Equitable Top-k Results for Long Tail Data

MD MOUINUL ISLAM, New Jersey Institute of Technology, USA
MAHSA ASADI, New Jersey Institute of Technology, USA
SENJUTI BASU ROY, New Jersey Institute of Technology, USA

For datasets exhibiting *long tail* phenomenon, we identify a fairness concern in existing top-k algorithms, that return a "fixed" set of k results for a given query. This causes a handful of *popular* records (products, items, etc) getting overexposed and always be returned to the user query, whereas, there exists a long tail of *niche* records that may be equally desirable (have similar utility). To alleviate this, we propose θ-**Equiv-top-**k**-MMSP** inside existing top-k algorithms - instead of returning a fixed top-k set, it generates all (or many) top-k sets that are *equivalent* in utility and creates a probability distribution over those sets. The end user will be returned one of these sets during the query time proportional to its associated probability, such that, after many draws from many end users, each record will have as equal exposure as possible (governed by uniform selection probability). θ-**Equiv-top-**k**-MMSP** is formalized with two sub-problems. (a) θ-**Equiv-top-**k**-Sets** to produce a set S of sets, each set has k records, where the sets are *equivalent in utility* with the top-k set; (b) **MaxMinFair** to produce a probability distribution over S, that is, $PDF(S)$, such that the records in S have uniform selection probability. We formally study the hardness of θ-**Equiv-top-**k**-MMSP**. We present multiple algorithmic results - (a) An exact solution for θ-**Equiv-top-**k**-Sets**, and **MaxMinFair**. (b) We design highly scalable algorithms that solve θ-**Equiv-top-**k**-Sets** through a random walk and is backed by probability theory, as well as a greedy solution designed for **MaxMinFair**. (c) We finally present an adaptive random walk based algorithm that solves θ-**Equiv-top-**k**-Sets** and **MaxMinFair** at the same time. We empirically study how θ-**Equiv-top-**k**-MMSP** can alleviate a equitable exposure concerns that group fairness suffers from. We run extensive experiments using 6 datasets and design intuitive baseline algorithms that corroborate our theoretical analysis.

CCS Concepts: • **Information systems** → **Top-k retrieval in databases**.

Additional Key Words and Phrases: equal exposure, fairness, long tail data, top-k retrieval, optimization algorithm

ACM Reference Format:
Md Mouinul Islam, Mahsa Asadi, and Senjuti Basu Roy. 2023. Equitable Top-k Results for Long Tail Data. *Proc. ACM Manag. Data* 1, 4 (SIGMOD), Article 240 (December 2023), 24 pages. https://doi.org/10.1145/3626727

1 INTRODUCTION

The proliferation of e-commerce platforms such as Amazon.com, Netflix, and Spotify.com has given rise to the so-called "infinite-inventory", which offer an order of magnitude more records (products, movies, songs) than their brick-and-mortar counter-parts [5]. This results in a long-tail market, where a handful of records get heavily exposed to the end users and a long tail of "niche" records remain relatively unknown. As a concrete example, the top-1000 highest rated movies in IMDB [31]

Authors' addresses: Md Mouinul Islam, New Jersey Institute of Technology, Newark, New Jersey, USA, 07102, mi257@njit.edu; Mahsa Asadi, ma2266@njit.edu, New Jersey Institute of Technology, Newark, New Jersey, USA, 07102; Senjuti Basu Roy, New Jersey Institute of Technology, Newark, New Jersey, USA, 07102, senjutib@njit.edu.

2836-6573/2023/12-ART240
https://doi.org/10.1145/3626727

follow a long tail distribution in terms of number of views (refer to Y-axis in Figure 1), even though they all have highly similar (average rating between 8.34 and 7.9)"utility" (IMDB ratings).

In Section 2.1 we describe the current process with a running example on the aforementioned IMDB-1000 datasets, how it leads to inequitable exposure of movies, and how we intend to redesign existing top-k algorithms to circumvent that. Our proposed solution advocates to return one of the *equivalent* top-k sets to the end users in a probabilistic manner, such that, after many such draws by many end users, the exposure of the records are as equitable as possible. The same static answer could still be returned if the application warrants - but when users pose generic queries [37] (e.g., top-3 movies, books) on long tail data, this will unveil interesting movies, songs, and products, that the users will not experience otherwise. *To the best of our knowledge, we are the first to study this aspect of unequal exposure inside top-k algorithms that is agnostic to any specific scoring functions.*

Problem Motivation and Models. We adapt a political theory, namely, the *Sortition Act* [15, 45] and redesign existing top-k algorithms to have them compute *a set S of multiple top-k sets that are equivalent in utility* as opposed to a fixed top-k set. Given S, an end user still draws one of the sets at random. Hence, the goal is to assign a probability distribution over S, i.e., $PDF(S)$, such that after many such draws from many end users, the records returned inside the top-k sets have as uniform selection probability as possible. We formalize **θ-Equiv-top-k-MMSP** that produces $PDF(S)$ for a given query and a scoring function $\mathcal{F}$. Each set $s \in S$ contains k number of records whose score is at most θ% (a tunable application dependent input parameter) smaller than the optimum top-k score, and the $PDF(S)$ is computed such that the selection probabilities of the records in it are as uniform as possible. Enabling equal selection probabilities promotes equal exposure of the records. **θ-Equiv-top-k-MMSP** is rooted on maxmin fairness theory that maximizes the minimum exposure. We are aware of a few related works that we borrow inspirations from. [7] studies how to enable equal exposure in similarity search by returning points within distance r from the given query with the same probability. The bulk of the algorithmic fairness literature deals with group fairness along the lines of demographic parity[32, 48]: this is typically expressed by means of some fairness constraint requiring that the top-k results (for any k) to contain enough records from some groups that are protected from discrimination based on sex, race, age, etc. In practice these group fairness constraints hurt equitable exposure [9, 21, 25] owing to differential participation rates across sub population. Both [21, 25] study how group fairness alone can hurt equitable exposure of the records and thus define computational frameworks to promote equal selection probability in group fairness. These existing works do not have any easy extension to top-k algorithms. We study how **θ-Equiv-top-k-MMSP** alleviates exposure based fairness concerns that demographic parity based group fairness (e.g., top-k parity [32], proportionate fairness [48]) give rise to.

Technical Contributions. We formalize key definitions, such as, θ-equivalent top-k sets, selection probability of records, and present **θ-Equiv-top-k-MMSP** that has two steps **(Section 2)**. (A) **θ-Equiv-top-k-Sets** generates S, the set of θ equivalent top-k sets (where θ is a tunable parameter that can control how much change is desirable across different top-k sets for different applications), (B) **MaxMinFair** computes $PDF(S)$ such that the minimum selection probability of a record is maximized. We prove that the counting problem involved in **θ-Equiv-top-k-Sets** is #P-hard, which makes **θ-Equiv-top-k-MMSP** an NP-Complete problem.

In **Section 3**, we first present an exact algorithm **OptTop-k-θ** that produces S, all θ-equivalent top-k sets and is exact in nature. We also study efficient alternatives later, which only computes a few θ equivalent top-k sets (as opposed to all). The exact algorithm is inspired by the celebrated NRA algorithm [19] but *not an easy adaptation, because of the exponential nature of **θ-Equiv-top-k-Sets***. At the heart of the process, **OptTop-k-θ** intends to maintain a set of candidate top-k sets, efficiently compute and maintain their best and worst possible scores through upper and lower bounds, and decide if it is safe to terminate and produce the exact S without having to read any more

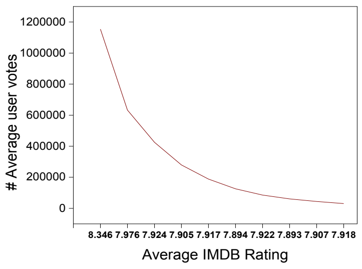


Fig. 1. Viewership distribution of top-1000 IMDB movies

records. However, because the number of possible size-k sets increases exponentially with new records being read, **OptTop-k-θ** leverages an efficient data structure based on the concept of item lattice that allows efficient computation of the possible size-k sets and incremental updates of their score bounds by reusing previously calculated scores. For producing $PDF(S)$, we present a linear programming-based exact solution **Opt-SP**. For **OptTop-k-θ** , the storage space and computational cost of this lattice is $O(\binom{N}{k})$, which is the theoretical lower bound, but the same structure could be made significantly lightweight, if approximation is allowed, as we discuss in Section 4.
In **Section 4.1**, we present **RWalkTop-k-θ** that is highly scalable to solve both **θ-Equiv-top-k-Sets** and **MaxMinFair**. It makes use of the same item lattice structure described above, but builds it only partially on the go, making it significantly lightweight. **RWalkTop-k-θ** is a *probabilistic algorithm* based on random walk on the lattice that is backed by the Good Turing Test [24]. Good Turing Test is often used in population studies to estimate the number of unique species in a large unknown population [24], which we use to determine when **RWalkTop-k-θ** could stop and still discover all θ-equivalent top-k sets with high probability. Given S, **RWalkTop-k-θ** calls a highly efficient greedy solution **Gr-SP** to produce a probability distribution over it.
In **Section 4.2**, we finally design **ARWalkTop-k-θ**, an adaptive random walk based approach that solves **θ-Equiv-top-k-Sets** and **MaxMinFair** at the same time. The intuition comes from the fact (that we formally prove in the paper) that if S contains records that only appears in one and exactly one set $s \in S$, then $PDF(S)$ is a uniform probability distribution which ensures equal selection probabilities for all records. **ARWalkTop-k-θ** is similar to the random walk described in **RWalkTop-k-θ**, except it performs the random walk adaptively, by lowering the probability of the records that are already part of some valid s, and boosting the probability of the remaining records that have not been part of any valid s yet. After that, $PDF(S)$ becomes a uniform probability distribution over the sets produced during the adaptive random walk.
Experimental Evaluations (Section 5). Our final contributions are empirical. As discussed above, equal exposure is orthogonal to demographic parity based group fairness, such as, top-k-parity or proportionate fairness [32, 48], but we show it can further alleviate biases that group fairness alone gives rise to. Our results corroborate that when **θ-Equiv-top-k-MMSP** is integrated inside top-k-parity, the returned results satisfy both equitable exposure as well as group fairness criteria, which top-k parity alone is unable to promote. We use 4 different large scale real world datasets and two large synthetic datasets to extensively evaluate our designed solutions and compare them against several intuitive baseline algorithms. Our experimental evaluations also corroborate our theoretical analysis, it terms of the quality and the scalability of the designed solutions.

Section 6 contains the related work and we conclude in Section 7, giving future research directions.

2 DATA MODEL & PROBLEM DEFINITION

In this section, we present a running example, introduce key notations used throughout the paper (Table 1), describe our data model, present key definitions, formalize θ-**Equiv-top-**k**-MMSP**, and study its hardness.

