Fair Top-k Query on Alpha-Fairness

Hao Liu[1], Zheng Zhang[2], Raymond Chi-Wing Wong[1], Min Xie[3], Bo Tang[4]
[1]*The Hong Kong University of Science and Technology* [2]*Individual* [3]*Shenzhen Institute of Computing Sciences* [4]*Southern University of Science and Technology*
{hliubs, raywong}@cse.ust.hk, mooooochaa@gmail.com, xiemin@sics.ac.cn, tangb3@sustech.edu.cn

***Abstract*—The traditional top-k query was proposed to obtain a small subset from the database according to the user preference, which is explicitly expressed as a ranking scheme (i.e., utility function). However, a poorly-designed utility function may create discrimination, which in turn may cause harm to minority groups, e.g., women and ethnic minorities, and thus, fairness is becoming increasingly important in many situations, e.g., hiring and admission decisions. Motivated by this, we study fair ranking to alleviate discrimination. We design a fairness model, called α-*fairness*, to quantify the fairness of utility functions. We propose an efficient exact framework with a basic implementation and an improved implementation to find the fairest utility function with the minimum modification penalty. We conducted extensive experiments on both real and synthetic datasets to demonstrate our effectiveness and efficiency compared with the prior studies.**

I. Introduction

It is often hard for users to find desired tuples from a large database for decision-making, e.g., hiring. We can use a traditional top-k query [1], [2], [3], where a user provides a preference function, called the *utility function*, and an output size, k. A larger weight on an attribute in the utility function means that this attribute is more important. Based on the utility function, the *utilities* of tuples can be computed as the weighted sum of attribute values. Then, the tuples in the database are ranked based on the utilities and the *top-k set* (i.e., the set of k tuples with the highest utilities) is returned. However, it is hard for users to specify a good utility function in most cases.

A poor *ranking scheme* may create *discrimination*, i.e., people are treated in a different and especially bad way simply because of their membership in a group (e.g., women and African Americans) that is often discriminated against and denoted as the *protected group* defined by the *protected attribute* (e.g., gender and race). In ranking problems, discrimination means people are ranked lower simply because of their membership in a protected group. For example, women have a lower success rate for loans than men [4], so women are the protected group in this case. Men can also be the protected group in some situations. By DATA USA [5], in 2017, 11.5% of female applicants are accepted to MIT, as opposite to 5.25% for males. Such unfairness is also reflected in other aspects, e.g., race [6]. In this paper, we aim at helping users design a fair ranking scheme. In particular, we focus on the fairness in the top-k sets.

Following a typical example [7], [8], we have 9 applicants for a college in Table I. The admission office needs a ranking scheme to give each applicant a score and then admits the top-7 ones. Assume that the original utility function f_0 does not consider any protected attribute (e.g., gender and race) during its design and it simply ranks applicants by computing the score of each applicant t as $f_0(t) = 0.1 \times \text{TOEFL} + 0.1 \times \text{GRE} + 0.8 \times \text{GPA}$, without involving the protected attributes in the score computation. This scheme may lead to bias, e.g., the top-7 set is marked 1 in column "f_0" of Table I, including 6 males but only 1 female (where the proportion of females is 14.3% in the top-7 set, much lower than 33.3% in the entire set). Similarly, while there are 4 out of 9 (i.e., 44.4%) African-Americans, the number of African-Americans in the top-7 set is 2 out of 7 (i.e., 28.6%). Clearly, this f_0 is unfair to females and African-Americans. In the worst case, this scheme may recruit many students from one gender or one race, violating laws and regulations [9], [10], since it neither considers protected attributes explicitly in the score computation nor considers protected attributes implicitly during the design of the ranking scheme. Consider the top-7 set of another function $f_1(t) = 0.6 \times \text{TOEFL} + 0.1 \times \text{GRE} + 0.3 \times \text{GPA}$ in column "f_1". As will be shown later, f_1 is designed by considering proportions of females and African-Americans in the resulting top-7 set, which now become 42.9% and 57.1%, respectively, closer to their proportions in the entire dataset. Thus, f_1 is considered fairer than f_0 by involving protected attributes in its formulation.

TABLE I: Information of Applicants

ID	Gender	Race	TOEFL	GRE	GPA	f_0	f_1
1	Male	Others	110	335	3.5	1	1
2	Male	African-American	108	330	4.0	1	1
3	Male	Others	115	330	3.2	1	1
4	Female	African-American	105	320	3.7	1	1
5	Male	Others	87	310	3.9	1	0
6	Female	African-American	88	315	2.8	0	1
7	Male	Others	87	310	4.0	1	1
8	Female	African-American	88	310	2.6	0	1
9	Male	Others	87	310	3.8	1	0

Top-k queries can also be used in other data analytics tasks in information retrieval [11] and machine learning [12]. For instance, the admission office might train a model, using the top-k students in previous years as positive examples, to predict the academic performance of the current students. If the top-k sets used for training are biased, the model can easily create discrimination [13], e.g., if the top-7 set w.r.t. function f_0 is labeled with "good performance", the model trained will tend to predict females or African-Americans as "bad performance", which further worsens the fairness.

This illustrates our problem which helps users design new utility functions that give fair top-k sets for its increasing importance [14]. Moreover, we do not want the new utility function to deviate much from the original (user-specified) one.

Related studies on fair top-k queries (or ranking) can be classified into two branches, i.e., *intervention with the ranking result* and *intervention with the ranking scheme*. Studies on the former [4], [15] assume that the utilities of all tuples have been computed by a pre-existing ranking scheme. Then, they focus on designing a procedure to select top-k tuples that are fair and have as high utilities as possible. However, these methods may select some tuples with lower utilities to ensure fairness, which causes individual discrimination to higher-utility tuples. In our problem, we always return the top-k tuples under a fair utility function, which could alleviate such individual discrimination.

Studies on the latter branch [7], followed by this paper, focus on finding a ranking scheme (i.e., a utility function) that is both fair and close to the user input utility function f_0 (so that it is easily accepted by the user). The only prior work [7] in this branch returns a *feasible* utility function that is the closest to f_0, where a utility function is feasible if its top-k set satisfies a fairness constraint, specified by the user under an arbitrary (group) fairness model, e.g., the proportion of the protected group in the returned top-k set must be at least a given threshold. However, returning only a feasible utility function may not suffice, and the *fairest* top-k set could be more interesting to the user. Worse still, if the constraint is set too hard, leading to no feasible utility function, the user needs to repeatedly try some looser constraints to obtain the result. Another issue of [7] is its inefficiency when handling a large number of hyperplanes, for testing all feasible utility functions, since for each pair of tuples, a hyperplane has to be inserted and processed with time-consuming geometric operations.

Motivated by the deficiency of fairness constraints like [7], we focus on the (group) fairness measurement in this paper so that if a top-k set is fair, its fairness value is large by the measurement. In particular, we aim to find a utility function, whose top-k set *maximizes* the fairness measurement (i.e., it gives the *fairest* top-k set). Along this line, there are also some existing studies in fair ranking that explore how to measure the group fairness of the ranking result (i.e., a top-k set). [16] proposes the *proportional* model which considers one "binary" protected attribute (e.g., gender defining two groups, females and males). This model yields better fairness if one group (e.g., females) has closer proportions between the top-k set and the entire dataset. [17] extends the proportional model when the protected attribute defines more than two groups or the protected groups are defined by multiple attributes, by considering the KL-divergence between the proportions of all concerned groups in the top-k set and in the entire dataset. However, the KL-divergence only works well when any two groups are not overlapping (i.e., non-intersectional) and all groups form the entire dataset (i.e., complete), while the recent studies [18] also raise the importance of intersectional and incomplete groups.

In light of this, we propose a group fairness model called α-*fairness* to measure the fairness of a top-k set for arbitrarily defined groups (e.g., intersectional or incomplete groups). As will be formalized in Section III, we extend the proportional model [16], to quantify the fairness of a top-k set via the L_p distance between the proportions of all groups (e.g., males and females) in the top-k set and their counterpart in the entire dataset in a latent space called *proportion space*, where each dimension of this space corresponds to a user-specified group.

Based on the above fairness measurement, we want to return a utility function that not only gives the fairest top-k set, but is also close to the input utility function f_0 (i.e., it has small modification penalty). We achieve a trade-off between the fairness and the penalty via a parameter α that controls the strictness of the fairness measurement. If α is larger, the measurement is less strict and more top-k sets are considered as the fairest, so that the returned utility function can be closer to f_0. Note that our parameter α is different from the threshold of a fairness constraint in [7] (even though our fairness model can also be used as the criterion for setting fairness constraints in [7]). This is because for a strict α and a strict constraint, we always return the fairest utility function, but [7] might return no result satisfying the strict constraint. For a loose α and a loose constraint, we can still give sufficiently fair results since we always return the fairest top-k set, but [7] could return a result of poor fairness due to the loose constraint.

To help users find the *fairest* top-k set under our α-fairness model, we propose an efficient exact framework and its basic implementation called **FairTQ-Exact** (Fair Top-k Query). The main idea is to enumerate all top-k sets, and then check (a) their fairness in terms of our α-fairness model and (b) the modification penalty (computed via quadratic programming).

We also propose a fast algorithm called **FairTQ-Exact-BnB** to reduce the top-k sets to be checked via branch-and-bound. To achieve this, we explore an effective and efficient upper bound of α-fairness for all possible top-k sets in a branch, by leveraging geometric properties of the α-fairness model in the proportion space. Based on the upper bounds, some branches could be pruned during an effective best-first search scheme. In our experiments, **FairTQ-Exact-BnB** only needs to process less than 5% of the total number of hyperplanes handled in [7], which uses an *inefficient* hyperplane-based approach.

The major contributions of this work are listed as follows.

- We design a fairness model called α-fairness to measure the fairness of top-k sets, inspired by the proportional model.
- We use a parameter α to effectively trade-off the fairness with modification penalty for the returned utility function.
- We design an efficient exact framework and its basic implementation called **FairTQ-Exact** to find the fairest utility function with the minimum modification penalty.
- We propose another fast exact algorithm **FairTQ-Exact-BnB** based on branch-and-bound to improve the basic one.
- Extensive experiments on real and synthetic datasets demonstrate our effectiveness and efficiency. Our algorithms beat the baselines by one to two orders of magnitude in response time and return fairer results. Our case study showcases that we provide fairer results compared with baselines.

The remainder of this paper is organized as follows. Section II reviews related work. Section III gives the problem definition. Section IV presents our framework and its basic implementation. Section V describes our fast algorithm. Section VI shows the experiments and Section VII gives our conclusion.

II. Related Work

Fairness Models in Ranking. Recent fairness models are commonly classified into *group fairness* [19], [20] and *individual fairness* [21], where group fairness requires treating different groups *equally* and individual fairness ensures equal treatment for similar individuals. We follow the mainstream studies and design a fair ranking scheme under group fairness. Nonetheless, we return a top-k set w.r.t. a concrete utility function, which can also alleviate individual discrimination compared to other approaches that could favor smaller-utility tuples in one group over larger-utility tuples in another group.

