arXiv:2011.04219v1 [cs.CY] 9 Nov 2020

Mitigating Bias in Set Selection with Noisy Protected Attributes

Anay Mehrotra
Yale University

L. Elisa Celis
Yale University

November 10, 2020

Abstract

Subset selection algorithms are ubiquitous in AI-driven applications, including, online recruiting portals and image search engines, so it is imperative that these tools are not discriminatory on the basis of protected attributes such as gender or race. Currently, fair subset selection algorithms assume that the protected attributes are known as part of the dataset. However, attributes may be noisy due to errors during data collection or if they are imputed (as is often the case in real-world settings). While a wide body of work addresses the effect of noise on the performance of machine learning algorithms, its effect on fairness remains largely unexamined. We find that in the presence of noisy protected attributes, in attempting to increase fairness without considering noise, one can, in fact, *decrease* the fairness of the result!

Towards addressing this, we consider an existing noise model in which there is probabilistic information about the protected attributes (e.g., [19, 32, 56, 44]), and ask *is fair selection is possible under noisy conditions?* We formulate a "denoised" selection problem which functions for a large class of fairness metrics; given the desired fairness goal, the solution to the denoised problem violates the goal by at most a small multiplicative amount with high probability. Although the denoised problem turns out to be **NP**-hard, we give a linear-programming based approximation algorithm for it. We empirically evaluate our approach on both synthetic and real-world datasets. Our empirical results show that this approach can produce subsets which significantly improve the fairness metrics despite the presence of noisy protected attributes, and, compared to prior noise-oblivious approaches, has better Pareto-tradeoffs between utility and fairness.

Contents

1 Introduction

The subset selection problem arises in various contexts including online job portals (where an algorithm shortlists candidates to show to the recruiter), university admissions (where a panel admits a subset of students), and online search (where the platform selects a subset of the results in response to a user query) [26, 48, 64, 46]. The basic problem is as follows: There are m *items*, and each item $i \in [m]$ has a utility $w_i \geq 0$, i.e., the value it adds to the subset. The goal is to select a subset of $n \ll m$ items which has the largest total utility. Given the pervasiveness of subset selection tasks, it is crucial to ensure that subset selection algorithms do not propagate social biases. Consequently, there has been extensive work on developing fair algorithms for selection (and for the related problem of ranking); see [26, 12] for an overview. Many of these approaches ensure that the number of individuals selected from different socially salient groups (e.g., those defined by gender or race) satisfy some fairness constraints and/or improve along a given fairness metric. Towards this, these algorithms assume (exact) access to the corresponding *protected attributes* of individuals.

However, in practice, these attributes can be erroneous, unavailable for some individuals, or missing entirely [50, 25, 62]. For instance, in healthcare, patients' ethnic information can be incorrectly recorded [62] or left blank [25].[1] When this data is missing, probabilistic methods based on other proxy information are used to "impute" these protected attributes [28, 30, 27, 22]. For instance, when assessing if lenders comply with fair lending policies, the Consumer Financial Protection Bureau uses last name and geolocation to impute consumers' race [8]. Similar approaches have also been used in the context of healthcare [49, 31]. Additionally, online job platforms (such as, LinkedIn) use a user's data to infer their demographic information based on the data they have on other users [54]. Furthermore, in some cases, such as with images on the internet, protected attributes are missing for the entire datasets (and labeling all images is not viable). Inferring protected attributes is bound to have errors, which can affect the groups differently [7]. Thus, using imputed attributes as a black-box in subsequent fair algorithms, without accounting for their noise, can have an unexpected (and adverse) impact on the fairness achieved. For instance, [52, 14] observe that (noise oblivious) fair algorithms do not satisfy their fairness guarantee in the presence of noise.

To gain some intuition, consider the setting where we are given a set of candidates and would like to ensure proportional representation across individuals with different skin-tones, coded as White and non-White. Assume that the utilities of all candidates have a similar distribution, and so picking candidates with top n utilities proportionately represents them. Further, assume that the labels have a higher amount of noise for non-Whites than Whites.[2] Any "fair algorithm" which assumes that these noisy labels are correct, and selects a proportionate number of White and non-White candidates based on them, would violate proportional representation. In this case, adding fairness constraints increased the disparity. This leads us to the question addressed in this paper:

> *Can we develop a framework for selection which outputs an approximately fair subset despite noisy protected attributes?*

1.1 Our contributions

Building on prior work on fairness constraints [64, 16], we develop a framework for fair selection in the presence of noisy protected attributes. This framework allows for multiple and intersectional groups, and, given access to (unbiased[3]) probabilistic information about the true protected attributes, it can satisfy a large class of fairness constraints (including, demographic parity, proportional representation, and the 80% rule) with high probability.

Formally, we would like to solve an ideal optimization problem (Program Target) which satisfies the fairness constraints for the true (and unknown) protected attributes. Such problems have been studied by prior works, e.g., [63, 64, 70]. However, since we do not have the true protected attributes, we cannot solve it directly using their approaches. Instead, we formulate a "denoised" problem (Program Denoised); such that, an optimal solution of Program Denoised has an optimal utility for Program Target and violates

[1] Recently, this received public attention when attempting to estimate the racial disparities in COVID19 infections showed large discrepancies [5].

[2] For instance, as observed in commercial image-based gender classifiers [7].

[3] Here, unbiased refers to the statistical notion of an unbiased estimator.

the fairness constraints of Program Target by at most a small multiplicative factor with high probability (Lemma 3.4). Although Program Denoised turns out to be **NP**-hard (Theorem 3.6), we develop a linear-programming based approximation algorithm for it. This, in turn, implies an approximation algorithm for Program Target.

We empirically study the fairness achieved by this approach with respect to standard fairness metrics (e.g., risk difference) on both synthetic and real-world datasets. We also study the performance of existing fair algorithms in the presence of noise and benchmark our approach with them. We observe that our approach achieves the highest fairness and has a Pareto-optimal tradeoff between utility and fairness (on changing the strength of constraints). Interestingly, these observations also hold in our empirical results where, unlike what our theoretical results assume, we have skewed probabilistic information of the noisy attributes. Finally, our empirical results hint at potential applications of this approach, e.g., in online recruiting portals and image search engines.

1.2 Related work

Mitigating bias. An extensive body of work strives to mitigate bias and improve diversity in subset selection and the closely related ranking problem. We refer the reader to [26] for a comprehensive overview of work on diverse selection, and an excellent talk [12] which discusses work on curtailing bias in rankings. Closest to our setting, are approaches which use protected attributes to impose fairness constraints on algorithms for selection [64, 46] and ranking [17, 70, 63, 33, 67]. However, if the attributes are noisy, these could even increase the bias.

A different approach is to learn "unbiased utilities" by either using a *causal model* to capture the relation between attributes and utilities [51, 68] or by casting it as a multi-objective unconstrained optimization problem [69, 71]. The former approach explicitly uses the protected attributes to generate *counterfactuals*; thus, it can lead to unfair outcomes in the presence of noise (also see, Section 4.3). And the latter approach can lead to sub-optimal fairness if noisy data is not accounted for, as shown by works on fair classification [4, 52, 14].

In [35], it is empirically shown that when protected attributes are missing, proxy attributes can be used to improve fairness in classification. However, they do not consider how necessary noise resulting from the proxy attributes affects the fairness or accuracy.

Mitigating bias with noise. Works on curtailing bias with noisy information are relatively recent. Closest to this paper are those which consider noise in the protected attributes. In [4], conditions on the noise under which the popular post-processing method for fair classification by [39] reduces bias in terms of equalized odds are characterized. However, they only consider noise in the training samples and assume that the test samples are not noisy, which often doesn't hold in practice. In [52], an in-processing approach to fair classification is suggested; they show that applying tighter fairness constraints in existing fair classification frameworks can mitigate bias in terms of equalized odds and statistical parity with binary protected attributes. However, this approach does not extend to nonbinary protected attributes and to other definitions of fairness. In [14], an in-processing approach for fair classification which can mitigate bias with nonbinary and noisy protected attributes is developed. However, they assume that the noise only depends on the (unknown) underlying protected attributes, whereas, we also allow the noise to vary with nonprotected attributes and utility. Furthermore, [4, 52, 14] mitigate bias in classification tasks, and it is not clear how to extend these methods to subset selection.

In [19, 44], methods to reliably assess disparity in the setting where the protected attributes are entirely missing are proposed. We consider a similar noise model as the one they propose; however, the problem is fundamentally different as their goal is assessment rather than mitigation.

Noise models in literature. Several works in the machine learning literature consider noise in the predicted labels as opposed to in attributes, protected or otherwise [32, 3, 55, 56]. In this paper, we consider a noise model that arises from this line of work, but applied to the protected attributes rather than the label.

