o(r)$ since the other terms are just constants. We first bound the elements of this summation. Using the facts that $p = \frac{\gamma \log l}{l}$, $lc/a = (1+\epsilon)r$ and that $i < r$ throughout the summation, we get the following bound on each term:

$$
\begin{aligned}
(1-p)^{l-\frac{ia}{c}} &\leq \left(1-\frac{\gamma \log l}{l}\right)^{l-\frac{ia}{c}} \\
&\leq \exp\left(-\frac{\gamma \log l}{l}\left(l-\frac{ia}{c}\right)\right) \\
&= \exp\left((-\log l)\left(\gamma-\frac{ia}{lc}\right)\right) \\
&= l^{-\gamma+\frac{ia}{lc}} = l^{-\gamma+\frac{i}{(1+\epsilon)r}} \\
&\leq l^{-1+\frac{1}{1+\epsilon}} = l^{-\frac{\epsilon}{1+\epsilon}}
\end{aligned}
$$

Finally, we can evaluate the whole sum:

$$
\begin{aligned}
(lp)^{a-1}\sum_{i=0}^{r-1}(1-p)^{l-\frac{ia}{c}} &\leq \left(\log^{a-1} l\right)\sum_{i=0}^{r-1} l^{-\frac{\epsilon}{1+\epsilon}} \\
&\leq \left(\log^{a-1} l\right) r l^{-\frac{\epsilon}{1+\epsilon}} \\
&= \left(\log^{a-1} l\right)\frac{c}{(1+\epsilon)a} l^{1-\frac{\epsilon}{1+\epsilon}} = o(l)
\end{aligned}
$$

However, since r is a constant times l, any function that is $o(l)$ is also $o(r)$ and this proves the claim.

□

2.4.3 The Partition Algorithm

To motivate the partition algorithm, we first define optimal solutions for the recommendation subgraph problem.

Perfect Recommendation Subgraphs: We define a *perfect* (c, a)-recommendation subgraph on G to be a subgraph H such that $deg_H(u) \leq c$ for all $u \in L$ and $deg_H(v) = a$ for $\min(r, \lfloor cl/a \rfloor)$ of the vertices in R.

The reason we define perfect (c, a)-recommendation subgraphs is that when one exists, it's possible to recover it in polynomial time using a min-cost b-matching algorithm (matchings with a specified degree b on each vertex) for any setting of a and c. However, implementations of b-matching algorithms often incur significant overheads even

over regular bipartite matchings. This motivates a solution that uses regular bipartite matching algorithms to find an approximately optimal solution given that a perfect one exists.

We do this by proving a sufficient condition for perfect (c, a)-recommendation subgraphs to exist with high probability in a bipartite graph G under the **Erdös-Renyi model** [39] where edges are sampled uniformly and independently with probability p. This argument then guides our formulation of a heuristic that overlays matchings carefully to obtain (c, a)-recommendation subgraphs.

Theorem 2.9. *[36] Let G be a bipartite graph drawn from $G_{n,n,p}$. If $p \geq \frac{\log n + \log\log n}{n}$, then as $n \to \infty$, the probability that G has a perfect matching approaches 1.*

We will prove that a perfect (c, a)-recommendation subgraph exists in random graphs with high probability by building it up from a matchings each of which must exist with high probability if p is sufficiently high. To find these matchings, we identify subsets of size l in R that we can perfectly match to L. These subsets overlap, and we choose them so that each vertex in R is in a subsets. While the theorem is stated for the case when $a \leq c$, it applies equally well to the $a > c$ case by partitioning L instead of R in the following proof.

Theorem 2.10. *Let G be a random bipartite graph drawn from $G_{l,r,p}$ with $p \geq a\frac{\log l + \log\log l}{l}$ then the probability that G has a perfect (c, a)-recommendation subgraph tends to 1 as $l, r \to \infty$.*

This theorem guarantees the existence of an optimal recommendation subgraph in sufficiently dense subgraphs, and provides a constructive proof of this fact that is also the basis of our partition algorithm.

Proof. We start by either padding or restricting R to a set of $\frac{lc}{a}$ before we start our analysis. If $r \geq \frac{lc}{a}$, then we restrict R to an arbitrary subset R' of size $\frac{lc}{a}$. Since induced subgraphs of Erdös-Renyi graphs are also Erdös-Renyi graphs, we can instead apply our analysis to the induced subgraph. Since the optimal solution has size bounded above by $\frac{lc}{a}$ a perfect (c, a)-recommendation subgraph in $G[L, R']$ will imply a perfect recommendation subgraph in $G[L, R]$.

On the other hand, if $r \leq \frac{lc}{a}$, then we can pad R with $\frac{lc}{a} - r$ dummy vertices and adding an edge from each such vertex to each vertex in L with probability p. We call the resulting right side of the graph R'. Note that $G[L, R']$ is still generated by the Erdös-Renyi process. Further, since the original graph $G[L, R]$ is a subgraph of this new

graph, if we prove the existence of a perfect (c, a)-recommendation subgraph in this new graph, it will imply the existence of a perfect recommendation subgraph in $G[L, R]$.

Having picked an R' satisfying $|R'| = \frac{lc}{a}$, we pick an enumeration of the vertices in $R' = \{v_0, \ldots, v_{lc/a-1}\}$ and add each of these vertices into a subsets as follows. Define $R_i = \{v_{(i-1)l/a}, \ldots, v_{(i-1)l/a+l-1}\}$ for each $1 \leq i \leq c$ where the arithmetic in the indices is done modulo lc/a. Note both L and all of the R_i's have size l.

Using these new sets we define the graphs G_i on the bipartitions (L, R_i). Since the sets R_i are intersecting, we cannot define the graphs G_i to be induced subgraphs. However, note that each vertex $v \in R'$ falls into exactly a of these subsets.

Therefore, we can uniformly randomly assign each edge in G to one of a graphs among $\{G_1, \ldots, G_c\}$ it can fall into, and make each of those graphs a random graph. In fact, while the different G_i are coupled, taken in isolation we can consider any single G_i to be drawn from the distribution $G_{l,l,p/a}$ since G was drawn from $G_{l,r,p}$. Since $p/a \geq (\log l + \log\log l)/l$ by assumption, we conclude by Theorem 2.9, the probability that a particular G_i has no perfect matching is $o(1)$.

If we fix c, we can conclude by a union bound that except for a $o(1)$ probability, each one of the G_i's has a perfect matching. By superimposing all of these perfect matchings, we can see that every vertex in R' has degree a. Since each vertex in L is in exactly c matchings, each vertex in L has degree c. It follows that except for a $o(1)$ probability there exists a (c, a)-recommendation subgraph in G. □

Approximation Algorithm Using Perfect Matchings: The above result now enables us to design a near linear time algorithm with a $(1 - \epsilon)$ approximation guarantee to the (c, a)-recommendation subgraph problem by leveraging combinatorial properties of matchings. In particular, we use the fact a matching that does not have augmenting paths of length $> 2\alpha$ is a $1 - 1/\alpha$ approximation to the maximum matching problem. We call this method the Partition Algorithm, and we outline it below.

Theorem 2.11. *Let G be a bipartite random graph drawn from $G_{l,r,p}$ where $p \geq a\frac{\log l + \log\log l}{l}$. Then Algorithm 3 finds a $(1-\epsilon)$-approximation in $O(\frac{|E|}{\epsilon})$ time with probability $1 - o(1)$.*

Proof. Using the previous theorem, we know that each of the graphs G_i has a perfect matching with high probability. These perfect matchings can be approximated to a $1 - \epsilon/c$ factor by finding matchings that do not have augmenting paths of length $\geq 2c/\epsilon$ [31]. This can be done for each G_i in $O(|E|c/\epsilon)$ time. Furthermore, the union of unmatched vertices makes up an at most $c(\epsilon/c)$ fraction of R', which proves the claim. □

Data: A bipartite graph $G = (L, R, E)$
Result: A (c,a)-recommendation subgraph H
$R' \leftarrow$ a random sample of $|L|c/a$ vertices from R;
Choose $G[L, R_1], \ldots, G[L, R_c]$ as in Theorem 2.10;
for *i in [1..c]* **do**
  $M_i \leftarrow$ A matching of $G[L, R_i]$ with no augmenting path of length $2c/\epsilon$;
end
$H \leftarrow M_1 \bigcup \ldots \bigcup M_c$;
return H;

Algorithm 3: The partition algorithm

Notice that if we were to run the augmenting paths algorithm to completeness for each matching M_i, then this algorithm would take $O(|E||L|)$ time. We could reduce this further to $O(|E|\sqrt{L})$ by using Hopcroft-Karp. [40]

Assuming a sparse graph where $|E| = \Theta(|L| \log |L|)$, the time complexity of this algorithm is $\Theta(|L|^{3/2} \log |L|)$. The space complexity is only $\Theta(|E|) = \Theta(|L| \log |L|)$, but a large constant is hidden by the big-Oh notation that makes this algorithm impractical in real test cases.

2.5 Generalized Models of Recommendation Graphs

Even though we studied the fixed-degree uniform model in detail, recommendation systems based on relevance in practice will not have edges that are spread uniformly at random. Items that are about specific topics are much more likely to interlink within themselves than to those outside that topic, leading to clusters of recommendations. To understand this clustering in underlying graphs in practice, we compiled results from several e-commerce retailers that have been aggregated and anonymized in the table shown below. For each retailer, we compiled the product ontology present within the site that places a product in this tree-like categorization. E.g., a juicer called "Breville Juice Fountain Plus" is in the tree path: Home → Juicers → High Speed Juicers → Breville Juice Fountain Plus. We then examined the recommendations from products at different depths of the hierarchy. In the table in Figure 2.6 we show the edges adjacent to products at depth 4 or greater. We calculated the percentage of edges connecting to products that had different least common ancestors (LCA) with the current product. We then randomized the edges so that we can compare how the graph would have looked if there was no clusters and re-calculated the distribution of the edges and the LCA levels. We noticed that the uniform distribution had edges that had very shallow LCA indicating that most edges did not follow the product hierarchy while in reality, the

endpoints of edges recommended had much deeper LCA meaning recommendation edges were clustered based on the product hierarchy. This led us to formalize this new model of input graphs that we study in Subsection 2.5.1 as the *hierarchical tree model*. In a

LCALevel	0	1	2	3	4	5	6
Uniform	13.4	69.7	12.5	2.6	1.2	0.6	0.0
Hierarchical	7.1	1.9	8.0	24.9	52.3	5.5	0.2

FIGURE 2.6: Percent edges for depth-4 products by LCA of endpoints in reality (from hierarchical data) and simulated uniform distribution of edges.

second analysis, we simply truncated the product hierarchy at depth 3 and collected the resulting disjoint clusters in the hierarchy. We then examined all the recommendations and partitioned them into those going between each pair of these clusters. In a uniform distribution, we would expect the edges to be equally likely to span across each pair of clusters (if clusters are equal sized). But what we observed was that different pairs of clusters had different edge-densities. For instance, an Espresso Machine might point more to other Coffee Machines or Coffee Beans (note that Coffee Beans and Espresso Machine might share no LCA apart from the root) than to other clusters. These results are shown in Figure 2.7 which clearly demonstrates that the pairs of clusters responsible for the most number of recommendation edges produce many more edges than the uniform model would predict. This motivated us to define and study the *cartesian product model* in Subsection 2.5.2 which is orthogonal to the uniform and hierarchical tree models.

The way we performed the analyses for these new models are similar to those that we carried out for the uniform model. While we only present results for the approximation of $(c, 1)$-recommendation subgraphs for brevity, these results can be extended to the more general problem of finding (c, a)-recommendation subgraphs as done in Section 2.4.1.

2.5.1 Hierarchical Tree Model

In this model, the vertex sets L and R are the leaf sets of two trees T_L and T_R of depth D where there is a 1-to-1 correspondence between the subtrees of these two trees. We also assume that each branching in both T_L and T_R splits the nodes evenly into the two subtrees. As in the previous sections, we set $|L|/|R| = k$, and require that this ratio is still k if we divide the size of any subtree on the left and that of its corresponding subtree on the right. For simplicity of notation, we will use a subtree and its leaf set interchangeably. We assume that the trees are fixed in advance but the bipartite recommendation graph $G = (L, R, E)$ is generated probabilistically according to the

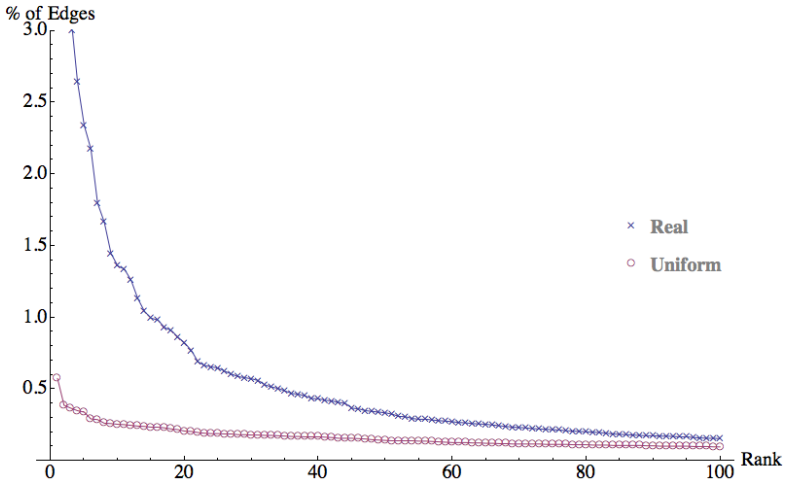


FIGURE 2.7: Histogram of percent edges between pairs of clusters. Each point on x-axis is a pair of clusters. The x-axis has no inherent order but they have been sorted by number of edges for easier visualization. The tail is omitted.

following procedure. Let $u \in L$ and $T_L^0, \ldots T_L^{D-1}$ be the subtrees it belongs at depths $0, \ldots, D-1$. Also, let $T_R^0, \ldots, T_R^{D-1}$ be the subtrees on the right that correspond to these trees on the left. We let u make a recommending edge to d_{D-1} of the vertices in T_R^{D-1}, d_{D-2} edges to the vertices in $T_R^{D-2} \backslash T_R^{D-1}$ and so on. The d_i edges out of u are chosen uniformly from $T_R^i \setminus T_R^{i+1}$. Let $d = d_0 + \ldots + d_{D-1}$.

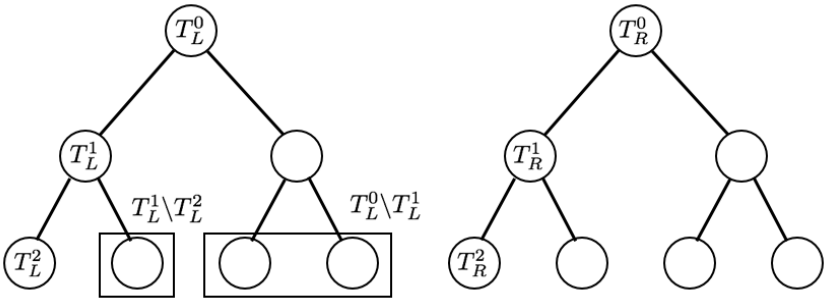


FIGURE 2.8: This diagram shows the notation we use for this model and the 1-to-1 correspondence of subtrees.

For the $a = 1$ case, our goal now is to find a b-matching [27] in this graph that is close to optimal in expectation. That is, our degree upper and lower bounds on vertices in L and R are c and 1 respectively. Let $c = c_0 + \ldots + c_{D-1}$ be similar to how we defined d. To combine the analysis of the randomness of the algorithm and the randomness of the graph, the algorithm will pick c_i edges uniformly from among the d_i edges going to each level of the subtree to form a $(c, 1)$-recommendation subgraph H. This enables us to think of H as being generated by the same process that generated G but with fewer neighbors selected. With this model and parameters in place, we can have the following analog of our main theorem for $a = 1$ for the hierarchical model.

Theorem 2.12. *Let S be the subset of edges $v \in R$ such that $\deg_H(v) \geq 1$ in the hierarchical tree model. Then*

$$\mathrm{E}[S] \geq r(1 - \exp(-ck))$$

Proof. Let $v \in R$ and let $T_L^{D-1}, T_L^{D-2} \backslash T_L^{D-1}, \ldots, T_L^0 \backslash T_L^1$ be the sets it can take edges from. Since T_L and T_R split perfectly evenly at each node the vertices in these sets will be chosen from $r_{D-1}, r_{D-1}, r_{D-2}, \ldots, r_1$ vertices in R as neighbors, where r_i is the size of subtree of the right tree rooted at depth i. Furthermore, each of these sets described above have size $l_{D-1}, l_{D-1}, l_{D-2}, \ldots, l_1$ respectively, where l_i is the size of a subtree of T_L rooted at depth i. It follows that the probability that v does not receive any edges at all is at most

$$\begin{aligned}
\Pr[\neg X_v] &= \left(1 - \frac{1}{r_{D-1}}\right)^{c_0 l_{D-1}} \prod_{i=1}^{D-1} \left(1 - \frac{1}{r_i}\right)^{c_{D-i} l_i} \\
&\leq \exp\left(-\frac{l_{D-1}}{r_{D-1}} c_0\right) \prod_{i=1}^{D-1} \exp\left(-\frac{l_i}{r_i} c_{D-i}\right) \\
&= \exp\left(-(c_0 + \ldots + c_{D-1})k\right) \\
&= \exp(-ck)
\end{aligned}$$

Since this is an indicator variable, it follows that

$$\mathrm{E}[S] = \mathrm{E}\left[\sum_{v \in R} X_v\right] \geq r\left(1 - \exp(-ck)\right)$$

□

Note that this is the same result as we obtained for the fixed degree model in Section 2.4.1. In fact, the approximation guarantees when $ck \ll 1$ or $ck \gg 1$ hold exactly as before.

The algorithmic sampling of H is convenient in this model because we separated out the edge generation process at a given depth from the edge generation process at deeper subtrees. If we superimpose T_L and T_R, then an edge between $u \in L$ and $v \in R$ must have come from an edge generated by the process corresponding to the lowest common ancestor of u and v in the same hierarchy. This way, the algorithm can actually sample intelligently and in the same way that the graph was generated in the first place, which is also the key to our simple analysis. Note that we do not have to assume that the trees

T_L and T_R are binary. We only need the trees to be regular and evenly divided at each vertex since the proof only relies on the proportions of the sizes of the subtrees in T_L and T_R.

2.5.2 Cartesian Product Model

In this model, we assume that L has been partitioned into t subsets $L_1, \ldots, L_t$ and that R has been partitioned into t' subsets $R_1, \ldots, R_{t'}$. For convenience, we let $|L_i| = l_i$ and $|R_j| = r_j$. Given this, for each $1 \leq i \leq t$ and each $1 \leq j \leq t'$, we let $G[L_i, R_j]$ be an instance of the fixed degree model with $d = d_{ij}$. This allows us to assume different densities of edges between different pairs of clusters. However, we require that for all i, we have $\sum_{j=1}^{t'} d_{ij} = d$ for some fixed d. We also require that we have fixed in advance $c_{ij} \leq d_{ij}$ for each $1 \leq i \leq t$ and $1 \leq j \leq t'$ that satisfy $\sum_{j=1}^{t'} c_{ij} = c$ for all i for some fixed c. To sample H from G, we sample c_{ij} neighbors from R_j for each $u \in L_i$. Letting S be the set of vertices in $v \in R$ that satisfy $\deg_H(v) \geq 1$, we can show the following theorem.

Theorem 2.13. *Let S be the subset of edges $v \in R$ such that $\deg_H(v) \geq 1$ in the cartesian product model. Then*

$$\mathrm{E}[S] \geq r - \sum_{j=1}^{t'} r_j \exp\left(-\sum_{i=1}^{t} c_{ij} \frac{l_i}{r_j}\right)$$

Proof. Let $v_j \in R_j$ be an arbitrary vertex and let X_{v_j} be the indicator variable for the event that $\deg_H(v_i) \geq 1$. The probability that none of the neighbors of some $u_i \in R_i$ is v_j is exactly $(1 - \frac{1}{r_j})^{c_{ij}}$. It follows that the probability that the degree of v_j in the subgraph $H[L_i, R_j]$ is 0 is at most $(1 - \frac{1}{r_j})^{c_{ij} l_i}$. Considering this probability over all R_j gives us:

$$\Pr[X_{v_i} = 0] = \prod_{i=1}^{t} \left(1 - \frac{1}{r_j}\right)^{c_{ij} l_i} \leq \exp\left(-\sum_{i=1}^{t} c_{ij} \frac{l_i}{r_j}\right)$$

By linearity of expectation $\mathrm{E}[S] = \sum_{i=1}^{t'} r_i \mathrm{E}[X_{v_i}]$, so it follows that

$$\mathrm{E}[S] \geq \sum_{j=1}^{t'} r_j \left(1 - \exp\left(-\sum_{i=1}^{t} c_{ij} \frac{l_i}{r_j}\right)\right) = r - \sum_{j=1}^{t'} r_j \exp\left(-\sum_{i=1}^{t} c_{ij} \frac{l_i}{r_j}\right)$$

□

A powerful aspect of this model and the algorithm we described for sampling H is that we are free to select c_{ij}. In particular, c_{ij} can be chosen to maximize the approximation

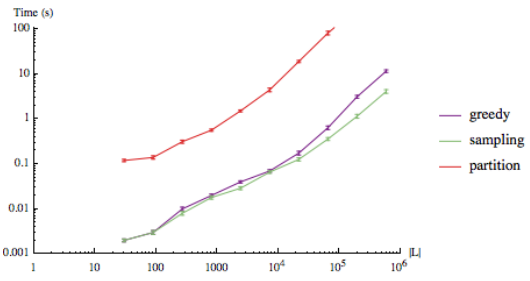


FIGURE 2.9: Time needed to solve a (10,3)-recommendation problem in random graphs where $|R|/|L| = 4$ (Notice the log-log scale.)

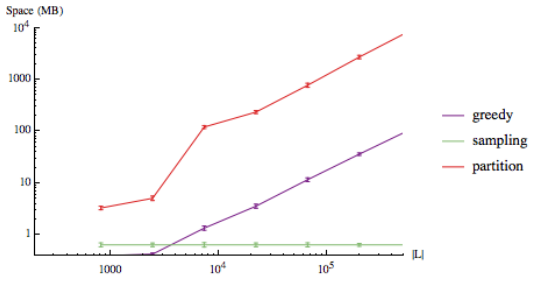


FIGURE 2.10: Space needed to solve a (10,3)-recommendation problem in random graphs where $|R|/|L| = 4$ (Notice the log-log scale.)

guarantee in expectation we obtained above using gradient descent or other first order methods prior to running the recommendation algorithm to increases the quality of the solution.

2.6 Experimental Results

2.6.1 Simulated Data

We simulated performance of our algorithms on random graphs generated by the graph models we outlined. In the following figures, each data point is obtained by averaging the measurements over 100 random graphs. We first present the time and space usage of these algorithms when solving a $(10, 3)$-recommendation subgraph problem in different sized graphs. In all our charts, error bars are present, but too small to be noticeable. Note that varying the value of a and c would only change space and time usage by a constant, so these two graphs are indicative of time and space usage over all ranges of parameters.

Recall that the partition algorithm split the graph into multiple graphs and found matchings (using an implementation of Hopcroft-Karp [40]) in these smaller graphs which were then combined into a recommendation subgraph. For this reason, a run of the partition algorithm takes much longer to solve a problem instance than either the sampling or greedy algorithms. It also takes significantly more memory as can be seen in Figures 5 and 6. Compare this to greedy and sampling which both require a single pass over the graph, and no advanced data structures. In fact, if the edges of G is pre-sorted by the edge's endpoint in L, then the sampling algorithm can be implemented as an online algorithm with constant space and in constant time per link selection. Similarly, if the edges of G is pre-sorted by the edge's endpoint in R, then the greedy algorithm can be

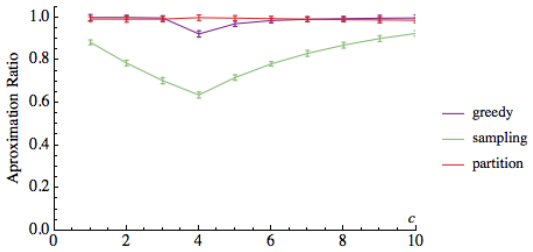


FIGURE 2.11: Solution quality for the $(c, 1)$-recommendation subgraph problem in graphs with $|L| = 25$k, $|R| = 100$k, $d = 20$

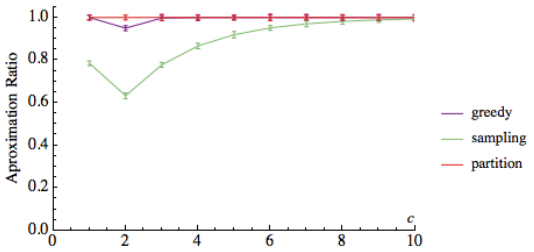


FIGURE 2.12: Solution quality for the $(c, 1)$-recommendation subgraph problem in graphs with $|L| = 50$k, $|R| = 100$k, $d = 20$

implemented so that the entire graph does not have to be kept in memory. In this event, greedy uses only $O(|L|)$ memory.

Next, we analyze the relative qualities of the solutions each method produces. Figures 2.11 and 2.12 plot the average performance ratio of the three methods compared to the trivial upper bounds as the value of c, the number of recommendations allowed is varied, while keeping $a = 1$. They collectively show that the lower bound we calculated for the expected performance of the sampling algorithm accurately captures its behavior when $a = 1$. Indeed, the inequality we used is an accurate approximation of the expectation, up to lower order terms, as is demonstrated in these simulated runs. The random sampling algorithm does well, both when c is low and high, but falters when $ck = 1$. The greedy algorithm outperforms the sampling algorithm in all cases, but its advantage vanishes as c gets larger. Note that the dip in the graphs when $cl = ar$, at $c = 4$ in Figure 2.11 and $c = 2$ in Figure 2.12 is expected and was previously demonstrated in Figure 2.3. The partition algorithm is immune to this drop that affects both the greedy and the sampling algorithms, but comes with the cost of higher time and space utilization.

In contrast to the case when $a = 1$, the sampling algorithm performs worse when $a > 1$ but performs increasingly better with c as demonstrated by Figures 2.13 and 2.14. The greedy algorithm continues to produce solutions that are nearly optimal, regardless of the settings of c and a, even beating the partition algorithm with increasing values of a. Our simulations suggest that in most cases, one can simply use our sampling method for solving the (c, a)-recommendation subgraph problem. In cases where the sampling is not suitable as flagged by our analysis, we still find that the greedy performs adequately and is also simple to implement. These two algorithms thus confirm to our requirements we initially laid out for deployment in large-scale real systems in practice.

To summarize, our synthetic experiments show the following strengths of each algorithm:

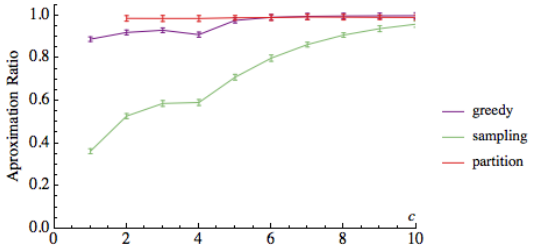


FIGURE 2.13: Solution quality for the $(c, 2)$-recommendation subgraph problem in graphs with $|L| = 50$k, $|R| = 100$k, $d = 20$

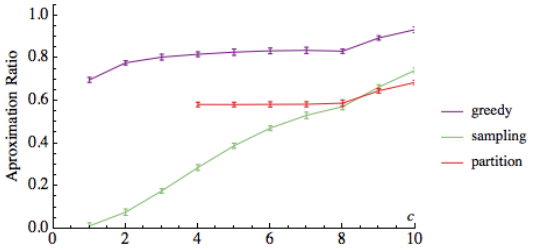


FIGURE 2.14: Solution quality for the $(c, 4)$-recommendation subgraph problem in graphs with $|L| = 50$k, $|R| = 100$k, $d = 20$

Sampling Algorithm: Sampling uses little to no memory and can be implemented as an online algorithm. If keeping the underlying graph in memory is an issue, then chances are this algorithm will do well while only needing a fraction of the resources the other two algorithms would need.