2.1 Running Example

Consider the IMDB-1000 datbase D. The attributes are movie name, IMDB rating, year, genre, and director. Assume that a user writes a query (q) to search for top-3 movies ($k = 3$) released in year 2022.

Imagine only 5 movies as described in Table 2 are released in 2022 and they have highly similar IMDB ratings. Let the scoring/utility function $\mathcal{F}$ be the weighted relevance and max sum diversity (WRMSD in short), as proposed below (with $\lambda = 0.5$). Let IMDB ratings reflect the relevance scores of the records and diversity be computed considering genre and director values. The sorted pairwise diversity is given in Table 3.

In the set $s1 = \{r_2, r_3, r_5\}$, the utility score of r_2, r_3, r_5 are 6.75, 6.65, 6.45, leading to the maximum utility score of top-3 movies to be 19.85, as shown in 4. Static top-k algorithms will always return $\{r_2, r_3, r_5\}$, whereas, $s2 = \{r_1, r_2, r_3\}$, $s3 = \{r_2, r_3, r_4\}$, $s4 = \{r_1, r_3, r_5\}$, may also be equally desirable (all have items with high utility, leading to high set score above 19). However, if only $s1$ is always returned, this leads to little to no exposure of movies r_1, r_4.

We advocate for an alternative process, where, there exists a tunable parameter θ, which will empower the application designer to introduce variability in the top-k results to the end users (if the application warrants the same static answer, θ could be set to 0). For long tail data with generic queries [37], this process may bring forth additional interesting movies, products, songs to the end users. If $\theta = 0.03$, the goal is to create a set S of top-k sets, such that each $s \in S$ has utility score $\geq (19.85 - [0.03 \times 19.85]) = 19.25$. It is easy to notice that even with only 5 records, there are three additional sets $\{s2, s3, s4\}$ that satisfy this condition (Table 4).

The top-k interface however still allows users to see only one set of k results. Thus, given S, our goal is to create a probability distribution over it, $PDF(S)$. A user draws one s from S corresponding to its associated probability, such that, after many draws from many end users, the movies in S have as uniform selection probabilities as possible. Creating $PDF(S)$ is non-trivial - if one associates uniform probability (0.25) to each of the 4 sets, then, r_3 will always be over exposed (quantified by its selection probability, which is also formalized in this section), as it will always be returned to the end users, leading to 1 selection probability, whereas, r_4 will be heavily underexposed. The selection probabilities of $r_1 = 0.5$, $r_2 = 0.75$, $r_5 = 0.5$, and that of r_4 is only 0.25, as r_4 is present in only $s3$ out of the 4 sets. Our effort here is thus to produce $PDF(S)$ such that the movies in S have as uniform selection probabilities as possible.

2.2 Data Model

Database. A database D contains N records, where each record is represented as r.
Top-k **Query.** A top-k query q intends to return k answers from D. We are especially interested in generic queries (e.g., top vacation spots, top movies, good books, etc).

2.2.1 *Utility Based Scoring Functions.*

Given a query q and D, a utility based scoring function $\mathcal{F}$ scores each record with utility value $\mathcal{F}(r, q)$ and produces $\mathcal{F}(s, q), r \in s, |s| = k$, which is the the aggregated utility score of set s with k records.

- Relevance: $\mathcal{F}(r, q) = Rel(r, q)$, where Rel is the *relevance* between record r and query q.
- Diversity: Diversity is the dissimilarity between any two records, $Div(r_i, r_j)$ that is used to capture results that are representative of the population.

The attributes of the records could be used to calculate these values. Tables 2, 3 have some of those for Example 2.1.

Representative $\mathcal{F}$. Some representative utility functions appear as follows.

- Sum-relevance. $\mathcal{F}(s, q) = \Sigma_{r \in s} Rel(r, q)$
- Weighted relevance and max sum diversity (WRMSD). $\mathcal{F}(s, q) = \lambda \times \Sigma_{r \in s} Rel(r, q) + (1 - \lambda) \times \Sigma_{r \in s} Max_{r, r_j \in \{s-r\}}$ $Div(r, r_j)$, where λ is a weight between $[0, 1]$.
- Maximal marginal relevance [12] or MMR. $\mathcal{F}(s, q) = \lambda \times \Sigma_{r \in s} Rel(r, q) + (1 - \lambda) \times \Sigma_{r \in s} Min_{r, r_j \in \{s-r\}} Div(r, r_j)$

The proposed framework is generic and extensible to any utility function, however, as we shall see later that the exact solution θ-**Equiv-top-**k**-Sets** requires the function to be monotonic.

2.2.2 *Top-k Algorithms.* Given D, q, and an integer k, return a set s of k records from D that has the highest $\mathcal{F}(s, q)$, i.e.,

- $|s| = k$;
- s has the highest utility score, i.e., for any other set of k records s', $\mathcal{F}(s, q) \geq \mathcal{F}(s', q)$.

2.2.3 *Promoting Fairness inside Top-k Algorithms.* It is easy to see that there could be more than one set of k-records that have highly similar utility score. To that end, we define the notion of equivalent size-k sets.

Definition 2.1. **Equivalent size k sets.** Given a threshold θ, a query q and size k, two sets s_i and s_j each with k records are equivalent if the score of the set with lower score is not smaller than *a predefined threshold θ%* of that with the higher score, i.e.,
$s_i \equiv s_j$ if $\mathcal{F}(s_i, q) \geq (1 - \theta) \times \mathcal{F}(s_j, q)$, when $\mathcal{F}(s_i, q) < \mathcal{F}(s_j, q)$

Running Example. In the context of example 2.1, when WRMSD is considered as the scoring function and $\theta = 0.03$, two equivalent size k sets with scores 19.85 and 19.7 are $s1 = \{r_2, r_3, r_5\}$, $s2 = \{r_1, r_2, r_3\}$, respectively.

Definition 2.2. **Probability Distribution over size k sets.** Given a set S of sets, each with k records, a probability distribution $PDF(S)$ assigns a probability $P(s)$ to each $s \in S$, such that $\sum_{s \in S} P(s) = 1$.

Definition 2.3. **Selection probability of a record.** Given a probability distribution $PDF(S)$ of a set S containing many size k sets, the selection probability [21] of a record r is the sum of probability values of all the sets that contain r.

$$\mathcal{P}(r) = \sum_{r \in s, s \in S} P(s) \tag{1}$$

Running Example. Considering the running example, uniform probability distribution $P(s_1) = P(s_2) = P(s_3) = P(s_4) = 1/4$, leads to selection probability $\mathcal{P}(r_4) = P(s3) = 1/4$, whereas, $\mathcal{P}(r_3) = P(s1) + P(s2) + P(s3) + P(s4) = 1$. Indeed, no matter which set the end users draw, r_3 will always be returned, whereas, r_4 will be returned only $1/4$ of the time.

2.3 Problem Definition & Hardness

Our overarching goal is to produce top-k set of sets that are "equivalent" in utility w.r.t. the set with the highest utility (i.e., the optimum top-k set), and ensure that all records present in any of the equivalent top-k sets have an equal selection probability. Generally speaking, we adapt the Egalitarian Social Welfare notion [17], which maximizes the lowest selection probability of a record present in any top-k sets.

Problem Definition 1. (θ-Equiv-top-k-MMSP) Maximize Minimum Selection Probability in θ-Equivalent Top-k Sets.
Given a database D with N records, scoring function $\mathcal{F}$, threshold θ, query q, and integer k, produce a set S of equivalent top-k sets and a probability distribution $PDF(S)$ over S, such that, the minimum selection probability of a record present in any $s \in S$ is maximized. Specifically, we define the following two sub-problems.

- **θ-Equiv-top-k-Sets.** *Produce a set S of all θ-equivalent top-k sets, such that, $s \in S$ satisfies: $\mathcal{F}(s,q) \geq (1-\theta) \times argmax_{s' \in S}\mathcal{F}(s',q)$*
- **MaxMinFair.** *Compute probability distributions S such that the smallest selection probability $\mathcal{P}(r)$ of a record $r \in s, s \in S$ is maximized. That is:*

$$Maximize\ Min\ \mathcal{P}(r), r \in s, s \in S, \tag{2}$$

In general, our proposed framework can accommodate any scoring function. However, when the scoring function is non-monotone, such as, MMR [12], the designed solutions become approximation.

Theorem 2.4. *The problem of finding the number of θ-**Equiv-top-k-Sets** is #P-hard.*

Proof. We show a polynomial time reduction from the problem of computing all maximal frequent itemsets of size at most t [27, 49] to the problem of computing all θ-equivalent top-k sets, that has a simple mapping between the number of solutions. This suffices since the problem of finding the number of σ-frequent maximal itemsets (threshold $\sigma \in [0,1]$) with at most t items of a given 0-1 database D is known to be #P-hard [49].

We take an instance of such 0-1 database with m transactions over N items. The σ is set to be $1/m$. Given one such instance of a 0-1 database, we create an instance of our problem as follows: each item becomes a unique record r, such that $\mathcal{F}(r,q) = 1$, for an arbitrary query q. $\mathcal{F}(s,q) = \Sigma_{\forall r \in s}\mathcal{F}(r,q)$. θ is set to be any number between $[0,1]$. A set of items is σ-frequent maximal itemset of size at most k, iff the set of records corresponding to the itemset forms a set s with score $\mathcal{F}(s,q) = k$. Therefore, the number of θ-equivalent top-k sets is at least as many as the number of σ frequent maximal itemsets of size at most k. This completes the reduction. □

Theorem 2.5. *The θ-**Equiv-top-k-MMSP** problem is NP-Complete.*

Proof. (sketch) We omit the details for brevity. Intuitively, the hardness comes from the fact that θ-**Equiv-top-k-MMSP** needs to enumerate all θ-equivalent top-k sets, which is at least as hard as counting all such sets that is proved to be #P-hard. □

Symbol	Definition
N	# records in D
k, q	size of result sets, query
θ,s, S	equivalence threshold, a top-k set, θ-equivalent top-k sets
C, $\mathcal{L}$, $\mathcal{F}$	candidate set, sorted input lists, scoring function
$\mathcal{P}(r)$	selection probability of record r

Table 1. Table of notations

3 EXACT ALGORITHMS

We first describe an exact solution that solves both the sub-problems θ-**Equiv-top-k-Sets** and **MaxMinFair** exactly, thereby ensuring exact solution for θ-**Equiv-top-k-MMSP**.