2 Model

For a natural number $n \in \mathbb{N}$ by $[n]$ we denote the set $\{1, 2, \ldots, n\}$, and for a real number $x \in \mathbb{R}$ by $\exp(x)$ we denote e^x. We use $\mathbb{I}[\cdot]$ to denote the indicator function, $o(1)$ to denote $O(1/n)$, and $\mathcal{U}(a, b)$ to denote the uniform distribution on interval $[a, b]$. Given a natural number $p \in \mathbb{N}$, Δ^p denotes the standard p-simplex.

2.1 Selection problem and noise model

Selection problem. In the classical selection problem, one is given m *items*, where each item $i \in [m]$ has a utility $w_i \geq 0$. An item's utility is the *value* it adds to the selection. The goal is to find a subset of n items which has the most total value. It is convenient to encode a subset with a binary selection vector $x \in \{0, 1\}^m$. Then, the classical selection problem is

$$\max_{x \in \{0,1\}^m} \sum\nolimits_{i=1}^{m} w_i x_i \quad \text{s.t.,} \ \sum\nolimits_{i=1}^{m} x_i = n. \tag{1}$$

Protected attributes. We consider $s \in \mathbb{N}$ protected attributes (such as, gender or race), where for $k \in [s]$, the k-th protected attribute can take $p_k \in \mathbb{N}$ values (such as, different genders or races). Let $\mathcal{X}$ be the domain of all other nonprotected attributes. Fix a joint distribution over $\mathcal{D} := \mathbb{R}_{\geq 0} \times [p_1] \times \cdots \times [p_s] \times \mathcal{X}$. Then, each item $i \in [m]$ is represented by the tuple

$$(w_i, z_i^{(1)}, \ldots, z_i^{(s)}, a_i) \in \mathbb{R}_{\geq 0} \times [p_1] \times \cdots \times [p_s] \times \mathcal{X},$$

and is drawn independently from this joint distribution. We observe the utility w_i and nonprotected attributes a_i, but do *not* observe the protected attributes $(z_i^{(1)}, \ldots, z_i^{(s)})$. Instead, we observe a *noisy version* $(\widehat{z}_i^{(1)}, \ldots, \widehat{z}_i^{(s)})$ of them (for each $i \in [m]$).

For each attribute-value pair $k \in [s]$ and $\ell \in [p_k]$, there is a (*unknown*) group $G_\ell^{(k)} \subseteq [m]$: items whose k-th attribute has value ℓ:

$$G_\ell^{(k)} := \left\{ i \in [m] \colon z_i^{(k)} = \ell \right\}. \tag{Groups – unknown}$$

For example, if the k-th protected attribute is race, then for different values of $\ell \in [p_k]$, $G_\ell^{(k)}$ is the subset candidates whose race is ℓ. However, we only have noisy information about the protected attributes of each item; so, only noisy information of this subset.

Intersectional groups. In the above model, each protected attribute takes a unique value. It may appear that this does not allow for intersectional groups, e.g., say multiracial candidates. But this is only a matter of encoding, and is remedied by considering attributes such as '`has-raceA?`' and '`has-raceB?`', which take `Yes` or `No` values.

Definition 2.1 (**Noise**). *For each item $i \in [m]$ and $k \in [s]$, we have a probability vector $q_i^{(k)} \in \Delta^{p_k}$, such that, the k-th protected attribute of item i takes value $\ell \in [p_k]$ with probability $q_{i\ell}^{(k)}$ conditioned on $(w_i, \widehat{z}_i^{(1)}, \ldots, \widehat{z}_i^{(s)}, a_i)$:*

$$q_{i\ell}^{(k)} := \Pr\left[i \in G_\ell^{(k)} \mid (w_i, \widehat{z}_i^{(1)}, \ldots, \widehat{z}_i^{(s)}, a_i) \right]. \tag{2}$$

The event that $(i \in G_\ell^{(k)})$ is independent of all other items $j \in [m] \backslash \{i\}$ and all other attributes in $[s] \backslash \{k\}$. Note that for all $i \in [m]$ and $k \in [s]$, $\sum_{\ell \in [p_k]} q_{i\ell}^{(k)} = 1$.

Discussion of the noise model. The above model says that given the utility (w_i), noisy protected attributes $(\widehat{z}_i^{(1)}, \ldots, \widehat{z}_i^{(s)})$, and nonprotected attributes (a_i) of an item i, there is probabilistic information about its protected attributes. If items represent candidates for a job and the protected attribute is race, then we can use the candidate's last name (encoded in a_i) to derive probabilistic information about their race. This has been used in practice, e.g., by [27]. We could also consider multiple nonprotected attributes such as both last name and geolocation, e.g., as used by [30, 28]. As discussed in the introduction, this could be relevant for online hiring platforms, which may not have demographic information of some or all of its users [54], and image search engines where the images do not have gender labels.

2.2 Target problem

Studies have found that, in the absence of other constraints, the selection problem (1), can overrepresent individuals with certain protected attributes at the expense of others [45, 24]. Towards mitigating this bias, we consider lower bounds and upper bounds on the number of items of a given protected attribute selected.

Formally, the constraints ensure that for each attribute-value pair $k \in [s]$ and $\ell \in [p_k]$, the selection has at least $L_\ell^{(k)} \geq 0$ and at most $U_\ell^{(k)} \geq 0$ items from $G_\ell^{(k)}$. Then, a selection $x \in \{0,1\}^m$ satisfies the (target) fairness constraints if: for all $k \in [s]$ and $\ell \in [p_k]$

$$L_\ell^{(k)} \leq \sum\nolimits_{i \in G_\ell^{(k)}} x_i \leq U_\ell^{(k)}. \qquad \text{(Fairness constraints; 3)}$$

Constraints similar to Equation (3) have been studied by several works in algorithmic fairness [20, 21, 17], and are rich enough to encapsulate a variety of fairness and diversity metrics (e.g., see [13]). Thus, for the appropriate L and U, the subset satisfying constraints (3) would be fair for one from a large class of fairness metrics.

Overall, our *constrained subset selection problem* is:

$$\begin{aligned}
\max_{x \in \{0,1\}^m} \quad & \sum\nolimits_{i=1}^m w_i x_i && \text{(Target)}\\
\text{s.t.} \quad & L_\ell^{(k)} \leq \sum\nolimits_{i \in G_\ell^{(k)}} x_i \leq U_\ell^{(k)}, \quad \forall\, k \in [s],\ \ell \in [p_k], && (4)\\
& \sum\nolimits_{i=1}^m x_i = n. && (5)
\end{aligned}$$

If we know the protected attributes, and in turn $G_\ell^{(k)}$ (for each $k \in [s]$ and $\ell \in [p_k]$), then we can hope to solve Program Target directly. Indeed, prior works consider similar problems in rankings [17], or its generalization, to multiple Matroids constraints [21]. However, with only noisy information about protected attributes, we can not even verify if a selection vector x is feasible for Program Target. To overcome this, we must go beyond exact algorithms which always satisfy fairness constraints.

2.3 Denoised problem

The difficulty in solving Program Target is that we do not know the constraints (as we do not know $G_\ell^{(k)}$). We propose to solve a different problem, Program Denoised, which uses the noise estimates q to approximate the constraints of Program Target. For some small $\delta \in (0,1)$, we define the denoised program as the following

$$\begin{aligned}
\max_{x \in \{0,1\}^m} \quad & \sum\nolimits_{i=1}^m w_i x_i && \text{(Denoised)}\\
\text{s.t.} \quad & L_\ell^{(k)} - \delta n \leq \sum\nolimits_{i=1}^m q_{i\ell}^{(k)} x_i \leq U_\ell^{(k)} + \delta n, \quad \forall\, k \in [s],\ \ell \in [p_k], && (6)\\
& \sum\nolimits_{i=1}^m x_i = n. && (7)
\end{aligned}$$

Here, $\sum_{i=1}^m q_{i\ell}^{(k)} x_i$ is the expected number of items selected by x whose k-th protected attribute is ℓ. Then, intuitively, we can see Program Denoised that satisfies the constraints of Program Target in expectation, where the expectation is taken over the noise.

However, just satisfying constraints in expectation is not sufficient. For instance, this would allow algorithms that, in each use, violate the fairness constraints by a large amount, but "average out" their errors in aggregate. Instead, our goal is to find an algorithm which violates the constraints by at most a small amount, almost always. Before presenting our theoretical results, we discuss an alternate noise model and why it is not suitable in our setting.

2.4 Group-level noise model

Recent works on noisy fair classification [14, 4, 52] consider a different noise model, which adapted to out setting, uses the following probabilities

$$\overline{q}_{i\ell} \coloneqq \Pr[i \in G_\ell \mid (\widehat{z}_i^{(1)}, \ldots, \widehat{z}_i^{(s)})].$$

Notice that unlike Definition 2.1, $\overline{q}_{i\ell}$ does not condition on the utility w_i or nonprotected attributes a_i. Thus, its estimates are the same for all items with the same set of noisy protected attributes. We call this the *group-level noise model* (GLN). In the next example, we discuss why GLN is not sufficient to mitigate bias in subset selection.