In recent studies of fair ranking, various fairness models have been applied [22], [23], [24], [16], [25], [15]. They include pairwise fairness, maxmin fairness, exposure-based fairness, proportion-based model and diversity constraints.

Pairwise fairness [22] quantifies fairness by counting the tuple-pairs (t_i, t_j) where t_i (scoring higher than t_j) ranks lower than t_j in the result. *Maxmin fairness* [23] prioritizes treatment for the most unfairly treated individual while satisfying some group fairness constraints. Although these models attempt to grant individual-level fairness, they assume the ranking scheme and the tuple scores are known and fixed, while our problem is to find the fairest ranking scheme. *Exposure-based fairness* [24] measures group fairness by comparing the sum of *exposure* of two groups, where the exposure of a tuple in position p represents its "position bias" (i.e., higher-ranked positions have higher exposure, since users favor top positions). Since we focus on the top-k query, we consider the *set-based* fairness which only considers the representation of different groups in the top-k set, without considering their ranks.

To measure the set-based group fairness in a top-k set T, the widely-applied *proportional model* [16] quantifies how close the *representational proportion* of the protected group (i.e., the proportion of the protected tuples among all top-k tuples) and its *populational proportion* (i.e., the proportion of the protected tuples among the entire dataset) are. However, [16] only considers two groups (i.e., a binary protected attribute) while recent studies have raised the importance of multinary protected attribute, multiple protected attributes and even intersectional groups [26], [18]. The KL-divergence-based measurement compares the distributions of all groups (which could be defined by multinary and multiple protected attributes) in the top-k tuples and in the entire dataset. Given m groups, we denote by P_D^i (resp. R_T^i) for $i \in [1, m]$ the populational proportion (resp. representational proportion) of the i-th group. Then, the fairness measurement is the KL-divergence between the two distributions, i.e., $\sum_i P_D^i \log \frac{P_D^i}{R_T^i}$. However, the KL-divergence measurement only works well on non-intersectional and complete groups (i.e., $\sum_i P_D^i = 1$ and $\sum_i R_T^i = 1$ [17]). *Diversity constraints* [25], [15] can be defined on arbitrary groups so that the number of tuples in T from each concerned group satisfy a range constraint. As mentioned in Section I, selecting suitable constraints is tricky. In this work, we aim to design set-based group fairness measurements for arbitrarily defined groups by adapting the proportional model.

Algorithms on Fair Ranking. Major algorithms on fair ranking can be classified into the following two categories.
(1) *Intervention with the ranking result* [4], [15], where the score for each tuple is fixed and the ranking result with these scores lead to systematic bias. *FA*IR* [4] ensures the minimal proportion for a *single* protected group, by iteratively selecting tuples into the top-k set satisfying the fairness constraint and the "in-group" utility monotonicity. However, FA*IR could discriminate unprotected groups and could not guarantee the utility monotonicity across groups. [15] selects the top-k set by maximizing the total "worthiness" (where a value is assigned to each tuple t and each rank r, representing the worthiness of the r-th rank tuple t), subject to a fairness constraint, which sets bounds on the proportion of each group in the top-k set to grant fairness to all groups. However, both [4] and [15] could out-rank lower-utility tuples to satisfy group fairness, leading to *ranking violation*, where a tuple t ranks higher than t' but t is dominated by t' (i.e., t' is better than t on every attribute). In this case, no utility function can give such a top-k set.
(2) *Intervention with the ranking scheme* [7], which is followed in this paper. [7] aims to find the utility function closest to the input utility function f_0 such that some group fairness constraints are satisfied. It proposes an exact algorithm called *SATREGIONS*, which adopts an inefficient hyperplane-based approach. Specifically, their hyperplanes are constructed in an angle coordinate system, which split the space into regions. For each region, they check whether there is a feasible utility function satisfying the fairness constraint, and the feasible utility function closest to f_0 is returned. The geometric relationships between hyperplanes and regions are determined by time-consuming linear programming (LP) in *SATREGIONS*.

Algorithms on Related Problems. Supervised learning approaches are widely applied in fair *learn-to-rank* [27], [24], and in decision-making tasks like fair classification [28], [29], [30], [31], [32], [19]. However, they need well-labeled data for training. Besides, recent studies also explore fairness in other decision-making problems similar to ranking [33], [34], [18].

III. Preliminary and Problem Definition

Preliminary: Top-k Set. Given a set D with n tuples, each tuple $t \in D$ has d scoring attributes, denoted by $t[1], \ldots, t[d]$, each of which is a non-negative real value. For example in Table I, each tuple has 3 scoring attributes (i.e., TOEFL, GRE and GPA). Following typical settings in [35], [3], given a d-dimensional *utility vector* $w = (w[1], \ldots, w[d])$, the user preference on t is captured by a linear *utility function* w.r.t. w, denoted by f_w, where $f_w(t) = \sum_{i=1}^{d} w[i] \cdot t[i]$. For simplicity, we also denote f_w by w. We define the *utility* of tuple t w.r.t. w to be $f_w(t)$. In particular, we can rank all tuples in D in a descending order of their utilities (ties broken arbitrarily). The *top-k set* in D w.r.t. w, denoted by $\mathbf{top}\text{-}k_w(D)$, is defined to be the set of k tuples in D with the highest utilities w.r.t. w. A top-k query w.r.t. w thus returns $\mathbf{top}\text{-}k_w(D)$. When D and w are clear in the context, we also denote $\mathbf{top}\text{-}k_w(D)$ by $\mathbf{top}\text{-}k_w$ or T for simplicity. Consider our running example in Table I. Given $w = (0.1, 0.1, 0.8)$ and $k = 7$, the top-k query

w.r.t. w returns a top-k set containing applicants of ID 1-5, 7 and 9, since they have the highest utilities w.r.t. w.

We assume w.l.o.g. that $\sum_{i=1}^{d} w[i] = 1$ where $w[i] \in [0, 1]$, since the norm of w does not affect the top-k results [36]. Note that $w[d] = 1 - \sum_{i=1}^{d-1} w[i]$. Thus, we could reduce the last dimension from the domain of w (i.e., from d-dimension to $(d-1)$-dimension) for more efficient processing [36]. We define the *utility space*, denoted by Ω, to be the $(d-1)$-dimensional subspace containing all utility functions after the dimension reduction. In our example in Table I, the utility space is a subspace in a 2-dimensional space, as shown in the shaded triangular region of Figure 1(a), bounded by the two axis and the straight line, represented by $w[1] + w[2] = 1$.

We summarize the frequently used notations in [37].

Fairness Model. Following prior studies [7], [4], we focus on *group fairness*, which requires that each group (e.g., the female group) receives equal treatment when its members are considered for the top-k set [35]. Specifically, we adapted the widely-used "proportional" model [16], which requires that the *representational proportion* of each group in the top-k set is close to its *populational proportion* in the dataset D.

We differ from the group fairness models of existing fair ranking studies in the following aspects. (1) We form fairness measurement to grant group fairness (based on the proportional model) instead of taking a constraint which is difficult to select suitably. (2) We consider not only the groups defined by a single protected attribute, but also more generalized groups that could be incomplete or intersectional (including groups defined by multiple protected attributes), while existing group fairness measurements [16] only focus on two disjoint groups defined by one protected attribute. (3) We use a parameter to control the strictness of the fairness measurement so that there is a trading-off between fairness and penalty (i.e., how close the returned utility function is to the input utility function).

Consider our dataset D. Each tuple $t \in D$ is associated with one or multiple protected attributes (e.g., gender or race), each of which uses categorical values to define the group membership of t (e.g., females or African-American). The group membership could also be defined by multiple protected attributes (e.g., African-American females). We are given a list G of m group memberships with which the user is concerned. For instance, when the user only desires equal treatment to different genders, G could be {Male, Female}; when the user wants to consider multiple protected attributes, G might be {Female, African-American}. In some cases, the user might require a set G including all combinations of groups with multiple protected attributes, e.g., {African-American female, African-American male, Other-race female, Other-race male}. In this work, we adopt the group fairness measurement to accommodate all the above cases with the proportional model.

Let G_j denote the j-th group membership in list G, and let D_j (for $j \in [1, m]$) denote the set of all tuples in D with the group membership G_j. We define the *populational proportion* of G_j in D, denoted by $P_D(G_j)$, to be the proportion of D_j among D, i.e., $P_D(G_j) = |D_j|/|D|$. We define the *overall population* of G in D, denoted by $P_D(G)$, to be an m-tuple whose j-th value is defined to be the populational proportion of G_j in D, i.e., $P_D(G) = (P_D(G_1), \ldots, P_D(G_m))$.

Given a top-k set T, we also denote the set of tuples in T with group membership G_j by T_j. Then, the *representational proportion* of G_j in T, denoted by $R_T(G_j)$, is the proportion of T_j among T (i.e., $R_T(G_j) = |T_j|/|T|$). We also define the *overall representation* of G in T, denoted by $R_T(G)$, to be an m-tuple whose j-th value is the representational proportion of G_j in T, i.e., $R_T(G) = (R_T(G_1), \ldots, R_T(G_m))$.

Consider the set D in Table I. The populational proportions of group memberships Female and African-American are 0.333 and 0.444, respectively. For the top-7 set w.r.t. f_0 (i.e., applicants of ID 1-5, 7 and 9) denoting T, the representational proportions of Female and African-American in T are 0.143 and 0.286, respectively. If G is {Female, African-American}, the overall population of G in D is $(0.333, 0.444)$, while the overall representation of G in T is $(0.143, 0.286)$.

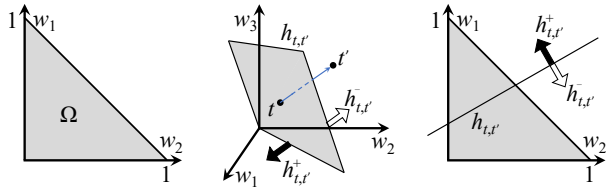
Note that both $P_D(G)$ and $R_T(G)$ can be regarded as points in the m-dimensional space. To better formalize the geometric properties, we introduce a concept called the *proportion space*. Consider an m-dimensional space. We define the *proportion space*, denoted by Θ, to be the region in the m-dimensional space containing all m-dimensional points whose value of each dimension is in $[0, 1]$. In Figure 3(a), we show an example of Θ in the 2-dimensional space (as marked in light gray) when $G = \{\text{Female, African-American}\}$ (following our running example). Note that Θ is different from the utility space Ω, since each dimension of Ω corresponds to a value in a utility function, while here each dimension of Θ corresponds to the proportion of a group membership in G (see G_j in Figure 3(a)). Both $R_T(G) = (0.333, 0.444)$ and $P_D(G) = (0.143, 0.286)$ are the points in Θ (as plotted in Figure 3(a)).