Partition Algorithm: This algorithm does well, but only when a is small. In particular, when $a = 1$ or 2, partition seems to be the best algorithm, but the quality of the solutions degrade quickly after that point. However this performance comes at expense of significant runtime and space. Since greedy performs almost as well without requiring large amounts of space or time, partition is best suited for instances where a is low the quality of the solution is more important than anything else.

Greedy Algorithm: This algorithm is the all-round best performing algorithm we tested. It only requires a single pass over the data thus very quickly, and uses relatively little amounts of space enabling it run completely in memory for graphs with as many as tens of millions of edges. It is not as fast as sampling or accurate as partition when a is small, but it has very good performance over all parameter ranges.

2.6.2 Real Data

We now present the results of running our algorithms on several real datasets. In the graphs that we use, each node corresponds to a single product in the catalog of a merchant and the edges connect similar products. For each product up to 50 most similar products were selected by a proprietary algorithm of BloomReach that uses text-based features such as keywords, color, brand, gender (where applicable) as well as user browsing patterns to determine the similarity between pairs of products. Such algorithms are commonly used in e-commerce websites such as Amazon, Overstock, eBay etc to display the most related products to the user when they are browsing a specific product.

Two of the client merchants of BloomReach presented here had moderate-sized relation graphs with about 10^5 vertices and 10^6 input edges (candidate recommendations); the remaining merchants (3, 4 and 5) have on the order of 10^6 vertices and 10^7 input edges between them. We estimated an upper bound on the optimum solution by taking the minimum of $|L|c/a$ and the number of vertices in R of degree at least a. Figures 2.15, 2.16 and 2.17 plot the average of the optimality percentage of the sampling, greedy and partition algorithms across all the merchants respectively. Note that we could only run the partition algorithm for the first two merchants due to memory constraints.

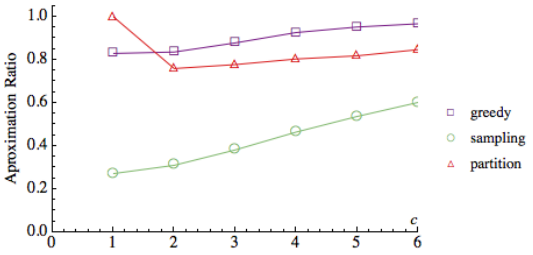


FIGURE 2.15: Solution quality for the $(c, 1)$-recommendation subgraph problem in retailer data

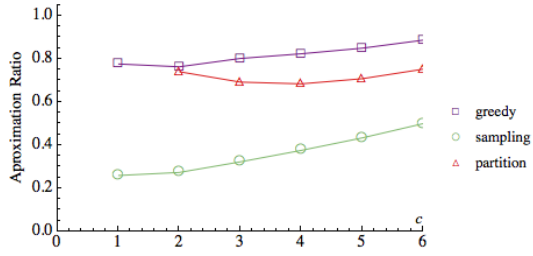


FIGURE 2.16: Solution quality for the $(c, 2)$-recommendation subgraph problem in retailer data

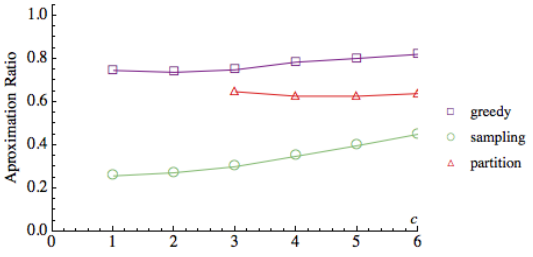


FIGURE 2.17: Solution quality for the $(c, 3)$-recommendation subgraph problem in retailer data

From these results, we can see that that greedy performs exceptionally well when c gets even moderately large. For the realistic value of $c = 6$, the greedy algorithm produced a solution that was 85% optimal for all the merchants we tested. For several of the merchants, its results were almost optimal starting from $a = 2$.

The partition method is also promising, especially when the a value that is targeted is low. Indeed, when $a = 1$ or $a = 2$, its performance is comparable or better than greedy, though the difference is not as pronounced as it is in the simulations. However, for larger values of a the partition algorithm performs worse.

The sampling algorithm performs mostly well on real data, especially when c is large. It is typically worse than greedy, but unlike the partition algorithm, its performance improves dramatically as c becomes larger, and its performance does not worsen as quickly when a gets larger. Therefore, for large c sampling becomes a viable alternative to greedy mainly in cases where the linear memory cost of the greedy algorithm is too prohibitive.

2.7 Conclusions and Future Work

We have presented a new class of structural recommendation problems cast as computationally hard subgraph selection problems, and analyzed three algorithmic strategies to solve these problems. The sampling method is most efficient, the greedy approach trades off computational cost with quality, and the partition method is effective for smaller problem sizes. We have proved effective theoretical bounds on the quality of these methods, and also substantiated them with experimental validation both from simulated data and real data from retail web sites. Our findings have been very useful in the deployment of effective structural recommendations in web relevance engines that drive many of the leading websites of popular retailers.

We believe that our work lends itself to promising future work in two directions. The first is that through a better understanding of the underlying graph's topology, more precise or complex models can be used. This would require an empirical validation of the proposed graph model and the adaptation of our methods to different random graph models.

The second is that most of our algorithms aren't particularly suited for the weighted setting. While our sampling result carries over to the weighted regime as seen in Theorem 2.5, our other algorithms don't, and even this result is weak compared to the unweighted result we presented in full. The problem presented by ignoring weights is that some really high value recommendations might be ignored by the randomness of the algorithm by chance. In practice, it's possible to mitigate this issue by hardcoding in the really desirable edges, and using seeding either the greedy or sampling algorithm with these edges. While this can work well in practice, it would be nonetheless be valuable to prove strong approximation guarantees in the weighted regime as well.

Acknowledgments: We thank Alan Frieze and Ashutosh Garg for helpful discussions. This work was conducted jointly with R Ravi, and Srinath Sridhar.

Chapter 3

Discrepancy

TABLE 3.1: Notation for Chapter 3

L	Set of users.
R	Set of items.
$G(L, R, E)$	Bipartite graph of candidate recommendations.
H	A subset of G denoting the recommendations made by the system.
u_i	The i^{th} user.
v_j	The j^{th} item.
c_i	The display constraint for the i^{th} user.
a_j	The target degree for the j^{th} item.
$N(u_i)$	The set of c_i recommendations made to the i^{th} in H.
$T(u_i)$	The set of held-out recommendations for the i^{th} user in the test set.

3.1 Motivation

Collaborative filtering has long been a favored approach in recommender systems since its recommendations are derived mainly from the record of interactions between users and items. However, a key concern of CF systems is the filter bubble, the idea that recommendation systems that focus solely on accuracy lead to echo chambers that amplify "rich-get-richer" effects among the recommended items [5, 12, 41, 42]. This problem stems from the way these systems are designed since they can only make confident recommendations on items that have had a lot of engagement, and hence increase their importance. This is the main motivation to diversify the recommendations of such CF systems.

Recent ethical concerns about algorithms have focused on similar issues about algorithmic results inherently being biased [43, 44]. One approach to counter the status quo, also advocated by Karger [44] is to explicitly design algorithms that do not discriminate by designing an appropriate objective function that will increase diversity in CF recommendations.

The importance of diversifying recommendations for the sake of the user arises from their intrinsic appreciation for novelty and serendipity, a view that is supported by psychological studies [45]. Conversely, research in recommendation systems [15] has shown that focusing solely on ratings hurts user satisfaction. This has led to a subfield of recommendation systems that focuses on improving diversity for the benefit of the user [16, 46].

A third motivation is the business need for diversifying recommendations: long-tail catalogs that are frequent in the internet [13, 47] as well as media distributors with all-you-can-play business models [14] require that the recommendations influence users to consume diverse content by driving more traffic to different portions of the site.

Roadmap: In this chapter , we address the non-diverse nature of CF recommendations, the needs of a long-tail business to shape traffic on its own site, and diversifying recommendations for the benefit of the users. We define a notion of diversity conducive to all these needs based on the degree-properties of the graph defined by the recommendations (Sections 3.2, 3.3). After reviewing related work (Section 3.4), we show that the design problem for this notion can be solved efficiently both in theory and practice using network flow techniques (Section 3.5). We validate our method by showing how to adapt standard collaborative filtering algorithms with an efficient post-processing step to optimize for our measure of diversity by sacrificing very little on the recommendation quality on standard data sets (Section 3.6).

3.2 A New Graph Optimization Problem

We model all the user-item recommendations provided by a CF system as a bipartite graph, and the choice of recommendations actually given to the users as a subgraph selection problem in this graph. The constraints on the number of items that can be recommended to a user put bounds on the out-degree of the user nodes. Following our earlier work [17], we model the problem of achieving diversity among the items as specifying target in-degree values for each item and then finding a subgraph that satisfies these constraints as closely as possible. We develop the resulting graph optimization problem next.

Input: A bipartite graph $G(L, R, E)$, a vector of display constraints $\{c_i\}_{i=1}^{l}$, a vector of target degrees $\{a_i\}_{i=1}^{r}$.
Output: A subgraph $H \subseteq G$, of maximum degree c at any node in L, which minimizes the discrepancy from the given target distribution.

FIGURE 3.1: The definition of the MIN-DISCREPANCY problem.

We start our discussion by reviewing the well known b-matching problem on bipartite graphs [48]. In the b-matching problem, an underlying bipartite graph $G = (L, R)$ with edge set E is given, along with a nonnegative weight g on the edges, and two vectors of non-negative integers $(c_1, \ldots, c_l)$ and $(a_1, \ldots, a_r)$ (degree bounds) such that $\sum_{i=1}^{l} c_i = \sum_{i=1}^{r} a_i$. The goal is to find a maximum g-weight (or minimum g-cost) subgraph H where the degree of vertex $u_i \in L$ in H is c_i for every $1 \leq i \leq l$ and the degree of vertex $v_j \in R$ in H is a_j for every $1 \leq j \leq r$. This problem generalizes the well-known maximum weight perfect matching problem, which can be obtained as a special case if we set all target degrees to 1. Like the perfect matching problem in bipartite graphs, the b-matching problem can be solved by a reduction to a network flow model by adding a source with arcs to L, a sink with arcs from R and using the degree constraints as capacities on these respective arcs.

Assume that the degree bounds are given. The vector $(c_1, \ldots, c_l)$ will be taken as a vector of hard constraints that must be met exactly, based on display constraints for the users. In other words, we consider a subgraph H to be a feasible solution if and only if $\deg_H^+(u_i) = c_i$ for all $u_j \in L$. The vector $(a_1, \ldots, a_r)$ will be specified by the recommendation system designer to reflect the motivations described above (increase the coverage of items in CF results, increase the novelty to users on average, or shape traffic to some items). However, this target degree bounds may be unattainable (i.e. there is no feasible b-matching for these degree bounds). To handle this potential infeasibility, we incorporate them in the objective. We call the vector $(a_1, \ldots, a_r)$, the target degree distribution. We now define the objective for a given feasible solution H,

$$D(H) = \sum_{v_j \in R} |deg_H^-(v_j) - a_j|$$

which is simply the sum of the violations (in both directions) of the degree constraints for R. We call this objective the discrepancy between H and the degree distribution $(a_1, \ldots, a_r)$, and we name the problem of meeting the hard constraints $(c_1, \ldots, c_l)$ while minimizing this objective the MIN-DISCREPANCY problem.

The MIN-DISCREPANCY problem defined above generalizes the b-matching problem, and has objective value zero iff there is a feasible b-matching for the given degree bounds.

Input: A weighted bipartite graph $G(L, R, E)$, a vector of display constraints $\{c_i\}_{i=1}^{l}$, a vector of target degrees $\{a_i\}_{i=1}^{r}$.
Output: A maximal weight subgraph $H \subseteq G$, of maximum degree c at any node in L, chosen among the subgraphs which have minimal discrepancy from the given target distribution.

FIGURE 3.2: The definition of the MAX-WEIGHT-MIN-DISCREPANCY problem.

Like the weighted b-matching problem mentioned above, we can adapt our problem to the weighted setting where each edge has a real-valued weight. In this setting, the objective to maximize the total weight of the chosen edges, among graphs that have the minimal discrepancy possible from the given targets. We call this variant of the problem the MAX-WEIGHT-MIN-DISCREPANCY problem.

3.3 Post-processing a CF Recommender

We now show how we can apply the graph optimization problem defined above to post-processing the results of a CF recommendation system. As input to a CF system, we have a set of items I, a set of users U, and list of known ratings given by each user to different subsets of the items. The CF system outputs a relevance function $rel : U \times I \to [a, b]$ that takes pairs of users and items to a predicted recommendation quality in some interval on the real line. (If g can be thought of as a similarity measure, then $\frac{1}{a}(b - rel(u, i))$ can be thought of as a dissimilarity measure.) Without any extra information on the problem domain, CF systems employ user-based filtering, item-based filtering, matrix factorization, or other methods to arrive at these predicted rating qualities. For the rest of this paper, this rating function will be considered to be given as a black-box since its implementation details have no consequence on our model, even though we will experiment with various options in our empirical tests.

A Surplus of Candidate Recommendations. To generate the graph G that will serve as input to our optimization problem, we choose only the recommendations for which the CF recommender predicts a rating above a certain threshold – we enforce this by constraining each user's recommendation list to their top-k candidate recommendations. This is a standard approach used in recommendation diversification [46, 49] and ensures that none of our candidate recommendations are below a certain quality level, thus establishing a quality baseline for our algorithm.

We apply our max-weight min-discrepancy method on this graph with the given predicted ratings as weights and given degree bounds as a post-processing step for the CF

system results to increase their diversity.

3.3.1 Summary of Contributions

1. Following our earlier work [17], we model the problem of post-processing recommendations from a CF system to increase diversity as a maximum-weight degree-constrained subgraph selection problem to minimize the discrepancy from a target distribution.

2. We demonstrate that the problem of finding maximum-weight min-discrepancy subgraph can be reduced to the problem of finding minimum cost flows. In particular, this shows that the discrepancy between a recommendation system and *any* desired indegree distribution can be minimized in polynomial time. The abundance of fast solvers [50] for this problem makes our model not just theoretically interesting, but also practically feasible. Moreover, we prove that aggregate diversity maximization can be implemented special case of the discrepancy minimization problem. This generalizes the work of Adomavicius and Kwon on maximizing aggregate diversity [51] while simultaneously maximizing recommendation quality, while matching the same asymptotic runtimes.

3. We conduct experiments on standard datasets such as MovieLens-1m, and Netflix Prize data. By feeding our discrepancy minimizer as a post-processing step on the undiversified recommendation networks created by standard collaborative filtering algorithms, we measure the trade-off our algorithm makes between discrepancy and recommendation quality under a variety of parameter settings. We compare against baselines and other diversification approaches, and find that our diversifier makes more relevant recommendations despite achieving higher diversity gains, as measured not only by our discrepancy measure, but also by standard sales diversity metrics such as the Gini index or aggregate diversity.

3.4 Related Work

First we review related work on various collaborative filtering approaches, and then discuss various extant notions of diversity already considered in the recommender system literature.

3.4.1 Collaborative Filtering

Collaborative filtering is the most versatile and widely accepted way of building recommender systems. The main idea behind collaborative filtering is to exploit the similarities between different users or between different items using user feedback. While there are many different methods for doing this, we constrain our evaluation to three representative approaches.

1. Matrix factorization approaches assume the existence of D latent features which describe both users and items, and seeks to find two rank D matrices whose product approximates the matrix of all known rankings. The advantage is that D is typically much smaller than the number of users or items. In our work, we experiment with a version of this approach due to Hu [1].

2. Another popular approach is neighborhood based recommenders, which can either be user-based or item-based. These approaches define a distance between pairs of users and pairs of items respectively, using measures like cosine similarity or Pearson correlation [52]. The user-based approach then predicts the unknown rating from user u to item i by taking a distance weighted linear combination of the ratings of similar users on item i. The item-based approach operates similarly, but instead takes a weighted combination of the ratings of user u on items similar to item i. We use the implementations of these methods in RankSys [53] in our experiments.

3. Finally, we consider a graph based recommender strategy due to Cooper et. al. [54]. This method considers a bipartite graph of known user and item interactions, ignoring all rating information. In this graph, a random walk of length 2 from a user u corresponds to the selection of a user similar to u, in the sense that both u and any user reachable from u in 2 steps have at least one item as a common interest. Therefore, a random walk of length 3 corresponds to sampling an item liked by a similar user, and recommendations for a user u are ranked according to how many random walks of length 3 starting at that user terminate at a given item. Since this method is both simple to state and implement on small to medium sized datasets, we use our own implementation of this method in our empirical comparisons. While less commonly used than the first two types of recommenders we discussed, this approach is still representative of a large class of recommendation strategies such as UserRank [55], ItemRank [56], or other other random walk based techniques [57].

3.4.2 Sales Diversity

User-Focused Diversity: User novelty has been called intra-list diversity [58], with the list referring to the list of recommendations made to a particular user. The need for novelty from the user's point of view is a psychological one [45]. While lack of intra-list diversity was a particularly bad shortcoming of early recommender systems [59], these problems have since been addressed in many works [60]. In this work, we do not consider diversity at a user level, and instead take a system level view, which motivated more by business needs than user needs.

The Need for Sales Diversity: As mentioned above, the need for system-level diversification in recommenders is a business related one. Since the internet enables businesses with low inventory costs, focusing on making more recommendations in the long tail can be an effective retail strategy. This view is most clearly expressed by Anderson, who advocates selling "selling less of more" [25]. Interestingly, recommender systems rarely capitalize on this opportunity, and often compound the problems observed with popularity bias. Indeed, Zhou et. al. find that YouTube's recommendation module leads to an increase in popularity for the most popular items [42]. Similarly, Celma et. al. report similar findings for music recommendations on Last.fm [41]. Hosanagar and Fleder show that this popularity bias can lead to subpar pairings between users and items, potentially hurting customer satisfaction [12], and McNee reports that a focus on accuracy alone has hurt the user experience of recommender systems [15]. Since recommendations have an outsized impact on customer behavior [51, 61], businesses have a need to control the distribution of recommendations that they surface in their recommenders.

Metrics for Sales Diversity: There are several well-established metrics for measuring sales diversity, and we focus our attention on three.

1. The most popular among these is the aggregate diversity, which is the total number of objects that have been recommended to at least one user. Under this name, this measure has been used notably by Adomavicius and Kwon [49, 51] and by Castells and Vargas [60]. It has also been used as a measure of system-wide diversity under the name of coverage [62, 63]. While easy to understand and measure, the aggregate diversity leaves a lot be desired as a measure of distributional equality. In particular, aggregate diversity treats an item which was recommended once as well-covered as an item which was recommended thousands of times. For example, imagine a system that recommends each item in a set of n items twice. This network will have the same aggregate diversity as a network which recommends one of the items n times, and every other item only once, even though this system is much more biased than the first. Moreover, aggregate diversity can be a

misleading measure of diversity when the number of users far outnumbers the size of the catalog. Under these circumstances, even very obscure items may get recommended at least once. Therefore, while aggregate diversity is a good baseline, more refined measures are needed to evaluate the equitability of the distribution of recommendations.

2. An example of a more nuanced metric is provided by the Gini index. This measure is most popularly used in economics, as a quantization of wealth or income inequality. The Gini Index can be adapted for the recommendation setting by considering the number of recommendations an item gets as its "wealth" in the system. The Gini index defines the most equitable distribution to be the one where every item is recommended an equal number of times. Given the actually realized distribution of recommendations, it aggregates the difference between the number of recommendations the bottom nth percentile gets in the system and the number of recommendations they would have obtained under the uniform distribution where n ranges from 0 to 100%. The measure we propose is a particularly good proxy for the Gini index, since both measure a notion of distance from the uniform distribution. Since recommender systems produce distributions even more unequal than the typical wealth distributions within a country, this metric has found widespread acceptance in the recommendation community [62, 64–66].

3. Finally, we consider the entropy of the distribution of recommendations. Entropy has its roots in physics and information theory, where it is used to measure the amount of information contained in a stochastic process. For every item, we can define a probability of being surfaced by the recommender by counting what fraction of recommendations (made to any user) point to this item. As with the Gini index, optimal entropy is achieved if and only if the recommendation distribution is uniform. While less common than either aggregate diversity or the Gini index, the entropy of the recommender system has also been used by many researchers [64, 67].

We measure the diversification performance of our methods and the baselines we test in our experimental section by all three of these metrics - aggregate diversity, Gini index and entropy.

Approaches for Increasing Sales Diversity: Attempts at increasing sales diversity fall into two approaches: optimization and reranking.

1. The optimization approach has been taken up most notably by Adomavicius and Kwon [51], who consider heuristic and exact algorithms for improving aggregate

diversity. Their flow based solution is approximate, while their exact solution to this problem relies on integer programming and has exponential complexity. Our work in this chapter subsumes these approaches by giving an exact polynomial algorithm for aggregate diversity maximization. To the best of our knowledge, neither the Gini index nor the entropy of the degree distribution can be optimized in an exact sense.

2. The reranking based approaches are by far the more popular choice in increasing sales diversity. Here, we consider three different approaches by Castells and Vargas, spread across two different papers. The first two approaches model discovery in a recommender system by associating each user-item pair with a binary random variable called *seen* which represents the event that a user is familiar with the given item, with the assumption that an unseen item is novel to the user. Given a training dataset θ which maps pairs of users from U and items from I to known rating values, the authors use a maximum likelihood model to estimate the probability that an item is seen by a user. In particular, they set the probability of an item i already being known to a user as the fraction of users who have rated that item

$$p(seen|i,\theta) = \frac{|\{u \in U | r(u,i) \neq \emptyset\}|}{|U|}$$

Given these probabilities, the popularity complement similarly defines the novelty of an item i as $\text{nov}_{PC}(i|\theta) = 1 - p(seen|i,\theta)$. The free discovery method of measuring novelty similarly defines the novelty of an item i to a user u as $\text{nov}_{FD}(i|\theta) = -\log_2(p(seen|i,\theta))$. After these probabilities are estimated from the training data, the candidate list of recommendations are reranked according to the a score which is the average of the predicted relevance of the item and the novelty of the item [46].

Instead of combining novelty and relevance components in the same function, the Bayes Rule method by the same authors explicitly adjusts the rating prediction function. In particular, let $rel : U \times I \to [0,1]$ be the function which predicts the strength of a recommendation between user and item pairs. The authors suggest that the probability that an item is relevant to a user is proportional to its predicted rating, and verify this claim experimentally. Combining this assumption with Bayesian inversion, they come up with a revised prediction function $rel_{BR}(u,i) = rel(u,i)\left(\sum_{u'} rel(u',i)\right)^{-\alpha}$, and rerank the recommendations according to this function. Note that predicted quality of a recommendation only differs from the original prediction by a factor of $\left(\sum_{u'} rel(u',i)\right)^{-\alpha}$. This term is a function of the item i, and grows larger as the sum of the predicted ratings of i goes

lower. Therefore, rescoring in this manner dampens the predicted ratings of popular items, while increasing the predicted ratings of less popular items [68].

Constrained Recommendations: Finally, our problem is one of constrained recommendation, and this problem has been studied in contexts where the data is both large and small. In the small data end of the of the spectrum, recommenders have been developed to solve problems such as matching students to courses based on prerequisites and requirements [69] or matching reviewers to paper submissions [70, 71]. These types of models perform well because they are built specifically for the task at hand, but they are not suited for general purpose recommendation tasks for increasing diversity.

On the other end of the spectrum, in a context of matching buyers with sellers with the goal of maximizing revenue [72], the problem has been modeled as a matching problem to maximize the number of recommended edges (rather than their diversity). Similarly, large-scale matching problems modeling recommendations have been tackled in a distributed setting [73] using degree bounds, but these methods are unable to accurately enforce degree lower bounds on the items begin recommended (so they are simply set to zero). Here again, the objective is to maximize the number of edges chosen rather than any measure of diversity. While our methods do not scale to the same level, we are able to model diversity with degree bounds more accurately.

3.5 Algorithms

In this section, we prove that discrepancy from a target distribution can be minimized efficiently by reducing this problem to one invocation of a minimum cost flow problem. This result holds regardless of the target in-degree distribution and the required out-degree distribution.

3.5.1 Construction of the Flow Network

Let $G = (L, R, E)$ be the input bipartite graph which contains candidate recommendations. We construct a flow network out of G such that the min-cost feasible flow will have cost equal to the min-discrepancy. Our network will have $|V|+2$ nodes: two special sink nodes t_1 and t_2, as well as a copy of each node in G (See Figure 3.3). We set the supply of each node u_i to c_i (its specified out-degree), and the demand of the sink t_2 to $\sum_{j=1}^{r} a_j = \sum_{i=1}^{l} c_i$. Next, for each arc $(u_i, v_j) \in G$, we create an arc (u_i, v_j) in the flow network, with unit capacity and zero cost. For each node v_j, we create an arc to

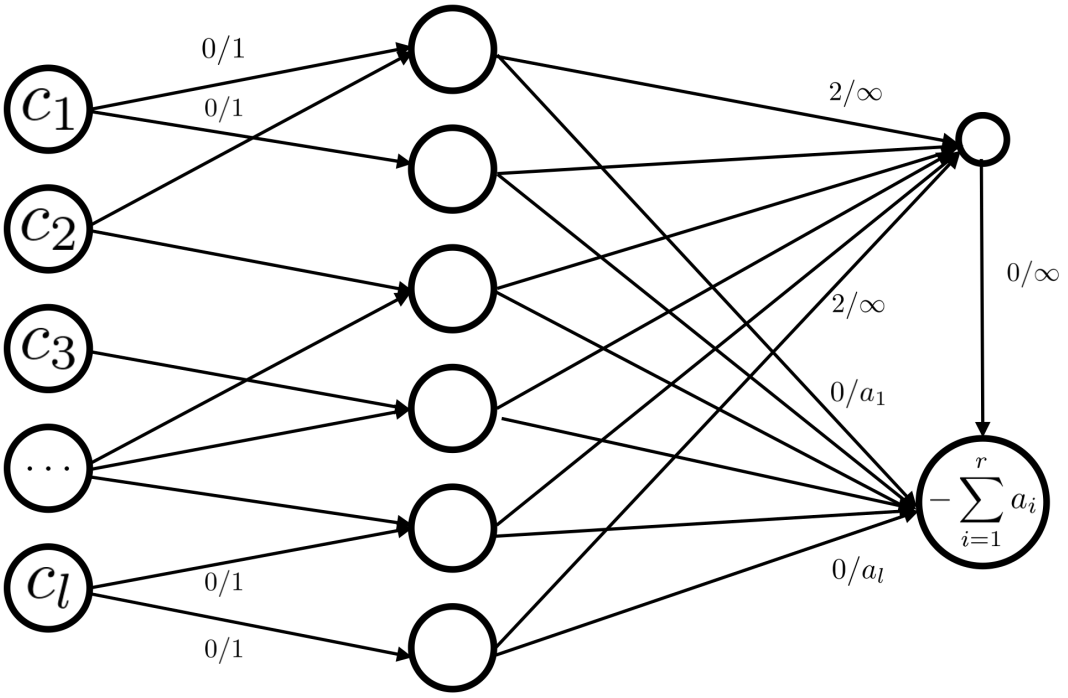


FIGURE 3.3: The network flow model for the MIN-DISCREPANCY problem with nodes labelled with their supply and arcs labeled with their cost/capacity. Unlabelled nodes have zero supply.

each sink. We add the arc (v_j, t_1) of capacity a_j (its target out-degree) and zero cost, and the arc (v_j, t_2) of infinite capacity and cost 2. We finally add to our network an arc (t_1, t_2) between the two sinks, with infinite capacity and zero cost. Our assumptions ensure that total supply, $\sum_{i=1}^{l} c_i$ meets total demand $\sum_{j=1}^{r} a_j$, and that a feasible flow exists since each node u_i in L can send as much as $deg_G^+(u_i) \geq c_i$ flow to the sink t_2 via any c_i different neighbors. Note that there are $|E| + 2|R| + 1$ arcs in total in our flow network. The complete flow network constructed this way is shown in Figure 3.3.