Toy example. Consider a setting where there is one protected attribute which takes two values (i.e., $s = 1$ and $p_1 = 2$), and the relevant fairness metric is equal representation. Let the two groups (unknown) be $A, B \subseteq [m]$, and their observed noisy versions be $\widehat{A}$ and $\widehat{B}$.[4]

According to $\overline{q}$, each candidate $i \in \widehat{B}$ has the same probability of being in A. In this noise model, these candidates are indistinguishable apart from their utilities, so, if one picks $n_b \in \mathbb{N}$ candidates from $\widehat{B}$, they would naturally be the ones with the highest utility. However, suppose that most individuals in A have a higher utility than most individuals in B.[5] In this case, the probabilities $\overline{q}$ will be "distorted" by the utilities, such that, candidates with higher utility in $\widehat{B}$ are more likely to be in A than those with lower utility in $\widehat{B}$. In fact, if m is much larger than n, then most of the top n_b candidates in $\widehat{B}$ would, in fact, be from A. This example can be extended to more realistic settings, with more than two groups and a smaller amount of "bias" in the utilities. Even then, to overcome this distortion in probabilities, one needs to consider a stronger noise model, in which the noise estimate varies with utility (as in Definition 2.1); either implicitly through proxy nonprotected attributes (a_i) or explicitly with utility (w_i).

Remark 2.2. *In Section 3 and 4, for the sake of simplicity, we only consider the setting with one protected attribute (i.e., $s = 1$) which takes $p \geq 1$ values. We defer the general case to Section 6—where we obtain analogous results. When $s = 1$, we let $p \coloneqq p_1$ and drop superscripts (representing the protected attribute) from all variables.*

3 Theoretical results

Our main algorithmic result is an efficient approximation algorithm for Program Target.

Theorem 3.1 (An approximation algorithm for Program Target). *There is an algorithm (Algorithm 1) that given an instance of Program Target for $s = 1$ and noise q from Definition 2.1, outputs a selection $x \in \{0,1\}^m$, such that, with probability at least $1 - 4p\exp\left(-\delta^2 n/3\right)$ over the noise in the protected attributes of each item, the selection x*

1. *has a value at least as $\underline{\text{high}}$ as the optimal value of Program Target,*
2. *violates the cardinality constraint* (5) *by at most p (additive), and*
3. *violates the fairness constraints* (4) *by at most $(p + 2\delta n)$ (additive).*

The algorithm runs in polynomial time in the bit complexity of the inputs.

As desired, the algorithm outputs subset which violates the constraints of Program Target by at most a small amount, with high probability.

Note that the approximation is only in the constraints and not in the value: with high probability, x has an higher value than the optimal solution, say $x^\star$, of Program Target, i.e.,

$$\sum\nolimits_{i=1}^{m} x_i w_i \geq \sum\nolimits_{i=1}^{m} x_i^\star w_i.$$

[4]Formally, $A = \{i \colon z_i^{(1)} = 1\}$, $B = \{i \colon z_i^{(1)} = 2\}$, $\widehat{A} = \{i \colon \widehat{z}_i^{(1)} = 1\}$ and $\widehat{B} = \{i \colon \widehat{z}_i^{(1)} = 2\}$.

[5]Such an bias in utilities is one reason why we need fairness constraints in the first place [48].

In most real-world contexts p is a small constant. Here, Theorem 3.1 implies that x violates the fairness constraints (Equation (4)) by a multiplicative factor of at most $(1+2\delta+o(1))$ and the constraint Equation (5) by a multiplicative factor of at most $(1+o(1))$ with high probability.[6] If p is large, then x (from Theorem 3.1) can violate the constraints by a large amount. However, in this case it is **NP**-hard to even check if there is a solution to Program Denoised which violates the constraints by a constant additive factor (let alone finds an optimal solution for Program Target); see Theorem 3.6.

Algorithm 1 crucially uses the Program Denoised problem: it first solves the linear-programming relaxation of Program Denoised, and then, "rounds" this solution to integral coordinates. In the next section, overview the proof of Theorem 3.1. We defer the proof of Theorem 3.1 to Section 5.1 due to space constraints.

Remark 3.2. *We can prove an analogous version of Theorem 3.1, when $s \geq 1$. It appears as Theorem 6.1 in Section 6.*

Remark 3.3. *We can strengthen Theorem 3.1 to guarantee that Algorithm 1 finds an $x \in \{0,1\}^m$ which does not violate the lower bound fairness constraint (left inequality in Equation (4)) and violates the upper bound fairness constraints by at most $(p+2\delta n)$ (without changing other conditions). In particular, this shows that, if one places only lower bound fairness constraints, then subset found by Algorithm 1 would never violate the fairness constraints.*

Algorithm 1: Algorithm for Program Target

```
Input: A number $n \in \mathbb{N}$, probability matrix $q \in [0,1]^{m \times p}$, utility vector $w \in \mathbb{R}^m$, constraint vectors
       $L, U \in \mathbb{R}^p_{\geq 0}$.
1. Solve $x \leftarrow$ Find a basic feasible solution to linear-programming relaxation
                of Program Denoised with inputs $(n, q, w, L, U)$.
2. Set  $x'_i \coloneqq \lceil x_i \rfloor$  for all $i \in [m]$.   //Round solution
3. Return $x'$.
```

3.1 Proof overview and hardness results

In this section, we present an overview of proof of Theorem 3.1. The complete proof appears in Section 5, and an extension of Theorem 3.1 for multiple protected attributes (i.e., $s \geq 1$) appears in Section 6.

The proof of Theorem 3.1 has two broad steps: First, we show that solving Program Denoised (even approximately) gives us a "good" solution to Program Target, and next, we develop an approximation algorithm along with matching hardness results for Program Denoised. To prove the former, we bound the difference between the true and the expected number of candidates from any one group G_ℓ.

Lemma 3.4. *For all $\delta \in (0,1)$ and $x \in [0,1]^m$, s.t., $\sum_{i=1}^m x_i = n$:*

$$\forall\ \ell \in [p], \qquad \left| \sum\nolimits_{i \in G_\ell} x_i - \sum\nolimits_{i \in [m]} q_{i\ell} x_i \right| \leq n\delta$$

holds with probability at least $1 - 2p \exp\left(-\delta^2 n/3\right)$ over the noise in the protected attributes of each item.

The proof of this lemma appears in Section 5.1.1.

Using Lemma 3.4, we can show that any solution that violates the constraints of Program Denoised by a small amount, with high probability, violates the constraints of Program Target by at most a small amount. Let $x^\star$ be an optimal for Program Target. Using Lemma 3.4, we can show that $x^\star$ is feasible for Program Denoised with high probability. It follows any solution x which is optimal for Program Denoised has value at least as large as $x^\star$, i.e.,

$$\sum\nolimits_{i=1}^m x_i w_i \geq \sum\nolimits_{i=1}^m x_i^\star w_i.$$

These suffice to show that, solving Program Denoised gives a "good" solution for Program Target—which satisfies the claims in the Theorem 3.1.

[6]Using $L_\ell, U_\ell \leq n$; if not, we can set L_ℓ to $\min(L_\ell, n)$ and U_ℓ to $\min(U_\ell, n)$.

It remains to solve Program Denoised. However, unfortunately, even checking if Program Denoised is *feasible* is **NP**-hard; see Theorem 3.6 (a constant-factor approximation (in utility) to the problem is also **NP**-hard). We overcome this hardness by allowing solutions to violate the constraints of Program Denoised by a small additive amount (p). Towards this, consider the linear-programming relaxation of Program Denoised (for $s = 1$). We show that any *basic feasible solution* (BFS) of LP-Denoised has a small number of fractional entries (Lemma 3.5).

$$\max_{x\in[0,1]^m} \quad \sum_{i=1}^{m} w_i x_i \qquad \text{(LP-Denoised for } s=1\text{)}$$

$$\text{s.t.} \qquad \forall\, \ell \in [p], \qquad L_\ell - \delta n \le \sum_{i=1}^{m} q_{i\ell} x_i \le U_\ell + \delta n, \tag{8}$$

$$\sum_{i=1}^{m} x_i = n. \tag{9}$$

Lemma 3.5 (An optimal solution with p fractional entries). *Any basic feasible solution $x \in [0,1]^m$ of LP-Denoised has at most* $\min(m,p)$ *fractional values, i.e.,* $\sum_{i=1}^{m} \mathbb{I}[x_i \in (0,1)] \le \min(m,p)$.

The proof follows by specializing well-known properties of BFSs to LP-Denoised. We remark that this result is tight; see Fact 5.1.