According to the proportional group fairness model which requires that the populational proportion of each group is equal (or close) to its representational proportion, we form our fairness measurement of a top-k set T which returns a small value for good fairness based on the closeness of the overall population of D and the overall representation of T. Specifically, we measure the fairness of T by the L_p *distance* between $R_T(G)$ and $P_D(G)$ in the m-dimensional space where p is a real value at least 1, since it could cover the commonly applied measurements. For instance, when $p = 2$, it is the Euclidean distance which is the most common distance metric; when $p = 1$, it is similar to the absolute difference form of group fairness as defined in [16]. Formally, following the common definition, we define the L_p distance between $R_T(G)$ and $P_D(G)$, denoted by $L_p\langle R_T(G), P_D(G)\rangle$, as follows.

$$L_p\langle R_T(G), P_D(G)\rangle = (\sum_{j=1}^{m} |R_T(G_j) - P_D(G_j)|^p)^{1/p}$$

Continuing our example of $G = \{\text{Female, African-American}\}$. If $p = 2$, the L_p distance between $R_T(G)$ and $P_D(G)$ is 0.247. In Figure 3(a), this distance is illustrated geometrically as the length of the line segment connecting $R_T(G)$ and $P_D(G)$.

Moreover, we control the strictness of the fairness measurement with parameter α, a real number in $[0, 1]$. Intuitively,



(a) Utility Space (b) 3D Hyperplane (c) 2D Hyperplane

Fig. 1: Examples of Utility Space and Hyperplane

when two proportions, says r_1 and r_2, are close (i.e., the difference of r_1 and r_2 is no more than a threshold), r_1 and r_2 could be regarded as being equal. We thus introduce a real number $\alpha \in [0, 1]$ as this threshold, and we define the *diminished difference* of r_1 and r_2 on α, denoted by $dif_\alpha\langle r_1, r_2\rangle$, to be $dif_\alpha\langle r_1, r_2\rangle = \max\{0, |r_1 - r_2| - \alpha\}$. Intuitively, if $|r_1 - r_2| < \alpha$, the diminished difference is 0. Otherwise, it is $|r_1 - r_2| - \alpha$.

Combining the L_p distance with the strictness parameter α, we define the α-L_p distance between $R_T(G)$ and $P_D(G)$, denoted by α-$L_p\langle R_T(G), P_D(G)\rangle$, as follows.

$$\alpha\text{-}L_p\langle R_T(G), P_D(G)\rangle = [\sum_{j=1}^{m}(dif_\alpha\langle R_T(G_j), P_D(G_j)\rangle)^p]^{1/p}$$

For example, if $\alpha = 0.2$, the diminished difference of R_T(Female) and P_D(Female) is 0 ($|R_T(G_j) - P_D(G_j)| < 0.2$). It can be verified that when $G = \{$African-American, Female$\}$, the α-L_p distance between $R_T(G)$ and $P_D(G)$ is 0.

Note that when $\alpha = 0$, $dif_\alpha\langle R_T(G_j), P_D(G_j)\rangle = |R_T(G_j) - P_D(G_j)|$ and thus, α-$L_p\langle R_T(G), P_D(G)\rangle$ is exactly the L_p distance between $R_T(G)$ and $P_D(G)$. In this case, the fairness measurement is the strictest since α-$L_p\langle R_T(G), P_D(G)\rangle = 0$ only if the overall representation in T are exactly the same as the overall population in D. A larger value of α results in a less strict measurement, since it is more likely for T to have representation proportions of some groups "rounded" to 0. Note that setting α too large (e.g., close to 1) could lead to a case where all top-k sets T satisfy α-$L_p\langle R_T(G), P_D(G)\rangle = 0$, which indicates the same fairness for all top-k sets.

Finally, given a top-k set T in dataset D, a list G of m group memberships, a real number α in $[0, 1]$ and a real number p with value at least 1, we define the α-fairness of T in D w.r.t. G and p, denoted by α-$FN_{G,p}(T|D)$, as follows.

$$\alpha\text{-}FN_{G,p}(T|D) = 1 - \frac{1}{m^{1/p}} \cdot \alpha\text{-}L_p\langle R_T(G), P_D(G)\rangle$$

The above formation first normalizes the α-L_p distance into $[0, 1]$ by being divided by a factor of $m^{1/p}$ (the maximum possible α-L_p distance). Then, since a smaller α-L_p distance means T is fairer, we use 1 minus the α-L_p distance such that larger α-fairness (e.g., closer to 1) indicates better fairness.

Fair Ranking Problem. The goal of this work is to design a fair ranking scheme for the top-k query. Recall that the core of a ranking scheme is a utility function in the utility space Ω. Given a user-input utility function $w_0 \in \Omega$, we want to return a new utility function $w \in \Omega$ such that w does not deviate too much from w_0 and the set $\mathbf{top}$-k_w is fair based on our fairness model. To quantify how far w deviates from w_0, we define the *modification penalty* of w from w_0, denoted by $m(w, w_0)$ to be the L_2 distance (a widely applied metric) between w and w_0, i.e., $m(w, w_0) = \sqrt{\sum_{i=1}^{d}(w[i] - w_0[i])^2}$.

Problem 1 (Fair Ranking)**.** *Given a dataset D, a real number α, a positive integer k, a list G of group memberships, a real number p and a utility function w_0, our goal is to find a utility function w^* in the utility space Ω such that the α-fairness of $\mathbf{top}$-k_{w^*} in D w.r.t. G (i.e., α-$FN_{G,p}(\mathbf{top}\text{-}k_{w^*}|D)$) is maximized, while under the maximized α-fairness, the modification penalty of w^* from w_0 is minimized. Mathematically,*

$$w^* = \arg\min_{w' \in \Omega} m(w', w_0)$$
$$s.t. \quad \alpha\text{-}FN_{G,p}(\mathbf{top}\text{-}k_{w^*}|D) = \max_{w \in \Omega} \alpha\text{-}FN_{G,p}(\mathbf{top}\text{-}k_w|D).$$

In our running example in Table I, assume that $\alpha = 0.1$, $k = 7$, $G = \{$Female, African-American$\}$, $p = 2$ and $w_0 = (0.1, 0.1, 0.8)$. We want to find a utility function w^* such that α-$FN_{G,p}(\mathbf{top}\text{-}k_{w^*}|D)$ is maximized, while the modification penalty of w^* from w_0 is minimized. It can be verified that the utility function w^* returned for this problem instance is $(0.33, 0.51, 0.16)$ which gives the fairest top-k set containing applicants ID 1-7 with the maximum α-fairness and the minimum modification penalty equal to 1 and 0.79, respectively.

IV. Exact Algorithm

In this section, we present our framework for our fair ranking problem and a basic implementation called **FairTQ-Exact**, which guarantees to find the fairest set with minimum penalty.

A. Framework

We begin with the framework overview. Note that there are an infinite number of utility functions in Ω. It is infeasible to enumerate all to compute the top-k sets. Alternatively, we adapt the hyperplane-based approach (a standard approach in the literature) to obtain the top-k sets [36], [3], which is more efficient than the naive way of checking all $\binom{|D|}{k}$ subsets of D.

It works in the following three steps.

- **Step 1 (Feasible Top-k Sets Enumeration)**: We enumerate all *feasible* top-k sets in D (which is much smaller than the set of all possible $\binom{|D|}{k}$ subsets of D) using an efficient hyperplane-based approach with effective optimizations.
- **Step 2 (α-fairness Computation)**: For each feasible top-k set T, we compute its α-fairness in D w.r.t. G and p (i.e., α-$FN_{G,p}(T|D)$) according to our definition of α-fairness.
- **Step 3 (Best Utility Function)**: For those top-k sets with the maximized α-fairness (possibly many), we find the *best* utility function w^* with the minimum modification penalty (among all those top-k sets) using *quadratic programming*.

Below we present how these are done in **FairTQ-Exact**.

B. Step 1 (Feasible Top-k Sets Enumeration)

We enumerate all top-k sets in D with a hyperplane-based approach. The intuition is to divide Ω into a number of small regions by the *arrangement* of a set $\mathcal{H}$ of hyperplanes, where each cell of the arrangement is a region bounded by some hyperplanes in $\mathcal{H}$ and it corresponds to a unique top-k set. Thus, to enumerate all top-k sets, we can enumerate all cells in this arrangement and find the top-k set for each cell.

We first formalize some geometric concepts. For each tuple t in D, it can be represented as a point in the d-dimensional space. Given two tuples t and t', we define the hyperplane between t and t', denoted by $h_{t,t'}$, to be the hyperplane in the d-dimensional space passing through the origin with $t'-t$ as its normal vector, i.e., $h_{t,t'} = \{w \in \mathbb{R}^d | (t'-t) \cdot w = 0\}$. In Figure 1(b), we give an example of hyperplane $h_{t,t'}$ (shown as a shaded plane) between tuple t and t' in the 3-dimensional space, where the normal of $h_{t,t'}$ is $t'-t$ (shown as a blue arrow). Recall that we reduced one dimension and consider the $(d-1)$-dimensional space for simplicity. We thus represent the hyperplane $h_{t,t'}$ in the $(d-1)$-dimensional space, by discarding the last dimension, e.g., in Figure 1(c), $h_{t,t'}$ is represented as a straight line in the (reduced) 2-dimensional space.

It can be observed that $h_{t,t'}$ divides the $(d-1)$-dimensional space into two halfspaces [3]. The halfspace containing all utility functions w such that $f_w(t) \geq f_w(t')$ (resp. $f_w(t) < f_w(t')$) is denoted by $h^+_{t,t'}$ (resp. $h^-_{t,t'}$), i.e., $h^+_{t,t'} = \{w \in \mathbb{R}^d | (t'-t) \cdot w \leq 0\}$ and $h^-_{t,t'} = \{w \in \mathbb{R}^d | (t'-t) \cdot w > 0\}$. In Figures 1(b) and (c), we indicate $h^+_{t,t'}$ and $h^-_{t,t'}$ with solid and hollow arrows, respectively, in both the (original) 3-dimensional space and the (reduced) 2-dimensional space.

A simple hyperplane-based approach can be implemented as follows. Firstly, for each pair of tuples t and t' in D, we create a hyperplane $h_{t,t'}$ and insert it to a set $\mathcal{H}$. Secondly, we compute an arrangement in the $(d-1)$-dimensional space of all hyperplanes in $\mathcal{H}$, bounded by the utility space Ω. We denote this arrangement by $\mathcal{A}(\mathcal{H}, \Omega)$. Thirdly, for each cell of $\mathcal{A}(\mathcal{H}, \Omega)$ (i.e., a region in Ω, say ρ), we randomly pick a utility function w in ρ and compute the top-k set w.r.t. w (i.e., $\mathbf{top}\text{-}k_w$) and insert it into the result set. The correctness of this simple implementation is guaranteed by the following lemma, which says that the top-k set w.r.t. any utility function in ρ is the same; in this case (i.e., the top-k set is unique), we simply call it the top-k set w.r.t. ρ, denoted by $\mathbf{top}\text{-}k_\rho$. For the sake of space, the detailed proof can be found in our technical report [37].

Lemma IV.1. *If w and w' are two utility functions in the same cell of arrangement $\mathcal{A}(\mathcal{H}, \Omega)$, then $\mathbf{top}\text{-}k_w = \mathbf{top}\text{-}k_{w'}$.*

Unfortunately, this implementation is inefficient since there are $O(|D|^2)$ hyperplanes in $\mathcal{H}$ and computing the arrangement of $\mathcal{H}$ takes $O(|\mathcal{H}|^{d-1})$ time [36]. Below we present a more efficient approach for computing the top-k sets. It is mainly inspired by [36], but we explore effective optimizations.