The framework is described in Algorithm 1. To solve θ-**Equiv-top-k-Sets**, it runs in a loop and finds the i-th best top-k set in the i-th iteration - that is, $\mathcal{F}(s,q) = \textbf{TopkSets}(i) \geq \mathcal{F}(s',q) = \textbf{TopkSets}(j)$, where $i < j$. It maintains all records that are seen throughout. This process continues

Record	Movie Name	IMDB Score
r1	Top Gun: Maverick	8.6
r2	K.G.F: Chapter 2	8.5
r3	Everything Everywhere All at Once	8.3
r4	RRR	8.1
r5	The Batman	7.9

Table 2. Records with sorted relevance (Example 2.1)

Pair of records	**(r2,r3)**	**(r3,r5)**	**(r1,r3)**	**(r3,r4)**	**(r1,r4)**	**(r4,r5)**	**(r1,r2)**	**(r2,r4)**	**(r2,r5)**	**(r1,r5)**
Diversity Score	5	5	4	4	2	2	2	2	1	1

Table 3. Sorted diversity list based on Example 2.1

sets	**s2:{r1,r2, r3}**	**{r1,r2, r4}**	**{r1,r2,r5}**	**{r1,r3,r4}**	**s4:{r1,r3,r5}**	**{r1,r4,r5}**	**s3:{r2,r3,r4}**	**s1:{r2,r3,r5}**	**{r2,r4,r5}**	**{r3,r4,r5}**
Utility Score	19.7	15.6	14.5	18.5	19.4	15.3	19.45	19.85	15.25	19.15

Table 4. WRMSD scores of all set of sets, each with 3 movies

Algorithm 1 Generic Framework for θ-**Equiv-top-**k**-MMSP**

```
    Inputs: $q$, $k$, $\theta$, database $D$, $\mathcal{F}$
    Outputs: $PDF(S)$: probability distribution over a set $S$ of top-k sets
 1: $flag = 0$
 2: $Opt = \infty$
 3: $s = \textsc{TopkSets}(1)(\mathcal{F}, D, k)$
 4: $Opt = s.score$, $Score = Opt$
 5: $S \leftarrow \{s\}$
 6: i $\leftarrow$ 2
 7: while $(Score \geq (1\text{-}\theta) \times Opt) and (flag \neq 1)$ do
 8:     $s = \textsc{TopkSets}(i)(\mathcal{F}, D, k)$
 9:     $S \leftarrow S \bigcup s$
10:     $Score = s.score$, $i \leftarrow i + 1$
11: end while
12: $PDF(S) \leftarrow \textsc{MaxMinFair}(S)$
```

until the utility score of a top-k set falls θ% below from the optimum top-k. After that, it calls the **MaxMinFair** S to produce $PDF(S)$.

In Section 4.2, we will show how these two steps could be combined to design a highly scalable solution.

3.1 Algorithm for θ-Equiv-top-k-Sets

Our proposed algorithm **OptTop-k-**θ runs in a loop by performing sorted accesses over the input lists through a cursor movement by calling **getNext**, gradually produces **TopkSets**(i) sets whose scores monotonically decreases, and finally terminates when all θ equivalent top-k sets are found. θ-**Equiv-top-**k**-Sets** requires the scoring functions to be monotonic, we demonstrate **OptTop-k-**θ using one of the representative function WRMSD described in Section 2.2.1.

(1) Generates and maintains a candidate set (C, i, j) of top-k sets as it reads j-th records from the cursors. (C, i, j) is needed for deciding **TopkSets**(i).

(2) Local stopping: if the **TopkSets**(i) is present in (C, i, j).
(3) Global stopping: if all θ Equivalent top-k Sets are found.

OptTop-k-θ borrows inspiration from the celebrated **NRA** (No Random Access) algorithm [19]. However, it is an *not an easy adaptation of **NRA**, because of the exponential nature of θ-**Equiv-top-k-Sets***. The algorithm leverages an efficient data structure based on the concept of item lattice that allows efficient computation of the possible size-k sets and incremental updates of their score bounds by reusing previously calculated scores, as described in Sections 3.1.2 and 3.1.3, respectively.

3.1.1 Generate i-th best top-k set. The first two operations are done inside Algorithm **TopkSets**(i), whose pseudo-code is presented in Algorithm 2. **TopkSets**(i) is responsible for generating the i-th best top-k set. For the ease of exposition, we assume there exists only one unique top-k set in each round, although ties could be handled seamlessly in the framework. Given the set $\mathcal{L}$ of sorted input lists, the algorithm sets a cursor on each list, and fetches the next record from those lists through $\mathcal{L}$ **getNext** calls. As an example, if the input lists consist of both relevance and diversity, then **getNext** fetches the next record from *sortedRelList* list as well as that from the *sortedDivList* list and their corresponding scores. The cursor points to the current position in the lists (let us assume that position to be j). It keeps track of the all seen records upto j-th position. Then **createNewSets** creates all possible size-k sets (lines 1-4).

In order to accomplish (2), the other challenge involves score computations of size-k sets that are encountered so far. Since, **OptTop-k-θ** performs only sorted accesses, it may not be able to produce the exact score of a set of k records immediately - rather has to consider upper and lower bounds of score to argue if this set is a possible candidate for **TopkSets**(i). Upper bound score of a set s, $ub(s)$ (similarly lower bound score $lb(s)$) is the maximum possible (similarly the smallest) possible score s can get. Moreover, when more records are being read, these bounds are to be updated as well. Section 3.1.2 describes how that could be done efficiently.

Lower and upper bound score of a set. Clearly, the lower bound (upper bound) score of a set s, $lb(s)$ (similarly $ub(s)$) is the minimum (similarly maximum) possible score of s that **LowerBound** and **UpperBound** calculate. **LowerBound**(s) is calculated based on an objective function $\mathcal{F}$ and using the scores of any unseen component of $\mathcal{F}(s)$ by the smallest possible value. **UpperBound**(s) is done analogously, except the unseen component is replaced by the cursor reading at the j-th position. Lines 5-7 do that task.

Illustration using WRMSD. Imagine $\mathcal{F}$ is (weighted rel, max div). In that case $\mathcal{L}$ consists of two lists - a sorted relevance list *sortedRelList* and a sorted pairwise diversity lists *sortedDivList* in decreasing order of relevance and diversity values, respectively. Imagine the cursor is at the 2nd position of both these lists (i.e., $j = 2$)- therefore, so far it has seen $rel(r_1)$, $rel(r_2)$, $div(r_2, r_3)$, $div(r_3, r_5)$. Clearly, 4 records are seen so far, but all of their scores are not known - 4 different size-k ($k = 3$) sets could be produced. But, because of sorted access, the score of none of these sets could be calculated exactly. As an example, $ub(r_1, r_2, r_3) = 8.6 + 8.5 + 8.5 + 5 + 5 + 5$ if the weight λ is ignored. However, when the cursor reads another record, either from the relevance or from the diversity list, the ub of all sets need to be updated.

Deciding the i-th top-k set. Line 8 of **TopkSets**(i) produces and maintains a *threshold* and lines 9-12 decide if it needs to continue the computation any further or it is safe to terminate.

Definition 3.1. Threshold is the maximum utility score of any unseen top-k set. $threshold[j] = Max[ub(C, i, j)]$

Given the cursor is at the j-th position of the input lists, if $threshold[j]$ falls below $Opt \times$ (1-θ), there is no point of looking any further,**TopkSets**(i) can terminate by returning the best set present in (C, i, j).

Algorithm 2 TopkSets (i)

```
    Inputs: a set $\mathcal{L}$ of input lists, $i$, $\mathcal{F}$, $k$, **TopkSets**$(i-1).score$, $\theta$, $Opt$
    Outputs: $nextBest$: $i$-th best set
 1: $cursor \leftarrow 0$, $seenR \leftarrow \emptyset$
 2: for $j = cursor$ to $Max_{l \in \mathcal{L}} Len(l)$ do
 3:     $seenR = \{seenR \bigcup \textsc{getNext}(l_1(j)), \textsc{getNext}(l_{|\mathcal{L}|}(j))\}$
 4:     $(C, i, j) \leftarrow$ CREATENEWSETS(seenR[j])
 5:     for s in $(C, i, j)$ do
 6:         lb(s), ub(s) $\leftarrow$ LOWERBOUND($s$), UPPERBOUND($s$)
 7:     end for
 8:     threshold[j] $\leftarrow$ MAX(ub)
 9:     if $threshold[j] < Opt \times (1-\theta)$ then
10:         $nextBest = argmax(C, i, j), flag = 1$
11:         return $nextBest$
12:     end if
13:     for s in $(C, i, j)$ do
14:         if lb[s] $\geq$ max(ub($(C, i, j) - s$)) then
15:             $nextBest \leftarrow s$
16:             return $nextBest$
17:         end if
18:         if ub[$(C, i, j)$] < max(lb($(C, i, j) - s$)) then
19:             Prune $\{(C, i, j) - s\}$
20:         end if
21:     end for
22:     if max(lb[$(C, i, j)$ $\geq$ min(threshold[j], TopkSets$(i-1).score$ then
23:         $nextBest \leftarrow$ ARGMAX(lb($C, i, j$))
24:         Break
25:     end if
26:     $cursor \leftarrow j + 1$
27: end for
28: return $nextBest$
```

Lemma 3.2. *$s =$ **TopkSets**(i), if $s = argmax(lb(C, i, j))$ and $lb(s) \geq max(ub(C, i, j) - s))$*

Lines 13-17 make another key calculation based on Lemma 3.2. It checks if there exists a set s in (C, i, j) with the maximum lower bound, such that the $lb(s)$ is not smaller than the upper bound scores of all other remaining sets in (C, i, j). In that case, s is the i-th best set and **TopkSets**(i) terminates upon returning that set and its values. Indeed, when $\mathcal{F}$ is monotonic, no other unseen sets can have higher score than s.

Lemma 3.3. *$s =$ **TopkSets**(i), if $s = argmax(lb(C, i, j))$ and $lb(s) \geq$ **min**$(threshold[j]$, **TopkSets** $(i-1).score)$*

Similarly, based on Lemma 3.3, the algorithm makes another important decision in Lines 22-27. If the maximum $lb(s)$ of s is not smaller than the minimum of $threshold[j]$ and the score of the top-k set seen in the $i-1$-th iteration, then $lb(s)$ is the top-k set in the i-th iteration. This lemma holds good, since the scores of the returned top-k sets decrease monotonically over iterations.

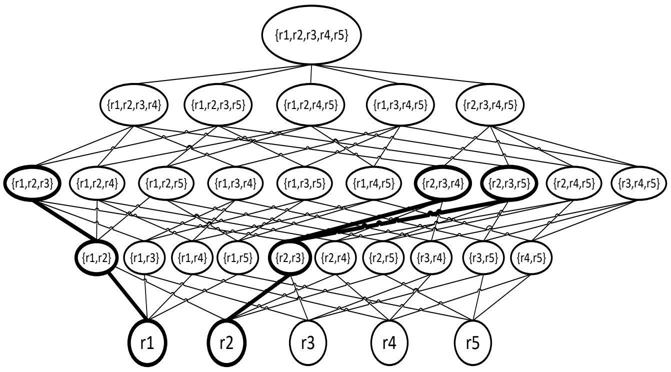

{r1,r2,r3,r4,r5}
{r1,r2,r3,r4} {r1,r2,r3,r5} {r1,r2,r4,r5} {r1,r3,r4,r5} {r2,r3,r4,r5}
{r1,r2,r3} {r1,r2,r4} {r1,r2,r5} {r1,r3,r4} {r1,r3,r5} {r1,r4,r5} {r2,r3,r4} {r2,r3,r5} {r2,r4,r5} {r3,r4,r5}
{r1,r2} {r1,r3} {r1,r4} {r1,r5} {r2,r3} {r2,r4} {r2,r5} {r3,r4} {r3,r5} {r4,r5}
r1 r2 r3 r4 r5

Fig. 2. A complete lattice based on Example 2.1

Pruning sets. Even when **TopkSets**(i) can not terminate, it checks if all sets in (C, i, j) are potential candidates to be the i-th best set - clearly, if the upper bound score of a set s in (C, i, j) is not larger than the lower bound scores of all other sets in C, s could be pruned.