Proof sketch of Theorem 3.1. Using Lemma 3.4, we can show that $x^\star$ is feasible for Program Denoised with probability at least $1 - 2p\exp\left(-\delta^2 n/3\right)$. Assume that this event happens. Then, $x^\star$ is also feasible for Program Denoised. Consider the basic feasible solution x to LP-Denoised from Step 1 of Algorithm 1. Since x is optimal for LP-Denoised, it follows that x has a value at least as large as $x^\star$, i.e.,

$$\sum_{i=1}^{m} x_i w_i \ge \sum_{i=1}^{m} x_i^\star w_i.$$

Further, since $w \ge 0$, the rounded solution x' from Step 2 of Algorithm 1 only increases the utility of x. Thus,

$$\sum_{i=1}^{m} x_i' w_i \ge \sum_{i=1}^{m} x_i^\star w_i.$$

This establishes the first claim in Theorem 3.1.

It follows from Lemma 3.5 that x' picks at most p more elements than x. Thus, x' violates Equation (7), and so, Equation (5) by at most p. By the same argument, x' violates the fairness constraints of Program Denoised by at most p (additive). Combining this with Lemma 3.4, we can show that, with probability at least $1-2p\exp\left(-\delta^2 n/3\right)$, x' violates the fairness constraints of Program Target by at most $2\delta n+p$ (additive). This establishes the last two claims in Theorem 3.1 (conditioning on the two events described above).

The run time follows since there are polynomial time algorithms to find a basic feasible solution of a linear program. Finally, taking a union bound over over the two events completes the proof.

3.1.1 Hardness results

Lastly, we present our hardness results; their proofs appear in Section 5.2.

Theorem 3.6 (Hardness results—Informal). *Consider variants of Program Denoised for values of p.*

1. *If $p \ge 2$, then, deciding if the problem is feasible is* **NP***-hard.*
2. *If $p \ge 3$, then, the problem is* **APX***-hard.*
3. *If $p = \mathrm{poly}(m)$ and $s > 1$, then for every constant $c > 0$, the following* violation gap *variant of Program Denoised is* **NP***-hard.*
   - *Output YES if the input instance is satisfiable.*
   - *Output NO if there is no solution which violates every upper bound constraint at most an additive factor of c.*

4 Empirical results[*]

We evaluate our approach on utilities and noise derived from both synthetic and real-world data. We consider the following algorithms:

Baseline.

- Blind: As a baseline, we consider the Blind algorithm which selects n candidates with the highest utility. Note that Blind has the optimal unconstrained utility.

Noise aware.

- FairExpec is our proposed approach (see Theorem 3.1).
- FairExpecGrp is the same as FairExpec but uses the probabilities $\overline{q}_{i\ell} \coloneqq \Pr[i \in G_\ell \mid \widehat{z}_i]$ from the group-level noise model (Section 2.4).

Noise oblivious.

Impute protected attributes Bayes-optimally from $q \in [0,1]^{m \times p}$ as:

$$\forall\ i \in [m],\ \ell \in [p], \quad q'_{i\ell} \coloneqq \begin{cases} 1 & \text{if } \ell \in \operatorname{argmax}_{j \in [p]} q_{ij}, \\ 0 & \text{otherwise.} \end{cases} \tag{10}$$

If $\operatorname{argmax}_j q_{ij}$ is not unique, pick one at random. Then, we consider the following noise oblivious algorithms which take the imputed protected attributes q' as input:

- Thrsh solves Program Target defined on q'. This is equivalent to the ranking algorithms of [17, 63] adapted to subset selection.
- MultObj is a multi-objective optimization algorithm inspired by [69]'s approach for ranking. Let $t \in \Delta^p$ be the target distribution of protected attributes in the selection. For example, if the target is equal representation, then $t \coloneqq ({}^1/_p, \ldots, {}^1/_p) \in \mathbb{R}^p$. Given a constant $\lambda > 0$, MultObj solves[7]

$$\max_{x \in [0,1]^n \colon \sum_i x_i = n} \quad w^\top x - \lambda \cdot D_{\mathrm{KL}}\left(\frac{(q')^\top x}{n}, t\right) \cdot \frac{w^\top 1_m}{m},$$

  where $1_m \in \mathbb{R}^m$ is the all one vector. The first term $w^\top x$ is the value of x, $({}^x/_n)$ is the distribution of noisy protected attributes in x, and entire second term is a penalty on x for being far from the target distribution t.[8]

4.1 Setup and metrics

4.1.1 Setup

We consider one protected attribute ($s = 1$) which takes p disjoint values (we use $p = 2$ and $p = 4$). Our simulations either target equal-representation, where $t = ({}^1/_p, \ldots, {}^1/_p) \in \mathbb{R}^p$, or proportional representation, where $t = ({}^{|G_1|}/_m, \ldots, {}^{|G_p|}/_m)$.

In each simulation, we do the following

- FairExpec, FairExpecGrp, and Thrsh: Set $L_\ell = 0$ and $U_\ell = n(1-\alpha) + n\alpha t_\ell$, and vary α from 0 to 1. Notice that $\alpha = 0$ enforces no constraints on the subset, the constraints become tighter as α increases, and $\alpha = 1$ ensures the subset chooses exactly $n = t_\ell$ candidates from the ℓ-th group.

[*]The code for the simulations is available at `https://github.com/AnayMehrotra/Noisy-Fair-Subset-Selection`.

[7]Given two vectors $x, y \in \Delta^p$, $D_{\mathrm{KL}}(x, y)$ denotes the Kullback–Leibler divergence of x and y defined as $D_{\mathrm{KL}}(x,y) \coloneqq \sum_{i=1}^p x_i \log({}^{x_i}/_{y_i})$

[8]We scale the second term by the average utility $\sum_{i=1}^m {}^{w_i}/_m$. This is not necessary, but ensures that λ does not (heavily) depend on the scale of the utility.

- MultObj: Vary λ from 0 to a large value. Here, $\lambda = 0$ enforces no penalty on the objective, the penalty increases as λ increases, and $\lambda = \infty$ forces MultObj to satisfy the target distribution exactly (on the noisy attributes).

Let (α_r, λ_r) be the r-th choice of α and λ. For each (α_r, λ_r), we draw a set M of m individuals or items from the dataset. For each element $i \in M$, we have $q_i \in \Delta^p$ and $w_i \in \mathbb{R}$. We give the details of drawing M and fixing q_i, w_i with each simulation.

4.1.2 Fairness metric

Given subset $S \in [m]$ and target $t \in [0,1]^p$, let the *risk difference* $\mathcal{F}(S,t) \in [0,1]$ of S for target t be

$$\mathcal{F}(S,t) \coloneqq 1 - \min_{\ell \in [p]} t_\ell \ \cdot \ \max_{\ell,k \in [p]} \left(\frac{|S \cap G_\ell|}{n \cdot t_\ell} - \frac{|S \cap G_k|}{n \cdot t_k} \right). \tag{11}$$

Here, a risk difference 1 is the most fair and 0 is the least fair. When the target is proportional representation, $\mathcal{F}(S,t)$ reduces to the usual definition of risk difference (up to scaling).[9] Let $\mathcal{A}(w,q) \subseteq [m]$ be the subset selected by algorithm $\mathcal{A}$ on input (w,q). We report

$$\mathcal{F}_{\mathcal{A}} \coloneqq \mathbb{E}\left[\mathcal{F}(\mathcal{A}(w,q),t)\right],$$

where the expectation is over the choices of (w,q). We drop the subscript $\mathcal{A}$ when the algorithm is not important or clear from context.

4.1.3 Utility metric

Let $U_{\mathcal{A}}$ to be the average utility obtained by $\mathcal{A}$:

$$U_{\mathcal{A}} \coloneqq \mathbb{E}\left[\sum\nolimits_{i \in \mathcal{A}(w,q)} w_i\right],$$

where the expectation is over the choices of (w,q). We report the utility ratio $\mathcal{K}_{\mathcal{A}} \in [0,1]$ for different algorithms $\mathcal{A}$, defined as

$$\mathcal{K}_{\mathcal{A}} \coloneqq \frac{U_{\mathcal{A}}}{U_{\mathsf{Blind}}}. \tag{Utility ratio}$$

We drop the subscript $\mathcal{A}$ when the algorithm is not important or clear from context.

4.2 Synthetic data with disparate error-rates

In this section, we consider the setting where noise levels are different for different groups. This has been observed, for instance, in commercial image-based gender classifiers [7].