Efficient Hyperplane-based Approach. Our approach mainly has two steps. The first step is a *filtering step*, which extracts a subset D' of D, such that $|D'|$ is significantly smaller than $|D|$ and only tuples in D' can appear in a top-k set. The second step is a *refinement step* which enumerates all feasible top-k sets in much smaller D' rather than D, which is very efficient.

The filtering step is handled by the *k-skyband* query [38]. Specifically, given tuples t and t', t is said to *dominate* t' if $t[i] \geq t'[i]$ for each $i \in [1, d]$ and there exists $i \in [1, d]$ such that $t[i] > t'[i]$. Clearly, if t dominates t', t has a higher utility and thus, a higher *rank* than t' w.r.t. any utility function in Ω. A tuple $t \in D$ is called a *k-skyband* [38] of D if t is dominated by fewer than k tuples in D. A k-skyband query returns the set D' of all k-skybands in D. It is easy to verify that if a tuple t is not a k-skyband of D (i.e., t is dominated by at least k tuples), then t cannot appear in any top-k set, since its rank is lower than at least k tuples in D w.r.t. any utility function in Ω. In the remaining steps, we merely focus on D' instead of D.

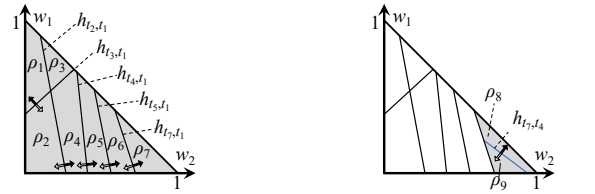
Example IV.1. *Consider D in Table I where $k = 3$. For better illustration, we denote tuple t_i to be the applicant of ID i. Consider t_9, which is dominated by 3 tuples in D (i.e., t_2, t_5 and t_7), and thus, t_9 is not a k-skyband of D. After other tuples are processed, the set D' of all k-skybands of D is $\{t_1, \ldots, t_7\}$.* □

In the refinement step, the core idea is to select an *anchor* from a set of *candidate* tuples (which is initially D'). Then, a region (which is initially set to Ω) is decomposed into sub-regions based on a "local" arrangement (i.e., an arrangement bounded by this region) of a set of hyperplanes each of which is a hyperplane between a *competitor* in the candidate set and the anchor. This above process is recursively executed for all sub-regions until we decompose the whole utility space Ω into desired regions where the top-k set w.r.t. each region is determined. Intuitively, by processing a region with an anchor, we can prune many candidates for sub-regions inside this region and reduce the number of hyperplanes to be processed.

The refinement step is recursively done via *partitioning*. Initially, the region we consider is the whole utility space Ω and the *candidate* set, denoted by $\mathcal{C}_\Omega$, is initialized to be the set D' of all k-skybands of D. To perform partitioning, we choose a tuple in $\mathcal{C}_\Omega$, says t_A, to be the *anchor* (the strategy of selecting an anchors presented later). Our goal is to determine the *rank* of the anchor among all tuples in $\mathcal{C}_\Omega$. We then perform the following steps to accomplish this goal. For the ease of understanding, we will explain each step using concrete examples.

Firstly, we find all *competitors* of anchor t_A from $\mathcal{C}_\Omega$, where a tuple $t_C \in \mathcal{C}_\Omega$ is a competitor of t_A if neither t_A is dominated by t_C nor t_A dominates t_C. Note that for a non-competitor of t_A, we already know whether it ranks higher/lower than t_A based on the dominance relation. Secondly, for each competitor t_C of t_A, we create the hyperplane h_{t_C,t_A} and insert it into a set $\mathcal{H}$ which is initialized to an empty set. We then compute the arrangement of all hyperplanes in $\mathcal{H}$ bounded by Ω, denoted by $\mathcal{A}(\mathcal{H}, \Omega)$. Thirdly, for each cell of $\mathcal{A}(\mathcal{H}, \Omega)$ (which is a region, say ρ), we find all the *dominating competitors* of t_A w.r.t. ρ where a competitor t_C is said to be a dominating competitor of t_A w.r.t. ρ if ρ is completely contained inside $h^+_{t_C,t_A}$. Intuitively, by the definition of $h^+_{t_C,t_A}$, $f_w(t_C) > f_w(t_A)$ for any utility function w in ρ. In other words, the rank of t_C is higher than t_A w.r.t. any utility function w in ρ and thus, t_C is a "dominating" competitor.

Example IV.2. *Assume that we select t_1 in $D' = \{t_1, \ldots, t_7\}$ to be the anchor. The competitors of t_1 are t_2, t_3, t_4, t_5 and t_7 (but not t_6 since t_1 dominates t_6). We insert 5 hyperplanes, namely h_{t_2,t_1}, h_{t_3,t_1}, h_{t_4,t_1}, h_{t_5,t_1} and h_{t_7,t_1}, to $\mathcal{H}$, which are shown as line segments in Figure 2(a). For each hyperplane,*



(a) Initial Partitioning (b) Recursive Partitioning

Fig. 2: Examples of Partition

we also use the solid and hollow arrows to differentiate its two halfspaces (e.g., the solid arrow for h_{t_3,t_1} represents $h^+_{t_3,t_1}$). Clearly, we form an arrangement $\mathcal{A}(\mathcal{H},\Omega)$ with 7 cells, namely regions $\rho_1,\ldots,\rho_7$. Here, t_3 is a dominating competitor of t_1 w.r.t. ρ_1, since ρ_1 lies completely inside $h^+_{t_3,t_1}$. Similarly, we can find the dominating competitors w.r.t. other regions. □

Then, for a given region ρ, we can obtain the rank of t_A w.r.t. *any* w in ρ, denoted by k'. Intuitively, k' is equal to 1 plus the number of tuples that rank higher than t_A w.r.t. w. Let $\overline{\mathcal{C}}$ be the set of tuples in $\mathcal{C}_\Omega$ dominating t_A (and thus, they are *non-competitors*) and $\overline{\mathcal{C}}^{omp}$ be the set of all dominating competitors of t_A w.r.t. ρ. Then, k' is computed to be $1+|\overline{\mathcal{C}}|+|\overline{\mathcal{C}}^{omp}|$, since only tuples in $\overline{\mathcal{C}}\cup\overline{\mathcal{C}}^{omp}$ rank higher than t_A w.r.t. w. As a by-product, this gives the top-k' set w.r.t. ρ in $\mathcal{C}_\Omega$: $\mathbf{top}\text{-}k'_\rho(\mathcal{C}_\Omega) = \{t_A\}\cup\overline{\mathcal{C}}\cup\overline{\mathcal{C}}^{omp}$, where the k'-th rank tuple is exactly t_A.

Based on the relationship between k' and k, we classify the region ρ into 3 types, namely *equal-to* region, *less-than* region and *greater-than* region, which are defined to be a region (resulting from t_A) in which the rank of t_A w.r.t. any w in ρ (i.e., k') is equal to k, less than k and greater than k, respectively.

- *Equal-to region (i.e., $k'=k$):* If ρ is an *equal-to* region, the top-k set w.r.t. ρ (i.e., $\mathbf{top}\text{-}k'_\rho(\mathcal{C}_\Omega)$) is found, which is $\{t_A\}\cup\overline{\mathcal{C}}\cup\overline{\mathcal{C}}^{omp}$. Thus, we do not need further processing for ρ.
- *Less-than region (i.e., $k'<k$):* If ρ is a *less-than* region, the top-k' set w.r.t. ρ is found similarly. Moreover, since $k'<k$, we only need to know the remaining $k-k'$ tuples to return the top-k sets. To achieve this, we recursively partition ρ in a similar way as Ω, with the following modifications: (a) the region now we consider is ρ rather than Ω (i.e., future arrangement is bounded "locally" by ρ); (b) since the top-k' w.r.t. ρ is found, we exclude them and form a new candidate set $\mathcal{C}_\rho$, i.e., $\mathcal{C}_\rho = \mathcal{C}_\Omega\setminus(\{t_A\}\cup\overline{\mathcal{C}}\cup\overline{\mathcal{C}}^{omp})$; and (c) instead of finding the top-k sets, we only need to find the top-$(k-k')$ sets w.r.t. the utility functions in ρ in $\mathcal{C}_\rho$. Here, the number $k-k'$ is called the *rank quota* for this partitioning.
- *Greater-than region (i.e., $k'>k$):* If ρ is a *greater-than* region, we cannot find the top-k sets in ρ. However, we know that t_A and tuples ranked lower than t_A cannot be in any top-k set in ρ. Thus, we recursively partition ρ where the candidate set $\mathcal{C}_\rho$ is set to be $\overline{\mathcal{C}}\cup\overline{\mathcal{C}}^{omp}$, with the same rank quota k.

Example IV.3. *In Figure 2(a), ρ_3 and ρ_5 are two equal-to regions, and their corresponding top-k sets are $\{t_1,t_2,t_3\}$ and $\{t_1,t_2,t_4\}$, respectively, which can be directly returned.*

Region ρ_1 is a less-than region with its top-2 set $\{t_1,t_3\}$ identified, where the rank of anchor t_1 is $k'=2$. Then, ρ_1 will be further processed with rank quota 1 ($=3-2$) and the candidate set $\mathcal{C}_{\rho_1}=\{t_2,t_4,t_5,t_6,t_7\}$. Note that t_1 and t_3 are not in $\mathcal{C}_{\rho_1}$, since their ranks are already known. Our goal becomes finding the top-1 set in $\mathcal{C}_{\rho_1}$. To do this, we select t_2 to be the new anchor, which dominates all other tuples in $\mathcal{C}_{\rho_1}$, indicating that t_2 is exactly the top-1 set w.r.t. ρ_1 and ρ_1 will not be further partitioned. Combining $\{t_2\}$ with the previous top-2 set $\{t_1,t_3\}$, the final top-3 set returned for ρ_1 is $\{t_1,t_2,t_3\}$.

Region ρ_7 is a greater-than region since the rank k' of anchor t_1 is $5>k(=3)$, shown in shaded in Figure 2(b). Thus, ρ_7 is also recursively partitioned, with same rank quota $k=3$. The new candidate set $\mathcal{C}_{\rho_7}$ is $\{t_2,t_4,t_5,t_7\}$, with 3 tuples excluded: (a) anchor t_1 with rank $k'>k$, (b) t_6, which is dominated by t_1 and thus, ranks lower than t_1, and (c) t_3, which has a lower rank than t_1 since ρ_7 lies in $h^-_{t_3,t_1}$ (i.e., t_3 is not *a dominating competitor t_1 w.r.t. ρ_7). Assume that we select t_4 as the new anchor for ρ_7. Since t_4 is dominated by t_2 and t_4 dominates t_5, the only competitor of t_4 is t_7. We thus insert h_{t_7,t_4} (shown as a blue line segment), resulting in two sub-regions, namely ρ_8 and ρ_9. By the definition of h_{t_7,t_4}, ρ_8 (resp. ρ_9) contains all functions w such that t_7 ranks higher (resp. lower) than t_4 w.r.t. w. As a result, ρ_8 is an equal-to region with the top-3 set $\{t_2,t_4,t_7\}$, while ρ_9 is a less-than region with the top-2 set $\{t_2,t_4\}$ and it will be further partitioned.* □

For each recursive partitioning of region ρ, we need to choose a new anchor from the candidate set $\mathcal{C}_\rho$. To avoid the case where all sub-regions are less-than or greater-than regions (i.e., all of them need further processing), we select an anchor for each partitioning, so that there is at least one equal-to region. Specifically, we randomly pick a utility function w from ρ and find the top-k set w.r.t. w (i.e., $\mathbf{top}\text{-}k_w(\mathcal{C}_\rho)$). Then, we pick the k-th rank tuple in $\mathbf{top}\text{-}k_w(\mathcal{C}_\rho)$ as the new anchor.