*3.1.2 Subroutine **createNewSets**.* Given $N' < N$ number of items that are encountered by **TopkSets**(i) already, when a new item r is read through a **getNext** call, **OptTop-k-**θ has to perform some hefty tasks.

- It needs to update (C, i, j) by adding additional size k sets that involve r.
- More importantly, it needs to update the lower and upper bound scores of the sets in (C, i, j) - or see if the score could be calculated exactly, if all required scores are read.

A naive idea is to regenerate all size $\binom{(N'+1)}{k}$ sets from scratch, which is computationally wasteful and exponential. To that end, we abstract the representation of the size k sets over a hierarchically ordered space as a lattice, and store ub and lb scores of the record sets there. This data structure offers a great benefit for doing both of these aforementioned tasks efficiently enabling incremental computation.

Data Structure. Given N' seen records, the lattice data structure maintains all $\binom{N'}{1}, \binom{N'}{2}, \ldots \binom{N'}{k}$ sets, as well as their utility score. A node in the lattice represents a possible set, singletons, pairs, triples, ..., size k sets, and so on. An edge represents the membership between two size l and $l+1$ sets. In order to solve θ-**Equiv-top-**k**-Sets** exactly, this space requirement is indeed the lower bound. We also note the lattice structure could be made significantly lightweight (both computationally and storage-wise), if approximate solutions are acceptable, as we discuss in Section 4. A complete lattice for our running example is shown in Figure 2 given $N = 5$, although the data structure only stores information upto size k sets. The set $\{r_1, r_2, r_3\}$ at level three is created by union of three sets in level two, which are $\{r_1, r_2\}$, $\{r_1, r_3\}$, $\{r_2, r_3\}$. Hence the edges represent the connection between these sets in level l and $l + 1$.

Maintaining the structure. This data structure is updated incrementally as new records are read by **OptTop-k-**θ. Take the running example again and imagine $rel(r_1)$, and $div(r_2, r_3)$ is read. So far, the data structure have the following nodes r_1, r_2, r_3, $\{r_1, r_2\}$, $\{r_2, r_3\}$, $\{r_1, r_3\}$, and $\{r_1, r_2, r_3\}$. Next, imagine it reads $div(r_3, r_5)$, thus a new record r_5 is encountered. This creates a singleton, 3 new pairs, and 3 additional size-3 sets. Clearly, r_5 will include the following three additional size-k sets in (C, i, j), $\{r_1, r_2, r_5\}$, $\{r_2, r_3, r_5\}$, $\{r_1, r_3, r_5\}$.

3.1.3 Efficient bound computation and maintenance. Imagine the cursor on the diversity list now moves to the third position and reads $div(r_1, r_3) = 4$. The upper bound scores of all of these following sets $\{r_1, r_2, r_3\}$, $\{r_1, r_2, r_5\}$, $\{r_2, r_3, r_5\}$, $\{r_1, r_3, r_5\}$ are to be updated now. One can naively calculate these bounds from the scratch - but there exists an opportunity of reusing previously done computation that is clearly more efficient.

After reading $div(r_1, r_3) = 4$, our representation updates the score of the node $\{r_1, r_3\}$ in the lattice. All nodes that have a direct or indirect edge to $\{r_1, r_3\}$, their scores are also updated.

Similar situation occurs, when a new record r is encountered - the lattice representation allows us to quickly identify the new nodes that now contains r, as well as how to efficiently reuse the previously computed score of a set s' of size smaller than k to compute score of set $\{s' \bigcup r\}$.

$$\mathcal{F}(s' \bigcup r, q) = \mathcal{F}(s', q) + \mathcal{F}(r, q) \tag{3}$$

Formally, our effort is to study score update as an incremental process and reuse sub-computations that are done before. We express the score (lb, ub, or exact) of a set as a summation of scores over the subsets and retrieve the previously computed scores and reuse it, as opposed to calculating the scores from scratch every time. Indeed, the lattice representation over the seen records allows us to decompose the score of a set as an aggregation over the sub-sets and reuse what has been done before.

Score reuse for WRMSD. Imagine an instance of **OptTop-k-θ** and the **getNext** call has just returned the second row in the diversity list, namely $div(r_3, r_5) = 4$ and the goal is to produce top-k sets, where $k = 4$. A brand new record r_5 is just seen and this will add three additional size-3 sets $\{r_1, r_2, r_5\}$, $\{r_2, r_3, r_5\}$, $\{r_1, r_3, r_5\}$, three size-2 sets $\{r_1, r_5\}$, $\{r_2, r_5\}$, $\{r_3, r_5\}$, and one singleton r_5 on the lattice. The lattice structure facilitates score calculation of $WRMSD(\{r_1, r_2, r_3, r_5\})$ by reusing the scores that are calculated before. For the purpose of illustration, lets just consider the diversity component of the WRMSD calculation $WRMSD - Div(\{r_1, r_2, r_3, r_5\})$ and see how upper bound of scores could be calculated incrementally.

$$\begin{aligned} ub - div(\{r_1, r_2, r_3, r_5\}) = & Maxdiv[(r_1, \{r_2, r_3, r_5\})] \\ & + Maxdiv[(r_2, \{r_1, r_3, r_5\})] \\ & + Maxdiv[(r_3, \{r_1, r_2, r_5\})] \\ & + Maxdiv[(r_5, \{r_1, r_2, r_3\})]. \end{aligned}$$

Now consider $Maxdiv[(r_3, \{r_1, r_2, r_5\})]$ and note that this could simply be expressed as follows:

$$Maxdiv[(r_3, \{r_1, r_2, r_5\})] = Max(div(r_3, r_5), Maxdiv[(r_3, \{r_1, r_2\})] \tag{4}$$

$Maxdiv[(r_3, \{r_1, r_2\})]$ is pre-calculated, hence Equation 4 could be efficiently computed by taking a maximum over $Maxdiv[(r_3, \{r_1, r_2\})]$ score and $div(r_3, r_5)$. This allows sharing computation across sets.

3.1.4 *Global stopping.*

OptTop-k-θ halts when all θ-equivalent top-k sets are produced. This is checked by when one of the following two conditions is satisfied; (i). the last score received from **TopkSets**(i) is smaller than $(1 - \theta) \times Opt$, or (ii). the latest threshold fell below $(1 - \theta) \times Opt$ (which sets a flag to 1). It is guaranteed that there is no future unseen sets with score at most θ% smaller than the best top-k sets. At that point, **OptTop-k-θ** safely terminates and produces the exact solution.

THEOREM 3.4. ***OptTop-k-θ** is an exact solution for θ-**Equiv-top-k-Sets** .*

PROOF. (sketch). Given a monotonic scoring function, it is easy to see that **TopkSets**(i) produces the i-th best top-k set in the i-th iteration. **OptTop-k-θ** maintains all records across iteration, forms all potential top-k sets. Finally, when **OptTop-k-θ** terminates, the global stopping condition guarantees that no unseen set of k records will be θ-equivalent of the top-k set. Hence the proof. □

Running time of OptTop-k-θ. In Section 2, we prove that the counting problem involved in θ-**Equiv-top-k-Sets** is #P-hard. In reality, the running time is dominated by the number of records **OptTop-k-θ** reads before termination and is dominated by the factor $\binom{\#\text{ seen records}}{k}$, which is purely instance dependent. It could be proved that **OptTop-k-θ** is instance optimal.

3.2 Algorithm for MaxMinFair

The last line of Algorithm 1 calls Algorithm **MaxMinFair**, which maximizes the minimum selection probability of the records present in S. We propose a linear programming based optimum solution **Opt-SP** that takes the set of sets S as input, and produces $PDF(S)$, such that **MaxMinFair** optimizes. The problem is formally defined as,

$$\text{Maximize: } x$$

$$\text{subject to:}$$

$$\begin{aligned} \mathcal{P}(r_i) &= \sum_{\forall r_i \in s, s \in S} P(s) \\ \mathcal{P}(r_i) &\geq x, r_i \in s, s \in S \\ \sum_{\forall s \in S} P(s) &= 1 \end{aligned}$$

Given the linear objective function and constraints this could be solved using an off-the-shelf linear programming solver using Simplex or Ellipsoid method.

Running Time. Opt-SP involves solving a linear program using Simplex or Ellipsoid method. Since the feasible region of the objective function is a polytope, these algorithms take polynomial time to the input size N and $|S|$.

Running Example. Using Example 2.1, $PDF(S)$ is produced as follows: $P(s1) = 0, P(s2) = 0, P(s3) = 0.5, P(s4) = 0.5$, leading to selection probability of $r_3 = 1$, and the remaining all 4 records each will have 0.5 selection probability.

4 APPROXIMATION ALGORITHMS

We present two approximate solutions in this section. The first one is **RWalkTop-k-θ**. To solve θ-**Equiv-top-k-Sets**, instead of designing a deterministic exact solution that could be exponential, it leverages a random walk based approach on the item lattice that is highly efficient and is backed by probability theory. To solve **MaxMinFair**, it presents a highly efficient greedy solution **Gr-SP**. **ARWalkTop-k-θ** solves both θ-**Equiv-top-k-Sets** and **MaxMinFair** at the same time through an adaptive random walk. Both **RWalkTop-k-θ**and **ARWalkTop-k-θ** make use of the lattice structure described in Section 3, but it is computed only partially on the fly, making it significantly lightweight computationally and storage-wise.

4.1 Algorithm RWalkTop-k-θ

Algorithm 3 leverages probabilistic computation for producing θ-**Equiv-top-k-Sets** by making random walks on the item lattice. Following that, it solves **MaxMinFair** using a greedy technique.

Inputs to the algorithm are the query, k, objective function $\mathcal{F}$, θ, and the items in D. Additionally, it takes the optimum top-k set and its corresponding score from **TopkSets** 1. It starts by assigning each record a uniform probability of $1/N$. At each step it does uniform random sampling without replacement to select a record and repeats the process until a set has k records. This completes a single random walk on the item lattice, where the walk consists of the edges that are traversed. After it retrieves a size k set s, it computes $\mathcal{F}(s,q)$ and retains s, if $\mathcal{F}(s,q) \geq Opt - \theta$. It keeps repeating the process and stops when each retained s is visited atleast twice in the process.