Our main theorem shows that the minimum cost of a flow in this network is the same as the minimum discrepancy a subgraph of G has from our target in-degree distribution.

Theorem 3.1. *Suppose $G = (V, E)$ satisfies $deg_G^+(u_i) \geq c_i$ for all $u_i \in L$ and the degree distributions satisfy $\sum_{i=1}^{l} c_i = \sum_{i=1}^{r} a_i$. Then the minimum cost flow in the network constructed above has the same cost as the value of the MIN-DISCREPANCY problem and can be computed $O(|E||V|^2 \log(|V|))$ time.*

Proof. Consider a minimum cost flow in this network. Since the network's capacities and supplies are all integral, we may assume that this minimum cost flow is integral as well [74]. This means each edge crossing from L to R is either fully used or unused because it is unit capacity.

We let H be a subgraph of G defined by taking the edges of the form (u_i, v_j) where (u_i, v_j) is used in the flow. Since each such edge is either used or unused, and the supply of node u_i is c_i, we will satisfy the constraints of the form $\deg_H^+(u_i) = c_i$. To see that the cost of this flow is the same as the cost of our objective, note that we can partition the vertices in R into two halves: P for the vertices satisfying their degree requirement $\deg_H^-(v_j) \geq a_j$, and N for the vertices not satisfying their degree requirement. We can now write our objective as follows.

$$\sum_{v_j \in R} |\deg_H^-(v_j) - a_j| = \sum_{v_j \in P} \left(\deg_H^-(v_j) - a_j\right) + \sum_{v_j \in N} \left(a_j - \deg_H^-(v_j)\right)$$

However, note that our flow is feasible. Therefore, the total number of edges recommended is $\sum_{i=1}^{l} c_i$. It now follows that $\sum_{v_j \in R}(\deg_H^-(v_j) - a_j) = \sum_{v_j \in R} \deg_H^-(v_j) - \sum_{i=1}^{l} c_i = 0$ from our assumption that $\sum_{i=1}^{l} c_i = \sum_{j=1}^{r} a_j$. Adding this to the expression above gives the following.

$$\sum_{v_j \in R} |\deg_H^-(v_j) - a_j| = 2 \sum_{v_j \in P} \left(\deg_H^-(v_j) - a_j\right)$$

In our formulation, we only pay for the flow going through a node v_j if the flow is in excess of a_j. Since we pay 2 units of cost for each unit of this type of flow, and don't pay for anything else, our objective matches that of the flow problem.

By reversing our reduction, we can show that every subgraph H with the desired properties induces a flow with the same cost as well. Therefore, the minimum discrepancy problem can be solved by a single invocation of a minimum cost flow algorithm, on a network with $|L| + |R| + 2$ nodes and $2|R| + |E| + 1$ edges with capacity bounded by $|V|$. This can be solved in $O(|E||V|^2 \log(|V|))$ time, using capacity scaling or other competitive methods [75]. □

Aggregate Diversity. Recall that aggregate diversity is the total number of items recommended by a recommender system. Aggregate diversity does not correspond to discrepancy from any target distribution, however it can be maximized by our model as well.

Theorem 3.2. *Suppose $\sum_{i=1}^{l} c_i \geq r = |R|$. Aggregate diversity is maximized by the minimum cost flow solution in the network constructed for the MIN-DISCREPANCY$(G, \{c_i\}_{i=1}^{l}, \{1\}_{j=1}^{r})$.*

Proof. The sufficiency of our condition is obvious as it is needed to make sure that the supply of the nodes in L can be absorbed by the sink node. Now suppose that a recommender system achieves aggregate diversity r. A total of $\sum_{i=1}^{l} c_i$ units of flow make it to the sink, and each has to travel through an arc of cost 0 or 1. Since there are r different items in R, and each can send 1 unit of flow without cost, this solution has cost $\left(\sum_{i=1}^{l} c_i\right) - r$. Conversely, suppose some solution obtains cost $\left(\sum_{i=1}^{l} c_i\right) - r$. The only way the cost can be reduced below $\left(\sum_{i=1}^{l} c_i\right)$ is achievable is through the use of 0 cost arcs. Since each such arc has capacity 1, at least r such arcs must be used in the solution. This implies that a solution of aggregate diversity r exists. □

3.5.2 Incorporating Recommendation Relevance

3.5.2.1 Cumulative Gain

Note that we have had to assign zero costs to all the edges crossing between the two sides of our bipartition in order for our reduction to work. Recommendation strengths can be taken into account by our flow based methods, and we can find the graph that has the highest total recommendation quality given a discrepancy value using an extra pass with a flow solver. This can be done accomplished as follows: first, we solve the regular discrepancy problem, and finding the lowest discrepancy value OPT attainable by the underlying G. Knowing this value, we can now fix the flow between t_1 and t_2 in the original flow network to $lc - OPT$, where lc is the total out-degree of the subgraph from L. This constrains the flow solver to choose subgraphs where exactly OPT of the recommendations go over the charged edges. We then keep all the other capacities the same and add new nonzero weights reflecting recommendation quality while removing all other costs. In a second pass, we find the highest cost flow in this network, which corresponds to the recommender graph with OPT discrepancy with the highest total recommendation quality. Therefore, we can solve the MAX-WEIGHT-MIN-DISCREPANCY problem with only two calls to a minimum cost flow solver. We call this approach the two-pass method[1]. Maximizing average recommendation quality in this fashion corresponds to the finding the recommendation subgraph with the highest cumulative gain.

In some cases, recommendation algorithms do not give each recommendation a score that falls in a uniform interval, and instead rank the resulting recommendations among themselves. In this case, we cannot use the average recommendation quality as a measure of the quality of our recommendations. An appropriate measure of quality in this

[1]This follows the goal-programming methodology for two-objective functions, popular in Operations Research.

case is the precision-in-top-c metric, where the quality of the recommendations made to the user u_i is measured as $|N(u_i) \cap B_k(u_i)|/c$, where $B_k(u_i)$ is the list of top-k recommendations for the user u. When c recommendations are made for each user, the average recommendation quality of the system is

$$\sum_{u_i \in L} \frac{|N(u_i) \cap B_k(u_i)|}{c} = \frac{1}{c} \sum_{u_i \in L} |N(u_i) \cap B_k(u_i)|$$

Note that the quantity inside the sum is simply a linear function of the recommendations made by the subgraph: recommendation edges which are in the top-k for a user u have weight 1, while every other edge has weight 0. Therefore, we can optimize the average of the accuracy-in-top-k using a flow model as well. The objective we maximize in this setting corresponds to the average cumulative gain in the binary relevance setting.

3.5.2.2 Discounted Cumulative Gain

Not every recommendation in a list is considered equally valuable, and the value of recommendation slot depends on its rank among the presented recommendations. Discounted cumulative gain accomplishes this by weighing the relevance of the jth slot by $1/\log(j+1)$. That is, if we let $v_1, \ldots, v_c$ be the recommendations made to user u_i and $rel(u_i, v_j)$ the relevance of the jth recommendation

$$CDG(u_i) = \sum_{j=1}^{c} \frac{rel(u_i, v_j)}{\log(j+1)}$$

We first show how to maximize CDG in the binary relevance case, which has a simpler construction than the general case.

Theorem 3.3. *The recommendation graph having the minimum discrepancy from a target distribution $\{a_i\}_{i=1}^{r}$ while having the highest cumulative discounted gain in the binary relevance setting and can be computed with two invocations to a minimum cost flow solver.*

Proof. We use the construction in Theorem 3.1 as our starting point and set the cost of each arc to 0. We fix the flow between t_1 and t_2 to $lc - OPT$ to constrain the solver to solutions which have the desired discrepancy as discussed above.

Note that when k relevant recommendations are made to a user, then the resulting discounted cumulative gain is $\sum_{j=1}^{k} \frac{1}{\log(j)}$. In order to be able to charge this quantity

in our flow model, we create create an intermediary node $n_{u,c}$. Every recommendation (u, v) which has binary relevance 1, now connects $n_{u,c}$ to v instead of u and c. We also connect $n_{u,c}$ to node u by using c parallel arcs with costs $-1, -1/\log(2), \ldots, -1/\log(c)$, each of capacity 1. This modification only adds an extra node and c arcs for each user. Furthermore, the cost of the flow is the negation of the CDG function summed across all users, and this can be minimized with a single invocation of a min-cost flow solver. □

The rationale behind discounting the value of later recommendations is based on a a model of the user's consumption of the recommendations: later recommendations are less likely to receive attention from the user, which diminishes their usefulness. However, the discounting serves another beneficial purpose in our model. In particular, the value of making k relevant recommendations in the binary relevance model is

$$\sum_{j=1}^{k} \frac{1}{\log(j)} = \Theta\left(\frac{k}{\log(k)}\right)$$

This function grows slower than linearly, which means that there are diminishing returns as more and more relevant recommendations are made to the same user. It is therefore, more advantageous to make a second relevant recommendation for a user u than to make a tenth relevant recommendation to a user u'. This can be used to ensure that no user gets a disproportionate share of irrelevant recommendations.

Theorem 3.4. *The recommendation graph having the minimum discrepancy from a target distribution $\{a_i\}_{i=1}^{r}$ while having the highest cumulative discounted gain can be computed with two invocations to a minimum cost flow solver.*

Proof. We use the construction in Theorem 3.1 as our starting point and set the cost of each arc to 0. We fix the flow between t_1 and t_2 to $lc - OPT$ to constrain the solver to solutions which have the desired discrepancy as discussed above.

Since cumulative discounted gain depends on the ranking of the recommendations made, we create c nodes for each user $n_{u,1}, \ldots, n_{u,c}$. We connect each of these to the user node u by an arc of cost 0 and capacity 1. For each candidate recommendation $(u, v) \in G$ we create c arcs in total. For each $1 \leq j \leq c$, the node $n_{u,j}$ is connected to node v by a cost $-rel(u, v)/\log(j + 1)$ capacity 1 arc. The rest of the construction remains the same, and these modifications add $|V|$ vertices to the construction, as well as $(c-1)|E|$ additional arcs since every recommendation edge now has c copies instead of just 1.

In a feasible solution, no more than c recommendations can be made to a user because the total capacity of the arcs coming out of node u is c. The single unit of flow that

is allowed to leave $n_{u,j}$ discounts the value of this recommendation by $\log(j+1)$ due to the way we set the costs. The sum of these contributions gives us the negation of the cumulative discounted gain for the system, which can be minimized with one extra min-cost flow invocation. □

Table 3.2 summarizes the number of arcs and nodes in each of our constructions.

	Arcs	Nodes
Cumulative Gain (Binary)	$2\|R\| + \|E\| + 1$	$\|L\| + \|R\| + 2$
Cumulative Gain	$2\|R\| + \|E\| + 1$	$\|L\| + \|R\| + 2$
Cumulative Discounted Gain (Binary)	$(c+1)\|L\| + 2\|R\| + \|E\| + 1$	$2\|L\| + \|R\| + 2$
Cumulative Discounted Gain	$\|L\| + 2\|R\| + c\|E\| + 1$	$c\|L\| + \|R\| + 2$

TABLE 3.2: The number of nodes and arcs in each of our difference relevance models. The non-discounted models are the most efficient, followed by the binary cumulative discounted model. The full discounted gain model is likely to be prohibitively expensive for most settings of c.

3.5.2.3 Bicriteria Optimization

In each of the constructions above, we needed to make an extra pass with a flow solver in order to find a solution with a high level of relevance. If we used these cost settings along with the cost settings we used in 3.1, we would no longer be optimizing only for ratings, or only for discrepancy. Instead, this results in a bicriteria objective of the form $\text{discrepancy}(H) - \mu \cdot \text{rel}(H)$, where μ can be any real number, and where relevance of a solution graph denotes the average relevance of the recommendations in H as predicted by the underlying CF recommender. We call this approach the weighted method, and demonstrate that while it is strictly worse than the two-pass method in theory, it yields acceptable results in practice while saving an extra pass of flow minimization. We discuss the performance differences in our experimental section.

3.5.3 Category Level Constraints

It is sometimes desirable to set multiple goals in an optimization problem. For example, a news website might have a target distribution for the articles in mind, but might also want to ensure that none of the different categories such as current events, politics, sports, entertainment, etc. are neglected. Alternatively, due to a payment from a sponsor, a vendor may wish to boost the profile of a specific subset of movies in the catalog, and might need to balance its own needs about the distribution of recommendations with their commitment to their sponsor.

This type of problem can be accommodated by our model in the following way. Let there be k categories $C_1, \ldots, C_k$ that partition the items in R with minimum targets $A_1, \ldots, A_k$ respectively. We will require have $A_t \leq \sum_{v_j \in A_t} a_j$, i.e., the category requirements are less stringent than the aggregate of the individual target requirements. For ease of notation, let $D_1, \ldots, D_k$ be the number of times an item from category $C_1, \ldots, C_k$ are recommended. In this setting, we can optimize the objective $\sum_{v_i \in R} |\deg_H^-(v_j) - a_j| + \sum_{i=1}^{k} \min(A_i - D_i, 0)$. Note that this objective is simply the discrepancy objective, plus another term which looks like discrepancy objective for categories. However, we must make an important distinction. The discrepancy objective penalizes low degree nodes both directly, and indirectly via the targets degrees for other nodes, since an extra recommendation for one node is one "stolen" from another node. The second term in our new objective penalizes low-degree categories, but does not necessarily penalize oversaturated categories.

Theorem 3.5. *Assume the conditions of theorem 1, and let $C_1, \ldots, C_k$ be a partition of L, with $A_1, \ldots, A_k$ an arbitrary sequence of non-negative integers. Then a flow network exists, whose flow cost equals the objective $\sum_{v_i \in R} |\deg_H^-(v_j) - a_j| + \sum_{i=1}^{k} \min(A_i - D_i, 0)$.*

Proof. The proof is similar to that of Theorem 3.1. For each category C_i, we create three nodes: t_1^i, t_2^i and t_3^i. The first two of these nodes are connected to the items in their category with arcs having the same costs/capacities as they did in the original network. Unlike the original network, where t_2^i would have non-zero demand, both nodes in this construction have zero demand for flow. We set the demand for t_i^3 to be the category constraint A_i, and connect it to t_2^i with an arc of capacity A_i and cost 0.

In addition to these nodes, we create two more nodes, common to all categories: a distributor node s_1 and a supersink s_2 that are common to all the categories. The distributor s_1 can accept any amount of flow from a node of type t_2^i at cost 1. It can also move any amount of flow to a node of type t_3^i at 0 cost. Finally, the supersink s_2 has demand $lc - \sum_{i=1}^{k} A_i$, and accepts unbounded flow from nodes of type t_3^i at no cost. Since we require that the category requirements sum to less than the total number of requirements, this node always has non-negative demand, and feasibility is ensured. Figure 3.4 illustrates the construction:

From the proof of Theorem 1, it follows that ignoring the cost of moving flow between nodes of type t_2 and t_3, the cost of the flow in this network is the discrepancy of the network from the target distribution. Therefore, we only need to account for the second term in our objective. The arc connecting t_2^i and t_3^i ensures that only A_i units of flow can go uncharged towards the sink t_3^i. Every other unit of flow must first go to the distributor, then to t_3^i to satisfy the demand of this node. Since the demand of this node

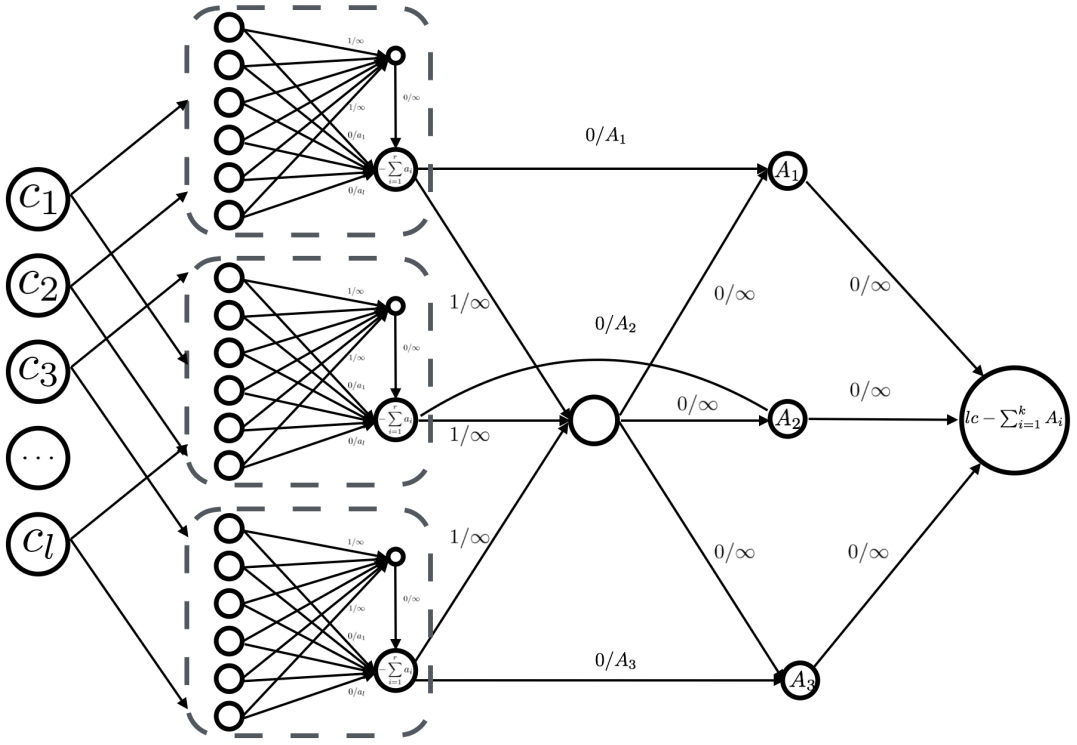


FIGURE 3.4: The network flow model for category targets with nodes labelled with their supply and arcs labeled with their cost/capacity. The central node with no supply or demand is the distributor s_1. The rightmost node is the supersink s_2.

is only A_i, we charge 1 unit of cost for every unit of flow we can't send to this category, but no cost for units of flow above the demand of this node. This gives us the term $\min(A_i - D_i, 0)$ for the contribution of this category to the objective, and completes the proof. □

We extend the results of this section in Chapter 4, where we present a more thorough analysis of the use category information in increasing sales diversity.

3.5.4 Greedy Algorithm

In this section we describe an alternative approach to solving the discrepancy minimization problem that does not require the use minimum cost flow solvers, which can be efficient in practice, but do not guarantee linear runtimes. Our greedy algorithm constructs the solution subgraph iteratively, making a discrepancy reducing recommendation whenever possible. If such an edge is not available, then we choose from all available recommendations. Our choice of recommendation is conditioned on the quality of this recommendation, as measured by our black-box relevance function *rel*.

Since the greedy algorithm considers all discrepancy reducing recommendations for a user at the same time, a large number of candidate recommendations may lead to the greedy algorithm making subpar selections, since almost every recommendation we consider early on will likely be a discrepancy reducing edge. In order to moderate this effect, we include a parameter $q > 1$ which we use to reweigh the relevance scores. The larger the q is, the more our greedy algorithm prefers making relevant recommendations. On the other hand, if we pick a q which is too large, then we overprioritize high relevance values, and the greedy algorithm effectively turns into the standard recommendation approach. To balance these concerns, we run the greedy algorithm with different settings of q, and select the solution with the highest predicted rating quality which has discrepancy at most 10% higher than the best solution we generate.

Data: $G = (L, R, E)$, the graph of candidate recommendations, a target indegree distribution $\{a_v\}_{v=1}^{r}$, tuning parameter q, and a relevance function $rel : L \times R \to [0, 1]$
Result: Solution subgraph H
$H \longleftarrow \emptyset$;
for $j = 1$ **to** c **do**
  foreach $u \in L$ **do**
    $D \longleftarrow \{v \in N_G(u) : \deg_H(v) < a_v\}$;
    if $D \neq \emptyset$ **then**
      Sample $e = (u, v)$ from D with $p(e) \propto rel(u, v)^q$;
    else
      Sample $e = (u, v)$ from $N_G(u)$ with $p(e) \propto rel(u, v)^q$;
    end
    $H \longleftarrow H \bigcup \{(u, v)\}$;
  end
end

Algorithm 4: The Greedy Algorithm for Discrepancy Minimization for a fixed value of q

3.6 Experiments

In this section, we put our model to the test. Our findings are summarized below, and we discuss each point further in the following subsections.

1. Our fast models perform well at optimizing for pre-existing notions of diversity such as aggregate diversity and the Gini index despite these measures not being explicitly referenced in our model. Conversely, we show that optimizing directly only for aggregate diversity (either by using heuristics or solving to optimality) does not yield results that are diverse by the other measures (See Tables 3.3, 3.4, 3.5).

2. Normalized discrepancy can often be reduced by more than 50%, at the cost of only a 15-30% change in average recommendation quality. Both the two-pass method, and the weighted method performed well in producing a smooth trade-off between recommendation quality and discrepancy reduction, with large gains in discrepancy being made for minimal recommendation quality loss (See Figure 3.6). The two-pass method is optimal, but the weighted-model provides a good approximation of two-pass method's output with less computational overhead.

3. Sales diversity maximization problems become easier as the display constraints are relaxed since there are more opportunities for the system to make unconventional recommendations. We show that the advantage our optimization based approach has over competing approaches gets bigger as display constraints are tightened, which is desirable for applications on mobile platforms where screen real estate is scarce (see Figure 3.5).

4. Using the uniform target distribution can lead the optimizer to pick subgraphs where degree constraints are violated by large margins at certain nodes. To remedy this, we advocate the use of target distributions that move towards resembling the underlying degree distribution rather than the uniform distribution.

Experimental Setup and Datasets. All of our experiments were conducted on a desktop computer with an Intel i5 processor clocked at 2.7GHz, and with 16GB of memory. We used three rating datasets to generate the graphs we fed to our flow solvers: MovieLens-1m, MovieLens-10m [76] and the Netflix Prize dataset.

We pre-process the datasets to ensure that every user and every item has an adequate amount of data on which to base predictions. This post processing leaves the MovieLens-1m data with 5800 users and 3600 items, the MovieLens-10m dataset with 67000 users and 9000 items, and the Netflix dataset with 8000 users and 5000 items. The use of these datasets is standard in the recommender systems literature. In this chapter, we consider the rating data to be triples of the form $(user, item, rating)$, and discard any extra information.

We used version 0.4.4 of the RankSys project to generate recommendations using standard collaborative filtering approaches [53]. The resulting network flow problems were optimized using a modified version of the MCFSimplex solver due to Bertolini and Frangioni [77]. Our choice of MCFSimplex was motivated by its open-source status and efficiency, but any other minimum cost flow solver such as CPLEX or Gurobi which accepts flow problems in the standard DIMACS format can also be used by our algorithms. Our discrepancy minimization code is available at Github.

Quality Evaluation. To evaluate the quality of our method, we employ a modified version of k-fold cross-validation. In particular, for each user in our datasets who has an high enough number of observed ratings, we divide the rating set into 10 equal sized subsets, and place each subset in one of 10 test sets. When creating the test sets, we filter out the items which received a rating of 1 or 2 and keep the items which received a rating of 3 or higher in order to ensure the relevance of our selections. We then define the precision of our recommendation list to be the number of items we recommend among all of our top-c recommendations which are also included in the test set. This provides an underestimate of the relevance of our recommendations, as there might be items which are relevant, but for which we have no record of the user liking. A simpler version of our hold-out method is utilized in other works [54, 68] where only a single random split is made. Using a 10-fold split of the test data enables us to run a signed rank test, and test whether the improvements made by our algorithms are statistically significant. This evaluation methodology will applied again in Chapter 4.

Our methodology stands in contrast with the methodology used by Adomavicius and Kwon to evaluate the effectiveness of their aggregate diversity maximization framework [51]. They use a metric called prediction-in-top-c, which measures the average predicted relevance of the c recommended items for each user. We believe that using predicted ratings for relevance evaluation purposes is flawed since these predictions are approximate in the first place. Furthermore, using the relevance values used by the recommender make comparisons across different recommenders difficult, as each recommender has its own scale.

Supergraph Generation. All of our optimization problems require that a supergraph of candidate recommendations be given. For each dataset we used, we generated 240 supergraphs in total. This is the result of using 10 training sets, 4 different recommender approaches, and 6 different quality thresholds enforced by picking the top 50, 100, 200, 300, 400 and 500 recommendations for each user. We use k to denote the number of candidate recommendations in the supergraph. The matrix factorization model we utilize [1] comes with three parameter settings: a regularization parameter λ, a confidence parameter α and the number of latent factors [1].

While the authors report an α value of 40 is suitable for most applications, we set a lower value of $\alpha = 30$ in order to obtain more diverse candidate recommendation lists. The regularization parameter λ was tuned with cross-validation as recommended by the authors, and the model was trained with 50 latent factors. For the neighborhood based methods, we consider neighborhoods of size 100 in both the item-based and user-based cases. For these recommenders we opted use the inverted neighborhood policy approach described in [68] in order to obtain more diverse candidate recommendation

lists. Instead of using the top 100 most similar items to an item i as the neighborhood, this approach uses the items which have item i in their top 100 neighborhoods. We also used Jaccard similarity in order to measure similarity between pairs of users and items in the neighborhood based methods.

The authors of the random walk recommender we implemented consider a parameter setting α which raises every element of the transition matrix to the power α and find that predictive accuracy is maximized for $\alpha = 1.5$ [54]. Since they conduct their experimental validation on the same datasets as ours, we also use this parameter in our tests. In our tables, we shorten the names of these recommenders as MF for the matrix factorization model, IB and UB for the (item- and user-) neighborhood based approaches, and RW for the random walk approach.

Trading Off Discrepancy and Rating Quality. The higher we set the number of candidate recommendations per user, the larger the input graph G will be, giving our algorithm more freedom to minimize discrepancy. On the other hand, in order to include more edges in G, we will have to resort to using more lower quality recommendations which will be reflected in our post-processed solution. Therefore, there is a trade-off between minimizing the discrepancy from the target distribution, and maintaining a high average recommendation quality. We will quantify this trade-off in our experiments and show that large discrepancy reductions can be made even with small compromises in recommendation quality.