Finally, the utility space Ω is partitioned into multiple equal-to regions. For each region ρ, we obtain the (unique) top-k set w.r.t. ρ, say $\mathbf{top}\text{-}k_\rho$. All these top-$k$ sets are returned.

Remark. We differ from [36] in the optimizations adopted below. (1) After obtaining the competitors of an anchor t_A for a region ρ, only *part* of competitors are processed in [36]. In contrast, we process *all* competitors, by creating a hyperplane for *each* competitor. Thus, we can deduce the dominance relationships between t_A and all its competitors. This allows us to apply our *speedup technique* (Section V-B) early. (2) When conducting recursive partitioning for a sub-region ρ' of region ρ, we consider not only the *static* relationships that hold globally in the dataset, but also the *dynamic* relationships (which varies for different sub-regions) deduced during previous partitioning, to reduce the candidates for consideration. In [36], only static relationships are used. (3) We represent a region ρ by its *vertices* (corner points) instead of its bounding hyperplanes. Given a hyperplane h', we determine the geometric relationship between ρ and h' efficiently by checking which halfspace the vertices of ρ reside. In [36], [7], similar checking is done by time-consuming linear programming.

C. Step 2 (α-fairness Computation)

For each feasible top-k set T obtained, we then compute its α-fairness in D w.r.t. G and p (i.e., $\alpha\text{-}FN_{G,p}(T|D)$). Specifi-

cally, we find the group membership of each top-k tuple in T, and then we obtain the overall representation of G in T. The α-fairness of T in D w.r.t. G and p is then computed according to the α-L_p distance between $R_T(G)$ and $P_D(G)$. After computing the α-fairness of each feasible top-k set, we can easily obtain a set $\mathcal{T}_{\max}$ of top-k sets with the maximum α-fairness.

D. Step 3 (Best Utility Function Finding)

After obtaining the set $\mathcal{T}_{\max}$ of all top-k sets with maximum α-fairness, we find the best function $w^* \in \Omega$ with the minimum modification penalty $m(w^*, w_0)$ from the given w_0.

Recall that for each set T in $\mathcal{T}_{\max}$, $T = \textbf{top-}k_\rho$ where ρ is a region in Ω. The problem is thus to find the nearest utility function of w_0 in ρ under the L_2 distance, which can be solved by *Quadratic Programming (QP)*. Denote by w_ρ the best utility function in region ρ, computed as follow.

$$
\begin{aligned}
w_\rho = \quad & \arg\min_{w} m(w, w_0) \\
\text{s.t.} \quad & (t' - t) \cdot w \leq 0, \quad \forall h_{t,t'} \in \mathcal{H}^+ && (1)\\
& (t' - t) \cdot w > 0, \quad \forall h_{t,t'} \in \mathcal{H}^- && (2)\\
& \textstyle\sum_{i=1}^{d} w[i] = 1 \text{ and } 0 \leq w[i] \leq 1, \forall i \in [1, d] && (3)
\end{aligned}
$$

where $\mathcal{H}^+$ (resp. $\mathcal{H}^-$) is the set of hyperplanes such that for each $h_{t,t'} \in \mathcal{H}^+$ (resp. $\mathcal{H}^-$), ρ is bounded by $h_{t,t'}$ and ρ lies in $h^+_{t,t'}$ (resp. $h^-_{t,t'}$). Here Constraints (1)-(2) ensure that w is in targeted region ρ and Constraint (3) specifies the utility space Ω. To solve this QP, we adopt the interior-point algorithm [39].

We solve the above QP for each top-k set in $\mathcal{T}_{\max}$. Finally, the best utility function w^* with the minimum penalty is returned and we obtain the optimal top-k set $T^* = \textbf{top-}k_{w^*}$.

V. Fast Algorithm

Although the basic implementation **FairTQ-Exact** adopts an efficient approach to obtain all the top-k sets, it is possible that there are still many top-k sets to process. In this section, we propose a fast exact algorithm under the same framework.

Our intuition is as follows. In Step 1 of feasible top-k sets enumeration (as introduced in Section IV-B), we may obtain a less-than region ρ during partitioning. Although we do not know the complete top-k set at this point for region ρ, we have determined a subset of the top-k set T from any utility function in ρ. With such subset, the α-fairness of T can be upper-bounded, and we can terminate the processing for ρ if ρ does not contain the fairest utility function based on this upper bound. Similarly, for a greater-than region ρ, the top-k set T of any utility function in ρ is selected from a smaller candidate set (i.e., the candidate set is a super-set of T). Thus, the α-fairness of T could also be upper-bounded.

In the following, we first formulate the upper bounds on the α-fairness of the top-k sets from less-than/greater-than regions, and then describe our fast algorithm utilizing these bounds.

A. Upper Bound of α-fairness

We first discuss the less-than regions, in which the resulting top-k sets could have their α-fairness upper-bounded based on a determined subset. Consider a top-k set T (= $\textbf{top-}k_w$ where w lies in a less-than region). Let T' be the subset of T that has been determined. We know the group membership of each tuple in T'. Although the remaining tuples in $T \setminus T'$ are not decided yet, the remaining number of tuples in each group is at most $k - |T'|$. Thus, for the "partially" determined top-k set T (containing subset T'), the representational proportion of each group G_j (for $j \in [1, m]$) in T (i.e., $R_T(G_j)$) satisfies

$$\frac{|T'_j|}{k} \leq R_T(G_j) \leq \frac{|T'_j| + k - |T'|}{k}. \quad (1)$$

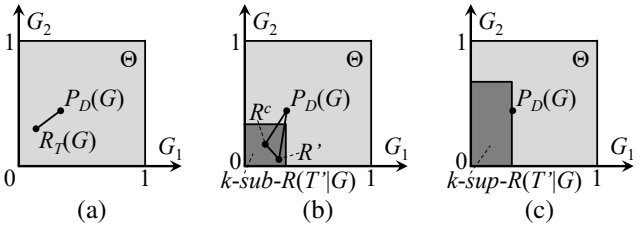


Fig. 3: Examples of (a) Proportion Space, (b) k-subset-extended Region and (c) k-superset-extended Region

We then explore from a geometric perspective to form the upper bound of the α-fairness of T. Recall that $R_T(G)$ and $P_D(G)$ are two points inside the proportion space Θ in the m-dimensional space. Equation 1 bounds a possible region containing $R_T(G)$ in Θ. Specifically, we define the *k-subset-extended region* of T' w.r.t. G, denoted by $k\text{-}sub\text{-}R(T'|G)$, to be the region in Θ containing all m-dimensional points, whose j-th value (for $j \in [1, m]$) is in $[\frac{|T'_j|}{k}, \frac{|T'_j|+k-|T'|}{k}]$.

Consider the less-than region ρ_2 in Figure 2(b) where $k = 3$. The top-2 set T' for this region is $\{t_1, t_3\}$ as identified in Example IV.3. For $G = \{\text{Female, African-American}\}$, it is easy to find that $|T'_1| = |T'_2| = 0$. Thus, $k\text{-}sub\text{-}R(T'|G)$ is a square region whose side length is 0.333 (both dimensions range from 0 to 0.333), as shown dark gray in Figure 3(b). In general, all k-subset-extended regions are m-dimensional boxes and thus we call each $k\text{-}sub\text{-}R(T'|G)$ as a *box region*.

With the k-subset-extended region, we can obtain the upper bound of the α-fairness of T, given that T has a subset T' determined. By the definition of α-fairness, finding the upper bound of the α-fairness of T is equivalent to finding the lower bound of the α-L_p distance between $R_T(G)$ and $P_D(G)$. Since $R_T(G)$ is in $k\text{-}sub\text{-}R(T'|G)$, one may want to find the exact α-L_p distance between $P_D(G)$ and a point in $k\text{-}sub\text{-}R(T'|G)$. This, however, is costly, especially for $\alpha > 0$ and arbitrary p.

In the following, we propose an efficient way to find an *approximate* lower bound of $\alpha\text{-}L_p\langle R_T(G), P_D(G)\rangle$. We first connect the α-L_p distance and the L_p distance as follows.

Lemma V.1. *Give a top-k set T of D,*

$$\alpha\text{-}L_p\langle R_T(G), P_D(G)\rangle \geq L_p\langle R_T(G), P_D(G)\rangle - m^{1/p} \cdot \alpha.$$

By Lemma V.1, to approximate the lower bound of the α-L_p distance, it suffices to compute a lower bound on the L_p distance. To do this, we leverage the triangle inequality of the L_p distance (for $p \geq 1$) [40]. We first choose the center point of the box in $k\text{-}sub\text{-}R(T'|G)$, says R^c. By triangle inequality, the L_p distance between $P_D(G)$ and any point R' in $k\text{-}sub\text{-}R(T'|G)$ is lower bounded by the difference

between $L_p\langle R^c, P_D(G)\rangle$ and $L_p\langle R', R^c\rangle$, i.e., $L_p\langle R', P_D(G)\rangle \geq L_p\langle R^c, P_D(G)\rangle - L_p\langle R', R^c\rangle$ (see Figure 2(b) for an example of Euclidean distance (i.e., $p = 2$)), where $L_p\langle R', R^c\rangle$ can be upper bounded by half the diagonal distance of the box, i.e., $L_p\langle R', R^c\rangle \leq m^{1/p} \cdot \frac{k-|T'|}{2k}$ (see details in [37]).

Putting these together, we have $\alpha\text{-}L_p\langle R_T(G), P_D(G)\rangle \geq L_p\langle R^c, P_D(G)\rangle - m^{1/p} \cdot \frac{k-|T'|}{2k} - m^{1/p} \cdot \alpha$ and the *subset-extended upper bound* of a less-than region ρ with a subset T' determined, denoted by $sub\text{-}UB(\rho, T')$, is defined to be

$$1 - \frac{1}{m^{1/p}} \cdot \max\{0, L_p\langle R^c, P_D(G)\rangle) - m^{1/p} \cdot \frac{k-|T'|}{2k} - m^{1/p} \cdot \alpha\}.$$

Clearly, given any top-k set T that has a subset T' determined, the α-fairness of T is at most $sub\text{-}UB(\rho, T')$.

We can also form the similar upper bounds for greater-than regions, which are based on the super-set determined. Consider a top-k set T ($= \mathbf{top}\text{-}k_w$ where w lies in a greater-than region). With a slight abuse of notation, also denote T' by the super-set of T that has been determined. Although it is unclear which k tuples in T' are eventually selected into T, the number of tuples in T with group membership G_j is at most $|T'_j|$, i.e., $|T_j| \leq |T'_j|$. Thus, the (representational) proportion of each group G_j in T is at least 0 and at most $\frac{|T'_j|}{k}$.