4.2.1 Data

We generate a synthetic dataset with one binary protected attribute ($p = 2$). This attribute partitions the (underlying) population into a minority group and a majority group. We assume that candidates in both groups have similar potentials, and so, sample utilities of all candidates (independently) from $\mathcal{U}(0,1)$. Next, we sample the probabilities q_i from a Gaussian mixture, such that, the resulting population has 40% minority candidates (in expectation), and the imputed attributes q' have a higher *false discovery rate* (FDR) for minority candidates (≈40%) compared to majority candidates (≈10%).[10] Formally, we sample q_i as follows:

$$q_{i0} \sim \frac{7}{11} \cdot \mathcal{N}(0.6, 0.05) + \frac{4}{11} \cdot \mathcal{N}(0.05, 0.05) \quad \text{and} \quad q_{i1} \coloneqq 1 - q_{i0},$$

where $\mathcal{N}(\mu, \sigma)$ is the truncated normal distribution on $[0,1]$ with mean μ and standard deviation σ.

[9] Some works also define risk difference as a measure of unfairness [11, 61], and set it equal to $1 - \mathcal{F}(S,t)$ with $t = (1/p, \dots, 1/p)$ (up to scaling).

[10] The difference of 30% in FDRs is comparable to 34% difference in FDRs between dark-skinned females and light-skinned men observed by [7] for a commercial classifier.

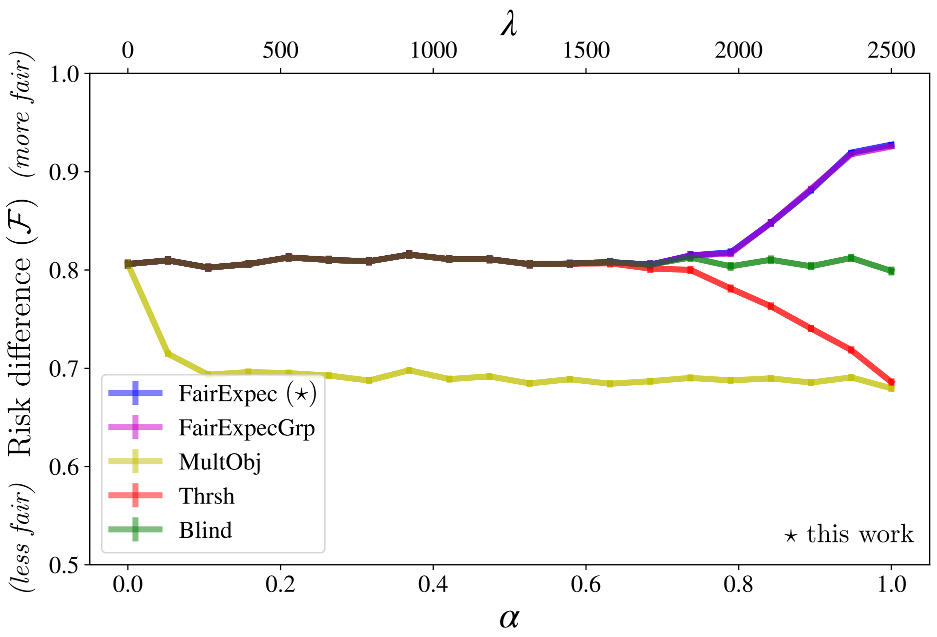


Figure 1: *Synthetic data with disparate error-rate (Section 4.2):* This simulation considers the setting where the minority group (40% of total) has a higher FDR (30%) compared to the majority group, whose FDR is 10%, and the utilities of all candidates are iid from the uniform distribution. The target is to ensure equal representation. The y-axis shows the risk difference $\mathcal{F}$ of different algorithms, and the x-axis shows the constraint parameters (α for FairExpec, FairExpecGrp, and Thrsh, and λ for MultObj); $\mathcal{F}$ values are averaged over 500 trials, and the error bars represent the standard error of the mean. We observe that increasing fairness constraints to noise oblivious algorithms (Thrsh and MultObj) worsens their risk difference! Whereas, the risk difference of noise aware algorithms improves (FairExpec and FairExpecGrp) on increasing fairness constraints.

4.2.2 Setup

In this simulation, we target equal representation between the majority group and the minority group, and fix $m = 500$ and $n = 100$.

We report the risk difference $\mathcal{F}$ of different algorithms as a function of α (for FairExpec, FairExpecGrp, and Thrsh) and as a function of λ (for MultObj) in Figure 1.

Remark 4.1. MultObj *does not guarantee a particular fairness-level for any fixed λ. Thus, one should consider the limiting value of $\mathcal{F}_{\mathsf{MultObj}}$ in Figure 1.*

4.2.3 Results

We observe that without any constraints (i.e., $\alpha = 0$ and $\lambda = 0$) all algorithms have similar risk difference ≈ 0.81. However, on adding fairness constraints Thrsh and MultObj become more *unfair*. In fact, for the strongest fairness constraint (i.e, $\alpha = 1$ and $\lambda = 2500$) they have the *lowest* risk difference (< 0.7). This is because, the imputed protected attributes have a higher FDR for the minority group (and so, the algorithms pick a higher number of candidates from the majority group).

In contrast, FairExpec and FairExpecGrp do not use the imputed protect attributes, and so, increasing fairness constraints to FairExpec and FairExpecGrp improves their risk difference, and for the strongest fairness constraint ($\alpha = 1$) they attain the highest risk difference (> 0.92). Finally, since we sample all utilities from the same distribution, it is not surprising that FairExpec and FairExpecGrp perform similarly.

4.3 Synthetic data with disparate utilities

In this section, we consider the setting where the utilities of different groups have different distributions. In particular, we assume that the minority group (unfairly) has a lower average utility, when in fact, the distributions of utilities should be the same for both the majority group and the minority group. (Contrast

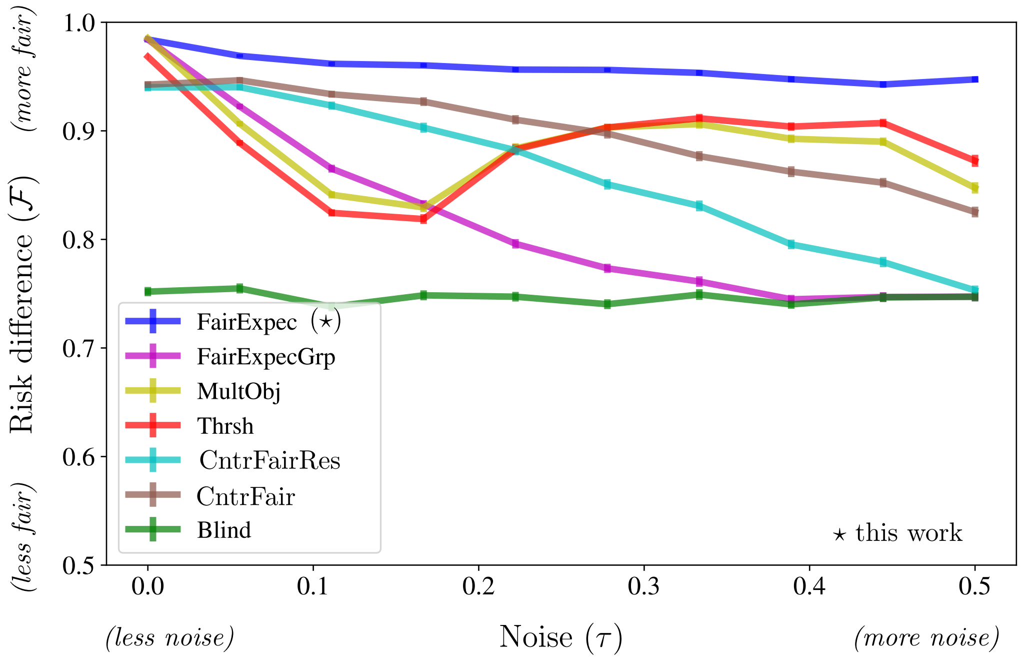


Figure 2: *Synthetic data with disparate utilities (Section 4.3):* This simulation considers the setting where the utilities of a minority group have a lower average than the majority group, and both groups have identical noise. The y-axis shows the risk difference $\mathcal{F}$ of different algorithms, and the x-axis shows the amount of noise added $\tau \in [0, 1/2]$; $\mathcal{F}$ values are averaged over 200 trials, and the error bars represent the standard error of the mean. We observe that the risk difference of all algorithms becomes poorer with noise ($\tau > 0$) than without it ($\tau = 0$). But, FairExpec has the highest risk difference for all values of noise. Finally, unlike Section 4.2, FairExpecGrp has a lower fairness than FairExpec (since, in this simulation, different groups have different distributions of utilities).

this with Section 4.2 where all utilities are identically drawn). Such differences in utility can manifest in the real world for many reasons, including, due to the implicit biases of the committee evaluating the candidates [66, 6, 48] and structural oppression faced by different groups [29].

Counterfactually fair approaches. One could also consider counterfactually fair approaches to mitigate bias in selection. (We refer the reader to [51] for an overview of counterfactual fairness). At a high-level, these approaches try to "unbias" the utilities across groups, and then use unbiased utilities in subsequent tasks (say, selection or ranking). In this simulation, we also consider two counterfactually fair algorithms by [68]: CntrFair and CntrFairResolving. (These correspond to non-resolving and resolving algorithms in [68].)