Post-Processing CF for a Diversified Recommender. To summarize the discussion so far, we start with an arbitrary rating function which can predict the relevance of any item to a given user. We use this rating function to generate a large number of candidate recommendations which we encode in a weighted bipartite graph. We use this graph along with the designer-specified degree constraints on users and items to create a discrepancy minimization problem. Finally, we measure the quality of the resulting solution by comparing the predictive accuracy on the held-out test data.

3.6.1 Comparison To Other Methods and Metrics

In this section we compare our discrepancy minimization framework to other similar approaches. In particular, we test 6 different approaches to diversifying the recommendation lists.

- **Top(TOP)**: The standard method considers the unmodified output of the underlying recommender, and makes the top-k recommendations for each user. This is the undiversified solution but provides the highest rating quality.

- **Two pass (GOL)**: The two-pass method first finds the lowest discrepancy value achievable with the given graph for the current target degrees, and then in a second pass, finds the highest rating solution which achieves this minimum.
- **Aggregate Diversity (AGG)**: The aggregate diversity maximizing method is also optimized using our own flow-based framework, by running our min-cost flow algorithms with the setting of $a_i = 1$ as described in Theorem 3.2.
- **PC Reranking (PC), FD Reranking (FD) and Bayes Rule Reranking (AB)**: These diversifiers are due Vargas and Castells [46, 68], and were discussed in detail in our related work section.
- **Greedy (GRD)**: This is an implementation of the greedy heuristic described in Section 3.5.4.

We evaluate these different approaches on the following metrics, all measured for the top-n recommendation task on both the MovieLens and Netflix data.

- **D@n:** Discrepancy from the uniform distribution, normalized to fit in the $[0, 1]$ range by dividing by the maximum discrepancy achievable, i.e. $2\sum_{i=1}^{l} c_i$.
- **A@n:** The fraction of items which received a recommendation.
- **G@n:** The Gini index of the degree distribution of items. If the degree distribution of the items is given as a sorted list $\{d_i\}_{i=1}^{r}$, then the Gini index is defined as follows:

$$G = \frac{1}{r}\left(r + 1 - 2\frac{\sum_{i=1}^{r}(r+1-i)d_i}{\sum_{i=1}^{r} d_i}\right)$$

- **E@n:** The entropy of the probability distribution formed by normalizing this degree distribution. Given the same degree distribution as above, entropy is defined as follows:

$$E = \sum_{i=1}^{r} -\frac{d_i}{\sum_{i=1}^{r} d_i} \log\left(\frac{d_i}{\sum_{i=1}^{r} d_i}\right)$$

- **P@n:** Precision, measured as the fraction of items in the recommendation list which are part of the test set.

Table 3.4 summarizes our results for the Netflix dataset and Table 3.3 does the same for the MovieLens-1m dataset.

We start our discussion with the results on the two medium sized datasets. The first thing to notice in these tables is that the undiversified recommendation lists perform

		k=50					k=250					k=500				
		P@10	A@10	G@10	E@10	D@10	P@10	A@10	G@10	E@10	D@10	P@10	A@10	G@10	E@10	D@10
MF	TOP	**0.433**	0.263	0.924	5.974	0.823	**0.433**	0.263	0.924	5.974	0.823	**0.433**	0.263	0.924	5.974	0.823
	AGG	***0.433***	**0.429**	0.919	6.024	0.816	*0.432*	**0.651**	0.906	6.094	0.804	*0.432*	**0.758**	0.896	6.146	0.796
	FD	0.340	0.385	0.826	6.819	0.712	0.316	0.478	0.792	7.000	0.658	0.330	0.528	0.802	6.952	0.659
	PC	0.362	0.331	0.858	6.619	0.754	0.335	0.347	0.843	6.715	0.736	0.321	0.356	0.836	6.756	0.728
	AB	0.257	**0.429**	**0.767**	**7.081**	0.661	0.167	0.649	0.648	**7.504**	0.503	0.204	0.735	0.660	7.472	0.507
	GOL	0.414	0.423	0.855	6.451	**0.658**	0.358	0.639	**0.646**	7.189	**0.414**	0.331	0.754	**0.483**	**7.581**	**0.282**
	GRD	0.269	0.418	0.815	6.863	0.670	0.125	0.620	0.603	7.583	0.449	0.083	0.734	0.455	7.852	0.324
IB	TOP	**0.410**	0.257	0.953	5.450	0.861	**0.410**	0.257	0.953	5.450	0.861	**0.410**	0.257	0.953	5.450	0.861
	AGG	*0.409*	**0.431**	0.947	5.496	0.855	*0.409*	**0.653**	0.933	5.573	0.843	*0.409*	**0.813**	0.918	5.640	0.833
	FD	0.366	0.352	0.906	6.202	0.795	0.321	0.465	0.842	6.730	0.708	0.296	0.633	0.781	7.035	0.622
	PC	0.368	0.317	0.921	6.028	0.819	0.338	0.332	0.902	6.261	0.796	0.323	0.348	0.889	6.383	0.779
	AB	0.282	0.429	**0.861**	**6.610**	0.742	0.196	0.641	**0.735**	**7.231**	0.591	0.214	0.782	0.695	**7.299**	0.528
	GOL	*0.404*	0.380	0.911	5.834	**0.735**	0.380	0.573	0.763	6.611	**0.524**	0.358	0.733	**0.605**	7.160	**0.375**
	GRD	0.258	0.383	0.898	6.215	0.753	0.135	0.556	0.747	7.163	0.568	0.090	0.695	0.610	7.577	0.437
UB	TOP	**0.412**	0.146	0.971	5.002	0.903	**0.412**	0.146	0.971	5.002	0.903	**0.412**	0.146	0.971	5.002	0.903
	AGG	***0.412***	**0.308**	0.967	5.062	0.895	***0.412***	**0.585**	0.952	5.176	0.878	***0.412***	**0.777**	0.936	5.262	0.866
	FD	0.380	0.282	0.902	6.242	0.797	0.335	0.498	0.840	6.674	0.686	0.333	0.619	0.833	6.630	0.665
	PC	0.391	0.240	0.922	6.029	0.833	0.360	0.275	0.903	6.252	0.809	0.350	0.279	0.898	6.300	0.802
	AB	0.291	0.307	**0.861**	**6.592**	0.761	0.195	0.566	0.747	**7.158**	0.593	0.210	0.696	0.703	7.240	0.528
	GOL	**0.412**	0.306	0.928	5.530	**0.760**	0.382	0.582	**0.732**	6.618	**0.486**	0.342	0.772	**0.507**	**7.358**	**0.297**
	GRD	0.265	0.303	0.900	6.210	0.766	0.136	0.566	0.703	7.302	0.519	0.091	0.746	0.519	7.757	0.353
RW	TOP	0.303	0.072	0.992	3.710	0.970	0.303	0.072	0.992	3.710	0.970	0.303	0.072	0.992	3.710	0.970
	AGG	*0.304*	**0.262**	0.989	3.786	0.960	*0.302*	**0.515**	0.976	3.909	0.943	0.297	**0.670**	0.967	3.965	0.936
	FD	0.343	0.232	0.964	5.203	0.898	0.289	0.466	0.911	5.968	0.780	0.284	0.523	0.908	5.792	0.761
	PC	**0.351**	0.205	0.967	5.147	0.905	**0.339**	0.364	0.933	5.819	0.829	**0.348**	0.404	0.929	5.842	0.819
	AB	0.287	0.261	**0.957**	**5.409**	0.889	0.209	0.488	**0.881**	**6.429**	0.766	0.183	0.550	0.846	**6.625**	0.705
	GOL	*0.319*	0.228	0.980	4.122	**0.887**	*0.302*	0.489	0.896	5.177	**0.689**	0.248	0.633	**0.780**	5.980	**0.536**
	GRD	0.213	0.238	0.966	5.129	0.889	0.130	0.478	0.856	6.627	0.708	0.094	0.613	0.743	7.206	0.574

TABLE 3.3: Comparison of different diversifiers systems on various diversification metrics for the 10 item recommendation task. The underlying dataset is the MovieLens-1m dataset and the candidate recommendations were generated using Matrix Factorization (MF), Item-Based (IB), User-Based (UB), and Random Walk Recommenders (RW). The bolded entries show the best values in each metric (ignoring the greedy algorithm), and italicized values show a statistically insignificant change from the baseline with $p < 0.01$.

		k=50					k=250					k=500				
		P@10	A@10	G@10	E@10	D@10	P@10	A@10	G@10	E@10	D@10	P@10	A@10	G@10	E@10	D@10
MF	TOP	**0.734**	0.364	0.893	6.311	0.771	**0.734**	0.364	0.893	6.311	0.771	**0.734**	0.364	0.893	6.311	0.771
	AGG	0.733	**0.515**	0.888	6.351	0.765	0.730	**0.743**	0.873	6.426	0.754	0.726	**0.870**	0.863	6.459	0.748
	FD	0.568	0.476	0.803	6.952	0.665	0.448	0.571	0.757	7.162	0.607	0.442	0.634	0.736	7.246	0.583
	PC	0.589	0.432	0.826	6.829	0.695	0.510	0.442	0.818	6.874	0.685	0.475	0.453	0.815	6.889	0.681
	AB	0.472	0.494	**0.765**	**7.122**	0.649	0.215	0.701	0.665	**7.480**	0.524	0.200	0.836	0.625	7.594	0.475
	GOL	0.694	0.503	0.812	6.759	**0.611**	0.554	0.722	**0.578**	7.452	**0.360**	0.440	0.860	**0.359**	**7.870**	**0.199**
	GRD	0.525	0.495	0.785	7.031	0.626	0.243	0.697	0.566	7.688	0.411	0.150	0.826	0.388	7.975	0.267
IB	TOP	0.681	0.079	0.984	4.436	0.954	0.681	0.079	0.984	4.436	0.954	**0.681**	0.079	0.984	4.436	0.954
	AGG	*0.682*	**0.133**	0.984	4.455	0.952	**0.682**	0.216	0.982	4.485	0.948	**0.681**	0.219	0.982	4.486	0.948
	FD	0.650	0.127	**0.948**	**5.584**	0.912	0.454	0.265	**0.881**	**6.437**	0.799	0.281	0.476	0.842	**6.712**	0.697
	PC	0.650	0.124	0.950	5.547	0.913	0.532	0.212	0.912	6.143	0.843	0.474	0.303	0.897	6.321	0.808
	AB	0.532	0.125	0.967	5.177	0.924	0.449	0.196	0.950	5.586	0.885	0.484	0.329	0.927	5.751	0.811
	GOL	**0.685**	0.130	0.979	4.635	**0.908**	0.670	**0.276**	0.943	5.207	**0.782**	0.620	**0.488**	**0.834**	6.041	**0.600**
	GRD	0.476	0.130	0.955	5.460	0.909	0.263	0.272	0.865	6.530	0.786	0.201	0.480	0.750	7.167	0.621
UB	TOP	0.701	0.101	0.976	4.863	0.931	0.701	0.101	0.976	4.863	0.931	0.701	0.101	0.976	4.863	0.931
	AGG	*0.702*	**0.192**	0.975	4.896	0.926	***0.703***	**0.433**	0.974	4.915	0.924	***0.702***	**0.652**	0.974	4.915	0.924
	FD	0.631	0.190	0.936	5.830	0.860	0.456	0.385	0.868	6.543	0.740	0.380	0.552	0.835	6.721	0.677
	PC	0.632	0.186	0.938	5.805	0.864	0.538	0.282	0.902	6.262	0.807	0.506	0.302	0.894	6.339	0.792
	AB	0.561	0.184	**0.933**	**5.888**	0.867	0.380	0.367	**0.866**	**6.565**	0.752	0.385	0.505	0.841	6.620	0.685
	GOL	***0.705***	**0.192**	0.961	5.183	**0.851**	0.658	0.419	0.873	5.963	**0.659**	0.563	0.630	**0.710**	**6.728**	**0.466**
	GRD	0.522	0.192	0.933	5.897	0.851	0.284	0.409	0.804	6.944	0.671	0.193	0.602	0.656	7.477	0.500
RW	TOP	0.615	0.027	0.990	3.901	0.979	**0.615**	0.027	0.990	3.901	0.979	**0.615**	0.027	0.990	3.901	0.979
	AGG	0.615	**0.069**	0.990	3.914	0.977	**0.615**	**0.194**	0.988	3.960	0.971	0.612	**0.329**	0.985	4.008	0.963
	FD	*0.659*	0.068	**0.966**	5.156	**0.941**	0.496	0.154	0.954	5.486	0.889	0.489	0.187	0.958	5.327	0.877
	PC	**0.661**	0.067	**0.966**	**5.157**	0.942	0.544	0.179	**0.930**	**5.928**	0.857	0.530	0.267	0.909	**6.190**	0.809
	AB	0.525	0.055	0.980	4.652	0.957	0.462	0.094	0.972	4.955	0.928	0.491	0.137	0.970	4.898	0.901
	GOL	*0.622*	**0.069**	0.988	4.042	**0.941**	0.611	0.193	0.963	4.529	**0.823**	0.568	0.327	**0.908**	5.125	**0.703**
	GRD	0.439	0.068	0.965	5.158	0.941	0.249	0.191	0.880	6.366	0.825	0.182	0.319	0.797	6.907	0.713

TABLE 3.4: Comparison of different diversifiers systems on various diversification metrics for the 10 item recommendation task. The underlying dataset is the Netflix dataset and the candidate recommendations were generated using Matrix Factorization (MF), Item-Based (IB), User-Based (UB), and Random Walk Recommenders (RW). The bolded entries show the best values in each metric (ignoring the greedy algorithm), and italicized values show a statistically insignificant change from the baseline with $p < 0.01$.

		k=50					k=250					k=500				
		P@10	A@10	G@10	E@10	D@10	P@10	A@10	G@10	E@10	D@10	P@10	A@10	G@10	E@10	D@10
MF	TOP	**0.451**	0.817	0.959	6.063	0.901	**0.451**	0.817	0.959	6.063	0.901	**0.451**	0.817	0.959	6.063	0.901
	AGG	***0.451***	**0.849**	0.955	6.094	0.881	0.447	**0.890**	0.947	6.124	0.873	0.446	**0.913**	0.939	6.147	0.868
	FD	0.361	0.831	0.925	6.881	0.852	0.355	0.840	0.911	7.069	0.820	0.374	0.849	0.921	6.912	0.828
	PC	0.382	0.823	0.939	6.639	0.876	0.353	0.823	0.933	6.747	0.868	0.341	0.819	0.930	6.798	0.864
	AB	0.267	0.837	**0.899**	**7.213**	0.829	0.188	0.867	**0.837**	**7.681**	0.710	0.205	0.887	0.834	**7.655**	0.682
	GOL	*0.439*	0.835	0.919	6.486	**0.822**	0.417	0.851	0.853	7.224	**0.645**	0.382	0.860	**0.792**	7.541	**0.556**
	GRD	0.275	0.834	0.926	6.837	0.832	0.128	0.859	0.847	7.653	0.707	0.087	0.880	0.775	8.041	0.617
IB	TOP	**0.388**	0.828	0.968	5.543	0.910	**0.388**	0.828	0.968	5.543	0.910	**0.388**	0.828	0.968	5.543	0.910
	AGG	*0.387*	**0.876**	0.966	5.615	0.905	*0.386*	**0.932**	0.959	5.654	0.903	*0.386*	**0.970**	0.951	5.684	0.899
	FD	0.359	0.844	0.953	6.160	0.879	0.338	0.873	0.920	6.842	0.812	0.306	0.911	0.898	6.938	0.759
	PC	0.358	0.838	0.959	5.972	0.891	0.331	0.842	0.953	6.198	0.882	0.321	0.847	0.949	6.306	0.876
	AB	0.275	0.859	**0.939**	**6.549**	0.856	0.232	0.915	**0.866**	**7.412**	0.745	0.231	0.960	0.823	**7.557**	0.661
	GOL	0.362	0.844	0.941	5.934	**0.853**	0.354	0.901	0.897	6.873	**0.659**	0.335	0.949	**0.820**	7.301	**0.580**
	GRD	0.239	0.844	0.950	6.257	0.863	0.131	0.880	0.892	7.206	0.749	0.095	0.915	0.819	7.744	0.640
UB	TOP	**0.440**	0.802	0.970	5.483	0.919	**0.440**	0.802	0.970	5.483	0.919	**0.440**	0.802	0.970	5.483	0.919
	AGG	***0.440***	**0.853**	0.968	5.619	0.919	*0.439*	**0.927**	0.954	5.681	0.909	0.437	**0.978**	0.941	5.732	0.893
	FD	0.387	0.828	0.938	6.596	0.851	0.344	0.904	0.886	7.166	0.749	0.337	0.934	0.867	7.201	0.718
	PC	0.399	0.817	0.951	6.308	0.879	0.373	0.838	0.943	6.494	0.867	0.365	0.832	0.942	6.539	0.864
	AB	0.317	0.832	**0.922**	**6.918**	0.834	0.251	0.912	**0.827**	**7.594**	0.666	0.237	0.951	0.737	**7.899**	0.548
	GOL	0.412	0.842	0.930	6.145	**0.833**	0.398	0.916	0.845	7.185	**0.592**	0.372	0.939	**0.723**	7.621	**0.467**
	GRD	0.279	0.831	0.940	6.472	0.835	0.154	0.907	0.836	7.553	0.652	0.115	0.942	0.716	8.087	0.509

TABLE 3.5: Comparison of different diversifiers systems on various diversification metrics for the 10 item recommendation task. The underlying dataset is the MovieLens-10m dataset and the candidate recommendations were generated using Matrix Factorization (MF), Item-Based (IB), User-Based (UB) recommenders. The bolded entries show the best values in each metric (ignoring the greedy algorithm), and italicized values show a statistically insignificant change from the baseline with $p < 0.01$.

very poorly with respect to all distributional measures. This is true with respect to even the simplest measure, aggregate diversity. The Random Walk Recommender in particular surfaces only 3% of the items in the Netflix catalog and 7% of the items in the MovieLens catalog. Other recommenders do not do particularly better, and only surface 15-20% of items to the users via top-10 recommendation lists.

We also note that an optimization based approach using aggregate diversity as the objective function is unlikely to make a large difference in the degree distribution of the underlying recommender system. Indeed, for all the recommenders and for both of the datasets, the aggregate diversity maximization approach only makes significant improvements with respect to its own metric. This is to be expected from a crude measure of system-level diversity such as aggregate diversity, as it is possible to have a very lopsided recommendation distribution even while achieving full coverage of the item catalog. As an extreme example, consider a recommender which seeks to make 2 recommendations per user, and suppose there are as many users as there are items. If the recommender recommends one unique item to each users, and one of the items to every user, aggregate diversity will be maximized. However, the Gini index will be high and the entropy of the degree distribution will be low. Aggregate diversity is still a valuable measure of the coverage properties of the recommender, as our baseline approaches do not achieve good values for even this simple metric. However, all of the other approaches we tested also make significant gains in aggregate diversity while also significantly impacting the other metrics at the same time. In particular, the Bayes Rule reranking approach and our own discrepancy minimization almost maximize aggregate diversity on the MovieLens-1m and Netflix datasets respectively. Therefore, we believe that the best approach to aggregate diversity maximization is to optimize it by proxy, by optimizing for a more refined measure of recommendation diversity.

As expected, our method performs the best with respect to minimizing discrepancy from the uniform distribution. However, the other methods we tested also make gains towards minimizing discrepancy, which lends credibility to the use of discrepancy from the uniform distribution as a metric for evaluating the system-wide diversity of recommendations.

Among the methods we tested, the most aggressive diversifier is the Bayes Rule reranker, which obtains the best scores with respect to every metric, in almost every instance. Among the metrics we tested, our method is best suited for optimizing for aggregate diversity, the Gini index, and entropy of the degree distribution, in that order. Even though the two-pass method does not always obtain the best diversity improvements, it does so at a very low predictive cost. Furthermore, as more edges are included in the list of permissible recommendations (the second column of Tables 3.3,3.4,3.5), the

advantage of the Bayes Rule reranker almost vanishes with respect to the Gini and aggregate diversity metrics, despite our recommender surfacing between 30% to 120% more relevant recommendations. The predictive accuracy of the Popularity Complement and Free Discovery reranking methods fall off at a slower rate than the Bayes Rule reranking method, but our method dominates those two methods in every metric except for entropy. When 500 candidate recommendations are included for each user, our recommender creates solutions with better diversity values and better precision values than all the other recommenders we tested, even when they were supplied with more accurate and shorter candidate recommendation lists.

Finally, we comment on the performance of the greedy discrepancy reducing technique. With a relatively small number of candidate recommendations per user like $k = 50$, the greedy algorithm can strike a good balance between discrepancy reduction and maintaining the relevance of recommendations. However, in this regime, the greedy diversifier is matched on all diversity metrics by the corresponding reranking based techniques, which also beat it in predictive accuracy. The greedy algorithm is much quicker to improve diversity as more candidate recommendations are added; However, in this regime the Bayes Rule Reranker matches it well in diversity measures while beating it in predictive accuracy. Since the predictive losses are too great, the greedy algorithm is an inadequate replacement for the full flow based methods.

3.6.1.1 Large Dataset

The large dataset we tested deserves a special discussion for two reasons. First, the MovieLens-10m dataset has 7 times as many users as it has items in its catalog, compared to only a 1.5 times ratio in the MovieLens-1m dataset. With 70 times as many recommendations as items, even the standard recommendation approach achieved 80% coverage of the item catalog. This renders aggregate diversity maximization ineffective as a method for increasing the equitability of the recommendation distribution. Second, the large dataset is too large to solve in one batch by our flow methods. Therefore for these experiments, we split the user base into 4 batches, find the minimum discrepancy graph in each of these problems and combine them. Naturally, this leads to a higher discrepancy and lower precision than solving the entire problem in one go. Despite these factors, our results for the precision, Gini index and entropy measurements follow the trends found in the medium sized datasets.

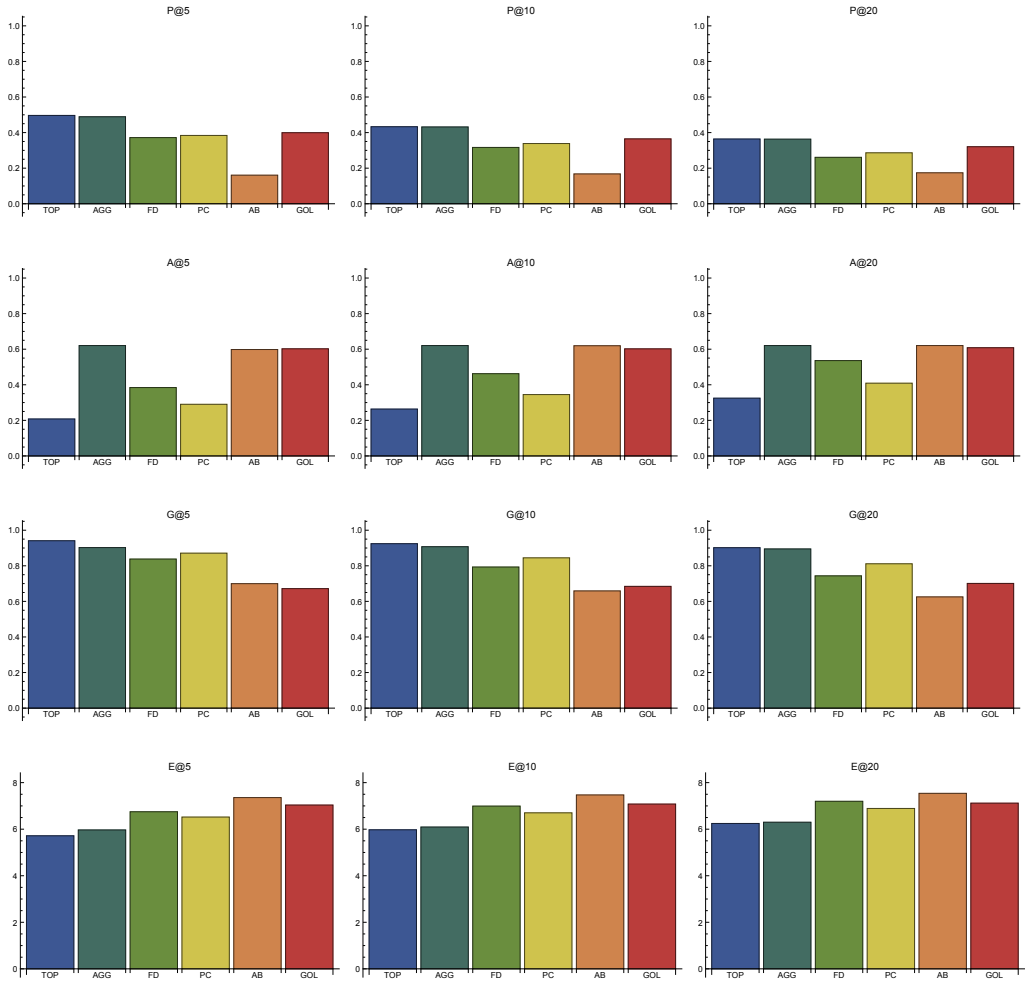


FIGURE 3.5: The effect of recommendation list length on distributional diversity measures in the MovieLens Matrix Factorization graph with 200 candidate recommendations.

3.6.2 Effect of Recommendation List Length

With more computer usage shifting from devices with larger displays like desktops and laptops to mobile devices like phones and tablets, display constraints that govern the number of recommendations we can make to user have gotten tighter. Therefore, it has become increasingly important for diversification approaches to be effective even when there is space for only a few recommendations to be made. In the set of graphs below, we fix the underlying subgraph to be the graph generated by our Matrix Factorization recommender with 200 candidate recommendations for each user, and measure the performance of the different diversifiers with display constraints set to $c = 5, 10$ and 20.

We note that our diversifier performs better as the display constraints get tighter. All approaches suffered precision loss as display constraints were tightened, which is to be expected. The lowest level of losses came from the Bayes Reranking approach, but this

approach had such low baseline precision that all the other algorithms kept outperforming it even as recommendation lists got longer. Moreover, our two-pass method increased both its absolute and relative edge over the Free Discovery and Popularity Complement diversifiers as c was set lower.

Our two-pass method also performed better in increasing diversity metrics when the (initial) recommendation lists were shorter. In particular, there was no significant changes in the aggregate diversity achieved by our algorithms as c was varied. This was also the case for Bayes Rule Reranking approach, but not for the novelty based reranking approaches FD and PC. These approaches would have needed lists of size $c = 50$ to achieve comparable aggregate diversity values, at which point the precision of these diversifiers suffer considerably.

For the top-5 recommendation task, our method effectively outperformed all other diversifiers. Our two-pass method loses out to the aggregate diversity and the baseline approaches in precision, which achieve low diversity values. It loses out to aggregate diversity maximization when considering the A@5 metric, but only by a small and statistically insignificant margin. We also achieve the highest G@5 value, while beating the runner-up Bayes Rule Reranker significantly in precision. We only achieve the second highest E@5 value, but once again beat our main competition in the Bayes Rule reanker in precision. Considering all of these factors, we conclude that an optimization based framework works better in applications where display constraints are particularly tight. The reranking approaches make recommendations for each user independently, whereas our optimization framework makes all of these recommendations at once, while optimizing explicitly for a diversity metric. Therefore, our two-pass method is better able to coordinate a small number of recommendations to make large gains in diversity while also keeping precision high.

3.6.3 Trading Off Discrepancy and Precision

In this section we explore the discrepancy/precision trade-offs made by our different models. Throughout the section, we consider the discrepancy from the uniform target distribution. This target indegree distribution sets $a_j = c \cdot l/r$ for each $v_j \in R$ for a fixed c which represents the display constraints. The discrepancy from this target can be thought of as an extreme measure of diversity, since we are measuring distance from the most equitable distribution.