Algorithm 3 RWalkTop-k-θ

```
    Inputs: query $q$, $D$, $k$, $\mathcal{F}$, $\theta$
    Outputs: $PDF(S)$
 1: while true do
 2:     $s = \{\}, S = \{\}$
 3:     while $|s| \leq k$ do
 4:         pick a uniform random $r \in \{D - s\}$,
 5:         $s \leftarrow \{s \bigcup r\}$
 6:     end while
 7:     if $\mathcal{F}(s, q) \geq (1 - \theta) \times Opt$ then
 8:         $S \leftarrow S \bigcup \{s\}$
 9:     end if
10:     $visit.s \leftarrow visit.s + 1$
11:     if $visit.s \geq 2, \forall s \in S$ then
12:         break
13:     end if
14: end while
15: $PDF(S) \leftarrow$ Gr-SP$(S)$
```

4.1.1 *Termination Condition of the Random Walk.*

The termination condition used for random walk is inspired by the Good Turing Test that is often used in population studies to determine the number of unique species in a large unknown population [24]. Consider a large population of individuals drawn from an unknown number of species with diverse frequencies, including a few common species, some with intermediate frequencies, and many rare species. Let us draw a random sample of N_{samp} individuals from this population, which results in $n1$ individuals that are the lone representatives of their species, and the remaining individuals belong to species that contain multiple representatives in the sample population. Then, $P0$, which represents the frequency of all unseen species in the original population can be estimated as follows:

LEMMA 4.1. *Lemma 1 (Good Turing Test).* $P0 = n1/N_{samp}$.

The assumption here is that the overall probability of hitting one rare species is high while the probability of hitting the same rare species is low. Therefore, the more the sample hits the rare species multiple times, the less likely there are unseen species in the original population. We apply Lemma 4.1 to the θ-equivalent top-k sets construction, where a valid θ-equivalent top-k sets maps to the species and the probabilities of finding each such set in **RWalkTop-k-θ** are the frequencies. The set of θ-equivalent top-k sets discovered during **RWalkTop-k-θ** is the sample population. By ensuring this process visits each constructed set at least twice, we are essentially ensuring that n_1 is 0. Thus, using Lemma 4.1, P_0 can be estimated to be 0, which means it is highly likely that all θ-equivalent top-k sets are discovered.

Illustrative Example. Figure 2 shows the complete lattice involving Example 2.1. To solve **θ-Equiv-top-k-Sets**, the algorithm uniform randomly adds a record and continues the process until a size-3 is obtained. This way the set $s1$:$\{r_1, r_2, r_3\}$ is formed. If $s1$ is a valid answer, it is retained. The process continues until all valid sets are discovered at least twice.

4.1.2 *Subroutine **Gr-SP**.*

Subroutine **Gr-SP** is designed by leveraging the following lemma.

LEMMA 4.2. *If every record r in S appears in only one set $s \in S$, the $PDF(S)$ is a uniform distribution that guarantees equal selection probability of the records.*

Proof. Lemma 4.2 demonstrates an ideal scenario, where a record $r \in s$, $s \in S$ appears in only one s. If the $PDF(S)$ is a uniform distribution, that is, $P(s) = 1/|S|, \forall s \in S$, by leveraging the definition of selection probability of a record (Definition 2.3), then, $\mathcal{P}(r) = 1/|S|$. Clearly, this guarantees that each records r to have the same selection probability. □

Basically, the greedy algorithm is iterative and attempts to select a subset of sets from S that contains different records. Those subset of sets become part of O and gets a non-zero probability value. Specifically, It selects a set s from S in each iteration and adds to O, which includes the highest number of records that are not yet present in O but present in S. The process terminates when O contains all records in S. After that, each set that is present in O gets uniform probability of $\frac{1}{|O|}$. Any set $s \in \{S - O\}$, gets probability 0. We conjecture that this simple yet highly efficient algorithm accepts a 2-approximation factor, the formal proof is left to be explored in the future.
Illustrative Example. Imagine S contains the following 5 sets ($k = 2$), $s1$: $\{r_1, r_2\}$, $s2$: $\{r_3, r_4\}$, $s3$: $\{r_1, r_5\}$, $s4$: $\{r_3, r_5\}$, $s5$: $\{r_1, r_3\}$. If **Gr-SP** first adds $s1$ to O, then, in the next iteration it will add $s2$, and finally $s3/s4$. One possible solution will be $O = \{s1, s2, s3\}$. Each of these sets will get a probability of $1/3$ and the remaining two sets will have probability 0. The minimum selection probability of the records will be $1/3$.
Running time. With an appropriate data structure, such as bucket queue, **Gr-SP** takes $O(N \times |S|)$ to run.

4.2 Algorithm ARWalkTop-k-θ

The last algorithm **ARWalkTop-k-**θ we discuss does not separately compute θ-**Equiv-top-**k**-Sets**, and then, **MaxMinFair** - instead, solves these two problems together. It makes use of Lemma 4.2 to design an adaptive random walk.

The adaptive random walk based algorithm **ARWalkTop-k-**θ is similar to the random walk part of **RWalkTop-k-**θ, except it performs the random walk adaptively, by lowering the probability of the records that are already part of some valid s, and boosting the probability of the remaining records that have not been part of any valid s yet. The goal is to discover θ-equivalent top-k sets where the same record r repeats as few times as possible across the sets - ideally appears in one and only one s. The stopping condition is still guided by the Good Turing Test as described above. Once the process terminates, each set s in S gets uniform probability, and accordingly the selection probability of the records are calculated.

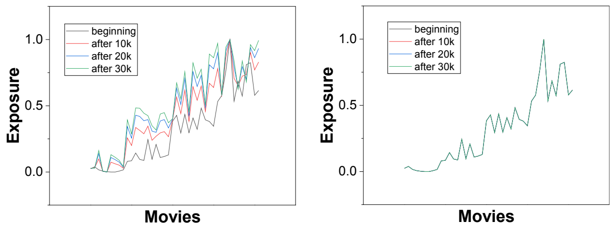
For each record $r \in N$, the algorithm keeps track of the sets in S that contain r ($r.seenCnt$). Instead of picking a record uniformly at random, it then, selects r with a probability that is inversely proportional to $r.seenCnt$. The intuition is that if a record r has already appeared in many $s \in S$, picking it again will hurt the minimum selection probability of other records r' that did not appear as frequently. Therefore, in the i-th iteration of the random walk, it is likely to discover a set of k records that contains new records that are not present in S yet.
Illustrative Example. Imagine Example 2.1 again and assume that $s1$: $\{r_1, r_2, r_3\}$ is discovered. After that, the $r_1.seenCnt$, $r_2.seenCnt$, $r_3.seenCnt$ are increased to 1, and the probabilities of these records are readjusted proportional to their $1/r.seenCnt$. Consequently r_1, r_2, r_3 now have smaller probabilities, whereas, r_4, r_5 have higher probability. Then the random walk is repeated again and the process terminates based on the Good Turing Test. Once S is obtained, each $s \in S$ is assigned uniform probability to produce $PDF(S)$.

5 EXPERIMENTAL EVALUATIONS

Our experimental evaluations have three primary goals.
Goal (1) (Section 5.1). How θ-**Equiv-top-**k**-MMSP** alleviates a limitation that demographic parity based group fairness, such as, top-k parity, or proportional fairness [32, 48] has. We measure how

Dataset	Size	Used Attributes
Yelp	112,686	latitude, longitude, review count
IMDB-top 1000	1,000	numVotes, genre,rating
IMDB	10,000	numVotes
Airbnb	39,882	price
Synthetic	10,000	random samples from uniform distribution
Makeblobs	1,000,000	random samples from guassian distribution

Table 5. Dataset statistics



(a) Only top-k parity **(b)** θ-**Equiv-top-k-MMSP** + top-k parity

Fig. 3. θ-**Equiv-top-k-MMSP** alleviates exposure concerns inherent in demographic parity

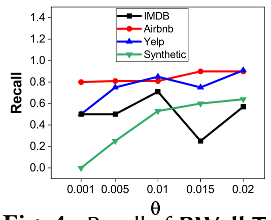


Fig. 4. Recall of **RWalkTop-k-θ** varying θ

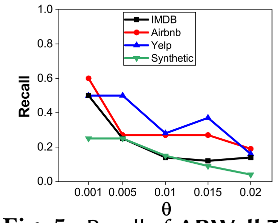


Fig. 5. Recall of **ARWalkTop-k-θ** varying θ

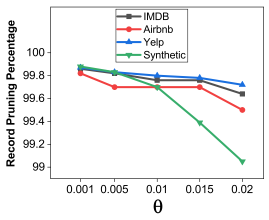


Fig. 6. Record pruning percentage **OptTop-k-θ**

top-k parity alone leads to inequitable exposure, but when θ-**Equiv-top-k-MMSP** is integrated inside top-k parity, they together promote both equal exposure while satisfying group fairness constraints. For that, we present (max-min normalized) exposure, which counts the number of times each record is returned in top-k.

Goal (2) (Section 5.2). Examine quality. For θ-**Equiv-top-k-Sets** , we present recall [28] of the efficient alternatives **RWalkTop-k-θ** and **ARWalkTop-k-θ** compared to **OptTop-k-θ**. For **MaxMinFair**, we present approximation factors (objective function of approximate solution/ objective function of exact solution) of **Gr-SP** and **H-SP** wrt **Opt-SP**.

Goal (3) (Section 5.3). Investigate scalability. For θ-**Equiv-top-k-Sets**, we present pruning capabilities of **OptTop-k-θ** , as well as study the scalability of the different algorithms designed for θ-**Equiv-top-k-Sets** and **MaxMinFair** varying pertinent parameters.

1. Experimental setup. All algorithms are implemented in Python 3.8. All experiments are conducted on a server machine with 128GB RAM memory, OS: windows server 2019 datacenter, version: 1809, CPU: Processor 11th Gen Intel(R) Core(TM) i9-11900K @ 3.50GHz, 3504 Mhz, 8 Core(s), 16 Logical Processor(s). Obtained results are the average of three separate runs. Github has further details [14].

2. Datasets. Experiments are conducted on six datasets, four real and two synthetic data. For real datasets, we use **Yelp** [51], **IMDB-top 1000** [31], **IMDB** [29], and **Airbnb** [4]. For one of the synthetic data, we generate random samples for relevance and diversity scores from uniform distributions. The other synthetic data is **MakeBlobs** [36] from the sklearn package that produces data points from a normal distribution. Table 5 has an overview.

3. Implemented Algorithms.
We note that existing works [7, 21, 25] do not have an easy extension to solve θ-**Equiv-top-**k-**MMSP** because the solution frameworks do not adapt to solve θ-**Equiv-top-**k-**Sets**.