Similarly, the region containing $R_T(G)$ in the proportion space Θ can be bounded. We define the *k-superset-extended region* of T' w.r.t. G, denoted by $k\text{-}sup\text{-}R(T'|G)$, to be the region in the proportion space Θ containing all m-dimensional points, whose j-th value (for $j \in [1, m]$) is in $[0, \min\{1, \frac{|T'_j|}{k}\}]$, where the minimum ensures that $k\text{-}sup\text{-}R(T'|G)$ is inside Θ.

Consider the greater-than region ρ_7 in Figure 2(b). As discussed in Example IV.3, the determined super-set $T' = \{t_2, t_4, t_5, t_7\}$, and thus $|T'_1| = 1$ for group Female and $|T'_2| = 2$ for group African-American. The region $k\text{-}sup\text{-}R(T'|G)$ has ranges $[0, 0.333]$ and $[0, 0.667]$ for the two dimensions, respectively, as also shown in dark gray in Figure 3(c).

Below we present the lower bound of $\alpha\text{-}L_p\langle R_T(G), P_D(G)\rangle$ for a top-k set T that has a super-set T' determined, based similarly on triangle inequality and the α-relationship.

Lemma V.2. *Given a top-k set T of D and its super-set T',*

$$\alpha\text{-}L_p\langle R_T(G), P_D(G)\rangle \geq L_p\langle R^c, P_D(G)\rangle - L_p\langle R^c, R^o\rangle - m^{\frac{1}{p}} \cdot \alpha.$$

where R^c is the center of $k\text{-}sup\text{-}R(T'|G)$, R^o is origin of Θ.

Slightly different from the less-than regions, there is a term $L_p\langle R^c, R^o\rangle$ in Lemma V.2. This is because the maximum L_p distance between R^c and a point in $k\text{-}sup\text{-}R(T'|G)$ is also bounded by half the diagonal distance of the box, which now is exactly $L_p\langle R^c, R^o\rangle$ since R^o is a vertex of the box.

Based on Lemma V.2, we define the *superset-extended upper bound* of a greater-than region ρ with a super-set T' determined, denoted by $sup\text{-}UB(\rho, T')$, to be

$$1 - \frac{1}{m^{1/p}} \cdot \max\{0, L_p\langle R^c, P_D(G)\rangle - L_p\langle R^c, R^o\rangle - m^{1/p} \cdot \alpha\}.$$

For any w in a greater-than region ρ, the α-fairness of $\mathbf{top}\text{-}k_w$ is at most $sup\text{-}UB(\rho, T')$. Note that the upper bounds in both greater/less-than regions could be computed in $O(m)$ time.

B. Fast Algorithm with BnB

In this section, we leverage the upper bounds formed in previous section and propose a fast algorithm based on the same framework as **FairTQ-Exact**. It follows the idea of branch-and-bound [41], and thus we call it **FairTQ-Exact-BnB**.

The main difference of **FairTQ-Exact-BnB** compared to basic implementation **FairTQ-Exact** is that it applies effective speedup techniques, by combining Step 2 (top-k set enumeration) with Step 3 (α-fairness computation), so as to terminate the processing for more regions than just equal-to regions.

In **FairTQ-Exact-BnB**, we maintain the currently best α-fairness, denoted by F^*, and terminate the processing for a greater/less-than region if it cannot contain a utility function w.r.t. which the top-k set has higher α-fairness than F^*. To achieve that, we maintain greater-than and less-than regions with an efficient structure. We adopt the priority queue in our implementation so as to support the best-first search strategy.

We present the details of **FairTQ-Exact-BnB**. Firstly, we follow the same filtering step to obtain a subset D' of candidate top-k tuples. Then, we maintain the following structures: (1) the current best α-fairness F^* (initialized to ∞), (2) the current best utility function w^* (whose fairness is F^*) with the smallest modification penalty, and (3) a priority queue $\mathcal{Q}$ of regions (initialized to an empty queue). Each region ρ in $\mathcal{Q}$ is associated with (a) a fairness upper bound $\mathcal{U}_\rho$, (b) a rank quota r, (c) a candidate set $\mathcal{C}_\rho$, and (d) a top-$(k-r)$ set T_ρ w.r.t. ρ. In other words, we only need to determine the remaining r tuples for ρ, to decide the desired top-k sets. Here part (a) is computed in Section V-A and parts (b)(c)(d) are computed as in Section IV.

For the refinement step, we first add the utility space Ω into $\mathcal{Q}$. Initially, we set $\mathcal{U}_\Omega = 1$, $r = k$, $\mathcal{C}_\Omega = D'$ and $T_\Omega = \varnothing$. Then we repeat the following sub-steps until $\mathcal{Q}$ is empty:

(1) We first pop a region, says ρ, that has the best (i.e., largest) fairness upper bound $\mathcal{U}_\rho$ from the priority queue $\mathcal{Q}$.

(2) We choose an anchor, says t_A, from $\mathcal{C}_\rho$. According to t_A, we split region ρ into a set of sub-regions. This step follows the same procedure as **FairTQ-Exact** in Section IV.

(3) For each sub-region, says ρ', we process it by its types.

- If ρ' is an *equal-to* region, we have determined the exact top-k set w.r.t. ρ', says T. Then, we compute the α-fairness of T (i.e., $\alpha\text{-}FN_{G,p}(T|D)$). There are three cases: (a) If the α-fairness of T is larger than the current best F^*, we update F^* and compute w^* accordingly. (b) If the α-fairness of T is equal to F^*, we find the best utility function in ρ' and update w^* if it incurs smaller penalty. (c) If the α-fairness of T is less than F^*, we do not make any updates.
- If ρ' is a *less-than* or *greater-than* region, we compute the upper bound of α-fairness of any top-k set T w.r.t. a utility function in this region (Section V-A). Specifically, for a *less-than* (resp. *greater-than*) region ρ', we can find a subset (resp. super-set) of the top-k set, says T', with which we can compute the subset-extended (resp. superset-extended) upper bound of ρ'. If this upper bound is smaller than the current best α-fairness F^*, we terminate the processing for ρ'; otherwise, we add ρ' into queue $\mathcal{Q}$ for further processing.

Correctness. The following theorem proves our correctness.

Theorem V.1 (Correctness)**.** *Given a dataset D, a real number α, a positive integer k, a list G of group memberships, a real number p and a utility function w_0, both **FairTQ-Exact** and **FairTQ-Exact-BnB** return the utility function w^* such that α-$FN_{G,p}(\textbf{top}\text{-}k_{w^*}|D)$, is maximized and meanwhile, $m(w^*, w_0)$ is minimized under the maximized α-fairness.*

VI. Experiment

We conducted extensive experiments[1] on synthetic and real datasets to evaluate our effectiveness and efficiency on a Linux PC with a 2.66 GHz CPU and 48 GB main memory.

Datasets. We used 3 real datasets, *XING*, *COMPAS*, and *DOT*, and 2 synthetic datasets, *Independent* and *Anti-correlated*.

XING [4] has 2,280 user profiles in a job finding website [42]. We used two indicators *work_experience* and *education_experience* as scoring attributes and *gender* as the protected attribute to prevent unfair treatment of women in hiring.

COMPAS [43] has the criminal offense information of 6,889 individuals for finding those are likely to recidivate. We set *race* as one protected attribute to prevent discrimination to African-Americans. To test multiple (and intersectional) protected attributes, we set *gender* as another protected attribute. We consider three groups, namely African-Americans, males and African-American males, using three scoring attributes, namely *c_days_from_compas, juv_other_count* and *start* [7].

DOT [44] is a flight on-time dataset with over 1M flights conducted by 12 US carriers in the first three months of 2016. Following [7], we set *op_unique_carrier* as the protected attribute (with 12 groups each corresponding to a carrier) and set three scoring attributes to be *dep_delay*, *taxi_in* and *arr_delay*.

For synthetic datasets, attribute values in *Independent* are generated independently, while in *Anti-correlated*, tuples with a large value in one attribute tend to have small values on other attributes. The protected attribute is randomly generated.

For each dataset, we normalized the values of each scoring attribute into range $[0, 1]$ via the min-max normalization.

Baselines. We compared **FairTQ-Exact** and **FairTQ-Exact-BnB** with four baselines: (1) **SR-Adapt**: We adapted *SATREGIONS* in [7] using their approach of handling hyperplanes to get the exact fairest top-k set. (2) **Greedy**: We adapted the idea of [15], which iteratively selects a tuple t from D such that t is not dominated by any remaining tuples in D (thus, no ranking violation) and meanwhile, t belongs to the group that contributes the least in the top-k set. A utility function with minimum penalty can be returned from a region, by intersecting the halfspaces formed by the top-k tuples (if any). (3) **Prop**, which preserves the proportion of each group in the top-k set, by first sorting tuples by their utilities w.r.t. w_0 and then forming the top-k set by selecting the top k_i tuples from the i-th group where k_i/k equals the populational proportion of the i-th group. (4) **FA*IR**: The original algorithm of [4]. Algorithm **FA*IR** was implemented in Python (using its original implementation) and others were implemented in C++.

[1]Code available at: https://github.com/satansin/FairTQ

Note that we implemented **FairTQ-Exact**, **FairTQ-Exact-BnB** and **SR-Adapt** with a simple trick (see details in [37]). Since the fairness is bounded by 1, if we find a utility function w with the maximum fairness (i.e., 1), its penalty is an upper bound of the optimal penalty. All regions (regardless of types) that do not contain any utility function with penalty smaller than that of w are pruned, without computing its exact fairness.

Parameter Settings. We studied the following parameters: (1) the dimensionality d (i.e., the number of scoring attributes), (2) the dataset size $|D|$, (3) the number of groups $m(= |G|)$, (4) the parameter k in the top-k query, (5) the fairness parameter α and (6) the parameter p in the α-L_p distance formation.

Our parameter settings mainly follow [7]. In real datasets, we used the original dataset by default. In synthetic datasets, $d = 3$, $|D| = 100$k and m = 2 by default. The default values of k, α and p are 10, 0.1 and 2, respectively, for all datasets.

Measurements. We adopted four measurements: (1) the response time, (2) the α-fairness, (3) the penalty of the returned utility function from w_0, (4) violation count: the sum of ranking violations occurred within the top-k set and between the top-k tuples and the remaining tuples (see Section II). We ran each experiment 10 times, with 10 random utility functions w_0, and report the average here. For the lack of space, we only report the results on some datasets; other results are consistent.

A. Results on Real Datasets

Effect of α. We first studied the effect of parameter α in our fairness model on fairness and modification penalty, by varying it from 0 to 0.2 on dataset *XING*; **Greedy**, **FA*IR** and **Prop** are excluded since they do not rely on α. Note that the α-fairness under different α values cannot be directly compared. We picked a reference value α_0 and compared the α_0-fairness of the returned top-k sets, where we set α_0 to be 0.1. Figure 4(a) give the results (note that all exact algorithms give the same results). It could be seen that when α increases, the returned set is less fair (since the α_0-fairness decreases), while the returned utility function has smaller modification penalty. This verifies the effect of parameter α to achieve the trade-off between fairness and modification penalty.