Roughly, they assume that there is a causal model $\mathcal{M}[\theta]$ (parameterized by θ), such that, given the attributes (z_i, a_i) of an individual, their utility is $w_i = \mathcal{M}[\theta](z_i, a_i)$. Then, roughly, they fix each individual's protected attributes to v and compute the "unbiased" utility as $\hat{w}_i := \mathcal{M}[\theta](v, a_i)$; this represents the utility of the individuals had they had the same protected attribute v.

4.3.1 Data

We consider a synthetic hiring dataset, generated with the code provided by [68]. In the data, each candidate i has one protected attribute $z_i \in \{0, 1\}$ denoting their race (0 if the candidate is Black and 1 otherwise) and two nonprotected attributes $a_{i1} \in \{0, 1\}$ and $a_{i2} \in \mathbb{R}$: a_{i1} is 1 if the candidate has prior work experience, and a_{i2} is denotes their qualifications (the larger the better).[11] We sample 2000 candidates independently from a fixed distribution defined in [68], which, is such that, the utility of Black candidates is (unfairly) lower than non-Black candidates.[12] The dataset has 37% Black candidates, 37% candidates without prior experience,

[11] This interpretation differs from [68] who interpret both z_i and a_{i1} as protected attributes.

[12] The only difference from [68] is that we increase the underlying bias (by reducing the mean utilities for Black candidates and candidates without prior experience). We do so because, the dataset already had a high risk difference (> 0.9) without

and 13.7% Black candidates without prior experience. The utilities are such that the Black candidates ($z_i = 0$) have a lower expected utility than non-Black candidates, candidates without prior experience ($a_{i1} = 0$) have a lower expected utility than those with prior work experience, and Black candidates without prior experience ($z_i = 0, a_{i1} = 0$) have the lowest expected utility.

Preprocessing. We sample a training dataset D' with $m = 2000$ candidates. Then, we use the code by [68] to get an approximation $\bar{\theta}$ of θ from D'. (CntrFair and CntrFairResolving use $\mathcal{M}(\bar{\theta})$ to generate counterfactual utilities). To generate the estimate of q, we partition candidates into 20 equally sized bins, $(b_1, \ldots, b_{20})$, by their utility. (Each bin has 100 candidates). Given a candidate i described by $(w_i, z_i, a_{i1}, a_{i2})$ such that $w_i \in b_k$, we define q_i as follows:

$$q_{i0} = \Pr_j[z_j = 0 \mid w_j \in b_k] \quad \text{and} \quad q_{i1} = 1 - q_{i0}, \tag{12}$$

where the probability is over the candidates j in D'. Note that the candidate i may not be in D'.

Adding noise. The dataset does not have noise to begin with. Given a noise level $\tau \in [0, 1/2]$, we generate noisy race $\widehat{z}_i \in \{0, 1\}$ of candidate i by flipping their race z_i (independently) with probability τ.

4.3.2 Setup

We target equal representation of race and vary τ over $[0, 0.5]$. For each noise level τ, we sample a new instance D of the dataset and add τ-noise to it. We fix the strongest constraints $\alpha = 1$ and $\lambda = 500$ for the algorithms.[13] Here, CntrFair and CntrFairResolving use $\mathcal{M}(\bar{\theta})$ (calculated in preprocessing), and FairExpec, FairExpecGrp, MultObj, and Thrsh using the q (or the imputed attributes q') (also calculated in preprocessing).

We report the risk difference $\mathcal{F}$ as a function of τ in Figure 2. We also report the selection rates from each group in Section A.5.

4.3.3 Results

We observe that all algorithms have the highest fairness when there is no noise ($\tau = 0$). Here, they have a similar risk difference (lying between 0.94 to 0.98). As the noise increases, we observe that the risk difference of FairExpecGrp and CntrFair approaches $\mathcal{F}_{\mathsf{Blind}} = 0.74$, and risk difference of MultObj, Thrsh, and CntrFairResolving approaches a value between $[0.82, 0.87]$. In contrast, FairExpec has a better risk difference, $\mathcal{F} > 0.94$, throughout. The risk difference of MultObj and Thrsh improves with τ at some values of τ — we give a possible explanation in Remark 4.2.

Notice at $\tau = 0.5$, for all candidates $i \in [m]$, the noisy label $\widehat{z}_i \in \{0, 1\}$ is chosen uniformly at random and provides no information about z_i. CntrFair and CntrFairResolving use $\widehat{z}_i$ to compute the counterfactual utilities, and so, performs poorly at $\tau \approx 0.5$. Further, FairExpecGrp uses the probabilities $\overline{q}_i$ which depend on $\widehat{z}_i$, but not on w_i. Since the utility of candidates of different races has a different distribution, $\overline{q}_i$ can be skewed (see Section 2.4).

We note that the utility of all algorithms decreases on adding noise. In particular, while FairExpec is able to satisfy the fairness constraints with noise, its utility decreases on adding noise; see Section A.5 for a plot of utility ratio ($\mathcal{K}$) vs noise (τ).

Remark 4.2. *The risk difference of* Thrsh *and* MultObj *is non-monotonic in the noise. This might be because the false discovery rate (FDR) of q' for Black candidates is non-monotonic. Specifically, the FDR first increases with τ (roughly, for $\tau \leq 0.2$), and then decreases. The decrease in FDR after $\tau = 0.2$ comes at the cost of fewer total positives (i.e., q' identifies fewer total Black candidates). The total number of total positives drop below $n/2$ for higher values of τ. Correspondingly, the $\mathcal{F}$ of* Thrsh *and* MultObj *first decreases as FDR reduces, then increases as FDR increases until the number of total positives is larger than, roughly, $n/2$, and finally, decreases as the number of total positives drops below $n/2$.*

adding any fairness constraints.

[13]We choose $\lambda = 500$ as MultObj's fairness $\mathcal{F}_{\mathsf{MultObj}}$ converges before $\lambda = 500$.

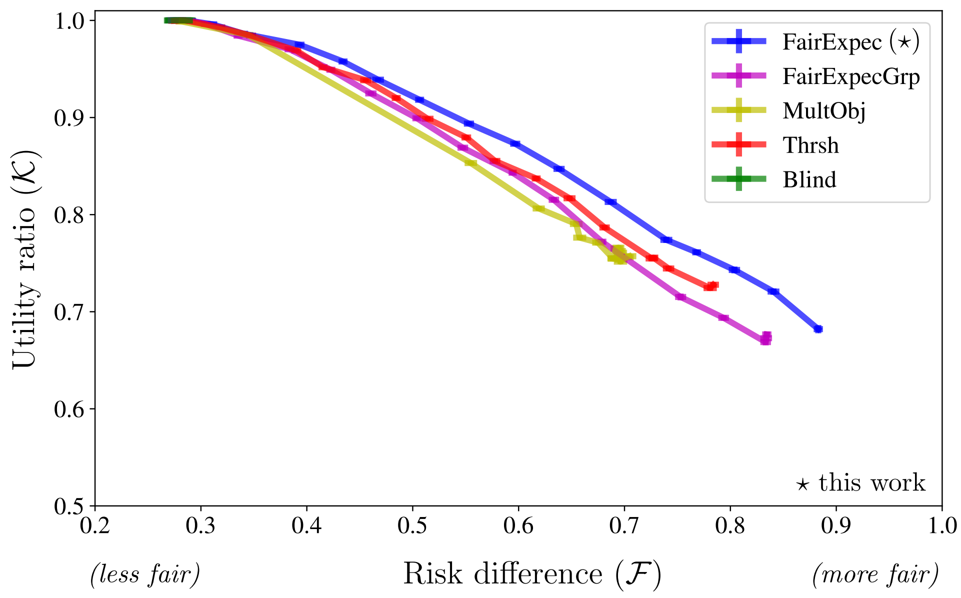


Figure 3: *Real-world data for candidate selection (Section 4.4):* This simulation considers race as the protected attribute which takes $p = 4$ values, where an individual's utility is drawn from different distributions depending on their race. The simulation uses last name as a proxy to derive noisy information of race. The y-axis shows the utility ratio $\mathcal{K}$ of different algorithms, and the x-axis shows the risk difference of different algorithms; both $\mathcal{K}$ and $\mathcal{F}$ values are averaged over 100 trials, and the error bars represent the standard error of the mean. We observe that FairExpec reaches the highest risk difference ($\mathcal{F} = 0.89$), and has a better tradeoff between utility and risk difference compared to other algorithms.