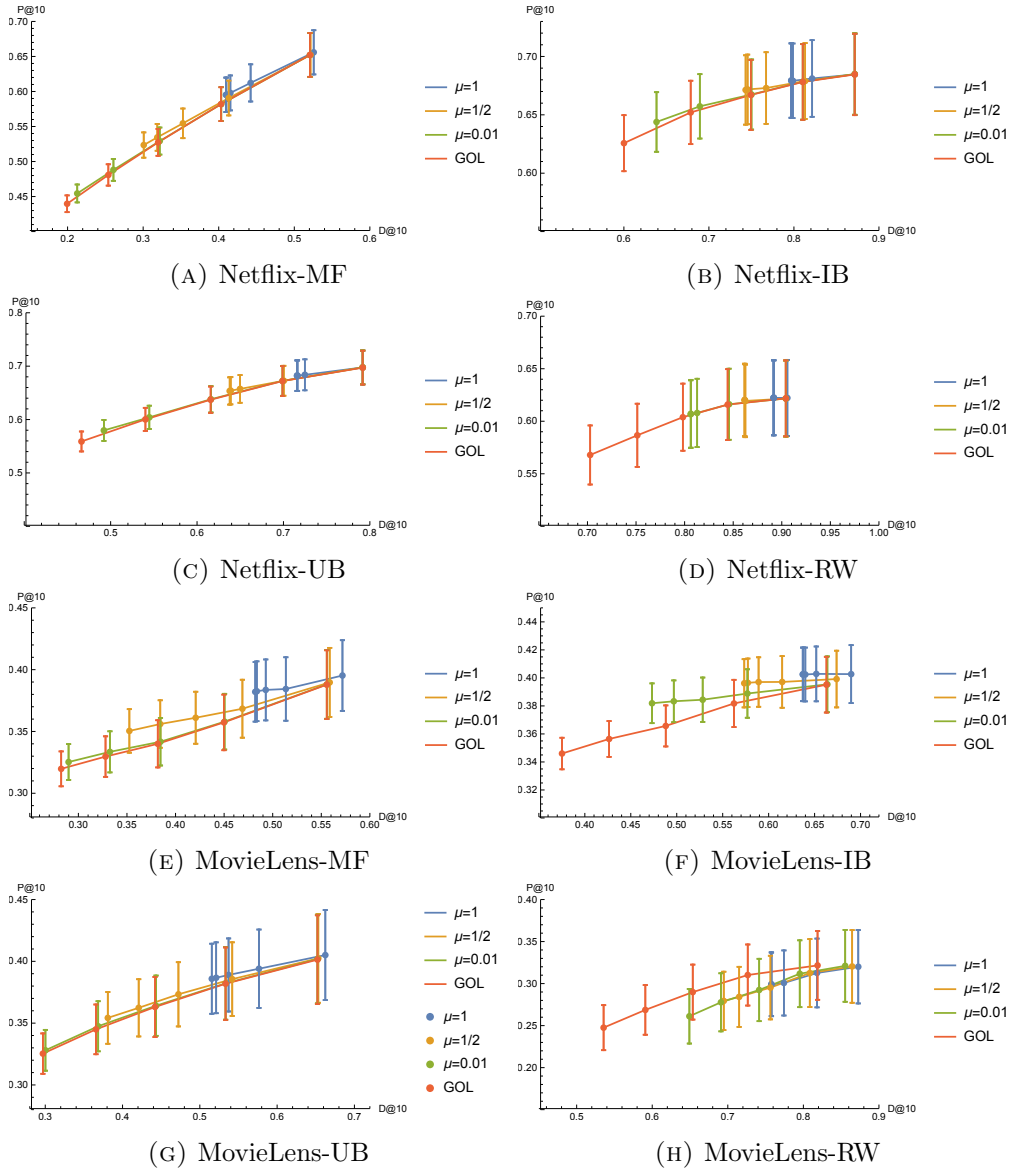
For the set of comparison graphs in Figure 3.6, we increase the number of candidate recommendations for each user from 100 to 500 in increments of 100, and show how

this affects the recommendation quality of our solution as well as the lowest discrepancy achievable. We plot the normalized discrepancy to the uniform target on the x-axis against precision in the y-axis. Discrepancy improves towards the left, and recommendation quality improves towards the top.

We consider 4 different approaches to reducing discrepancy. The first is our two-pass method, which optimizes for discrepancy first, then for average predicted relevance across the system. We also consider the weighted method with the settings $\mu = 1$, and $\mu = 1/2$ and $\mu = 0.01$. Recall that the weighted method optimizes the objective $\text{discrepancy}(H) - \mu \cdot \text{rel}(H)$, where $\text{rel}(H)$ denotes the average recommendation quality in the solution graph H we produce. Therefore, the weighted method does not optimize for discrepancy. Instead, it find a solution where the cost of reducing discrepancy by μ units is the same as reducing aggregate predicted relevance by 1 unit. When run on the same input graph, the predicted relevance of the two-pass method is always lower than the predicted relevance of the $\mu = 1$ model, and the predicted relevance of the weighted method with μ is always lower than the predicted relevance of the weighted method with $\mu' > \mu$. The discrepancies produced by these algorithms on the same graph are also ordered in the same way.

From these graphs we can immediately notice certain features. First, all three algorithms produce highly clustered data with initial normalized discrepancy from the uniform distribution always being over 0.8. Second, the fall-off in discrepancy happens first quickly, then slowly as more edges are included. Therefore, significant gains are possible even while including a small number of candidate recommendations. Third, the choice of recommender matters. It is harder to improve the discrepancy on graphs generated by the Random Walk recommender than on any of the other other graphs we generated.

Fourth, with a high enough number of edges, discrepancy to the uniform target can be driven down towards 0. However, this may come at significant quality loss in recommendation quality depending on the underlying graph. In particular, after the addition of 500 candidate recommendations for each user, we were able to reduce discrepancy from the uniform distribution on every one of the graphs except for the Random Walk Recommender by nearly 50% over the baseline. This drops the number of relevant recommendations surfaced by our method by 15%-30%. These precision losses are smaller than the losses produced the reranking diversifiers we tested, even when they were supplied with a shorter candidate recommendation lists which contain more relevant recommendations (see Table 3.3). Nonetheless, this level of loss in precision is not insignificant, and we suggest that users of our methods explore the full range of the trade-offs possible before putting the algorithms into commercial use.



(A) Netflix-MF (B) Netflix-IB

(C) Netflix-UB (D) Netflix-RW

(E) MovieLens-MF (F) MovieLens-IB

(G) MovieLens-UB (H) MovieLens-RW

FIGURE 3.6: Recommendation quality vs normalized discrepancy from the uniform target in MovieLens and Netflix generated graphs. In each series, the number of edges in the input graph increases towards the left.

Two-pass versus Weighted Methods. One notable difference between the weighted methods and the two-pass method is that in the weighted method, the discrepancy improvements start slowing down as more and more candidate edges are included in the supergraph. As mentioned above, the bicriteria objectives improve when discrepancy gains can be made which offset the fall in predicted relevance. As we enlarge the candidate set of recommendations, we enable discrepancy increases with edges that are less and less able to make up for the corresponding relevance losses. Therefore, the solution graph stops changing though lower discrepancies are possible. Where this limiting point lies depends on the structure of the graph and the distribution of relevance values assigned by the underlying recommender and is not easy to predict. We find that the

	d_0	$d_{0.2}$	$d_{0.4}$	$d_{0.6}$	$d_{0.8}$	d_1
d_0	0	0.17	0.38	0.59	0.81	0.86
$d_{0.2}$	0.17	0	0.21	0.42	0.64	0.85
$d_{0.4}$	0.38	0.21	0	0.22	0.43	0.64
$d_{0.6}$	0.59	0.42	0.22	0	0.22	0.42
$d_{0.8}$	0.81	0.64	0.43	0.22	0	0.21
d_1	0.86	0.85	0.64	0.42	0.21	0

TABLE 3.6: Pairwise discrepancy between different target distributions in the top-10 recommendation task in the MovieLens-1mItem-Based recommender and thresholded to 300 candidate recommendations

Matrix Factorization and User Based recommenders were more amenable to bicriteria optimization than the Item Based or Random Walk Based recommenders. We also find that for suitably low values of μ, the weighted method can adequately approximate the output of the two-pass method, achieving essentially the same trade-off curves.

3.6.4 Qualitative Parameter Tuning: Choice of Target Distribution

It might not be realistic to set the same target indegree for every item since the popular items are probably popular for a good reason. In order to create alternative distribution, we will use a convex combination of two different distributions. The first distribution we use is f, the uniform target indegree distribution. We also generate a more skewed distribution p by taking the indegree distribution of the candidate supergraph, and normalizing it so that it sums to the same value as the flat distribution. This distribution, while more diversified than the top c distribution, should still be significantly unbalanced. Finally, we generate the distribution $d_\alpha = \alpha f + (1-\alpha)p$ to be used as our target. Since both f and p satisfy our feasibility criterion $\sum a_i = \sum c_j$, so does d_α for all $\alpha \in [0,1]$. Furthermore, for $0 \leq \alpha \leq 1$, this function smoothly interpolates between f and p and produces a non-negative vector of target indegree values.

The particular setting of α is not special, and provided there are enough edges in the underlying graph, the two-pass method can achieve significant improvements regardless of the distribution used. To demonstrate this, fix the candidate supergraph to be the graph generated by applying the Item Based recommender to the MovieLens-1m dataset and thresholding candidate recommendations to the top 300 recommendations. Table 3.6 below shows that as we vary α, we get significantly different target distributions in the top-10 recommendation problem.

Table 3.7 shows that our diversifier achieves significant normalized discrepancy reduction in each case with the same modest rating loss.

	d_0	$d_{0.2}$	$d_{0.4}$	$d_{0.6}$	$d_{0.8}$	d_1
ΔP@10	0.097	0.100	0.105	0.109	0.105	0.103
ΔD@10	0.300	0.351	0.413	0.464	0.475	0.480

TABLE 3.7: Rating loss vs reduction in discrepancy given different target distributions when compared with the top 10 recommendations, in the MovieLens-1m Item-Based recommender and thresholded to 200 candidate recommendations

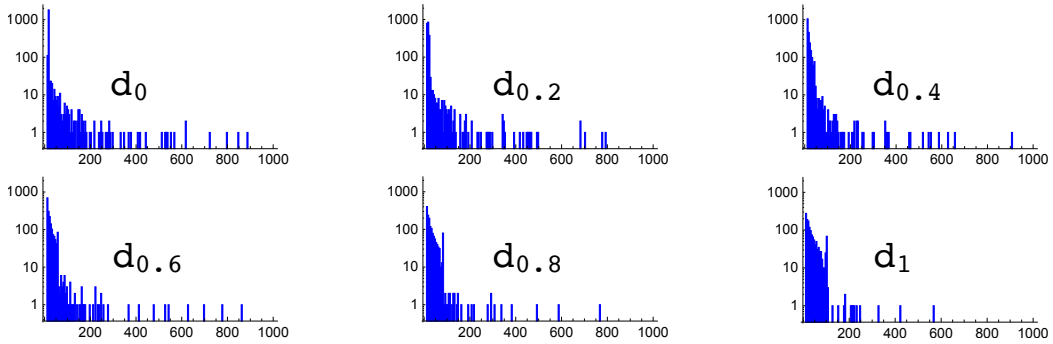


FIGURE 3.7: Degree distribution in a log-scale of the solution subgraphs as the α of the target distribution is varied in a top-10 recommendation task. The underlying supergraph is the MovieLens graph generated by Item Based recommender and thresholded to the top 200 recommendations. Note the presence of large outliers when the target distribution is close to uniform.

We have mentioned earlier that it is not always desirable to use the uniform target. In our modeling section, we gave a semantically motivated reason for this: a recommendation engine that recommends every item the same number of times would do a poor job of endorsing any item, even the ones that obviously deserve such endorsement. Here, we give an empirically motivated reason as well. When targeting the uniform distribution (d_0), discrepancy can do a poor job of producing a pleasing in-degree distribution at the items. We reduce the degrees of more vertices, more drastically this way, but this can mean that we produce a small group of vertices with really high indegree as well. In the Figure 3.7, we show the in-degree distributions of the items in the solutions produced by our two-pass method on the same underlying graph, but with varying target distributions. It demonstrates that the degree distribution is much smoother in the global sense when a proportional target is used, and that the number of outliers with very large degree are reduced.

3.6.5 Qualitative Parameter Tuning: Convex Cost Functions

As noted in the Algorithms section, we can change the way we charge for degree overruns and change the types of distributions that the flow model prefers. This can be used to remedy a particular behavior of the ℓ_1 norm, which is used in the discrepancy definition, to prefer sparse solutions. The flipside is that the majority of the contributions to

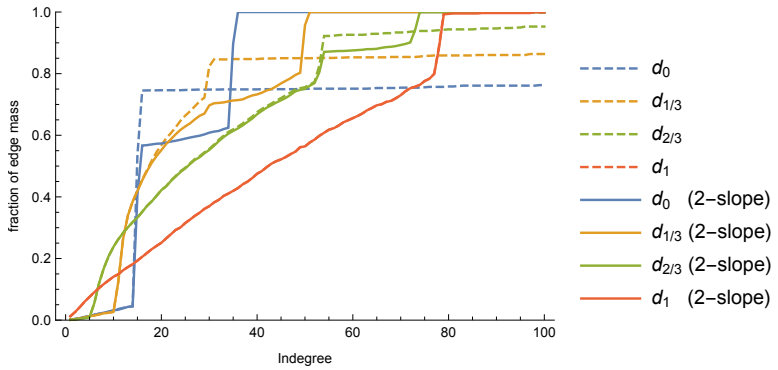


FIGURE 3.8: Cumulative degree distribution of the solution subgraph with different α and cost functions for a candidate supergraph on the Netflix data.

discrepancy come from relatively few vertices. Since the amount a vertex can contribute to discrepancy by undershooting its target indegree is bounded, these vertices must in fact be very high degree vertices. Therefore, while straightforward ℓ_1 norm minimization still has benefits in spreading degree more equitably among lower degree vertices, it does not necessarily cut down on the long tail. While it is possible to use another ℓ_p norm in the objective by approximating the costs by piecewise linear cost functions [78], common network flow solvers neither provide tools to convert arbitrary convex costs to piecewise linear costs, nor do they implement algorithms that can efficiently solve convex cost versions of flow problems.

We can overcome this problem by more strictly punishing degree overruns. To demonstrate this, we fix our attention to the supergraph generated on the Netflix data using User Based Filtering, thresholded to the top 300 recommendations. We then diversified the graph against the target indegree distributions $d_0, d_{1/3}, d_{2/3}, d_1$, as defined in the previous section. In one set of runs, we optimized our flow model directly for discrepancy. In another set of runs, labelled "2-slope" in the chart below, we added a second sink, which does not charge for the first a_v edges, charges a cost for the next 20 edges, and charges double the cost for any edges beyond that. The name is due to the fact that the cost of a network edge increases first with slope 1, then with slope 2, as opposed to the usual setting where it increases with slope 1 throughout. On the x-axis, we put the number of times an item is recommended. On the y-axis, we show the fraction of recommendations in the system which were made to items with fewer than a given number of recommendations.

As predicted, the 2-slope runs are much better at cutting down on the long tail. Solutions optimized for regular discrepancy tend to satisfy the indegree requirement of more vertices exactly. So when α is small, and the target distribution close to uniform, there tends to a large bump in the degree distribution near the average indegree of the target

distribution. When the extra sink is added, it causes a secondary bump. The location of this bump can be seen to be 20 units higher than the first bump, showing the reluctance of the optimizer to go above the threshold. Secondary bumps like this cause more of the long tail to be subsumed under given thresholds. However, it should be noted that this effect is not true across all target distributions. For example, in Figure 3.8, we can see that there is no difference between the 2-slope and regular runs against the d_1 target distribution. The reason for this is that when α is high, the target indegrees can all be met exactly, leading to no difference between the solutions produced by the two different schemes.

3.6.6 Resource Use

Since our methods are based on minimum cost flow, the problems that result from our reduction can be solved efficiently. The two graphs below show the cost of optimizing for uniform discrepancy with the two-pass method and the weighted method in the MovieLens-1m and Netflix graphs. We increase the number of candidate recommendations from 100 to 500, and report the average runtime across the different recommenders. The labels for this plot are identical to the labels we have been using so far, with the exception of WGT, which denotes a run of the weighted method. We did not find significant runtime differences between different settings of μ for this model, and present the results for a representative run with the setting $\mu = 1$. While our methods do not run as efficiently as reranking methods, they provide much better diversification, and even instances with tens of millions of candidate recommendations can be run on a desktop.

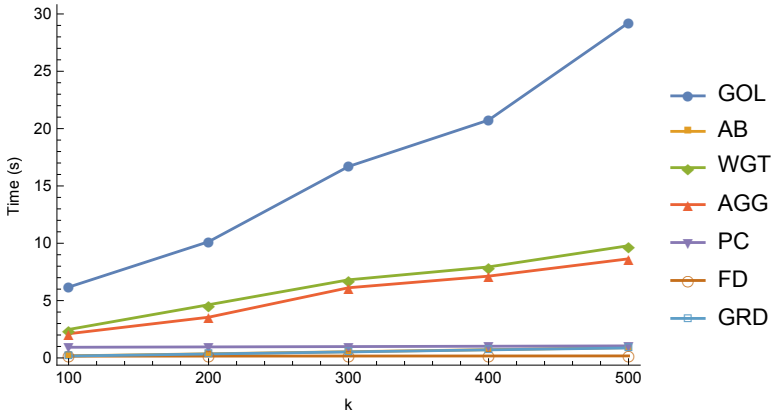


FIGURE 3.9: Time to optimize the top-10 recommendation task in MovieLens-1m based graphs in seconds ($|L|$=5800,$|R|$=3600)

We also note that the two-pass method takes noticeably longer than twice runtime the cost of the weighted method. It may be possible to improve this by a using a better implementation of the two-pass method, whereby the feasible solution found in the first

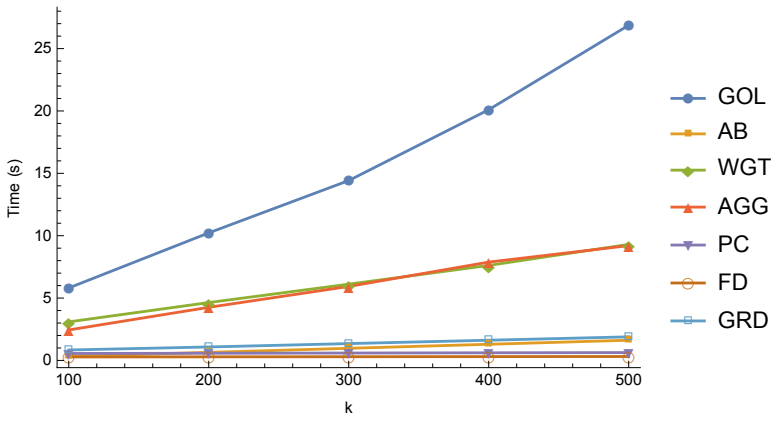


FIGURE 3.10: Time to optimize the top-10 recommendation task in Netflix based graphs in seconds ($|L|$=8000,$|R|$=5000)

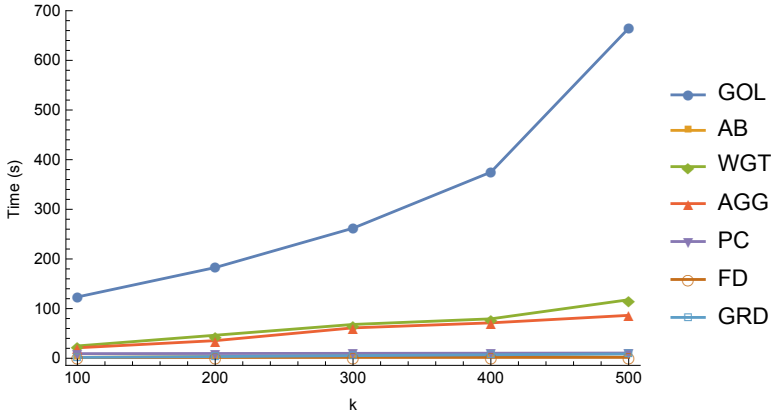


FIGURE 3.11: Time to optimize the top-10 recommendation task in Netflix based graphs in seconds ($|L|$=67000,$|R|$=9000)

pass is fed as a seed solution to the second pass. While we do not test this approach, we recommend its use to anyone who would like to use our framework in a commercial setting. Even if this improvement were to be made, the second pass of the two-pass method optimization is still a more challenging optimization problem than the first pass since it includes a stricter set of constraints. Moreover, the difficulty of the second pass is determined by how close the discrepancy target we set is to the lowest discrepancy achievable, which can lead to a significant increase in resource use. This problem can be dealt with in practice either by reducing the size of the discrepancy problems as mentioned in Section 3.6.1.1, or by using the weighted method instead of the two-pass method as described in Section 3.6.1.

3.7 Conclusions

We have proposed a new way of measuring how equitably a recommender system distributes its recommendations called discrepancy, and showed that it can be optimized for in polynomial time using network flow techniques. We validated the effectiveness and the efficiency of our method by conducting extensive tests on MovieLens and Netflix datasets, and showed it to improve diversity across a variety of measures. Our work demonstrates that distributional diversity measures like discrepancy can be efficiently optimized to allow information designers to have more control over their recommender systems.

Acknowledgements: This work was conducted jointly with my advisor, R Ravi.

Chapter 4

Category and Type Coverage

TABLE 4.1: Notation for Chapter 4

L	Set of users.
R	Set of items.
$\mathcal{L}$	Set of user types.
$\mathcal{R}$	Set of item categories.
L_b	A subset of the users denoting users of type b.
R_a	A subset of the items denoting items of belonging to category a.
$G(L, R, E)$	Bipartite graph of candidate recommendations.
H	A subset of G denoting the recommendations made by the system.
u_i	The i^{th} user.
v_j	The j^{th} item.
c_i	The display constraint for the i^{th} user.
a_j	The target degree for the j^{th} item.
$\lambda_j(L_b)$	The threshold for item j's recommendations from type b.
$\rho_i(R_a)$	The threshold for user i's recommendations from category a.
β	The relative weight of the user diversity term.
μ	The relative weight of the item diversity term.
$N(u_i)$	The set of c_i recommendations made to the i^{th} in H.
$T(u_i)$	The set of held-out recommendations for the i^{th} user in the test set.

4.1 Introduction

The traditional goal in the design of recommendation systems is the accuracy of predictions as measured by implied relevance of the recommended items. Often, such systems

operate on item catalogs and user bases that have natural clusterings into user types or item categories, by political affiliation, artistic genres, or product hierarchies. The single-minded focus on relevance fails to incorporate requirements of diversity of the recommendations among the item categories and item types. Our problem is motivated by three different considerations in incorporating such diversity in the design of recommender systems.

First, the individual users need to be presented with a diverse set of items. In real life datasets, users' interaction histories often show a broad interest in different categories of items. Unlike search tools, recommender systems do not necessarily have knowledge of the user's intent, and must hedge their bets on which types of items the user is interested in at a given point. Therefore, diversity of recommendations within a user's list a necessary consideration in making the system useful to the users. Moreover, this lack of knowledge about the user's intent provides an opportunity to broaden the user's taste profile by exposing her to previously unknown types of items. This motivates the incorporation of user-focused diversity metrics in optimization problems involving recommender systems.

The second consideration we look at is item-level diversity, that is, we wish to show each item to a diverse set of users. Each item usually has a standard type of user "audience" that it gets shown to based on those users' search queries and interaction history; horror movies are typically shown to users who display some sort of interest in that particular genre or theme. Similarly, certain brands of items on an e-commerce website are typically shown to customers who have purchased other items from that brand. If we show items to only that niche set of users, we are ultimately preventing those items from being discovered by new types of users and receiving new feedback. Restricting items to only their traditional "audience" also prompts the recommender to keep recommending canonically popular items, since those items reach a broader set of users, while failing to recommend items that are less popular or "long tail" items that have a more niche audience. This creates a "filter bubble" that allows the more popular items to keep getting popular while stunting the popularity and discoverability of less-viewed items [5]. Showing items to users that fall outside of their traditional "audience" of gives items the opportunity to collect new feedback (both positive and negative) from a more diverse set of users that could potentially help them be recommended to other users. User-level diversity fails to consider item-level diversity, since assigning recommendations based on user-satisfaction would still only show items to users who fall in their traditional "audience." This would result in bad item-level diversity but could still give high user-level diversity if recommendations are diverse enough.

Finally, the third consideration we have is system-level diversity, which involves aggregating the recommendations made to all users, and studying the resulting distribution of recommendations. The business running the recommender system often has concerns other than pure user satisfaction or item recommendation diversity, the two factors mentioned above. Examples of such concerns include achieving good coverage of the item catalog and avoiding the perpetuation of biases across the system such as popularity bias or filter bubbles. Despite the fact that these considerations are often studied under the same "diversity" umbrella, systems that optimize for user-level diversity do not necessarily score well in system-level diversity. For example, a recommender system that recommends the same set of 10 most diverse items achieves almost negligible catalog coverage when aggregated across all users. Conversely, a recommender that makes monotonous recommendations at the user-level can score very well in coverage-style metrics provided that the user base is diverse enough. Due to this lack of correlation between user-level diversity and system-level diversity, both measures must be explicitly factored into a recommender system in order for the system to score well in both dimensions.

Prior work has mainly focused on defining intent-aware metrics among categories and maximizing relevance of the resulting recommendations, which we review in the next section. Following that, we provide two new diversity metrics to simultaneously address the problems of diversifying the categories of items that each user sees, diversifying the types of users that each item is shown, and maintaining high recommendation quality. Following that, we model this problem as a subgraph selection problem on the bipartite graph of candidate recommendations between users and items. In Section 4.3, we consider the case of disjoint item categories and user types, and show that the resulting problems can be solved exactly in polynomial time, by a reduction to the minimum cost flow problem. Following that in Section 4.4, we address the case of non-disjoint categories and user types, we present efficient approximation algorithms using the submodularity of the objective, after proving NP-completeness of the objectives. Both of the above subgroup-based diversity metrics can also be generalized if we truncate the diversity contribution of any item category or user type by a threshold. These define thresholded versions of our metrics which we employ in our experiments to reflect the relative importance of various categories and types. Finally, in Section 4.6, we validate the effectiveness of our algorithms on the MovieLens-1m and Netflix datasets, and show that algorithms designed for our objective also perform well on sales diversity metrics, and even some intent-aware diversity metrics. Our experimental results justify the validity of our new item-based diversity metrics.

4.1.1 Related Work and Our Contributions

First we survey previous work on category-aware metrics for diversification, describing relevant new metrics defined by this body of work since we also compare our systems according to these metrics. Next, we survey work on sales diversity measures that are system-wide measures of diversification. Finally we review some related work on submodularity and crowd-sourcing.

4.1.2 Category-Aware Metrics

Others have previously investigated the use of category information in building recommender systems, and defined metrics for measuring the diversity contained in user lists. In our work, we focus on three, each of which informs one of the baseline algorithms we compare against in our experimental section.