We showcase a typical case study of this result. In dataset *XING*, the populational proportions of males and females are 26.4% and 73.6%, respectively, and k is set to 23 (i.e., $1\%|D|$). Consider the input utility function $w_0 = (0.32, 0.68)$. When $\alpha = 0$ (the strictest fairness measurement), the returned utility function w is $(0.45, 0.55)$, leading to a top-k set T with 17 males and 6 females (with representational proportions 26.1% and 73.9%, respectively). Although T is nearly perfectly fair, w deviates much from w_0. When $\alpha = 0.1$ instead, the returned function becomes $(0.34, 0.66)$, much closer to w_0, while the returned set has 16 males and 7 females (with representational proportions 30.4% and 69.6%, respectively), which is still fair.

Moreover, Figure 4(b) shows that when α increases, all algorithms run faster. Since larger α indicates a less strict fairness measurement, more top-k sets could reach the maximum fairness (i.e., 1), triggering the implementation trick. In this

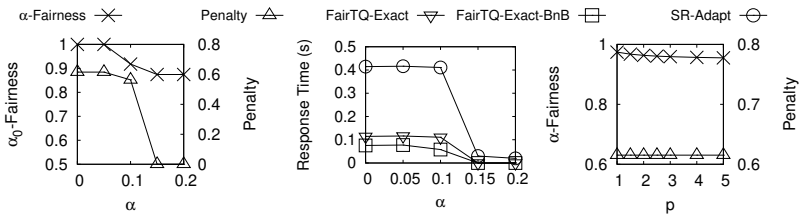


Fig. 4: Effect of α, p on *XING*

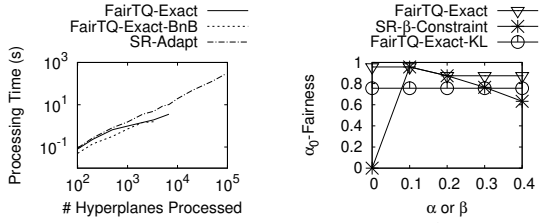


Fig. 5: # Hyperplanes & Fairness on *COMPAS*

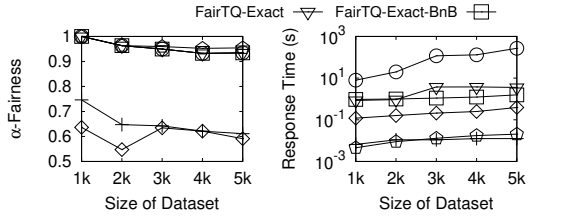


Fig. 6: Effect of $|D|$ on *COMPAS*

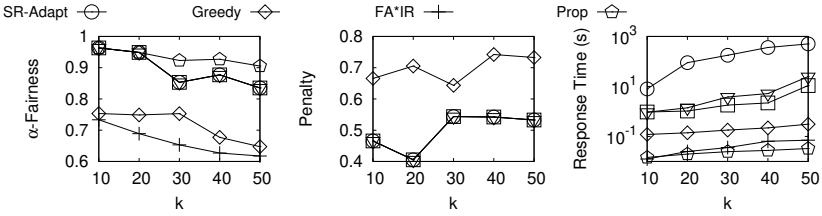


Fig. 7: Effect of k on *COMPAS*

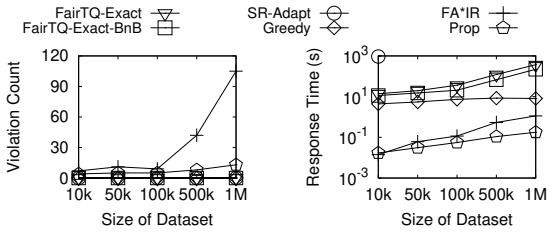


Fig. 8: Effect of $|D|$ on *DOT*

case, many regions are pruned without computing the exact fairness, leading to shorter time. Given many regions with the maximum fairness, the current best utility function is close to w_0, leading to a tight penalty bound and more regions pruned.

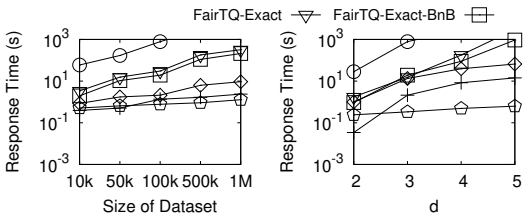
Effect of p in α-L_p distance. We also studied the effect of parameter p (**Greedy**, **FA*IR** and **Prop** are also excluded since they do not rely on p). Apart from the most commonly used p values of 1 and 2, we also tested some other values (note that $p \geq 1$). For a consistent comparison, we picked a reference value p_0 and compared the α-fairness under the α-L_{p_0} distance for different settings of p. Here, we set $p_0 = 2$. As shown in Figure 4(c), when p increases, the fairness of the returned result slightly decreases, which indicates that setting $p = 1$ leads to the strictest α-fairness measurement, while the response time is almost indifferent to the values of p (not shown).

Effectiveness of the fairness model. We studied the effectiveness of our problem setting that returns the fairest top-k set under our fairness model (i.e., α-fairness using the L_p distance). For comparison, we consider two additional baselines. The first baseline is the original *SATREGIONS* that aims to satisfy a fairness constraint. We set this constraint using our fairness model as the criterion, i.e., the α-fairness of the top-k set must be at least a threshold $(1 - \beta)$ (so that smaller β indicates a stricter constraint, consistent with our parameter α). We denote this baseline as **SR-β-Constraint**. The second baseline is to replace the L_p distance in our fairness model with the KL-divergence, denoted by **FairTQ-Exact-KL**.

As shown in Figure 5(b), we varied α/β from 0 to 0.4 on dataset *COMPAS* and compared the α_0-fairness ($\alpha_0 = 0.1$). Note that **FairTQ-Exact-KL** does not rely on α/β, and it has weaker fairness due to the intersectional and incomplete groups on this dataset. For **SR-β-Constraint**, when the constraint is set the strictest (i.e., $\beta = 0$), there is no top-k set in the dataset satisfying this constraint, and thus there is no output, while if the constraint is set too loose (i.e., $\beta > 0.3$), the returned result is unfair (i.e., the α_0-fairness lower than 0.6). In contrast, **FairTQ-Exact** consistently has the highest fairness.

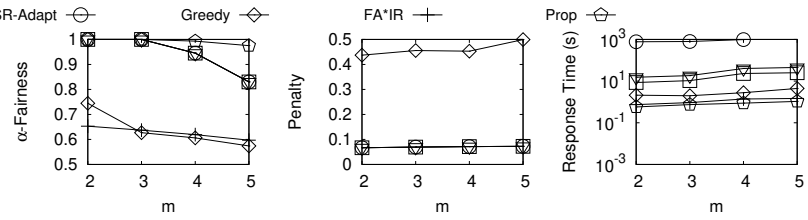
Effect of $|D|$. We studied the effect of dataset size $|D|$ by randomly selecting a subset of 1k to 5k tuples (resp. 10k to 1M tuples) from dataset *COMPAS* (resp. *DOT*) in Figure 6 (resp. Figure 8). As shown in Figures 6(b) and 8(b), when $|D|$ increases, the response time for all algorithms increases, as expected. Nevertheless, **FairTQ-Exact** and **FairTQ-Exact-BnB** outperform the exact baseline **SR-Adapt** by 1-2 orders of magnitude. Due to the effective branch-and-bound pruning, **FairTQ-Exact-BnB** further shortens the response time by around 40-50% compared with **FairTQ-Exact** on large datasets. However, on small datasets (e.g., 1k-2k tuples from *COMPAS*), **FairTQ-Exact** is slightly faster than **FairTQ-Exact-BnB**, which indicates that when the hyperplanes to be processed are not too many, **FairTQ-Exact** is still efficient. Although baseline **Greedy** has shorter response time than our algorithms, the top-k sets it returns are poor in fairness (e.g., around 0.6 in Figure 6(a)). Baseline **FA*IR** and **Prop** are also fast, but they incur ranking violations in Figure 8(a). In other words, **FA*IR** and **Prop** do not return a valid utility function (even though **Prop** always gives the theoretically highest fairness since it directly forms the top-k according to the populational proportion of each group), and their response time does not include the time for finding such utility function. Finally, we observe that the α-fairness of all algorithms slightly decreases when $|D|$ increases. This is because that in real datasets, the data distribution is imbalanced. Some tuples that have high attribute values will always be included in the result, even there are more candidate tuples to be selected, leading to discrimination to other tuples and smaller fairness.

More closely, we show how our algorithms outperform the most relevant baseline **SR-Adapt**. In Figure 5(a), we report the accumulated processing time under different numbers of



(a) (b)

Fig. 9: Effect of $|D|$ and d on *Independent*



(a) (b) (c)

Fig. 10: Effect of m on *Anti-correlated*

hyperplanes processed. As shown there, **FairTQ-Exact** and **FairTQ-Exact-BnB** not only process fewer hyperplanes than **SR-Adapt** (e.g., $< 10^4$ hyperplanes for **FairTQ-Exact**), due to our efficient top-k set enumeration, their average processing time of each hyperplane (i.e., the accumulated time divided by the number of hyperplanes) is also shorter. This is because we check the geometric relationship between hyperplanes and regions efficiently, without using the expensive linear programming as in **SR-Adapt**. Although **FairTQ-Exact-BnB** takes extra time of computing fairness bounds, it can prune more hyperplanes than **FairTQ-Exact** and thus, has the best efficiency.

Effect of k. When k is varied from 10 to 50 on dataset *COMPAS*, the response time of all algorithms increases, as shown in Figure 7(c). Similarly, **FairTQ-Exact-BnB** is 10-100 times faster than the exact baseline **SR-Adapt**, and outperforms **FairTQ-Exact** by 40% on large k values. Moreover, as shown in Figures 7(a) and (b), when k increases, it is also harder to achieve larger fairness in a given dataset, and thus, the fairness of returned results is smaller and fewer candidate sets can achieve the best fairness, leading the increased modification penalty to get the fairest top-k set. Nevertheless, baseline **Greedy** has large penalty and poor fairness although its response time is small, and the fastest baselines **FA*IR** and **Prop** do not return a valid utility function due to ranking violation.

Case Studies of Fairness. Setting G as {African-American, Male, African-American male}, dataset *COMPAS* has overall population $P_D(G) = (0.34, 0.72, 0.21)$. Our exact algorithm finds the fairest top-k set T with overall population $R_T(G) = (0.3, 0.7, 0.2)$, which is close to $P_D(G)$ and thus fair to all groups. Baseline **Greedy** returns a top-k set T' with $R_T(G) = (0.8, 0.7, 0.6)$, indicating its insufficiency of achieving fairness. Baseline **FA*IR** returns a top-k set where 90% of them are non-African Americans, since **FA*IR** controls the fairness by adding more non-African Americans into the top-k set. This, however, creates obvious discrimination to other races (since they are identified as the individuals who are likely to recidivate) while the African Americans are protected.