Remark 4.3. *We do not consider counterfactual approaches in Section 4.2 because, their the utilities are already unbiased, and so,* CntrFair *and* CntrFairResolving *reduce to* Blind*. Further,* CntrFair *and* CntrFairResolving *only ensure proportional representation. We find that the datasets considered in Sections 4.4 and 4.5 are already fair when the target is proportional representation; in both cases, $\mathcal{F}_{\mathsf{Blind}} > 0.9$. Therefore, we omit these algorithms from those simulations.*

4.4 Real-world data for candidate selection

In this simulation, we consider candidate selection under noisy information about a candidate's race: we use the candidate's last name to predict their race (similar to what has been used in various applications, e.g., [27]). We model the candidate's utility as their "previous-salary," which we model using the race-aggregated income dataset [10]. This dataset provides the income distribution of families of different races (elaborated below). This setting could be relevant in the context of an online hiring platform, which would like to display a race-balanced set of candidates, but only has noisy information about the race of each candidate [54].

4.4.1 Data

US 2010 Census dataset [65]. The dataset contains 151,671 distinct last names which occurred at least a 100 times in the US 2010 census (23,656 last names occur at least a 1000 times). For each last name i, the dataset has its total occurrences per 100k people $c(i) \in \mathbb{Z}$, and a vector $\hat{f} = (f_1(i), \ldots, f_6(i)) \in [0, 1]^6$ representing the fraction of individuals who are White, Black, Asian and Pacific Islander (API), American Indian and Alaskan Native only (AIAN), multiracial, or Hispanic respectively.

We do not use 'AIAN' and 'two or more races' categories (i.e., f_4 and f_5) as they do not occur in the income dataset. Then, for each last name i, we define the probability vector q_i as the normalized version of the vector $(f_1(i), f_2(i), f_3(i), f_6(i))$.

Race	Mean (USD per annum)
White	100,169
Black	70,504
Asian and Pacific Islander (API)	118,421
Hispanic	71,565

Figure 4: Statistics of $\mathcal{D}_r$ for different races r; $\mathcal{D}_r$ is the discrete distributions of incomes of families with race r derived from the income dataset [10].

Income dataset [10]. We use family income data aggregated by race [10]. This was compiled by the US Census Bureau from the Current Population Survey 2018 [9]. The dataset provides income data of 83,508,000 families. It considers four races (White, Black, Asian, and Hispanic), 12 age categories, and 41 income categories.[14] For each set of race, age, and income categories, the dataset has the number of families whose reference person (see definition here) belongs to these categories.

For each race r, we consider the discrete distribution $\mathcal{D}_r$ of incomes of families with race r derived from the income dataset [10]; see Figure 4 for some statistics of $\mathcal{D}_r$.

4.4.2 Setup

We consider race as a protected attribute with four labels ($p = 4$) and target equal representation based on race. Let $m = 1000$ and $n = 100$. Let (α_s, λ_s) be the s-th choice of α and λ. For each (α_s, λ_s), we draw a set M of m last names uniformly from the entire population with replacement: The i-th last name is drawn with probability proportional to $c(i)$. For each last name $i \in M$, we sample a ground-truth race r_i (unknown to the algorithms) according to the distribution q_i, and then, sample the income $w_i \sim \mathcal{D}_{r_i}$.

We report the utility ratio $\mathcal{K}$ as a function of the risk difference $\mathcal{F}$ for different algorithms in Figure 3. We report the risk difference $\mathcal{F}$ and $\mathcal{K}$ as a function of α (for FairExpec, FairExpecGrp, and Thrsh) and as a function of λ (for MultObj) in Section A.6.

4.4.3 Results

FairExpec reaches the highest risk difference of 0.89, followed by FairExpecGrp, which reaches a risk difference of 0.84. Thrsh reaches $\mathcal{F} = 0.79$, MultObj reaches $\mathcal{F} = 0.70$, and Blind has $\mathcal{F} = 0.28$.

We do not expect the algorithms to outperform the unconstrained utility (i.e., that of Blind). We observe that FairExpec has a better Pareto-tradeoff compared to other algorithms, i.e., for any desired level of risk difference, it has a better utility ratio ($\mathcal{K}$) than other algorithms. In contrast, while FairExpecGrp also has a high maximum risk difference, it is not Pareto-optimal.

All algorithms lose a large fraction of the utility (up to 33% This is because the unconstrained and constrained optimal are very different: without any constraints, we would roughly select roughly 7% candidates from some races, however, ensuring the equal representation requires selecting roughly 4 times as many candidates from these races. When the difference between unconstrained and constrained optimal is less extreme, we expect lose a smaller fraction of the utility.

4.5 Real-world data for image search

In this section, we consider image selection under noisy gender information: we derive noisy information about the gender of the person in an image using a CNN-based gender classifier, and use this information to select a gender-balanced set of images. This could be relevant in mitigating gender bias in image search engines, which have been observed to over represent the stereotypical gender in job-related searches [45, 15]. Here, one could first select a balanced subset of images to display on each page, and then order this subset in decreasing order of utility from top to bottom of each page.

[14]The age categories are: 15 to 24, 25 to 30, 30 to 35, ... The income categories are: $[0, 5000)$, $[5000, 10^4)$, ..., $[1.95, 2\cdot 10^5)$, and $(2 \cdot 10^5, \infty)$ in USD per annum.

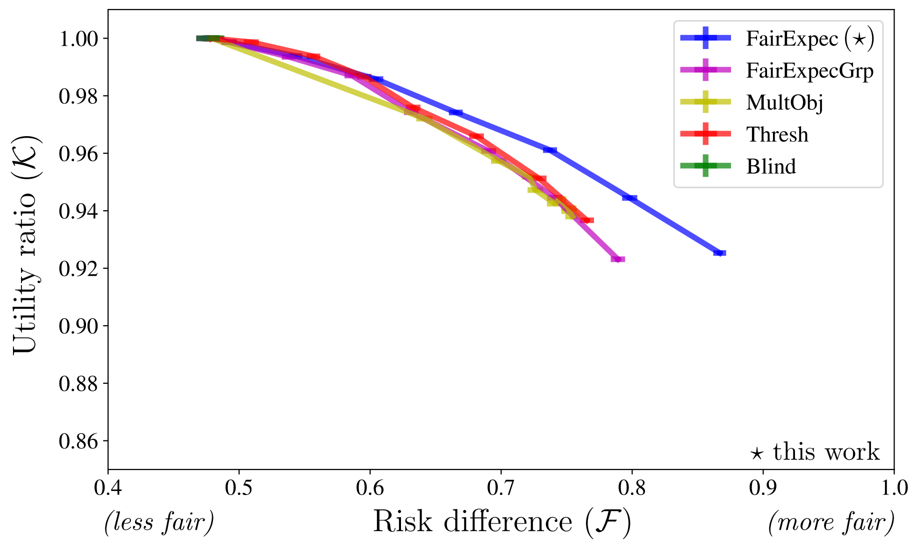


Figure 5: ***Real-world data for image search (Section 4.5)***: This simulation considers gender as the protected attribute and uses a CNN-based classifier to derive noisy information about the gender of the person in the image. The utility of a image at the r-th position in the dataset is $(\log(1+r))^{-1}$. The y-axis shows the utility ratio $\mathcal{K}$ of different algorithms, and the x-axis shows the risk difference of different algorithms; $\mathcal{K}$ and $\mathcal{F}$ values are averaged over 200 trials, and the error bars represent the standard error of the mean. We observe that FairExpec reaches the highest risk difference ($\mathcal{F} = 0.86$), and has a better tradeoff between utility and fairness compared to other algorithms.

	Male	Female	NA	Total
Dark	568	318	106	992
Light	2635	1987	386	5008
NA	198	119	3283	3600
Total	3401	2424	3775	

Figure 6: ***Statistics of the Occupations dataset [15].***

4.5.1 Data

We use a recent image dataset, Occupations Dataset, from [15]. The dataset contains top 100 Google Image Search results (from December 2019) for 96 occupations related queries. For each image, the dataset has the gender (coded as men, women, or other), skin-tone (coded as dark or light), and the image's position in the search result (an integer in [100]). We present aggregate statistics from the dataset in Figure 6.

Gender classifier. We use an off-the-shelf face-detector [1] to extract faces of people from the images, and then use a CNN-based classifier [60] to identify the gender of people from their faces. For each image i, the classifier outputs a prediction $f_i \in [0,1]$ (resp. $1 - f_i$) which is the (uncalibrated) likelihood that the image is of a man (resp. women) which we calibrate (as described next).[15]

To prepossess the images, we remove all images with a gender label NA, this leaves us 5,825 images. Next, we use face-detector to extract faces from the (5,825) images and remove the ones where the detector does not detect any face.[16] This gives us 4,494 images (77% of 5,825).

We calibrate the classifier by binning its values into $b = 20$ bins. For each bin $j \in [20]$, we calculate the classifier's accuracy $a(j) \in [0,1]$. This initial calibration is done only once. Given image i, we compute

[15]While there could be richer and nonbinary gender categories, many commercial classifiers, and the classifier by [60] categorizes images as either male or female.