- θ-**Equiv-top-**k-**Sets.** We compare the exact algorithm **OptTop-k-**θ with the two approximate solutions **RWalkTop-k-**θ and **ARWalkTop-k-**θ.
- **MaxMinFair.** We implement a simple baseline **H-SP** first. It goes over the sets in S one by one and checks if all records in a set s are present in other sets in $\{S - s\}$. If yes, s is deleted from S. After that, the remaining sets are returned, each associated with uniform probability. We compare the LP-based exact solutions **Opt-SP**, with approximate solutions **Gr-SP** and **H-SP**.

4. Representative utility functions.

(1) Maximize relevance. $\Sigma_{\forall r \in s} Rel(r, q)$
(2) Weighted relevance and max sum diversity (WRMSD)
Maximize $\lambda \times \Sigma_{r \in s} Rel(r, q) + (1 - \lambda) \times \Sigma_{r \in s} Max_{r, r_j \in \{s-r\}} Div(r, r_j)$, where λ is a weight between $[0, 1]$.
(3) Maximize diversity. Maximize $\Sigma_{r \in s} Max_{r, r_j \in \{s-r\}} Div(r, r_j)$

5. Query & Parameters. Queries are selected randomly. Unless specified, the default parameters are $N = 10k$, $k = 5$, $\mathcal{F}$ = WRMSD with $\lambda = 0.99$, $\theta = 0.01$.

5.1 Goal 1: θ-Equiv-top-k-MMSP and Top-k Parity

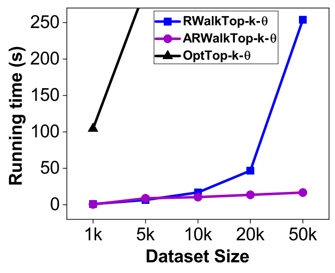
Figures 3 show (max-min normalized) exposure of IMDB-1000 movies considering group-fairness [43] top-k parity [32] alone and that of when θ-**Equiv-top-**k-**MMSP** is implemented along with top-k parity [32]. Group fairness considering top-k parity [32] is imposed using the genre attribute. It is clear when the top-k records are returned based on θ-**Equiv-top-**k-**MMSP**, while satisfying group-fairness [43], the exposure of the records remain unchanged (Figure 3(b)), whereas, records get inequitable exposure when only group-fairness is ensured. Thus, θ-**Equiv-top-**k-**MMSP** alleviates a shortcoming that group fairness suffers from.

5.2 Goal 2: Quality analysis

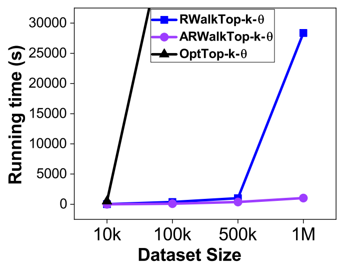
We first present the quality study related to the algorithms designed for θ-**Equiv-top-**k-**Sets** , following which, we present those results for the algorithms designed for **MaxMinFair** .

*5.2.1 Quality Analysis of θ-**Equiv-top-**k-**Sets** .* We study the algorithms designed for θ-**Equiv-top-**k-**Sets** from the quality standpoint. We present the *Recall* percentage [28], which is the percentage of equivalent top-k sets returned by the underlying algorithm w.r.t. the exact solution **OptTop-k-**θ(ground truth).

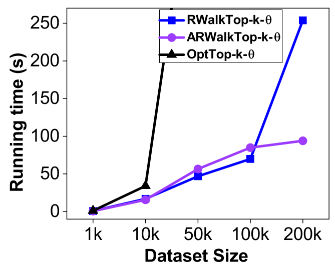
A. Recall of RWalkTop-k-θ . We measure the quality of **RWalkTop-k-**θ by presenting the *Recall* value as described above, which produces the ratio of the top-k sets returned by **RWalkTop-k-**θ compared to that of the exact solution **OptTop-k-**θ by varying θ. Figure 4 shows that the recall of **RWalkTop-k-**θ stays steady mostly (close to 80% for almost all real datasets) or increases with increasing θ. At some point it becomes as high as 91%. When data distribution is uniform (synthetic data), clearly **RWalkTop-k-**θ becomes more effective with increasing θ, which is unsurprising. These results also validate the applicability of the Good Turing Test for our studied problem, informing that for almost all of the datasets, the random walk is returning around 80% of the θ equivalent top-k sets, while being significantly computationally efficient.



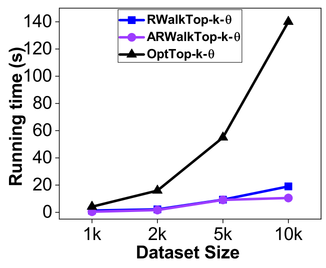
(a) Airbnb



(b) MakeBlobs

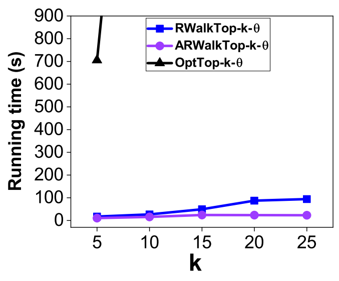


(c) Yelp

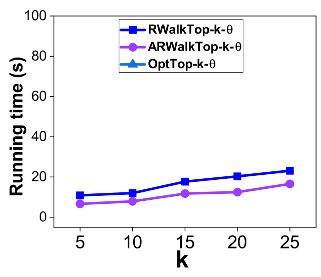


(d) IMDB

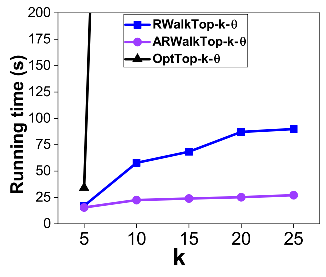
Fig. 7. RWalkTop-k-θ vs **ARWalkTop-k-**θ vs **OptTop-k-**θ scalability by varying dataset size N



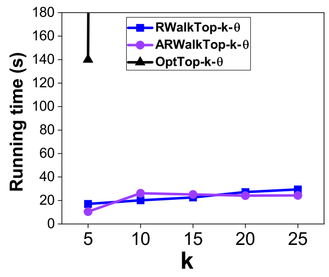
(a) Airbnb



(b) MakeBlobs



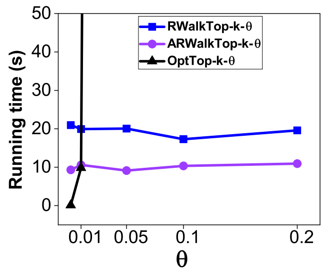
(c) Yelp



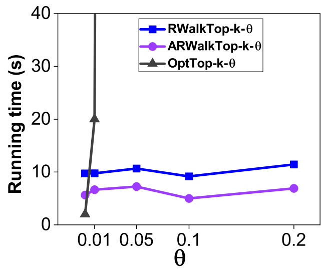
(d) IMDB

Fig. 8. RWalkTop-k-θ vs **ARWalkTop-k-**θ vs **OptTop-k-**θ scalability by varying k

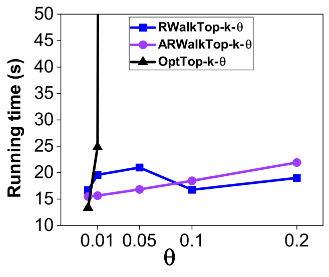
B. Recall of ARWalkTop-k-θ. Figure 5 shows the *Recall* value for the **ARWalkTop-k-**θ algorithm.



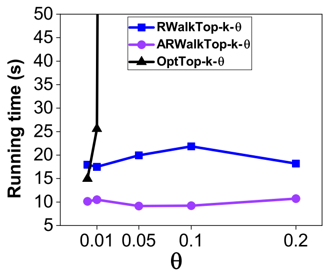
(a) Airbnb



(b) MakeBlobs

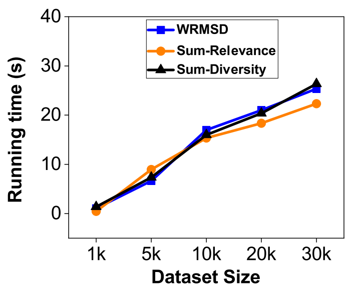


(c) Yelp

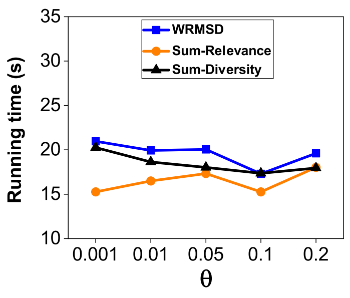


(d) IMDB

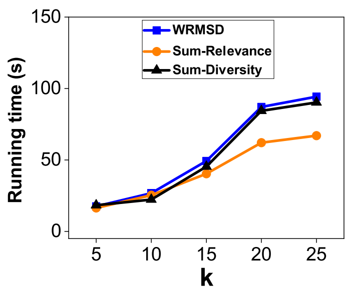
Fig. 9. RWalkTop-k-θ vs **ARWalkTop-k-θ** vs **OptTop-k-θ** scalability by varying θ



(a) Varying dataset size N



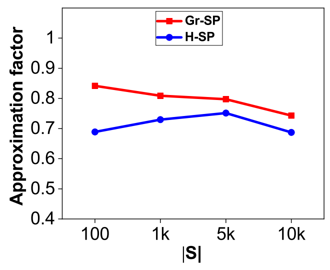
(b) Varying θ



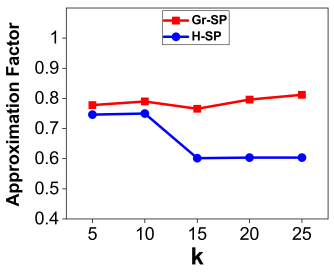
(c) Varying k

Fig. 10. RWalkTop-k-θ scalability for different utility functions

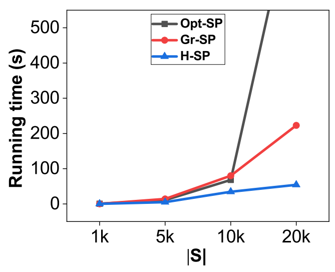
As expected, **ARWalkTop-k-θ** is inferior to solve θ-**Equiv-top-**k**-Sets** compared to **RWalkTop-k-θ** , as it only produces sets that are highly different from each other, giving rise to fewer number



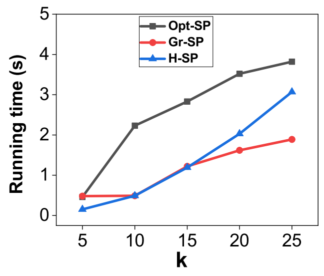
(a) Approx factor varying $|S|$



(b) Approx factor varying k



(c) Scalability by varying $|S|$



(d) Scalability by varying k

Fig. 11. **MaxMinFair** approx factor and scalability

of sets. **ARWalkTop-k-θ** reaches up to 60% recall for Airbnb dataset. *Recall* decreases with increasing θ here, since more top-k sets with common items become eligible with increasing θ, which **ARWalkTop-k-θ** does not return.