B. Results on Synthetic Datasets

Effect of $|D|$ and d. We also tested the effect of $|D|$ for the response time in synthetic dataset *Independent*, varying $|D|$ from 10k to 1M. As shown in Figure 9(a), the time efficiency of our algorithms are consistently superior as before. In particular, we are around 100 times faster than **SR-Adapt**. On the largest dataset of 1M tuples, **FairTQ-Exact-BnB** returns an exact result reasonably in 210s, while **SR-Adapt** cannot process more than 100k tuples in reasonable time (i.e., $< 1000s$).

We tested the effect of d in Figure 9(b). Due to the nature of computational geometry, the complexity of our problem rises with larger d, and all algorithms are slower. Nonetheless, **FairTQ-Exact** runs in more practical time than **SR-Adapt** (which fails to handle more than 3 scoring attributes in 1000s).

Effect of m. We varied the number of groups (i.e., m) from 2 to 5 in Figure 10 on dataset *Anti-correlated*. Since more groups lead to a harder condition of fairness, the α-fairness tends to decrease (Figure 10(a)), while the penalty increases slightly (Figure 10(b)), consistent as previous results. The harder condition also makes us more difficult to find a top-k set with the optimal fairness. Thus, the response time slightly increases (Figure 10(c)). Still, our algorithms (esp. **FairTQ-Exact-BnB**) is consistently faster than **SR-Adapt** and scales well w.r.t. m.

C. Summary

The experiments on real and synthetic datasets demonstrated the superiority of our algorithms. Specifically, our exact algorithms (which guarantees to return the fairest top-k set) achieves one to two orders of magnitude shorter response time compared with the exact baseline. When the dataset size scales to 1M, our algorithms have reasonable response time (i.e., around 210s), while the response time of the exact baseline is unacceptable (e.g., $>$ 1000s). With our case study, we also demonstrate the effectiveness of our fairness model, which strikes a balance between fairness and modification penalty.

VII. Conclusion

In this paper, we design a fairness model called α-fairness to measure the fairness of top-k results, and propose a framework to find a fair utility function. We propose two algorithms under this framework with speedup techniques to improve efficiency. We conducted experiments on real and synthetic datasets to demonstrate the effectiveness and efficiency of our algorithms.

As for future research, we would extend our fairness problem to different scenarios. For example, we could consider relative ranking among tuples into our fairness model. We could also explore other fairness measurements (e.g., some KL-divergence variants) to handle unknown protected groups.

Acknowledgment

Dr. Min Xie is supported in part by China NSFC 62202313, Guangdong Basic and Applied Basic Research Foundation 2022A1515010120. Dr. Bo Tang was partially supported by Shenzhen Fundamental Research Program (Grant No. 20220815112848002). He is also affiliated with the Research Institute of Trustworthy Autonomous Systems, Southern University of Science and Technology, Shenzhen, China.

REFERENCES

[1] M. Goncalves and M.-E. Vidal, "Top-k skyline: A unified approach," in *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*. Springer, 2005, pp. 790–799.

[2] J. Lee, G.-w. You, and S.-w. Hwang, "Personalized top-k skyline queries in high-dimensional space," *Information Systems*, vol. 34, no. 1, pp. 45–61, 2009.

[3] W. Wang, R. C.-W. Wong, and M. Xie, "Interactive search for one of the top-k," in *Proceedings of the 2021 International Conference on Management of Data*, 2021, pp. 1920–1932.

[4] M. Zehlike, F. Bonchi, C. Castillo, S. Hajian, M. Megahed, and R. Baeza-Yates, "Fa* ir: A fair top-k ranking algorithm," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, 2017, pp. 1569–1578.

[5] "Data usa: Massachusetts institute of technology," 2023. [Online]. Available: https://datausa.io/profile/university/massachusetts-institute-of-technology#admissions

[6] S. N. Bleich, M. G. Findling, L. S. Casey, R. J. Blendon, J. M. Benson, G. K. SteelFisher, J. M. Sayde, and C. Miller, "Discrimination in the united states: experiences of black americans," *Health services research*, vol. 54, pp. 1399–1408, 2019.

[7] A. Asudeh, H. Jagadish, J. Stoyanovich, and G. Das, "Designing fair ranking schemes," in *Proceedings of the 2019 international conference on management of data*, 2019, pp. 1259–1276.

[8] M. Zehlike, K. Yang, and J. Stoyanovich, "Fairness in ranking, part i: Score-based ranking," *ACM Computing Surveys*, vol. 55, no. 6, pp. 1–36, 2022.

[9] P. Jacobs, "Legacy admissions policies were originally created to keep jewish students out of elite colleges," *Business Insider*, 2013.

[10] J. Karabel, *The chosen: The hidden history of admission and exclusion at Harvard, Yale, and Princeton*. Houghton Mifflin Harcourt, 2005.

[11] Y. Hong, J. Vaidya, and H. Lu, "Search engine query clustering using top-k search results," in *2011 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology*, vol. 1. IEEE, 2011, pp. 112–119.

[12] S. Niu, J. Guo, Y. Lan, and X. Cheng, "Top-k learning to rank: labeling, ranking and evaluation," in *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, 2012, pp. 751–760.

[13] M. Hardt, E. Price, and N. Srebro, "Equality of opportunity in supervised learning," *Advances in neural information processing systems*, vol. 29, 2016.

[14] R. Courtland, "Bias detectives: the researchers striving to make algorithms fair," *Nature*, vol. 558, pp. 357–360, 2018.

[15] L. E. Celis, D. Straszak, and N. K. Vishnoi, "Ranking with fairness constraints," *arXiv preprint arXiv:1704.06840*, 2017.

[16] K. Yang and J. Stoyanovich, "Measuring fairness in ranked outputs," in *Proceedings of the 29th international conference on scientific and statistical database management*, 2017, pp. 1–6.

[17] S. C. Geyik, S. Ambler, and K. Kenthapadi, "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*, 2019, pp. 2221–2231.

[18] K. Cachel, E. Rundensteiner, and L. Harrison, "Mani-rank: Multiple attribute and intersectional group fairness for consensus ranking," in *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, 2022, pp. 1124–1137.

[19] M. Hardt, E. Price, and N. Srebro, "Equality of opportunity in supervised learning," *Advances in neural information processing systems*, vol. 29, 2016.

[20] F. Kamiran and T. Calders, "Data preprocessing techniques for classification without discrimination," in *Knowledge and Information Systems*, vol. 33, 2011, pp. 1–33.

[21] C. Dwork, M. Hardt, T. Pitassi, O. Reingold, and R. Zemel, "Fairness through awareness," in *Proceedings of the 3rd innovations in theoretical computer science conference*, 2012, pp. 214–226.

[22] A. Fabris, G. Silvello, G. A. Susto, and A. J. Biega, "Pairwise fairness in ranking as a dissatisfaction measure," in *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*, 2023, pp. 931–939.

[23] D. García-Soriano and F. Bonchi, "Maxmin-fair ranking: individual fairness under group-fairness constraints," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2021, pp. 436–446.

[24] A. Singh and T. Joachims, "Fairness of exposure in rankings," in *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, 2018, pp. 2219–2228.

[25] K. Yang, V. Gkatzelis, and J. Stoyanovich, "Balanced ranking with diversity constraints," *arXiv preprint arXiv:1906.01747*, 2019.

[26] J. Li, Y. Moskovitch, and H. Jagadish, "Detection of groups with biased representation in ranking," in *2023 IEEE 39th International Conference on Data Engineering (ICDE)*, 2023, pp. 2167–2179.

[27] A. Singh and T. Joachims, "Policy learning for fairness in ranking," *Advances in neural information processing systems*, vol. 32, 2019.

[28] B. Salimi, L. Rodriguez, B. Howe, and D. Suciu, "Interventional fairness: Causal database repair for algorithmic fairness," in *Proceedings of the 2019 International Conference on Management of Data*, 2019, pp. 793–810.

[29] P. Lahoti, K. P. Gummadi, and G. Weikum, "Operationalizing individual fairness with pairwise fair representations," *arXiv preprint arXiv:1907.01439*, 2019.

[30] C. Karako and P. Manggala, "Using image fairness representations in diversity-based re-ranking for recommendations," in *Adjunct Publication of the 26th Conference on User Modeling, Adaptation and Personalization*, 2018, pp. 23–28.

[31] E. Chzhen, C. Denis, M. Hebiri, L. Oneto, and M. Pontil, "Leveraging labeled and unlabeled data for consistent fair binary classification," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[32] A. Cotter, M. Gupta, H. Jiang, N. Srebro, K. Sridharan, S. Wang, B. Woodworth, and S. You, "Training well-generalizing classifiers for fairness metrics and other data-dependent constraints," in *International Conference on Machine Learning*. PMLR, 2019, pp. 1397–1405.

[33] J. Zheng, Y. Ma, W. Ma, Y. Wang, and X. Wang, "Happiness maximizing sets under group fairness constraints," *Proceedings of the VLDB Endowment*, vol. 16, no. 2, pp. 291–303, 2022.

[34] M. M. Islam, D. Wei, B. Schieber, and S. B. Roy, "Satisfying complex top-k fairness constraints by preference substitutions," *Proceedings of the VLDB Endowment*, vol. 16, no. 2, pp. 317–329, 2022.

[35] J. Lee, G.-w. You, and S.-w. Hwang, "Personalized top-k skyline queries in high-dimensional space," *Information Systems*, vol. 34, pp. 45–61, 2009.

[36] K. Mouratidis and B. Tang, "Exact processing of uncertain top-k queries in multi-criteria settings," *Proceedings of the VLDB Endowment*, vol. 11, no. 8, pp. 866–879, 2018.

[37] A. Author(s), "Fair top-k query on alpha-fairness (technical report)," 2023. [Online]. Available: https://github.com/satansin/FairTQ/FairTopK-TechnicalReport.pdf

[38] D. Papadias, Y. Tao, G. Fu, and B. Seeger, "Progressive skyline computation in database systems," *ACM Transactions on Database Systems (TODS)*, vol. 30, no. 1, pp. 41–82, 2005.

[39] R. D. Monteiro and I. Adler, "Interior path following primal-dual algorithms. part ii: Convex quadratic programming," *Mathematical Programming*, vol. 44, no. 1-3, pp. 43–66, 1989.

[40] M. A. Khamsi and W. A. Kirk, *An introduction to metric spaces and fixed point theory*. John Wiley & Sons, 2011.

[41] J. Clausen, "Branch and bound algorithms-principles and examples," *Department of Computer Science, University of Copenhagen*, pp. 1–30, 1999.

[42] "Find the right job for you," 2023. [Online]. Available: https://www.xing.com/

[43] J. Angwin, J. Larson, S. Mattu, and L. Kirchner. (2023) Machine bias: Risk assessments in criminal sentencing. ProPublica. [Online]. Available: https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing

[44] U. S. D. of Transportation. (2023) Bureau of transportation statistics. [Online]. Available: https://www.transtats.bts.gov/Tables.asp?QO_VQ=EFD&QO_anzr=Nv4yv0r%20b0-gvzr%20cr4s14zn0pr%20Qn6n0