Intra-list Distance: We define a recommendation set's intra-list distance (ILD) as the average pairwise distance among items [79]. This is used to measure the diversity of an individual user's recommendations and quantifies user-novelty. The distance $dist(v_k, v_j)$ between items we consider is measured using the cosine similarity between the items' category membership vectors. Given a list L of recommendations, defined by item lists of length c_u for user u, the intra-list distance is defined as follows (we use L to denote the left-side of the bipartite graph representation representing the users, and $N(u_i)$ to represent the neighbors of user u_i, which are items in the right-hand side recommended to her).

$$\frac{1}{|L|} \sum_{u_i \in L} \frac{1}{c_i(c_i - 1)} \sum_{(v_k, v_j) \in N(u_i)} dist(v_k, v_j).$$

Maximizing this objective enforces items in the recommendation list of a user to be dissimilar, but ILD does not influence the resulting distribution of categories in the resulting list. Furthermore, over-representation of certain categories is not explicitly punished by this metric. The MMR method [79] approximately optimizes the ILD metric, by greedily growing a recommendation list S. The next item to be added to the recommendation list is chosen to be the one which maximizes the quantity $\lambda rel(u_i, v_k) + (1 - \lambda) \min_{v_j \in S} dist(v_k, v_j)$, where λ is a trade-off parameter between 0 and 1, and $rel(u_i, v_k)$ represents the relevance score of item v_k to user u_i.

Intent-Aware Expected Reciprocal Rank (ERR-IA): The ERR-IA metric is the intent-aware version of Expected Reciprocal Rank metric, introduced by Chapelle et al [80]. ERR-IA considers the sum of each item category's weighted marginal relevance. To do so, we consider the quantity $p(R_i)$, which is the probability that the desired recommendation set's target category is R_i. Chapelle et al [80] formally define ERR-IA for some u's given recommendation set $N(u_i) = \{v_j\}_{j=1}^{c_i}$ as follows ($rel(v_k)$ denotes the relevance of item v_k to the given category):

$$\sum_{R_a \in \mathcal{R}} p(R_a) \sum_{k=1}^{c_i} \frac{1}{k} rel(v_k) \prod_{\ell=1}^{k-1} (1 - rel(v_\ell)).$$

ERR-IA is a personalized metric and aims for good coverage of relevant categories in the recommendation list. However, it does not explicitly penalize the over-representation of a particular category provided that it is well-covered. This metric is optimized by the xQuAD reranking strategy [81]. Similar to the MMR method, xQuAD greedily optimizes for its metric by greedily picking items which maximize the marginal change in the ERR-IA metric plus a relevance term.

Binomial Diversity: Binomial diversity is a diversity measure due to Vargas et al [82]. They model the number of times each category g is surfaced to a user as a binomial random variable X_g which depends on the display constraint, and the user's interest in that category. The probability of the user's interest in a given category is estimated from the training data, as the fraction of movies the user has shown interest in which belong to that category. Given a set of items S, which cover categories $g_1, \ldots, g_m$ out of a total of n categories $g_1, \ldots, g_n$, the Binomial coverage score of S is defined as

$$BinomialCov(S) = \left(\prod_{i=m+1}^{n} P(X_{g_i} = 0) \right)^{1/n}.$$

Given that S covers each of categories $g_1, \ldots, g_m$ a total of $k_1, \ldots, k_m$ times each, the binomial non-redundancy score is defined as

$$BinomialNonRed(S) = \left(\prod_{i=1}^{m} P(X_{g_i} > k_{g_i} | X_{g_i} > 0) \right)^{1/m}.$$

With these two measures in place, we can finally define

$$BinomialDiv(S) = BinomialCov(S) \cdot BinomialNonRed(S).$$

Binomial diversity punishes both under-representation and over-representation of a given category in a user's list, and strives for a balance between coverage and non-redundancy. The binomial coverage function obtains a high value when many categories which should appear at least once in a recommendation list do not make an appearance at all. The binomial non-redundancy term penalizes each genre for being either over or under-represented beyond its prevalence in the training data. Binomial diversity can be optimized for in the same way as xQuAD optimizes for the ERR-IA metric, and a thorough experimental evaluation of this method is carried out by [82]. However, due to the complexity of the metric, no explicit guarantees can be given for the performance of the algorithm.

4.1.3 Sales Diversity

In addition to adding diversity to a single user's recommendation list, we are also interested in surfacing content for increased feedback from the users. Since it is impossible for the users to gain feedback on items that are not surfaced adequately by the system, we measure our algorithms by two sales diversity measures as well. The first of these metrics is aggregate diversity, which counts the number of items that are shown to at least one user [49, 62, 63]. Our thresholded item diversity objective can be thought of as a refinement of aggregate diversity, where each item needs to be recommended to multiple different types of users instead of just once to anyone in the system. The second sales diversity measure that we employ in our experiments is the Gini index which is also widely employed in the recommender system community [62, 64–66, 83]. Category-aware metrics surveyed above try to solve the filter bubble problem for the users, while the type information can be used to solve the same problem for the business running the recommender system. We incorporate aggregate information symmetrically from both item-category and user-type information in our metrics to address this aspect.

4.1.4 Submodularity and NP-Completeness

The problem of maximizing a submodular set function have been extensively analyzed in the last 40 years, starting with Nemhauser et. al. [30] Many of the problems we pose reduce to maximizing such a function, which is NP-hard even when the function is monotone increasing. Nonetheless, the problem can be approximated using the greedy algorithm, which gives a $(1 - 1/e)$-approximation in the simplest case. Moreover, the

constraint of choosing a subset of a fixed size, which corresponds to a uniform matroid constraint, can be replaced by any other matroid constraint without affecting the approximation ratio [84]. Further advancements have made it possible to incorporate more types of constraints [85, 86] and yielded efficient algorithms for different types of computer architectures [87]. Since the coverage type objectives we define in this chapter are submodular, and our main type of constraints form a partition matroid constraint, we make extensive use of results in this area. Other researchers have considered the use of submodular functions in diversifying recommendations, but only over the set of a single user's recommendation set [88**?** –90].

4.1.5 Crowdsourcing

Finally, our work is related to some problems from the field of crowdsourcing. In crowdsourcing problems a large number of tasks need to be assigned to a number of human agents. The human agents have limited time, which gives rise to constraints similar to the display constraints we have in our recommender task. Moreover, every task needs to be attended to, making coverage an essential objective. There's a wealth of literature, some of which overlaps with the recommendation field, which seeks to solve these problems [91–93]. However, coverage of every item from every category is not a strict goal for us, and we seek only to increase coverage using post-processing while maintaining the relevance of the results. The relaxation of this constraint makes crowdsourcing results not directly applicable to the problems we pose.

4.2 Summary of Contributions

1. We introduce two new metrics for recommendation diversity that we call thresholded item-diversity ($TIDiv$) and thresholded user-diversity ($TUDiv$) which consider the distribution of user types among an item's recommended user set and the distribution of item categories in a user's recommendation item set respectively. $TUDiv$ is an objective that is similar to other category-aware diversity metrics, but $TIDiv$ is unique in considering diversity among an item's recommendees. TIDiv can be thought of as a sales diversity metric, and explicitly addresses the need of a business to collect feedback from different types of users.

2. In the case of disjoint types and categories, we model the problem of maximizing type diversity across all items and category diversity across all users as a subgraph selection problem. We reduce the resulting problem to a minimum cost flow problem and obtain exact polynomial time algorithms (Theorem 4.5) .

3. In the case of non-disjoint types and categories, we prove that the problem of maximizing the same objectives mentioned above is NP-complete (Theorem 4.6). While this rules out an exact polynomial time solution, we obtain a $(1 - 1/e)$-approximation using the submodularity of our objectives. We also show how to modify the algorithm to run in nearly linear time in the number of candidate recommendations (Theorem 4.11), making it very efficient.

4. Using the MovieLens dataset, we conduct experiments that consider both disjoint item categories and overlapping ones. We show that despite being flow based, our algorithms for the disjoint case can easily handle problems involving millions of candidate edges. We also show that the greedy algorithm we describe is competitive in efficiency with the reranking approaches we compare against (Subsection 4.7.1), and competitive with our optimal flow based approach when used with disjoint categories and types (Subsection 4.7.2). Our algorithms perform better than the baselines across the board on sales diversity metrics, and obtain good values for the other intent-aware metrics despite only optimizing for them by proxy.

4.3 Disjoint Types and Categories

We model the problem of making recommendations as a subgraph selection problem on a bipartite graph $G = (L \cup R, E)$ where the partition L represents a set of users and partition R represents a set of items. For each user u_i, we have a space constraint c_i, which is due to display space limitations on a given webpage. In this section, we model the case when the subgroups of users and items are disjoint.

We define a collection of subsets $\mathcal{L} = \{L_1, L_2, ...L_n\}$ on the user set L that represent different types of users and are mutually disjoint. Similarly, we define a collection of subsets $\mathcal{R} = \{R_1, R_2, ..., R_m\}$ on the item catalog which partition R to represent different categories or genres of items. This means there exists well-defined functions $type : L \to \mathcal{L}$, which maps users to their designated type, and $cat : R \to \mathcal{R}$, which maps items to their corresponding category. The edges between users and items in G represent possible recommendations that can be made. We wish to output a subgraph H of recommendations where each user u_i has c_i recommendations.

4.3.1 Global edge-wise diversity

Consider a recommendation edge (u_i, v_j) in the subgraph H. Let $\delta_i^H(cat(v_j))$ denote the number of neighbors user u_i has in v_j's category and $\delta_j^H(type(u_i))$ denote the number of

Input: A weighted bipartite graph $G(L, R, E)$, a vector of display constraints $\{c_i\}_{i=1}^{l}$, a collection of user types $\mathcal{L}$, a collection of item categories $\mathcal{R}$, real-valued parameters β, μ .
Output: A subgraph $H \subseteq G$, of maximum degree c_i at each node $u_i \in L$, and maximizing the objective $Div_{\beta,\mu}(H) + rel(H)$.

FIGURE 4.1: The definition of the MAX $- Div_{\beta,\mu}$ problem.

neighbors v_j has in u_i's type so far in H. In order to achieve a diverse set of recommendations, we would like each user to see a large number of categories, while also showing each item to a large number of user-types. To define a diversity metric that takes both of these considerations into account, we consider assigning the following weight to each edge where β and μ are real valued parameters

$$w_{ij} = \frac{\beta}{\delta_i^H(cat(v_j))} + \frac{\mu}{\delta_j^H(type(u_i))}.$$

A weighting like this is natural, since we are assigning less weight to recommendations that are not novel for either the user type or the item category that this recommendation serves. For instance, a recommendation edge that gives the user the only item from a category, and the item the only user from a type, will have the maximum weight of $\beta + \mu$. We can now define the diversity of a solution subgraph H as follows.

$$Div_{\beta,\mu}(H) = \sum_{u_i v_j \in H} w_{ij}$$

and subsequently maximize this objective for a highly diverse set of recommendations.

Proposition 4.1. *With the definition above*

$$Div_{\beta,\mu}(H) = \beta \sum_{u_i \in L} |a : R_a \cap N(u_i) \neq \emptyset| + \mu \sum_{v_j \in R} |a : L_a \cap N(v_j) \neq \emptyset|.$$

Proof. We can think of each edge weight as a user contributing a fractional value towards the category the user is hitting as well as an item contributing a fractional value of towards the user-type the item gets hit by. For example, if a user u_i has 4 edges to some category, the value of each $\frac{\beta}{\delta_i^H(cat(v_j))}$ for every item v in that category that u is connected to is $\frac{\beta}{4}$. If some item v_j has 3 edges coming from the same user-type, the value for $\frac{\mu}{\delta_j^H(type(u_i))}$ for each user v_j is connected to is $\frac{\mu}{3}$. This means that, $Div(H)$ gets a value of β for every category a user hits, and a value of μ for every user-type an item

hits:

$$\begin{aligned} Div_{\beta,\mu}(H) &= \sum_{u_i v_j \in H} \frac{\beta}{\delta_i^H(cat(v_j))} + \frac{\mu}{\delta_j^H(type(u_i))} \\ &= \sum_{u_i \in H} \sum_{R_a \cap N(u_i)} \frac{\beta}{|R_a \cap N(u_i)|} + \sum_{v_j \in H} \sum_{L_b \cap N(v_j)} \frac{\mu}{|L_b \cap N(v_j)|} \\ &= \beta \sum_{u_i \in L} |a : R_a \cap N(u_i) \neq \emptyset| + \mu \sum_{v_j \in R} |a : L_a \cap N(v_j) \neq \emptyset|. \end{aligned}$$

□

We can isolate both terms of this expression as their own objectives, which may be formalized as follows:

$$UserDiv(H) = \sum_{u_i \in L} \sum_{R_a} 1[\exists v_j \in R_a : u_i v_j \in H].$$

$$ItemDiv(H) = \sum_{v_j \in R} \sum_{L_a} 1[\exists u_i \in L_a : u_i v_j \in H].$$

Here, $UserDiv(H)$ will give us reward proportional to the number of categories hit for each user and $ItemDiv(H)$ will give us reward proportional to the number of user types hit for each item.

Ignoring type information, we first show that $UserDiv(H)$ can be optimized in polynomial time, since this construction is simpler to formulate and solve in practice.

Theorem 4.2. *The problem of maximizing $Div_{\beta,\mu}$ can be reduced to a minimum cost flow problem if the categories are disjoint, i.e. $R_a \cap R_b = \emptyset$ for all a, b.*

Proof. For each $u_i \in L$, we set supplies of c_i, and a demand of $\sum_{u_i \in L} c_i$ for a newly created sink node t. For each user u_i and category R_a such that $\exists v_j \in R_a$ such that $u_i v_j \in G$ we create nodes $n_{i,a}$ and $n'_{i,a}$. We will create an arc of capacity 1 and cost -1 between every u_i and $n'_{i,a}$. We will also add arcs of capacity 1 and cost 0 between every $n'_{i,a}$ and $n_{i,a}$ and arcs of unbounded capacity and cost 0 between u_i and $n_{i,a}$. For each edge $u_i v_j$ in G where $v_j \in R_a$ we create an arc of capacity 1 and cost 0 between $n_{i,a}$ and v_j. Finally, from each $v_j \in R$ we make an arc of unbounded capacity and cost 0 to the sink node t.

We let the solution subgraph H be the subgraph of G formed by using edges $u_i v_j$ for all arcs of the form $(n_{i,a}, v_j)$ used in the flow. Each node now gets to take one

recommendation in each new category for a cost of -1, Therefore, the cost of a flow defined by H is $-\sum_{u_i \in L} \sum_{R_a} 1[\exists v_j \in R_a : u_i v_j \in H]$. Minimizing this quantity is the same as maximizing $UserDiv(H)$, which proves the result. □

Proposition 4.3. *If every user is his own type, then subject to display constraints, $Div_{\beta,\mu}(H) \propto UserDiv(H)$, and $Div(H)$ can be maximized exactly in polynomial time.*

Proof. If every user is his own type, then the quantity $|a : L_a \cap N(v_j) \neq \emptyset|$ simply counts the number of edges incident on an item v_j. Therefore, we obtain

$$\begin{aligned} Div_{\beta,\mu}(H) &= \beta \sum_{u_i \in L} |a : R_a \cap N(u_i) \neq \emptyset| + \mu \sum_{v_j \in R} \delta^H(v_j) \\ &= \beta UserDiv(H) + \mu \sum_{u_i \in L} \delta^H(u_i) \\ &= \beta UserDiv(H) + \mu \sum_{u_i \in L} c_i. \end{aligned}$$

Since the quantity on the right is constant, the result follows from Theorem 4.2. □

Finally, we prove that the theorem in the most general case, by combining the objectives $UserDiv(H)$ and $ItemDiv(H)$. In fact, this is possible while incorporating rating relevance into the objective. In particular, let $rel(u_i, v_j)$ denote the relevance of item v_j to user u_i. Then the relevance based quality of the entire recommender system can be computed as $rel(H) = \sum_{(u_i,v_j) \in H} rel(u_i, v_j)$. We can now state the main result of this section.

Theorem 4.4. *The problem of maximizing $Div_{\beta,\mu}(H)$ can be reduced to a minimum cost network flow problem if both user types and item categories are disjoint, i.e. $R_a \cap R_b = \emptyset$ and $L_a \cap L_b = \emptyset$ for all a, b.*

Proof. Our network will have nodes for all users $u_i \in L$ and items $v_j \in R$ of $G(L \cup R, E)$, and a sink node t. The supply for each user u_i will be its corresponding space constraint c_i. For each user u_i, we will create two nodes for each item category R_a its edges hit, $n_{i,a}$ and $n'_{i,a}$. Let there be an arc of capacity 1 and cost $-\beta$ between u_i and $n'_{i,a}$ and an arc with capacity 1 and cost 0 between $n'_{i,a}$ and $n_{i,a}$. There will also be an arc with unbounded capacity and cost 0 between u_i and $n_{i,a}$. Similarly, for an item v_j, we will create two nodes for each user type L_b its incoming edges are from, $m_{j,b}$ and $m'_{j,b}$. Let there be an arc of capacity 1 and cost $-\mu$ between $m'_{j,b}$ and v_j, and an arc of capacity 1 and cost 0 between $m_{j,b}$ and $m'_{j,b}$. We will also add an arc of unbounded capacity and

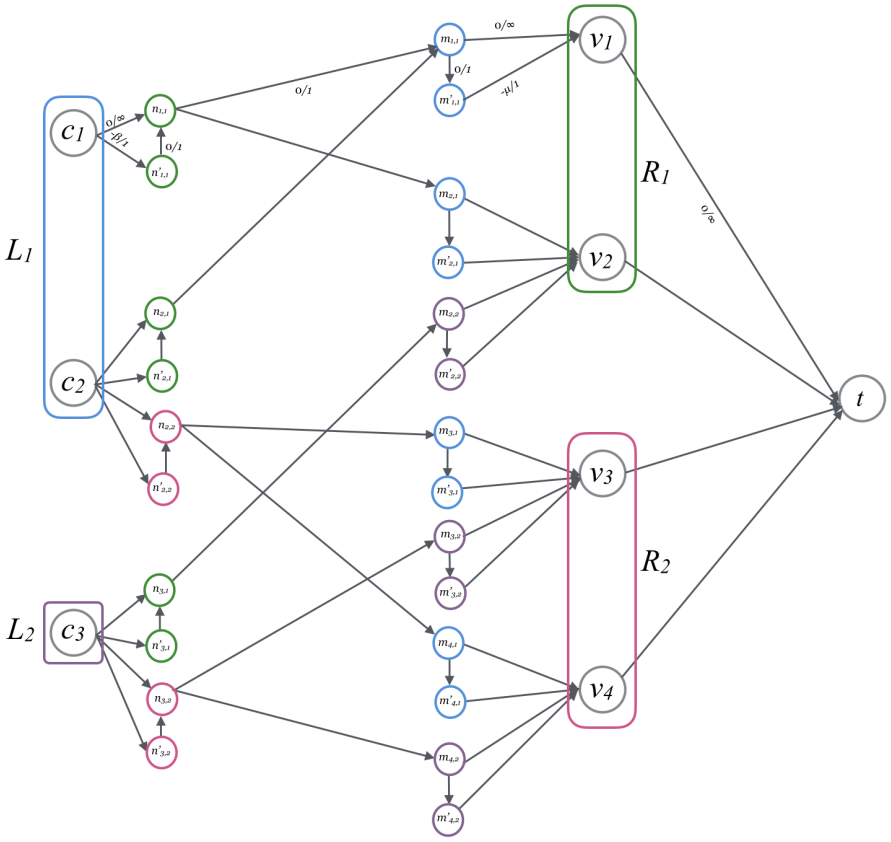


FIGURE 4.2: Construction of the flow problem in Theorem 4.4. The labels on the arcs denote cost/capacity.

cost 0 between $m_{j,b}$ and v. For each edge $(u_i, v_j) \in E$, where v_j is in category R_a and u_i is of type L_b, we will add an arc with cost $-rel(u_i, v_j)$ and capacity 1 between $n_{i,a}$ and $m_{j,b}$. Finally, there will be an arc from every item v_i to the sink t with unbounded capacity and cost 0. A diagram of the construction can be found in Figure 4.2.

We let the solution subgraph H be the subgraph of G formed by using edges (u_i, v_j) for all arcs of the form $(n_{i,a}, m_{j,b})$ used in the flow. The cost of the flow induced by H will therefore be $-\beta \sum_{u_i \in L} |a : R_a \cap N(u_i) \neq \emptyset| - \mu \sum_{v_j \in L} |a : L_a \cap N(v_j) \neq \emptyset| - rel(H)$, since for each new category a user u_i hits, we use one arc of cost $-\beta$ and similarly, we will use one arc of cost $-\mu$ for each new user type an item v_j gets hit by. Observe that this quantity is $-\beta UserDiv(H) - \mu ItemDiv(H) - rel(H)$. Therefore, minimizing this quantity is the same as maximizing $\beta UserDiv(H) + \mu ItemDiv(H)$. □

4.3.2 Diversity Thresholds

While increasing user and item diversity is important, one downfall to our method is that it fails to take into account the fact that the relevance of each category to a user

Input: A weighted bipartite graph $G(L, R, E)$, a vector of display constraints $\{c_i\}_{i=1}^{l}$, a collection of user types $\mathcal{L}$, a collection of item categories $\mathcal{R}$, user-category thresholds $\{\rho_i(R_a)\}_{i,a}$, item-type thresholds $\{\lambda_j(L_b)\}_{j,b}$, real-valued parameters β, μ .
Output: A subgraph $H \subseteq G$, of maximum degree c_i at each node $u_i \in L$, and maximizing the objective $TDiv_{\beta,\mu}(H) + rel(H)$.

FIGURE 4.3: The definition of the $\text{MAX} - TDiv_{\beta,\mu}$ problem.

may be different. It may not be beneficial for our recommender to show a user items from every different category possible, since that user may not be interested in some of those categories to begin with. The same can be said for the item side: item diversity may increase an item's popularity and help it collect feedback, however, an item should be shown to users in its target audience more than users outside its target audience.

To fix this, and help guide our algorithm to selecting more relevant recommendations for each user and item, we propose setting diversity thresholds for each user-category and item-type pair. For categories that the user cares a lot about, we can increase this threshold while setting it to zero for those that the user is not interested in at all. Let us denote $\rho_i(R_a)$ as user u_i's threshold for recommendations made to items in category R_a, and $\lambda_j(L_b)$ be an item v_j's threshold for recommendations made from users of type L_b. We now define two updated objectives that take these thresholds into account:

$$TUDiv(H) = \sum_{u_i \in L} \sum_{R_a} \min(\rho_i(R_a), \delta_i^H(R_a)).$$

$$TIDiv(H) = \sum_{v_j \in R} \sum_{L_b} \min(\lambda_j(L_b), \delta_j^H(L_b)).$$

We can again consider these two objectives together to form a single objective that will maximize the thresholded diversity of a solution subgraph H, where β and μ are real-valued parameters:

$$TDiv_{\beta,\mu}(H) = \beta TUDiv(H) + \mu TIDiv(H).$$

The main result of this section is that in the case where user types and item categories are disjoint, $TDiv(H)$ can still be optimized in polynomial time.

Theorem 4.5. *The $MAX - TDiv_{\beta,\mu}$ problem reduced to a minimum cost network flow problem if both user types and item categories are disjoint, i.e. $R_a \cap R_b = \emptyset$ and and $L_a \cap L_b = \emptyset$ for all a, b.*

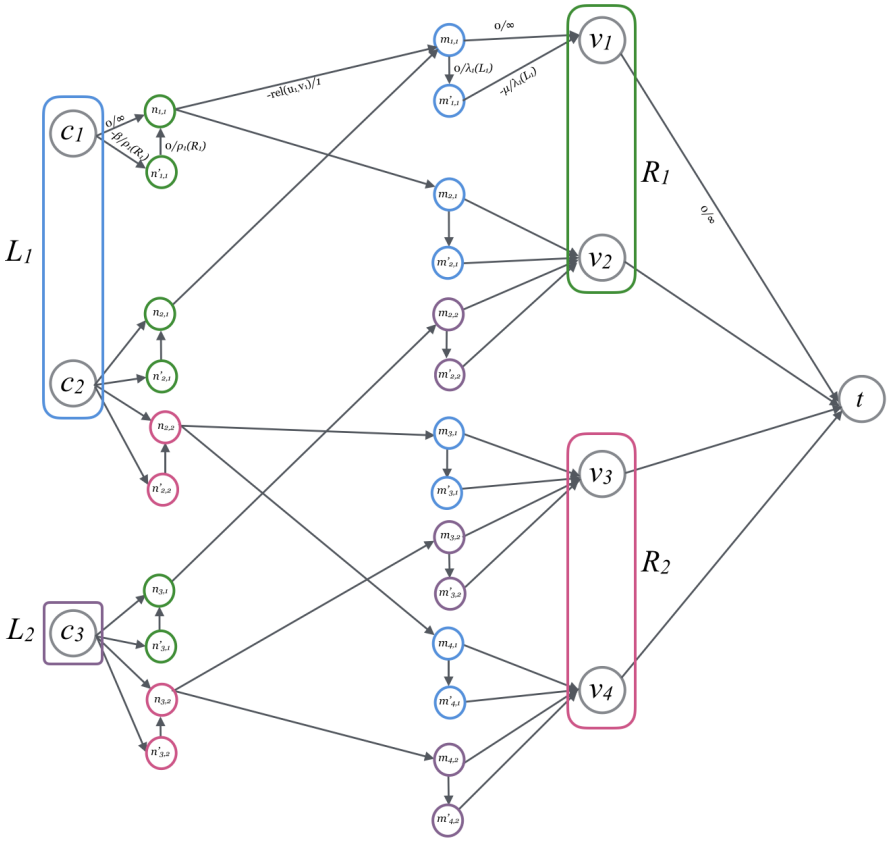


FIGURE 4.4: Construction of the flow problem in Theorem 4.5.

Proof. Our flow construction will be the same as that of Theorem 4.4. However let the capacity of the arc between all nodes u_i and $n'_{i,a}$ and between $n'_{i,a}$ and $n_{i,a}$ be $\rho_i(R_a)$ instead of just 1. Likewise, let the capacity between all nodes $m'_{j,b}$ and v_j and between $m_{j,b}$ and $m'_{j,b}$ be $\lambda_j(L_b)$. We extract the solution subgraph H the same way as in Theorem 4.4. The cost of the flow induced by H will therefore cost $-\beta \sum_{u_i \in L} \sum_{R_a} \min(\rho_i(R_a), \delta_i^H(R_a)) - \mu \sum_{v_j \in R} \sum_{L_b} \min(\lambda_j(L_b), \delta_j^H(L_b)) - rel(H)$, since we may use the $-\beta$ cost arc for each user-category pair and the $-\mu$ cost arc item-type pair up until they reaches capacity. This quantity is simply $-\beta TUDiv(H) - \mu TIDiv(H) - rel(H) = -TDiv_{\beta,\mu}(H) - rel(H)$, therefore, minimizing this quantity will maximize $TDiv_{\beta,\mu}(H) + rel(H)$. A diagram of our construction can be found in Figure 4.4.

□

In order for our algorithms to produce useful results, these thresholds must be set in a way that reflect the true category interests of the users and the type interests of items. Therefore, these thresholds must be set using the training data, and we leave a full discussion of how the thresholds are set in practice to Section 4.6 where we detail our experimental setup.

Our results about disjoint categories and type are useful in applications such as retail catalogs where the products (items) are split into natural retail categories according to product ontologies, and when the users are split according to natural mutually exclusive demographic types such as gender or age and income brackets. However, the more general case is the one when categories and items are not necessarily disjoint which we turn to next.

4.4 Overlapping Types and Categories

Although cases involving disjoint user-types and categories are solvable in polynomial time, in actual practice, categories of items are not necessarily disjoint, and users may be assigned to more than one user-type. When item categories and user types are non-disjoint, maximizing $TDiv(H)$ is NP-Hard, which can be seen in the following theorem.