5.2.2 *Quality analysis of **MaxMinFair** .* **A. Approximation Factor.** We calculate the approximation factor by dividing the minimum selection probability of the records returned by **Gr-SP** with that of **Opt-SP**. Since **MaxMinFair** is a maximization problem, hence the approximation factor is always ≤ 1. Similarly, the approximation factor of **H-SP** is also computed. As we shall demonstrate in Section 5.3.2, despite being an exact solution, **Opt-SP** is not highly scalable, since it involves a linear program. Figure 11 (a) shows the approximation factor using the sets returned by **RWalkTop-k-θ** algorithm for **Gr-SP** and **H-SP**. Since minimum selection probability for **Gr-SP** is higher than **H-SP**, its approximation factor is larger. The approximation factors demonstrate an encouraging facts. the minimum approximation factor value for **Gr-SP** is 0.74 and that of **H-SP** is 0.68, where as the maximum is 0.84 and 0.75, respectively. Figure 11 (b) present the approximation factor by varying k on 1000 sets returned by **RWalkTop-k-θ** algorithm for **Gr-SP** and **H-SP**. The minimum value of approximation factor of **Gr-SP** is 0.77, and for **H-SP** is 0.60, and the maximum values are 0.81 and 0.74, respectively.

5.3 Goal 3: Scalability Analysis

We first present the scalability study related to the algorithms designed for **θ-Equiv-top-k-Sets** , following which, we present those results for the algorithms designed for **MaxMinFair** .

5.3.1 *Scalability Analysis for **θ-Equiv-top-k-Sets** .* We compare the scalability aspects of the three designed algorithms by varying pertinent parameters.

A. Pruning Effectiveness. We show that **OptTop-k-θ** solves **θ-Equiv-top-k-Sets** by accessing a very few records in the sorted lists. Figure 6 shows effective record pruning of **OptTop-k-θ** varying θ. Record pruning percentage is = $\dfrac{(N - \text{number of seen records})}{N}$. **OptTop-k-$\theta$** is able to prune

99% of the dataset to exactly solve θ-**Equiv-top-k-Sets**. Also with θ, more equivalent sets are to be found, **OptTop-k-θ** needs to read more records, thereby pruning percentage slightly decreased by increasing θ.

B. Running time varying N. Figure 7 shows the scalability of the three proposed algorithms for θ-**Equiv-top-k-Sets** by increasing N. As expected, due to the exponential nature of θ-**Equiv-top-k-Sets** , **OptTop-k-θ** is not scalable over large value of N. In contrast, the other two proposed algorithms are scalable. **ARWalkTop-k-θ** is more scalable than **RWalkTop-k-θ** since it finds less number of sets because of its adaptiveness, it stops earlier. With 1M records in MakeBlobs, **ARWalkTop-k-θ** takes only a few minutes to finish.

C. Running time varying k. Figure 8 demonstrates the scalability of the three proposed algorithms by varying k. As expected, **OptTop-k-θ** does not scale well. Consider Figure 8(c) using Yelp dataset. When $k = 5$, **OptTop-k-θ** takes 34.02 seconds to run, and the number of seen records is 28. $\binom{28}{5} = 98280$ sets are generated and examined only to produce 12 final top-k sets. Now consider that it is increased to $k = 10$. This may end up producing $\binom{28}{10} = 13123110$ sets even with only 28 seen records, which is 133× larger than before. This exponential increase is expected due to the computational nature of θ-**Equiv-top-k-Sets** . On the other hand, **RWalkTop-k-θ** and **ARWalkTop-k-θ** are highly scalable, and not very sensitive to increasing k.

D. Running time varying θ. Figure 9 demonstrates the scalability of the three proposed algorithms by varying θ. Increasing θ increases the size of $|S|$. As expected, **OptTop-k-θ** is highly sensitive to this parameter and does not scale well. In comparison, the random walk based algorithms **RWalkTop-k-θ** and **ARWalkTop-k-θ** are less sensitive and scale reasonably well with increasing θ.

E. Running time varying $\mathcal{F}$. In Figure 10 we present the running time of **RWalkTop-k-θ** using the three representative utility functions, described at the beginning of the Section 5: Figure 10(a), Figure 10(b), Figure 10(c) demonstrate the scalability by varying parameters N, θ and k. As we can see, the running times of all three objective functions increase by increasing N, k, θ. However, the nature of the underlying objective function does not as such impact the running time. Similar observation holds for **ARWalkTop-k-θ** (the graphs are not presented for brevity). This is highly encouraging, as it demonstrates the effectiveness of our designed solutions across different objective functions.

*5.3.2 Scalability Analysis of **MaxMinFair** .* In this section, we present the scalability analysis of the three algorithms designed for **MaxMinFair**. We evaluate the scalability varying $|S|$, N, k.

A. Running time varying $|S|$. Figure 11 (c) shows running time of the **Opt-SP**, **Gr-SP**, **H-SP** with $k = 5$. The heuristic **H-SP** exhibits the highest scalability among all and the linear programming based exact algorithm **Opt-SP** has the least scalability, as expected. Similar observation holds when N is varied. Nevertheless, both **Gr-SP** and **H-SP** are highly scalable and the results corroborate their theoretical running time.

B. Running time varying k. Figure 11 (d) shows the scalability with varying k and $|S| = 1000$. Similar observation holds as before that agorithms **Gr-SP** and **H-SP** are highly scalable to increasing k. This observation is also consistent to their theoretical analysis.

5.4 Summary of Results

(a) Our first observation is θ-**Equiv-top-k-MMSP** alleviates a fairness limitation inherent to demographic parity based group-fairness [43]. (b) Our second observation demonstrates the computational effectiveness of **OptTop-k-θ** - despite the fact θ-**Equiv-top-k-MMSP** is computationally intractable, our designed solution **OptTop-k-θ** is highly effective in pruning the vast majority of the records from the input database to produce the exact solution for θ-**Equiv-top-k-Sets**. The pruning effectiveness is at times as high as 99%. (c) We experimentally observe that **RWalkTop-k-θ** is a highly scalable algorithm that is several order of magnitude faster than the exact solutions **OptTop-k-θ** and **Opt-SP**, yet the produced results are highly comparable qualitatively. This solution achieves high recall, sometime, as high as 91% recall value, while taking a few seconds to run. These results demonstrate the efficiency as well as effectiveness of **RWalkTop-k-θ** to be used and deployed inside real world applications. (d) Our final observation is that **ARWalkTop-k-θ** is a highly efficient solution that can easily scale to a very large N with millions of records, and is suitable for applications that can accommodate modest inaccuracy.

6 RELATED WORK

Group Fairness. Most approaches to algorithmic fairness interpret fairness as lack of discrimination [23] seeking *that an algorithm should not discriminate against its input entities based on attributes that are not relevant to the task at hand.* Such attributes are called protected, or sensitive, and often include among others gender, religion, age, sexual orientation and race. So far, most work on defining, detecting and removing unfairness has focused on classification algorithms [52, 54] used in decision making. W.r.t ranking and top-k results, the algorithmic fairness literature deals with group fairness along the lines of demographic parity this is typically expressed by means of some fairness constraint requiring that the $top - k$ results (for any k) to contain enough records from some groups that are protected [6, 21, 26, 30, 34, 40, 44, 48, 50, 53, 55]. A good survey on this could be found in [39].
Individual Fairness. Individual fairness, on the other hand, as proposed by Dwork et al [16], intends to ensure "similar individuals are treated similarly". Dwork et al. explain that a classifier is individually fair if the distance between probability distributions mapped by the classifier is not greater than the actual distance between the records [16]. Biega et al. propose measures that identify unfairness at the level of individual subjects considering position bias in ranking [8]. Mahabadi et al. study the individual fairness in k-clustering. Their goal is to develop a clustering algorithm of the records so that all records are treated (approximately) equally[35]. Patro et al. [38] investigate the fair allocation problem and study individual fairness in two-sided platforms consisting of producers and customers on opposite sides. Fish et al. study individual fairness in social network [20] to maximize the minimum probability of receiving the information for poorly connected users.
It has been recognized that group fairness alone has its deficiencies [22]. In two independent efforts, Flanigan et. al. [21] and Garcia-Soriano et. al. [25] study how to enable equitable selection probability of the records under group fairness constraints and propose maxmin-fair distributions of ranking. Zemel et al. develop a learning algorithm for fair classification that ensures both group fairness and individual fairness [54]. [7] studies individual fairness in similarity search to ensure points within distance r from the given query have the same probability to be returned.
In [13], the authors propose a new ranking function for search engines called "page quality" by estimating the intrinsic quality of a page. This solution deals with web pages with hyperlinks and alleviates the unequal exposure problem. Due to this specific nature, the solution does not extend to θ-**Equiv-top-k-MMSP** .
Top-k Algorithms. Given a user query, a top-k result contains k records that have the highest scores [41]. Scores are computed based on relevance, diversity, newness, serendipity, etc. Designing effective scoring functions as well as efficient algorithms [1, 2] lend to numerous applications in recommendation and search [3, 10, 11, 18, 33, 46, 47] and is an active area of research. The authors in [42] present a model-based approach to studying result diversity in search engines, exploring the interplay between diversity and quality and improving upon Google's performance on both

random and selective queries. We, on the other hand, do not propose any new scoring function or a new top-k algorithm, but study how to alleviate the unequal exposure problem in the existing algorithms.
*θ-**Equiv-top-k-MMSP** borrows inspiration from existing works, yet it is unique - we study existing top-k algorithms and redesign them to address a fairness concern that is prevalent in long tail data.*

7 CONCLUSION

We formalize θ-**Equiv-top-k-MMSP** to redesign existing top-k algorithms for long tail data to ensure fairness. Given a query, θ-**Equiv-top-k-MMSP** computes a set of top-k sets that are *equivalent* and assigns a probability distribution over these sets, such that, after many users draw a set from these sets according to its assigned probability, the selection probabilities of the records present in these sets are as uniform as possible. We present multiple algorithmic results with theoretical guarantees as well as present extensive experimental evaluation. We demonstrate how our proposed notion of fairness positively impacts compelling downstream applications, and complements group fairness.

One of the directions that we are currently exploring lies in understanding pre-processing techniques that can speed up the computation of θ-**Equiv-top-k-Sets**.

8 ACKNOWLEDGMENTS

This work is supported by the NSF CAREER Award #1942913, IIS #2007935, IIS #1814595, PPoSS:Planning #2118458, and by ONR Grants: #N000141812838, #N000142112966.