Theorem 4.6. *Finding an optimal solution to maximize $TDiv_{\beta,\mu}(H)$ with non-disjoint categories and types is NP-Hard.*

Proof. We fix $\beta = \mu = 1$ since proving the NP-Hardness of a special case is sufficient. We show that optimizing just $TUDiv_{\beta,\mu}(H)$ with a single user is NP-Hard with the following reduction from the Max-Cover problem, a well known NP-Hard problem: Given a set of elements $\{1, 2, 3, ..., n\}$, a collection of m sets $\mathcal{S}$, and an integer k, we want to find the largest number of elements covered by at most k sets.

We construct a bipartite graph $G(L \cup R, E)$ where $|L| = 1$ and $|R| = m$ with the items representing elements. The vertex $u \in L$ has an out-degree of m, one for each directed edge to the vertices in R. We then create subsets $R_1, R_2, R_3, ..., R_n \subseteq R$ with one such subset corresponding to each set S_i in $\mathcal{S}$: the subset of vertices in R_i correspond to the set of elements in the subset S_i. We also set $\{\rho_1(R_i)\}_{i=1}^{n} = 1$. We let $c_1 = k$. An optimal solution for $TUDiv(H)$ would give an optimal solution to Max-Cover, since finding the maximum number of categories hit with k edges out of L would find the maximum number of elements we can cover with k sets.

Since finding the optimal solution to $TUDiv_{\beta,\mu}(H)$ is NP-Hard, and $TUDiv(H)$ is a special case of $TDiv_{\beta,\mu}(H)$, we have shown that optimizing $UserDiv(H)$ will also be NP-Hard, thus proving the desired result. □

Since we are not able to maximize $TDiv_{\beta,\mu}(H)$ optimally, we can make use of the fact that $TDiv_{\beta,\mu}(H)$ is both monotone and submodular, which will allow us to apply a greedy algorithm which will yield a $(1 - 1/e)$-approximation ratio.

Proposition 4.7. *$TUDiv(H)$ is submodular.*

Proof. Let X and Y be two sets of edges such that $X \subseteq Y$ and let e be an edge not in X or Y. Consider the quantity $TUDiv(X \cup \{e\}) - TUDiv(X)$. Observe that this is the number of categories R_i that e will saturate (not including categories that have already reached their threshold). This will be at least as much as the number of categories e saturates in Y, since Y could contain edges that have already saturated categories that e would saturate. It follows that $TUDiv(X \cup \{e\}) - TUDiv(X) \geq TUDiv(Y \cup \{e\}) - TUDiv(Y)$. This satisfies the "diminishing returns" property of submodular functions. Therefore $TUDiv(H)$ is submodular. □

We get the following from a symmetric argument.

Proposition 4.8. *$TIDiv(H)$ is submodular.*

Proof. We again consider edge sets X and Y where $X \subseteq Y$ and let e be an edge not yet in X or Y. Observe that the quantity $TIDiv(X \cup \{e\}) - TIDiv(X)$ counts the number of types L_i that e saturates (not counting user types that have already reached their threshold). This will be at least as much as the number of types e saturates in Y, since Y could contain edges that saturate types that e saturates. This means that $TIDiv(X \cup \{e\}) - TIDiv(X) \geq TIDiv(Y \cup \{e\}) - TIDiv(Y)$. Therefore, $TIDiv(H)$ is submodular. □

Corollary 4.9. *$TDiv_{\beta,\mu}(H)$ is submodular.*

Proof. Since both $TUDiv(H)$ and $TIDiv(H)$ are submodular, and non-negative linear combinations of submodular functions are also submodular, $TDiv(H)$ will also be submodular. □

Corollary 4.10. *The objective function $rel(H)$ is submodular.*

Proof. The sum of the relevance values of all the recommendations is a linear function. Linear functions are modular and hence also submodular, and submodular functions are closed under addition, so this function is submodular as well. □

The monotonicity and the submodularity of the objective now allows us to write the following simple greedy algorithm:

Stated in its current form, the greedy algorithm takes $O(E^2)$ to run. However, it is possible to speed it up significantly by using better data structures.

Data: A bipartite graph $G = (L, R, E)$ and display constraint c
Result: A solution graph H maximizing $TDiv_{\beta,\mu}(H) + rel(H)$
while *some vertex* $u_i \in L$ *has* $deg_H(u_i) < c_i$ **do**
  $(u_i, v_j) = e \longleftarrow \arg\max_{e' \in E}\ TDiv_{\beta,\mu}(H \cup \{e'\}) - TDiv_{\beta,\mu}(H) + rel(e')$;
  if $deg_H(u_i) < c$ **then**
    $H \leftarrow H \cup \{e\}$
  end
end
return H;

Algorithm 5: The greedy algorithm for $TIDiv$ and $TUDiv$ maximization

Theorem 4.11. *Let $R_1, \ldots, R_k$ be the set of overlapping categories and $L_1, \ldots, L_p$ be the set of overlapping types for a $TDiv$ maximization problem. Then the greedy algorithm can be made to run in time, $O((E + \sum_{i=a}^{k} R_a + \sum_{i=b}^{p} L_b)\log(E))$.*

Proof. Let $u \in L$, $v \in R$, and let $u, v \in G$ be a candidate recommendation. The category contribution of this edge to a partial solution H is the number of categories R_i that v belongs to, for which $\rho(R_i) < \delta_L^H(R_i)$ is satisfied. Similarly, the type contribution of this edge is the number of types L_j that u belongs to, for which $\lambda(L_j) < \delta_R^H(L_j)$. While constructing the solution, both of these quantities can only decrease. Furthermore, we are only ever interested in the node with the highest marginal contribution.

Therefore, we can keep track of the potential contribution of each edge in a max-heap. Initially, the priority of each edge is set to be the number of categories and number types it covers. Each time an edge meets a category target, we decrease the priority of every unused edge incident on that category by β. Similarly, when a user type target is satisfied, we decrease the priority of every unused edge incident on that type by μ. Both operations take logarithmic time using a heap which supports the decrease-key operation. This operation is performed at most once for each type and category.

This means that we are maintaining a max-heap with $|E|$ elements, removing the maximal element $|E|$ times, and decreasing the key of some edge by at most $\sum_{a=1}^{k} R_a + \sum_{i=b}^{p} L_b$ times. Both of these operations can be done in $O(\log(E))$ time, which gives us the desired runtime. □

4.5 Category and Type Diversity

In this section, we lift the definition of user and item diversity to the level of types and categories. In other words, rather than looking at the diversity of individual users and items, we can instead consider the user type diversity for a specific category, i.e. the

number of user-types that a certain category receives recommendations from. Likewise, we may consider the category diversity for each user-type, as in the number of categories a certain user-type has recommendations to.

When looking at these quantities, just like before, it is useful to consider setting 'diversity thresholds' for types and categories, which represent the desired number of types hit by a specific category ($\rho(R_a)$) or categories hit by a specific type ($\lambda(L_b)$). If we take the minimum of $\rho(R_a)$ and the quantity $\delta^H_{\mathcal{L}}(R_a)$, which counts the number of different user types by which a category R_a is hit in H, we can make sure as many 'thresholds' of categories are met (as opposed to having some categories achieve great diversity in user-types while other categories remain significantly less diverse in comparison). The same can be done with $\lambda(L_b)$ and $\delta^H_{\mathcal{R}}(L_b)$, which counts the number of categories a user-type L_b hits in H. This motivates the following objective:

$$\Delta(H) = \sum_{R_a} \min(\rho(R_a), \delta^H_{\mathcal{L}}(R_a)) + \sum_{L_b} \min(\lambda(L_b), \delta^H_{\mathcal{R}}(L_b))$$

It is useful to consider both terms of $R(H)$ separately, by defining the following two sub-objectives:

$$\kappa(H) = \sum_{R_a} \min(\rho(R_a), \delta^H_{\mathcal{L}}(R_a))$$

$$\tau(H) = \sum_{L_b} \min(\lambda(L_b), \delta^H_{\mathcal{R}}(L_b))$$

4.5.1 Disjoint Categories

When item categories are disjoint, we are able to solve certain sub-cases of $R(H)$ optimally. For instance, if every user is his own type and diversity thresholds for each type is unbounded, it is the case that

$$R(H) = \kappa(H) + UserDiv(H).$$

Using similar flow structure as the previous section, we can construct a flow that will maximize $R(H)$ in this case in polynomial time.

Theorem 4.12. *The problem of maximizing the objective $\kappa(H) + UserDiv(H) + rel(H)$ subject to display constraints and category thresholds can be reduced to a minimum cost network flow problem if $R_a \cap R_b = \emptyset$ for all a, b.*

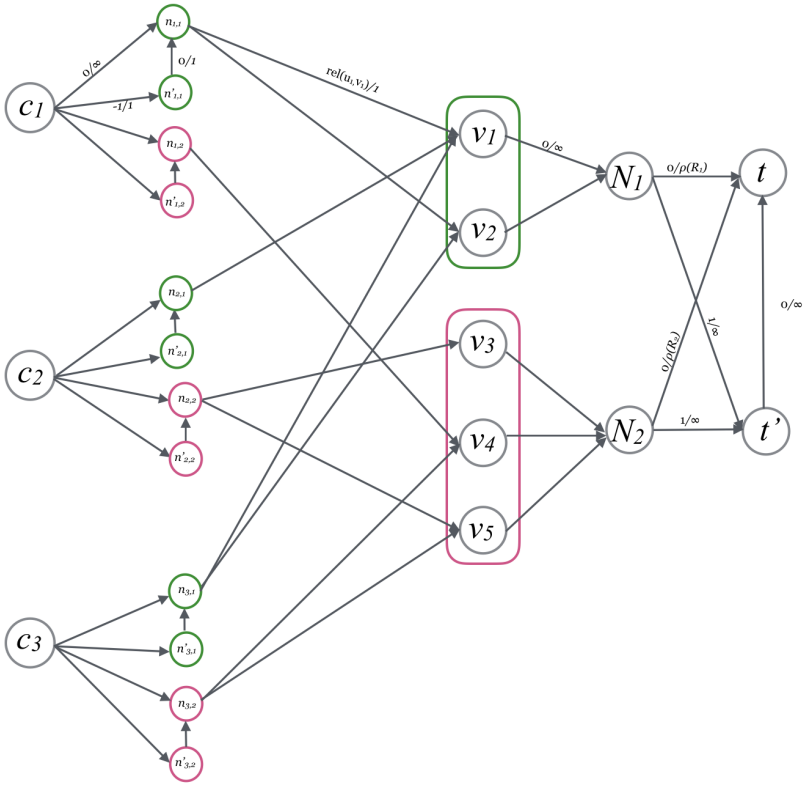


FIGURE 4.5: Construction of the flow problem in Theorem 4.12.

Proof. Similar to the network flow constructed in Theorem 4.2, we will set the supplies of each $u_i \in L$ to c_i and create a sink node t with a demand $\sum_i c_i$. For each user u_i and category R_a such that $v_j \in R_a$ and $u_i v_j \in G$ we create nodes $n_{i,a}$ and $n'_{i,a}$. We will create an arc of capacity 1 and cost -1 between every u_i and $n'_{i,a}$. We will also add arcs of capacity 1 and cost 0 between every $n'_{i,a}$ and $n_{i,a}$ and arcs of unbounded capacity and cost 0 between u_i and $n_{i,a}$. For each edge (u_i, v_j) in G where $v_j \in R_a$ we create an arc of capacity 1 and cost $-rel(u_i, v_j)$ between $n_{u,a}$ and v. For each category R_a we will create a collector node N_a and create arcs of unbounded capacity and cost 0 between every item $v_j \in R_a$ and N_a. From each collector node N_a we will create an arc of capacity $\rho(R_a)$ and cost 0 to the sink node. We will also create an arc of unbounded capacity and cost 1 between all N_a's and a second sink node, t'. Finally, we will create an arc of unbounded capacity and cost 0 from t' to t. A diagram of the construction can be found in Figure 4.5.

We let the solution subgraph H be formed by using edges (u_i, v_j) for all arcs of the form $(n_{i,a}, v_j)$ used in the flow. Each node will take one recommendation in each new category for -1 cost. For each category, we will incur a cost of $-\min(\rho(R_a), \delta^H_{\mathcal{L}}(R_a))$. Therefore, the cost of the flow induced by a solution subgraph H will be $-\sum_{R_a} \min(\rho(R_a), \delta^H_{\mathcal{L}}(R_a)) - \sum_{u_i \in L} \sum_{R_a} 1[\exists v_j \in R_a : u_i v_j \in H]$, which is $-\kappa(H) - UserDiv(H)$. Thus, minimizing

this quantity will maximize $\Delta(H)$ (when every user is his own type and categories are disjoint) in polynomial time. □

Similarly, if every item is its own category (and has unbounded diversity thresholds) and all user types are disjoint, our objective function becomes

$$\Delta(H) = ItemDiv(H) + \tau(H).$$

This can also be solved using the same flow structure as in Theorem 4.12.

Corollary 4.13. *Maximizing the objective $ItemDiv(H) + \tau(H) + rel(H)$ subject to display constraints and user-type thresholds can be reduced to a minimum cost network flow problem if $L_a \cap L_b = \emptyset$ for all a, b.*

Proof. We can construct a network that is essentially a reflection of the network flow in Theorem 4.12. The cost of the flow will be $-\sum_{L_b} \min(\lambda(L_b), \delta^H_{\mathcal{R}}(L_b)) - \sum_{v_j \in R} \sum_{L_a} 1[\exists u_i \in L_a : u_i v_j \in H]$, which is $-ItemDiv(H) - \tau(H)$. Therefore, minimizing this quantity will maximize $ItemDiv(H) + \tau(H)$ □

4.5.2 Non-disjoint Types and Categories

As before the more interesting practical case is when item categories and user types are non-disjoint, and in this case, maximizing $\Delta(H)$ is NP-Hard using the same reduction as in Theorem 4.6.

Theorem 4.14. *Finding an optimal solution to maximize $\Delta(H)$ is NP-Hard.*

Again we will show that $\Delta(H)$ is both monotone and submodular, which will allow us to apply a greedy $(1 - 1/e)$-approximation algorithm.

Proposition 4.15. *$\kappa(H)$ is submodular.*

Proof. Let X and Y be two sets of edges such that $X \subseteq Y$ and let e be an edge not in X or Y. Consider the quantity $\kappa(X \cup \{e\}) - \kappa(X)$. Observe that this is the number of categories R_i that e will contribute type-value to (not including categories that have already reached their threshold). This will be at least as much as the number of categories e will contribute to in Y, since Y could contain edges that already contribute to categories that e could contribute to. It follows that $\kappa(X \cup \{e\}) - \kappa(X) \geq \kappa(Y \cup \{e\}) - \kappa(Y)$. This satisfies the "diminishing returns" property of submodular functions. Therefore $\kappa(H)$ is submodular. □

A symmetric argument shows the following.

Proposition 4.16. *$\tau(H)$ is submodular.*

Corollary 4.17. *$\Delta(H)$ is submodular.*

Proof. Since both $\kappa(H)$ and $\tau(H)$ are submodular, and non-negative linear combinations of submodular functions are also submodular, $\Delta(H)$ will also be submodular. □

As before if each recommendation is assigned a non-negative rating w, we get the following.

Corollary 4.18. *The objective function $\Delta(H) + w(H)$ is submodular.*

The monotonicity and the submodularity of the objective now allows us to write a simple greedy algorithm for the category coverage maximization problem:

Since our degree constraints form a partition matroid, the greedy algorithm once again achieves a $(1 - 1/e)$ approximation ratio. The regular greedy algorithm takes $O(E^2)$ to run. However, it is possible to speed it up significantly by using better data structures as in Theorem 4.11.

Theorem 4.19. *Let $R_1, \ldots, R_k$ be the set of categories for a category coverage problem. Then the greedy algorithm can be made to run in time, $O((E + \sum_{i=a}^{k} R_a + \sum_{i=b}^{p} L_b) \log(E))$.*

4.6 Experimental Setup

4.6.1 Datasets

Category Data: In addition to the rating data, we use type and category data from the MovieLens-1m dataset and category data from IMDB. For disjoint user types in the MovieLens dataset, we use 3 the three different demographic data points included in the data: age-group (6 different values), gender (2 different values), and occupation (19 different values) each of which form a partition the user set. Since the Netflix dataset does not include demographic data, we partition the user set into clusters using the k-means clustering algorithm on the user feature matrix provided by the underlying matrix factorization algorithm.

Supergraph Generation: We used two rating datasets to generate the graphs we fed to our algorithms: MovieLens-1m [76] and the Netflix Prize dataset. We pre-processed

the datasets to ensure that every user and every item has an adequate amount of data on which to base predictions. This post processing leaves the MovieLens-1m data with 5800 users and 3600 items, and the Netflix dataset with 8000 users and 5000 items. The use of these datasets is standard in the recommender systems literature. In this chapter , we consider the rating data to be triples of the form $(user, item, rating)$, and discard any extra information.

Each dataset was processed in two different ways, once for experiments involving disjoint categories and once for experiments involving overlapping categories. In each case, the full dataset was filtered for items for which the category information was known. Each of these were then split 5-ways into holdout test sets and training sets. Only users for which more than 50 ratings were considered for inclusion in the test, and we denote this set of users by $L_T \subseteq L$. The training sets were then fed into a matrix factorization algorithm due to Hu et al. [1] with 50 latent factors. We set the input confidence value parameter α in their method to the recommended value of 40, as recommended by the authors, and grid searched for the best regularization parameter λ using 5-fold cross validation. Using the resulting user and item factor matrices, for each user we predicted the ratings of all the items for which the user did not provide feedback in the training test. Among these predicted ratings, we retained the 250 highest rated items along with their predicted ratings to feed into our reranking algorithms.

4.6.2 Quality evaluation

We measure the effectiveness of our algorithms and others' along several orthogonal dimensions. For relevance, we report precision values, i.e. the fraction of items in the recommendation set that match items given in the test set. Formally, if we denote the set of recommendations given to a user given H as $N(u_i)$ and the set of relevant held-out items for the user as $T(u_i)$, we define precision as

$$P = \frac{1}{|L_T|} \sum_{u_i \in L_T} \frac{|N(u_i) \cap T(u_i)|}{c_i}$$

In this paper, a held-out item is considered to be relevant to a user in our evaluation if its assigned rating was 3 or higher. Aside from relevance based metrics, we also report two sales diversity metrics: aggregate diversity and the Gini index.

The definitions of these metrics can be found in 3.

Finally, we report the objectives for which our methods explicitly optimize. These are ERR-IA for the xQuAD reranker, ILD for the MMR reranker, Binomial Diversity for

the Binomial Diversity reranker. Among these, only the Binomial Diversity reranking method takes a parameter α, which corresponds to a personalization parameter. The authors use the value $\alpha = 0.5$ in their experimental evaluation [82], and we also use this setting. We measure each of these metrics as well as our own $TUDiv$ and $TIDiv$ as they are measured among only the relevant items in the test set.

As mentioned in Subsection 4.3.2 the $TUDiv$ and $TIDiv$ metrics, we set the thresholds using the training data. In particular, for the case of disjoint categories, we count the number of times each category appears in a user's training set, normalize these values to sum to the display constraint, and round to integer values. For the case of overlapping categories, we perform the same operation, but normalize the thresholds to sum to the display constraint times the average number of categories for an item in the training set. In the case of disjoint types, we again set the type thresholds proportional to the distribution of types found in the training data, but normalize the distribution to sum to 20% of the average number of recommendations an item would have received if every item were equally promoted by the recommender system. This allows the measure to promote sales diversity among items, while respecting its interaction history with the users.

In our tables, we abbreviate the names of these metrics as P for Precision, A for aggregate diversity, G for the Gini index, BD for Binomial Diversity, and ILD for intra-list distance. The number next to each metric denotes the cutoff at which it was evaluated.

4.6.3 Baselines

We compare our method against 3 baselines methods: the Binomial Diversity reranker due to Vargas et al. [82], the MMR reranker due to Carbonell et. al [79], and the xQuAD algorithm due to Santos et al. [81]. Each method takes a parameter $\lambda \in [0, 1]$ which trades off relevance with the metric which is being optimized. For each of these methods, we grid searched for the best trade-off parameter, and report all the measurements for the setting which produced the best results for the method's corresponding metric. Since our algorithms have two trade-off parameters μ and β in the objective $rel(H) + TDiv_{\beta,\mu}$ corresponding to two different metrics, we grid search along both dimensions and report the two solutions which maximize $TIDiv$ and $TUDiv$ respectively. We additionally report the same metrics for the undiversified recommendation lists provided by the matrix factorization method under the heading "TOP".

4.6.4 Software

For the matrix factorization based recommender we trained, we used the implementation of Hu's matrix factorization method found in Ranksys [53]. The baseline methods we compare against are also implemented in the same library. Our methods and metrics were implemented in a way to be compatible with the same library, and can be found on Github. Additionally, we used a minimum cost network flow optimizer written by Bertolini and Frangioni [77]. Our choice of MCFSimplex was motivated by its open-source status and efficiency, but any other minimum cost flow solver such as CPLEX or Gurobi can be used as well.

4.7 Experiments

In this section we report our findings on diversifying recommendations in MovieLens derived recommendation problems. Our findings can be summarized as follows:

1. In the setting of overlapping item categories, the greedy algorithm leveraging the submodularity of the $TDiv$ objective obtains significant gains in the $TIDiv$ and $TUDiv$ recommendation diversity metrics. Our algorithm preserves or improves the accuracy of the baseline recommender system, while also increasing sales diversity metrics.

2. In the setting of disjoint item categories, we show that the flow based algorithm obtains solutions which have higher predictive accuracy and higher sales diversity measurements. However, the differences are small enough for the greedy algorithm to make a suitable replacement for the more expensive, flow-based optimization technique.

3. The greedy algorithm is faster than competing diversification techniques, making it suitable for large scale recommendation tasks, provided that the heap used in its implementation can fit in memory.

4.7.1 Experiments on Overlapping Categories

We first present our experiments on overlapping categories based on the artistic genre information. Our results are summarized in Tables 4.2,4.3,4.4 and the relative performance of the methods we tested can be seen in Figure 4.6. As expected each diversification method is best at maximizing their own objectives. In the case of our methods,

Method	P@20	ERR-IA@20	ILD@20	BD@20	TUDiv@20	TIDiv@20	A@20	G@20
TOP	0.244	0.191	0.157	0.203	0.169	0.178	0.386	0.082
xQuAD ($\lambda = 0.4$)	**0.264**	**0.236**	0.165	0.224	0.207	0.139	0.368	0.072
MMR ($\lambda = 0.4$)	0.233	0.183	**0.172**	0.227	0.158	0.139	0.371	0.075
BD ($\lambda = 0.6$)	0.227	0.208	0.156	**0.255**	0.152	0.176	0.384	0.084
Greedy ($\beta = 0.4, \mu = 4.0$)	0.252	0.201	0.163	0.238	**0.210**	0.189	0.420	0.087
Greedy ($\beta = 4.0, \mu = 0.2$)	0.245	0.200	0.158	0.237	0.203	**0.214**	**0.559**	**0.107**

TABLE 4.2: MovieLens diversifications for based on artistic genre based categories and age group based types. The best value in each metric is bolded.

Method	P@20	ERR-IA@20	ILD@20	BD@20	TUDiv@20	TIDiv@20	A@20	G@20
TOP	0.244	0.191	0.157	0.203	0.169	0.148	0.386	0.082
xQuAD ($\lambda = 0.4$)	**0.264**	**0.236**	0.165	0.224	0.207	0.139	0.368	0.072
MMR ($\lambda = 0.4$)	0.233	0.183	**0.172**	0.227	0.158	0.139	0.371	0.075
BD ($\lambda = 0.6$)	0.227	0.208	0.156	**0.255**	0.152	0.144	0.384	0.084
Greedy ($\beta = 0.4, \mu = 4.0$)	0.253	0.201	0.163	0.238	**0.210**	0.157	0.412	0.086
Greedy ($\beta = 4.0, \mu = 0.2$)	0.246	0.201	0.159	0.237	0.204	**0.181**	**0.545**	**0.104**

TABLE 4.3: MovieLens diversifications for based on artistic genre based categories and occupation based types. The best value in each metric is bolded.

Method	P@20	ERR-IA@20	ILD@20	BD@20	TUDiv@20	TIDiv@20	A@20	G@20
TOP	0.244	0.191	0.157	0.203	0.169	0.197	0.386	0.082
xQuAD ($\lambda = 0.4$)	**0.264**	**0.236**	0.165	0.224	0.207	0.139	0.368	0.072
MMR ($\lambda = 0.4$)	0.233	0.183	**0.172**	0.227	0.158	0.139	0.371	0.075
BD ($\lambda = 0.6$)	0.227	0.208	0.156	**0.255**	0.152	0.195	0.384	0.084
Greedy ($\beta = 0.4, \mu = 4.0$)	0.253	0.200	0.163	0.238	**0.210**	0.210	0.423	0.087
Greedy ($\beta = 4.0, \mu = 0.2$)	0.243	0.202	0.156	0.236	0.202	**0.233**	**0.564**	**0.111**

TABLE 4.4: MovieLens diversifications for based on artistic genre based categories and gender based type data. The best value in each metric is bolded.

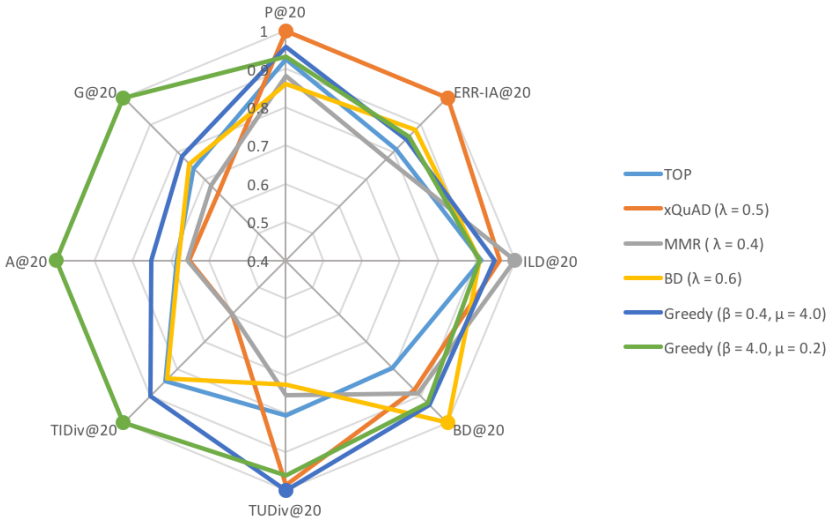


FIGURE 4.6: A radial graph showing the relative performance of the reranking methods we tested for MovieLens data and gender based diversification.

this is true for both $TIDiv$ and $TUDiv$. Among the metrics we tested, both our greedy algorithm and the xQuAD algorithms made minor improvements to the precision of the recommendation lists, while Binomial Diversity and MMR slightly deteriorated the precision values. However, these differences are minor and each algorithm was able to find a good trade-off between relevance and diversity under suitable parameter settings.

Among the intent-aware metrics we have tested, our algorithms provide a very good proxy for Binomial Diversity, while performing less well on the Intra-List Distance and ERR-IA metrics. This can be explained by the fact that Binomial Diversity, unlike the ERR-IA and ILD metrics, explicitly penalizes redundancy. In contrast, our metric $TUDiv$ is similar to binomial diversity in the sense that it sets thresholds which implicitly penalize over-redundancy by taking away the reward for hitting new categories. However, the converse is not true, and the Binomial Diversity reranking method achieves poor values for both the $TIDiv$ and $TUDiv$ metrics.