REFERENCES

[1] Sofiane Abbar, Sihem Amer-Yahia, Piotr Indyk, and Sepideh Mahabadi. 2013. Real-time recommendation of diverse related articles. In *Proceedings of the 22nd international conference on World Wide Web*. 1–12.
[2] Sofiane Abbar, Sihem Amer-Yahia, Piotr Indyk, Sepideh Mahabadi, and Kasturi R Varadarajan. 2013. Diverse near neighbor problem. In *Proceedings of the twenty-ninth annual symposium on Computational geometry*. 207–214.
[3] Pankaj K Agarwal, Stavros Sintos, and Alex Steiger. 2020. Efficient Indexes for Diverse Top-k Range Queries. In *Proceedings of the 39th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*. 213–227.
[4] Airbnb. 2023. Dataset. http://insideairbnb.com/get-the-data
[5] Robert Armstrong. 2008. The long tail: Why the future of business is selling less of more. *Canadian Journal of Communication* 33, 1 (2008), 127.
[6] Abolfazl Asudeh, HV Jagadish, Julia Stoyanovich, and Gautam Das. 2019. Designing fair ranking schemes. In *Proceedings of the 2019 international conference on management of data*. 1259–1276.
[7] Martin Aumuller, Sariel Har-Peled, Sepideh Mahabadi, Rasmus Pagh, and Francesco Silvestri. 2021. Fair near neighbor search via sampling. *ACM SIGMOD Record* 50, 1 (2021), 42–49.
[8] Asia J Biega, Krishna P Gummadi, and Gerhard Weikum. 2018. Equity of attention: Amortizing individual fairness in rankings. In *The 41st international acm sigir conference on research & development in information retrieval*. 405–414.
[9] Reuben Binns. 2020. On the apparent conflict between individual and group fairness. In *Proceedings of the 2020 conference on fairness, accountability, and transparency*. 514–524.
[10] Zhi Cai, Georgios Kalamatianos, Georgios J Fakas, Nikos Mamoulis, and Dimitris Papadias. 2020. Diversified spatial keyword search on RDF data. *The VLDB Journal* (2020), 1–19.
[11] David Campos, Tung Kieu, Chenjuan Guo, Feiteng Huang, Kai Zheng, Bin Yang, and Christian S Jensen. 2021. Unsupervised Time Series Outlier Detection with Diversity-Driven Convolutional Ensembles–Extended Version. *arXiv preprint arXiv:2111.11108* (2021).
[12] Jaime Carbonell and Jade Goldstein. 1998. 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*. 335–336.
[13] Junghoo Cho, Sourashis Roy, and Robert E. Adams. 2005. Page Quality: In Search of an Unbiased Web Ranking *(SIGMOD '05)*. Association for Computing Machinery, New York, NY, USA, 551–562. https://doi.org/10.1145/1066157.1066220
[14] Codes and data. 2024. https://anonymous.4open.science/r/FairSelectionInsideTopk-2F4F/README.md.
[15] Gil Delannoi and Oliver Dowlen. 2016. *Sortition: Thoery and Practice*. Vol. 3. Andrews UK Limited.

[16] Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard Zemel. 2012. Fairness through awareness. In *Proceedings of the 3rd innovations in theoretical computer science conference*. 214–226.
[17] Ulle Endriss. 2018. Lecture notes on fair division. *arXiv preprint arXiv:1806.04234* (2018).
[18] Mohammadreza Esfandiari, Ria Mae Borromeo, Sepideh Nikookar, Paras Sakharkar, Sihem Amer-Yahia, and Senjuti Basu Roy. 2021. Multi-Session Diversity to Improve User Satisfaction in Web Applications. In *Proceedings of the Web Conference 2021*. 1928–1936.
[19] Ronald Fagin, Amnon Lotem, and Moni Naor. 2003. Optimal aggregation algorithms for middleware. *Journal of computer and system sciences* 66, 4 (2003), 614–656.
[20] Benjamin Fish, Ashkan Bashardoust, Danah Boyd, Sorelle Friedler, Carlos Scheidegger, and Suresh Venkatasubramanian. 2019. Gaps in Information Access in Social Networks?. In *The World Wide Web Conference*. 480–490.
[21] Bailey Flanigan, Paul Gölz, Anupam Gupta, Brett Hennig, and Ariel D Procaccia. 2021. Fair algorithms for selecting citizens' assemblies. *Nature* 596, 7873 (2021), 548–552.
[22] Will Fleisher. 2021. What's Fair about Individual Fairness?. In *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society*. 480–490.
[23] Sorelle A Friedler, Carlos Scheidegger, and Suresh Venkatasubramanian. 2021. The (im) possibility of fairness: Different value systems require different mechanisms for fair decision making. *Commun. ACM* 64, 4 (2021), 136–143.
[24] William A Gale and Geoffrey Sampson. 1995. Good-turing frequency estimation without tears. *Journal of quantitative linguistics* 2, 3 (1995), 217–237.
[25] David García-Soriano and Francesco Bonchi. 2021. Maxmin-fair ranking: individual fairness under group-fairness constraints. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 436–446.
[26] Sahin Cem Geyik, Stuart Ambler, and Krishnaram Kenthapadi. 2019. Fairness-aware ranking in search & recommendation systems with application to linkedin talent search. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2221–2231.
[27] Dimitrios Gunopulos, Roni Khardon, Heikki Mannila, Sanjeev Saluja, Hannu Toivonen, and Ram Sewak Sharma. 2003. Discovering all most specific sentences. *ACM Transactions on Database Systems (TODS)* 28, 2 (2003), 140–174.
[28] Jiawei Han, Jian Pei, and Hanghang Tong. 2022. *Data mining: concepts and techniques*. Morgan kaufmann.
[29] IMDB. 2023. Dataset. https://www.kaggle.com/datasets/isaactaylorofficial/imdb-10000-most-voted-feature-films-041118
[30] Zhongjun Jin, Mengjing Xu, Chenkai Sun, Abolfazl Asudeh, and HV Jagadish. 2020. Mithracoverage: a system for investigating population bias for intersectional fairness. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. 2721–2724.
[31] Kaggle. [n. d.]. Top-1000 IMDB Movies. https://www.kaggle.com/datasets/harshitshankhdhar/imdb-dataset-of-top-1000-movies-and-tv-shows.
[32] Caitlin Kuhlman and Elke Rundensteiner. 2020. Rank aggregation algorithms for fair consensus. *Proceedings of the VLDB Endowment* 13, 12 (2020).
[33] Chang Li, Haoyun Feng, and Maarten de Rijke. 2020. Cascading Hybrid Bandits: Online Learning to Rank for Relevance and Diversity. In *RecSys 2020: The ACM Conference on Recommender Systems*. ACM, 33–42.
[34] Yunqi Li, Hanxiong Chen, Zuohui Fu, Yingqiang Ge, and Yongfeng Zhang. 2021. User-oriented fairness in recommendation. In *Proceedings of the Web Conference 2021*. 624–632.
[35] Sepideh Mahabadi and Ali Vakilian. 2020. Individual fairness for k-clustering. In *International Conference on Machine Learning*. PMLR, 6586–6596.
[36] Makeblobs. 2023. Dataset. https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_blobs.html
[37] Alberto O Mendelzon and Tova Milo. 1997. Formal models of web queries. In *Proceedings of the sixteenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*. 134–143.
[38] Gourab K Patro, Arpita Biswas, Niloy Ganguly, Krishna P Gummadi, and Abhijnan Chakraborty. 2020. Fairrec: Two-sided fairness for personalized recommendations in two-sided platforms. In *Proceedings of The Web Conference 2020*. 1194–1204.
[39] Evaggelia Pitoura, Kostas Stefanidis, and Georgia Koutrika. 2021. Fairness in rankings and recommendations: an overview. *The VLDB Journal* (2021), 1–28.
[40] Evaggelia Pitoura, Panayiotis Tsaparas, Giorgos Flouris, Irini Fundulaki, Panagiotis Papadakos, Serge Abiteboul, and Gerhard Weikum. 2018. On measuring bias in online information. *ACM SIGMOD Record* 46, 4 (2018), 16–21.
[41] Lu Qin, Jeffrey Xu Yu, and Lijun Chang. 2012. Diversifying top-k results. *arXiv preprint arXiv:1208.0076* (2012).
[42] Davood Rafiei, Krishna Bharat, and Anand Shukla. 2010. Diversifying web search results. In *The Web Conference*.
[43] Babak Salimi, Bill Howe, and Dan Suciu. 2020. Database repair meets algorithmic fairness. *ACM SIGMOD Record* 49, 1 (2020), 34–41.
[44] Ashudeep Singh and Thorsten Joachims. 2018. Fairness of exposure in rankings. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2219–2228.

[45] Peter Stone. 2016. Sortition, voting, and democratic equality. *Critical review of international social and political philosophy* 19, 3 (2016), 339–356.

[46] Sanne Vrijenhoek, Gabriel Bénédict, Mateo Gutierrez Granada, Daan Odijk, and Maarten de Rijke. 2022. RADio – Rank-Aware Divergence Metrics to Measure Normative Diversity in News Recommendation. In *RecSys 2022: The ACM Conference on Recommender Systems*. ACM.

[47] Lina Wang, Xuyun Zhang, Tian Wang, Shaohua Wan, Gautam Srivastava, Shaoning Pang, and Lianyong Qi. 2020. Diversified and scalable service recommendation with accuracy guarantee. *IEEE Transactions on Computational Social Systems* (2020).

[48] Dong Wei, Md Mouinul Islam, Baruch Schieber, and Senjuti Basu Roy. 2022. Rank aggregation with proportionate fairness. In *Proceedings of the 2022 International Conference on Management of Data*. 262–275.

[49] Guizhen Yang. 2004. The complexity of mining maximal frequent itemsets and maximal frequent patterns. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. 344–353.

[50] Ke Yang and Julia Stoyanovich. 2017. Measuring fairness in ranked outputs. In *Proceedings of the 29th international conference on scientific and statistical database management*. 1–6.

[51] Yelp. 2023. Dataset. https://www.yelp.com/dataset/documentation/main

[52] Muhammad Bilal Zafar, Isabel Valera, Manuel Gomez Rodriguez, and Krishna P Gummadi. 2017. Fairness beyond disparate treatment & disparate impact: Learning classification without disparate mistreatment. In *Proceedings of the 26th international conference on world wide web*. 1171–1180.

[53] Meike Zehlike, Francesco Bonchi, Carlos Castillo, Sara Hajian, Mohamed Megahed, and Ricardo Baeza-Yates. 2017. Fa* ir: A fair top-k ranking algorithm. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. 1569–1578.

[54] Rich Zemel, Yu Wu, Kevin Swersky, Toni Pitassi, and Cynthia Dwork. 2013. Learning fair representations. In *International conference on machine learning*. PMLR, 325–333.

[55] Hantian Zhang, Xu Chu, Abolfazl Asudeh, and Shamkant B Navathe. 2021. Omnifair: A declarative system for model-agnostic group fairness in machine learning. In *Proceedings of the 2021 International Conference on Management of Data*. 2076–2088.

Received April 2023; revised July 2023; accepted August 2023