Among the methods we tested, the best proxy for our $TUDiv$ metric was provided by the xQuAD approach and none of the algorithms we tested provided a good proxy for $TIDiv$. While this deficiency can be excused, as none of these algorithms take as input the various user type grouping we provide to our diversifiers, each of the other baselines also regressed or insignificantly changed the sales diversity metrics aggregate diversity and Gini index. This validates our hypothesis that $TIDiv$ is best thought of as a sales diversity measure and that being category-aware is not enough for a reranking algorithm to produce diverse results for items.

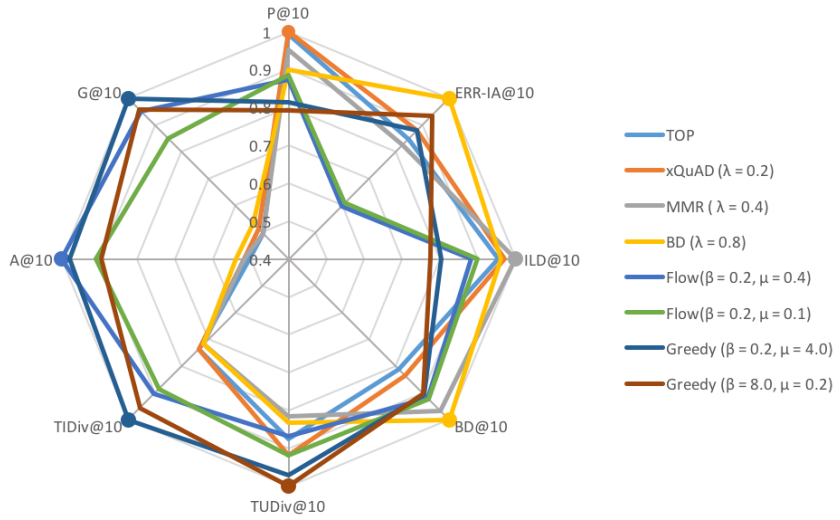


FIGURE 4.7: A radial graph showing the relative performance of the reranking methods we tested for MovieLens data and age group based diversification

4.7.2 Experiments on Disjoint Categories

In this section we present the diversification results for disjoint item categories derived from movie studio information. This is intended to simulate the scenario where a content aggregator would like to diversify recommendations among different content providers. Since we can apply both the greedy algorithm and the flow based algorithm in this case, we report results for both. Our results for the top-10 recommendation diversification task are summarized in Tables 4.2,4.3,4.4 . We note that once again, every reranker optimizes its metric the best, with the exception of the xQuAD, whose objective is actually maximized by the Binomial Diversity reranking method. We also note that the precision based effectiveness of our greedy algorithm is reduced in this setting, while its effectiveness in the sales diversity metrics is amplified.

Our flow-based method and greedy algorithm show several notable differences in experimental evaluation. First, we find that the greedy algorithm actually performs better than the flow-based method in our intent-aware metrics $TUDiv$ and $TIDiv$, although our flow based methods produce more accurate recommendation lists. The solution each algorithm produces creates a different split between the $TUDiv$ term of the objective, the $TIDiv$ term of the objective and the predicted relevance term of the objective. Therefore, they can produce qualitatively different solutions when measured with relevance-based metrics on hold-out data. In addition to this, both optimize Binomial Diversity equally well, while the flow based method increases intra-list distance and aggregate diversity better than the greedy algorithm. The two methods' overall results are

Method	P@10	ERR-IA@10	ILD@10	BD@10	TUDiv@10	TIDiv@10	A@10	G@10
TOP	0.315	0.078	0.266	0.530	0.119	0.167	0.346	0.070
xQuAD ($\lambda = 0.2$)	**0.317**	0.081	0.271	0.545	0.125	0.167	0.357	0.072
MMR ($\lambda = 0.4$)	0.302	0.076	**0.279**	0.631	0.111	0.163	0.356	0.070
BD ($\lambda = 0.8$)	0.285	**0.092**	0.268	0.652	0.113	0.163	0.372	0.075
Flow ($\beta = 0.2, \mu = 0.4$)	0.277	0.055	0.246	0.599	0.118	0.205	**0.689**	0.134
Flow ($\beta = 0.2, \mu = 0.1$)	0.281	0.056	0.251	0.601	0.125	0.201	0.627	0.120
Greedy ($\beta = 0.2, \mu = 4.0$)	0.258	0.081	0.224	0.590	0.132	**0.227**	0.674	**0.141**
Greedy ($\beta = 8.0, \mu = 0.2$)	0.251	0.086	0.216	0.589	**0.136**	0.217	0.616	0.135

TABLE 4.5: MovieLens diversifications based on movie studio based categories and age group based types. The best value in each metric is bolded.

Method	P@10	ERR-IA@10	ILD@10	BD@10	TUDiv@10	TIDiv@10	A@10	G@10
TOP	0.315	0.078	0.266	0.530	0.119	0.140	0.346	0.070
xQuAD ($\lambda = 0.2$)	**0.317**	0.081	0.271	0.545	0.125	0.139	0.357	0.072
MMR ($\lambda = 0.4$)	0.302	0.076	**0.279**	0.631	0.111	0.136	0.356	0.070
BD ($\lambda = 0.8$)	0.285	**0.092**	0.268	**0.652**	0.113	0.135	0.372	0.075
Flow ($\beta = 0.2, \mu = 0.4$)	0.276	0.055	0.245	0.598	0.117	0.174	**0.690**	0.133
Flow ($\beta = 0.2, \mu = 0.1$)	0.282	0.056	0.251	0.600	0.122	0.173	0.618	0.117
Greedy ($\beta = 0.2, \mu = 4.0$)	0.260	0.081	0.225	0.593	0.133	**0.196**	0.660	**0.137**
Greedy ($\beta = 8.0, \mu = 0.2$)	0.253	0.087	0.217	0.590	**0.137**	0.185	0.613	0.131

TABLE 4.6: MovieLens diversifications based on movie studio based categories and occupation based types. The best value in each metric is bolded.

Method	P@10	ERR-IA@10	ILD@10	BD@10	TUDiv@10	TIDiv@10	A@10	G@10
TOP	0.315	0.078	0.266	0.530	0.119	0.184	0.346	0.070
xQuAD ($\lambda = 0.2$)	**0.317**	0.081	0.271	0.545	0.125	0.185	0.357	0.072
MMR ($\lambda = 0.4$)	0.302	0.076	**0.279**	0.631	0.111	0.179	0.356	0.070
BD ($\lambda = 0.8$)	0.285	**0.092**	0.268	**0.652**	0.113	0.182	0.372	0.075
Flow ($\beta = 0.2, \mu = 0.4$)	0.277	0.055	0.247	0.599	0.117	0.220	**0.688**	0.136
Flow ($\beta = 0.2, \mu = 0.1$)	0.282	0.056	0.252	0.601	0.127	0.219	0.631	0.121
Greedy ($\beta = 0.2, \mu = 4.0$)	0.260	0.081	0.226	0.593	0.133	**0.246**	0.669	**0.142**
Greedy ($\beta = 8.0, \mu = 0.2$)	0.252	0.087	0.217	0.592	**0.137**	0.236	0.617	0.137

TABLE 4.7: MovieLens diversifications based on movie studio categories and gender based types data. The best value in each metric is bolded.

Method	Greedy	Flow	MMR	xQuAD	BD
Runtime (s)	5.83	20.3	8.18	11.25	31.3

TABLE 4.8: Running time of the 5 different rerankers on the diversification task in Table 4.5 .

similar enough that if precision is not as big a concern as intent-aware diversification, the two algorithms can be used interchangeably.

This is a significant finding as our flow-based algorithms, while more accurate, takes more time to run to completion. In particular, our greedy algorithms have runtime proportional to $O(|E|\log(|E|))$ where $|E|$ is the number of candidate edges, while our flow-based algorithms have complexity at least $O(|E|(|R|+|L|))$ and significantly higher overheads. While flow problems derived from medium-sized datasets such as MovieLens-1m or even MovieLens-10m can be solved in a matter of seconds on a desktop computer, the greedy algorithm can scale to much larger datasets. Moreover, it is the fastest among the methods we tested, which can be seen in Table 4.8. Since all edges are considered at once, the greedy and flow algorithms have higher space requirements than the other rerankers we tested. However, the simple form of our objectives makes it possible to make incremental updates to the priority of the edges in the priority queue used by the greedy algorithm (see Theorem 4.11), which avoids the problem of having to compute a complicated objective for every candidate recommendation for a given user in each iteration.

4.8 Conclusions and Future Work

In this work we introduced two new recommendation diversity metrics: $TUDiv$ which is similar to other category aware metrics such as Binomial Diversity, and $TIDiv$, which is a novel intent-aware diversity metric for increasing the diversity of users an item sees. We demonstrated mathematically that these metrics can be optimized either exactly or approximately, while keeping average recommendation quality high across the recommender system as measured by relevance-aware metrics. In the future, we would like to run an empirical analysis of our methods on news recommender data, where the idea of disjoint types and categories makes natural sense, and where our methods can increase the feedback received from different types of users, increasing the likelihood of flagging undesirable outcomes like the dissemination of fake news. Unfortunately, we could not find a suitable publicly available dataset on which we could perform these tests. Moreover, it is difficult to obtain any demographic data on the users in most publicly available datasets due to increased concerns of identifiability. MovieLens is a notable exception to

this trend. We hope that the curators of these datasets find a way to protect both the privacy of the their users and provide context-rich datasets which enable further lines of research.

Acknowledgements: This research for this chapter was conducted jointly with my advisor, R Ravi, and Tanvi Bajpai.

Chapter 5

Conclusions

In this work we have proposed three different optimization based approaches for increasing sales diversity. Our models provide answers to natural problems that modern businesses relying on recommender systems face:

1. How can we leverage the popular parts of an item catalog in order to spotlight less popular items? (Chapter 2)
2. How can we create a recommender system which produces recommendations with a prescribed distribution in order to satisfy business objectives? (Chapter 3)
3. How can we measure and alleviate the effects of echo chambers, and collect more reliable feedback from users by showing each recommended items to different types of users? (Chapter 4)

We address each of these questions with a unified framework of subgraph selection, and a strategy of explicit optimization for the various metrics we define. Since customers interact with recommender systems daily, and their consumption choices are dictated by the options they are presented by a business, there is a significant need for a business to be in complete control of the distribution of recommendations across their sites. Sales diversity is a notion that is distinct from diversity among a user's recommendation list, or the novelty of the recommendations a user faces. By conducting experiments on publicly available datasets, we demonstrate that this our approach is superior to previous attempts at increasing sales diversity which have focused on heuristic approaches and novelty at a user-level. We hope that our contributions motivate further research into sales diversity maximization.

Bibliography

[1] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, pages 263–272. IEEE, 2008.

[2] Yehuda Koren. The bellkor solution to the netflix grand prize. *Netflix prize documentation*, 81, 2009.

[3] Greg Linden, Brent Smith, and Jeremy York. Amazon. com recommendations: Item-to-item collaborative filtering. *Internet Computing, IEEE*, 7(1):76–80, 2003.

[4] A. S. Das, M. Datar, A. Garg, and S. Rajaram. Google news personalization: scalable online collaborative filtering. In *Proceedings of the 16th international conference on World Wide Web*, pages 271–280. ACM, 2007.

[5] Eli Pariser. *The filter bubble: How the new personalized web is changing what we read and how we think*. Penguin, 2011.

[6] Q Vera Liao and Wai-Tat Fu. Beyond the filter bubble: interactive effects of perceived threat and topic involvement on selective exposure to information. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 2359–2368. ACM, 2013.

[7] Gediminas Adomavicius, Jesse C Bockstedt, Shawn P Curley, and Jingjing Zhang. Do recommender systems manipulate consumer preferences? a study of anchoring effects. *Information Systems Research*, 24(4):956–975, 2013.

[8] Charles LA Clarke, Maheedhar Kolla, Gordon V Cormack, Olga Vechtomova, Azin Ashkan, Stefan Büttcher, and Ian MacKinnon. Novelty and diversity in information retrieval evaluation. In *Proc. of the 31st Internation ACM SIGIR Conf. on Research and development in information retrieval*, pages 659–666. ACM, 2008.

[9] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Trans. on Knowl. and Data Eng.*, 17(6):734–749, June 2005.

[10] Pearl Pu, Li Chen, and Rong Hu. Evaluating recommender systems from the users perspective: survey of the state of the art. *User Modeling and User-Adapted Interaction*, 22(4):317–355, 2012.

[11] Kensuke Onuma, Hanghang Tong, and Christos Faloutsos. Tangent: a novel, 'surprise me', recommendation algorithm. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 657–666. ACM, 2009.

[12] 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.

[13] Erik Brynjolfsson, Yu Hu, and Michael D Smith. Consumer surplus in the digital economy: Estimating the value of increased product variety at online booksellers. *Management Science*, 49(11):1580–1596, 2003.

[14] Daniel G Goldstein and Dominique C Goldstein. Profiting from the long tail. *Harvard Business Review*, 84(6):24–28, 2006.

[15] Sean M McNee, John Riedl, and Joseph A Konstan. Being accurate is not enough: how accuracy metrics have hurt recommender systems. In *CHI'06 Human factors in computing systems*, pages 1097–1101. ACM, 2006.

[16] Mi Zhang and Neil Hurley. Avoiding monotony: improving the diversity of recommendation lists. In *Proc. of the 2008 ACM Conf. on Rec. Systems*, pages 123–130. ACM, 2008.

[17] Arda Antikacioglu, R Ravi, and Srinath Sridhar. Recommendation subgraphs for web discovery. In *Proc. of the 24th Int. Conf. on World Wide Web*, pages 77–87. Int. World Wide Web Conf.s Steering Committee, 2015.

[18] BloomReach Inc. Inside the technology: Web relevance engine. URL http://go.bloomreach.com/rs/bloomreach/images/WP-Web-Relevance-Engine-4-12.pdf.

[19] J. B. Schafer, J. Konstan, and J. Riedi. Recommender systems in e-commerce. In *Proceedings of the 1st ACM conference on Electronic commerce*, EC '99, pages 158–166. ACM, 1999.

[20] P. Resnick and H. R. Varian. Recommender systems. *Communications of the ACM*, 40(3):56–58, 1997.

[21] D. Almazro, G. Shahatah, L. Albdulkarim, M. Kherees, R. Martinez, and W. Nzoukou. A survey paper on recommender systems. *arXiv preprint arXiv:1006.5278*, 2010.

[22] J. Hannon, M. Bennett, and B. Smyth. Recommending twitter users to follow using content and collaborative filtering approaches. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 199–206. ACM, 2010.

[23] G. Linden, B. Smith, and J. York. Amazon. com recommendations: Item-to-item collaborative filtering. *Internet Computing, IEEE*, 7(1):76–80, 2003.

[24] Robert M Bell, Yehuda Koren, and Chris Volinsky. The bellkor solution to the netflix prize.

[25] Chris Anderson. *The Long Tail: Why the Future of Business Is Selling Less of More*. Hyperion, 2006. ISBN 1401302378.

[26] Purnamrita Sarkar, Deepayan Chakrabarti, and Andrew W Moore. Theoretical justification of popular link prediction heuristics. In *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, volume 22, page 2722, 2011.

[27] H. Gabow. An efficient reduction technique for degree-constrained subgraph and bidirected network flow problems. In *Proceedings of the fifteenth annual ACM symposium on Theory of computing*, STOC '83, pages 448–456. ACM, 1983.

[28] Ran Duan and Seth Pettie. Approximating maximum weight matching in near-linear time. In *Foundations of Computer Science (FOCS), 2010 51st Annual IEEE Symposium on*, pages 673–682. IEEE, 2010.

[29] Christos Koufogiannakis and Neal E Young. Distributed fractional packing and maximum weighted b-matching via tail-recursive duality. In *Distributed Computing*, pages 221–238. Springer, 2009.

[30] George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. An analysis of approximations for maximizing submodular set functionsi. *Mathematical Programming*, 14(1):265–294, 1978.

[31] L Lovász and M. D. Plummer. *Matching theory*. North-Holland mathematics studies. Akadémiai Kiadó, 1986. ISBN 9789630541688.

[32] B. A. Huberman and L. A. Adamic. Internet: growth dynamics of the world-wide web. *Nature*, 401(6749):131–131, 1999.

[33] B. Du, M. Demmer, and E. Brewer. Analysis of www traffic in cambodia and ghana. In *Proceedings of the 15th international conference on World Wide Web*, WWW '06, pages 771–780. ACM, 2006.

[34] C. Kumar, J. B. Norris, and Y. Sun. Location and time do matter: A long tail study of website requests. *Decision Support Systems*, 47(4):500–507, 2009.

[35] J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. In *OSDI '04: Proceedings of the sixth conference on symposium on operating systems design and implementation*. USENIX Association, 2004.

[36] S. Janson, T. Luczak, and A. Rucinski. *Random Graphs*. Wiley Series in Discrete Mathematics and Optimization. Wiley, 2011. ISBN 9781118030967.

[37] A. Auger and B. Doerr. *Theory of Randomized Search Heuristics: Foundations and Recent Developments*. Series on Theoretical Computer Science. World Scientific Publishing Company, 2011. ISBN 9789814282666.

[38] R. M. Karp, U. Vazirani, and V. Vazirani. An optimal algorithm for on-line bipartite matching. In *Proceedings of the twenty-second annual ACM symposium on Theory of computing*. ACM, 1990.

[39] P. Erdös and A. Rényi. On random graphs, I. *Publicationes Mathematicae*, 6: 290–297, 1959.

[40] J. E. Hopcroft and R. M. Karp. An nˆ5/2 algorithm for maximum matchings in bipartite graphs. *SIAM Journal on computing*, 2(4):225–231, 1973.

[41] Òscar Celma and Pedro Cano. From hits to niches?: or how popular artists can bias music recommendation and discovery. In *Proc. of the 2nd KDD Workshop on Large-Scale Recommender Systems*, page 5. ACM, 2008.

[42] Renjie Zhou, Samamon Khemmarat, and Lixin Gao. The impact of youtube recommendation system on video views. In *Proc. of the 10th ACM SIGCOMM Conf. on Internet measurement*, pages 404–410. ACM, 2010.

[43] Cathy O'Neil. *Weapons of math destruction: How big data increases inequality and threatens democracy*. Crown Publishing Group (NY), 2016.

[44] Cecilia Mazanec. Will algorithms erode our decision-making skills?, 2016. URL http://www.npr.org/sections/alltechconsidered/2017/02/08/514120713/will-algorithms-erode-our-decision-making-skills. Accessed: 02/2016.

[45] Endel Tulving, Hans J Markowitsch, Shitij Kapur, Reza Habib, and Sylvain Houle. Novelty encoding networks in the human brain: positron emission tomography data. *NeuroReport*, 5(18):2525–2528, 1994.

[46] Saúl Vargas and Pablo Castells. Rank and relevance in novelty and diversity metrics for recommender systems. In *Proc. of the 5th ACM Conf. on Rec. Systems*, pages 109–116. ACM, 2011.

[47] Erik Brynjolfsson, Yu Hu, and Duncan Simester. Goodbye pareto principle, hello long tail: The effect of search costs on the concentration of product sales. *Management Science*, 57(8):1373–1386, 2011.

[48] Alexander Schrijver. *Combinatorial optimization: polyhedra and efficiency*, volume 24. Springer Science & Business Media, 2002.

[49] Gediminas Adomavicius and YoungOk Kwon. Improving aggregate recommendation diversity using ranking-based techniques. *IEEE Trans. on Knowl. and Data Eng.*, 24(5):896–911, 2012.

[50] Péter Kovács. Minimum-cost flow algorithms: an experimental evaluation. *Optimization Methods and Software*, 30(1):94–127, 2015.

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

[52] Christian Desrosiers and George Karypis. A comprehensive survey of neighborhood-based recommendation methods. In *Recommender systems handbook*, pages 107–144. Springer, 2011.

[53] Saúl Vargas. Novelty and diversity evaluation and enhancement in recommender systems, 2015.

[54] Colin Cooper, Sang Hyuk Lee, Tomasz Radzik, and Yiannis Siantos. Random walks in recommender systems: Exact computation and simulations. In *Proceedings of the 23rd International Conference on World Wide Web*, pages 811–816. ACM, 2014.

[55] Min Gao, Zhongfu Wu, and Feng Jiang. Userrank for item-based collaborative filtering recommendation. *Information Processing Letters*, 111(9):440–446, 2011.

[56] Marco Gori, Augusto Pucci, V Roma, and I Siena. Itemrank: A random-walk based scoring algorithm for recommender engines. In *IJCAI*, volume 7, pages 2766–2771, 2007.

[57] Jian-Guo Liu, Kerui Shi, and Qiang Guo. Solving the accuracy-diversity dilemma via directed random walks. *Physical Review E*, 85(1):016118, 2012.

[58] Pablo Castells, Saúl Vargas, and Jun Wang. Novelty and diversity metrics for recommender systems: choice, discovery and relevance. 2011.

[59] Kamal Ali and Wijnand Van Stam. Tivo: making show recommendations using a distributed collaborative filtering architecture. In *Proc. of the 10th ACM SIGKDD Int. Conf. on Knowl. discovery and data mining*, pages 394–401. ACM, 2004.

[60] Saúl Vargas. Novelty and diversity enhancement and evaluation in recommender systems. 2015.

[61] Sylvain Senecal and Jacques Nantel. The influence of online product recommendations on consumers online choices. *Journal of Retailing*, 80(2):159–169, 2004.

[62] 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*, pages 257–260. ACM, 2010.

[63] Panagiotis Adamopoulos and Alexander Tuzhilin. On unexpectedness in recommender systems: Or how to expect the unexpected. In *DiveRS@ RecSys*, pages 11–18, 2011.

[64] Guy Shani and Asela Gunawardana. Evaluating recommendation systems. In *Recommender systems handbook*, pages 257–297. Springer, 2011.

[65] 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.

[66] Xiaolong Ren, Linyuan Lü, Runran Liu, and Jianlin Zhang. Avoiding congestion in recommender systems. *New Journal of Physics*, 16(6):063057, 2014.

[67] Zoltán Szlávik, Wojtek Kowalczyk, and Martijn Schut. Diversity measurement of recommender systems under different user choice models. In *Fifth International AAAI Conference on Weblogs and Social Media*, 2011.

[68] 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. ACM, 2014.

[69] Aditya Parameswaran, Petros Venetis, and Hector Garcia-Molina. Recommendation systems with complex constraints: A course recommendation perspective. *ACM Trans. on Inform. Systems (TOIS)*, 29(4):20, 2011.

[70] David Mimno and Andrew McCallum. Expertise modeling for matching papers with reviewers. In *Proc. of the 13th ACM SIGKDD Int. Conf. on Knowl. discovery and data mining*, pages 500–509. ACM, 2007.

[71] Maryam Karimzadehgan and ChengXiang Zhai. Constrained multi-aspect expertise matching for committee review assignment. In *Proc. of the 18th ACM Conf. on Inform. and Knowledge Management*, pages 1697–1700. ACM, 2009.

[72] Cheng Chen, Lan Zheng, Venkatesh Srinivasan, Alex Thomo, Kui Wu, and Anthony Sukow. Conflict-aware weighted bipartite b-matching and its application to e-commerce. *IEEE Trans. on Knowl. and Data Eng.*, 28(6):1475–1488, 2016.

[73] Faraz Makari and Rainer Gemulla. A distributed approximation algorithm for mixed packing-covering linear programs. In *NIPS 2013 Workshop on Big Learning*. NIPS, 2013.

[74] Ravindra Ahuja, Thomas Magnanti, and James Orlin. *Network flows: theory, algorithms, and applications*. Prentice Hall, 1993.

[75] James B Orlin. A polynomial time primal network simplex algorithm for minimum cost flows. *Mathematical Programming*, 78(2):109–129, 1997.

[76] GroupLens. Movielens-1m data set. http://grouplens.org/datasets/movielens/1m/, 2015. Accessed: 03/2015.

[77] Antonio Frangioni and Luis Perez Sanchez. Searching the best (formulation, solver, configuration) for structured problems. In *Complex Systems Design & Management*, pages 85–98. Springer, 2010.

[78] László A Végh. Strongly polynomial algorithm for a class of minimum-cost flow problems with separable convex objectives. In *Proc. of the 44th annual ACM symposium on Theory of computing*, pages 27–40. ACM, 2012.

[79] Jaime Carbonell and Jade Goldstein. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 335–336. ACM, 1998.

[80] Olivier Chapelle, Donald Metlzer, Ya Zhang, and Pierre Grinspan. Expected reciprocal rank for graded relevance. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 621–630. ACM, 2009.

[81] Rodrygo Luis Teodoro Santos. Explicit web search result diversification, 2013.

[82] Saúl Vargas, Linas Baltrunas, Alexandros Karatzoglou, and Pablo Castells. Coverage, redundancy and size-awareness in genre diversity for recommender systems. In *Proceedings of the 8th ACM Conference on Recommender systems*, pages 209–216. ACM, 2014.

[83] Ben Carterette. An analysis of np-completeness in novelty and diversity ranking. In *Conference on the Theory of Information Retrieval*, pages 200–211. Springer, 2009.

[84] Zeinab Abbassi, Vahab S Mirrokni, and Mayur Thakur. Diversity maximization under matroid constraints. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 32–40. ACM, 2013.

[85] Baharan Mirzasoleiman, Ashwinkumar Badanidiyuru, and Amin Karbasi. Fast constrained submodular maximization: Personalized data summarization. In *Proceedings of The Thirty-Third International Conference on Machine Learning*, pages 1358–1367, 2016.

[86] Jon Lee, Vahab S Mirrokni, Viswanath Nagarajan, and Maxim Sviridenko. Non-monotone submodular maximization under matroid and knapsack constraints. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 323–332. ACM, 2009.

[87] Baharan Mirzasoleiman, Amin Karbasi, Rik Sarkar, and Andreas Krause. Distributed submodular maximization: Identifying representative elements in massive data. In *Advances in Neural Information Processing Systems*, pages 2049–2057, 2013.

[88] Maryam Habibi and Andrei Popescu-Belis. Enforcing topic diversity in a document recommender for conversations. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 588–599, Dublin, Ireland, August 2014. Dublin City University and Association for Computational Linguistics. URL http://www.aclweb.org/anthology/C14-1056.

[89] Onur Küçüktunç, Erik Saule, Kamer Kaya, and Ümit V Çatalyürek. Diversified recommendation on graphs: pitfalls, measures, and algorithms. In *Proceedings of the 22nd international conference on World Wide Web*, pages 715–726. ACM, 2013.

[90] Rodrygo LT Santos, Craig Macdonald, and Iadh Ounis. Exploiting query reformulations for web search result diversification. In *Proceedings of the 19th international conference on World wide web*, pages 881–890. ACM, 2010.

[91] David Geiger and Martin Schader. Personalized task recommendation in crowdsourcing information systemscurrent state of the art. *Decision Support Systems*, 65:3–16, 2014.

[92] Man-Ching Yuen, Irwin King, and Kwong-Sak Leung. Task matching in crowdsourcing. In *Internet of Things (iThings/CPSCom), 2011 International Conference on and 4th International Conference on Cyber, Physical and Social Computing*, pages 409–412. IEEE, 2011.

[93] Mejdl Safran and Dunren Che. Real-time recommendation algorithms for crowdsourcing systems. *Applied Computing and Informatics*, 2016.