[16]Note that we do not check if the detected faces are correct. This introduces some error, which is also expected in practice.

the classifier's output $f_i \in [0,1]$. Let f_i fall in bin j. Then, we set the noise-information q_i of image i to $q_i \coloneqq [a(j), 1-a(j)]$ (see Section A.1 for details).

Selecting occupations. Next, we infer the occupations which have considerable gender stereotype. Towards this, we fix a $\zeta \in [0,1]$ and partition the occupations into three sets:

- $\textsc{Sty}_f(\zeta)$: occupations for which at least ζ-fraction of images were labeled as women,
- $\textsc{Sty}_f(\zeta)$: occupations for which at least ζ-fraction of images were labeled as men, and
- all other occupations.

We fix $\zeta = 0.8$. This gives us $|\textsc{Sty}_f(\zeta)| = 12$ and $|\textsc{Sty}_m(\zeta)| = 29$ and 1877 images with occupations in $\textsc{Sty}_f(\zeta) \cup \textsc{Sty}_m(\zeta)$ (for a list of the occupations, see Table 17 in Section A.7).

Remark 4.4. *Note that we calibrate q on all occupations, only consider images in a subset of occupations (less than half). This means that qs may not be an unbiased estimate of the protected attributes—which is a hard case for* FairExpec.

4.5.2 Setup

In this simulation, we consider gender as the protected attribute with two values ($p = 2$), and fix $m = 500$ and $n = 100$.

We say gender is *stereotypical* if it is viewed as stereotypical for its occupation, e.g., men are stereotypical for occupations in $\textsc{Sty}_m(0.8)$ and women are stereotypical for occupations in $\textsc{Sty}_f(0.8)$. We call an image *anti-stereotypical* if it is not stereotypical. We would like to ensure equal representation between stereotypical and anti-stereotypical images.

For each (α_s, λ_s), we draw a subset M of m images uniformly from all images with occupation in $\textsc{Sty}_f(\zeta) \cup \textsc{Sty}_m(\zeta)$. For each image $i \in M$, let its rank be $r_i \in [100]$. We set its utility w_i to $w_i \coloneqq (\log(1+r_i))^{-1}$ and compute q_i as discussed earlier

We report $\mathcal{K}$ as a function of $\mathcal{F}$ for different algorithms in Figure 5. We also report the risk difference $\mathcal{F}$ of different algorithms as a function of α (for FairExpec, FairExpecGrp, and Thrsh) and as a function of λ (for MultObj) in Section A.7.

Remark 4.5. *Since the underlying application in this simulation is ranking image results, we also considered Normalized DCG [42] as the utility metric and observed similar results as utility ratio. For completeness, we present the plot in Section A.7. Furthermore, we tried other functions for utilities w_i, including $100 - r_i$ and ${}^{100}/_{r_i}$ and observed similar results.*

4.5.3 Results

The risk difference of Blind (i.e., without any interventions) is $\mathcal{F} = 0.48$. All algorithms reach a better risk difference than Blind. Among them, FairExpec reaches the highest risk difference of $\mathcal{F} = 0.86$, while the next best algorithm FairExpecGrp has $\mathcal{F} = 0.79$.

We do not expect the algorithms to outperform the unconstrained utility (that of Blind); hence, it is not surprising that $\mathcal{K} \le 1$. Among the algorithms we consider, we observe that FairExpec has the Pareto-optimal tradeoff between utility and fairness: it has a higher utility ratio compared to other algorithms for a given value of risk difference.

4.6 Additional empirical results

We present additional empirical results with selection lift in Section A. We observe that, indeed, FairExpec is able to mitigate discrimination with respect to selection lift and reaches the most-fair selection lift in all experiments. Further, it has the Pareto-optimal tradeoff between utility and fairness compared to other algorithms, in all but one case: in the simulation from Section 4.4, we find that while FairExpec has a lower utility than MultObj for some levels of selection lift. We believe this is because the constraint region induced by a level of selection lift is different from the constraint region of Program Target. One could correct this,

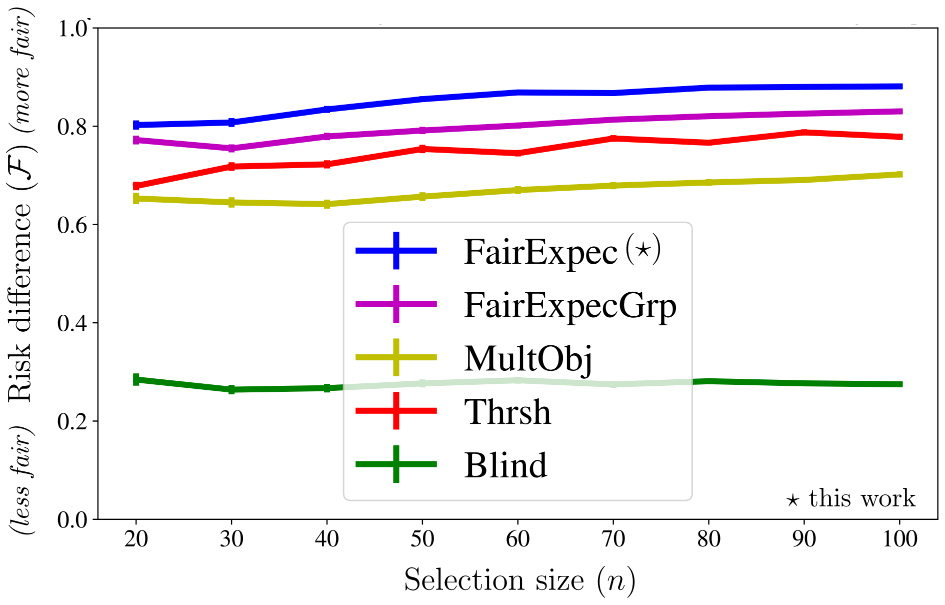


Figure 7: ***Risk difference on varying $^n/_m$ (Remark 4.6):*** Towards analyzing the robustness of our approach to the fraction of items selected $^n/_m$, we fix $m = 1000$ and vary n from 20 to 100 in Simulation 4.4. The y-axis shows the risk difference $\mathcal{F}$ of different algorithms, and the x-axis shows n; $\mathcal{F}$ values are averaged over 100 trials, and the error bars represent the standard error of the mean. We find that increasing on n the difference in the fairness of different algorithms remains roughly the same.

e.g., by using [13, Theorem 3.1] to reduce the problem of attaining a particular selection lift to that of satisfying multiple lower and upper bound constraints.[17]

Remark 4.6 (**Risk difference on varying $^n/_m$**). *Different applications could select different fractions of results from the available items. For example, an image search engine might select a very small fraction of available results, whereas a job platform can select a larger fraction. Towards analyzing the robustness of our approach to the fraction of items selected, we fixed m and varied n in simulations. We find that on increasing n (holding m fixed) the difference in the fairness attained by the algorithms remains roughly the same. See Figure 7 for a plot for the simulation in Section 4.4; we present the plots for other simulations in Section A.3.*

5 Proofs

In this section, we present the proof of Theorem 3.1 (Section 5.1), and present formal statements of our hardness results for Program Denoised and their proofs (Section 5.2).

5.1 Proof of Theorem 3.1

5.1.1 Proof of Lemma 6.4

Proof of Lemma 6.4. For all $i \in [m]$, define $Z_i \in \{0, 1\}$ to be the indicator random variable that $(i \in G_\ell)$. Notice that

$$\sum_{i \in G_\ell} x_i = \sum_{i \in [m]} x_i Z_i. \tag{13}$$

[17]This reduction uses multiple lower bound and upper bound constraints to provably approximate the constraints of selection lift.

From Definition 2.1 we have $\mathbb{E}[Z_i] = q_{i\ell}$. Using linearity of expectation we get $\sum_{i=1}^{m} x_i q_{i\ell} = \sum_{i=1}^{m} \mathbb{E}[Z_i]$. Now, we have

$$\begin{aligned}
\Pr\left[\sum_{i=1}^{m} x_i Z_i > \sum_{i=1}^{m} x_i q_{i\ell} + n\delta\right] &= \Pr\left[\sum_{i=1}^{m} x_i Z_i > \mathbb{E}[\sum_{i=1}^{m} Z_i] + n\delta\right] \\
&\le \exp\left(-\frac{(\delta n)^2}{3}\frac{1}{\mathbb{E}\left[\sum_{i=1}^{m} x_i Z_i\right]}\right) && \text{(Additive Chernoff bound [57])} \\
&\le \exp\left(-\frac{(\delta n)^2}{3}\frac{1}{\mathbb{E}\left[\sum_{i=1}^{m} x_i\right]}\right) && (\forall i \in [m],\ Z_i \le 1) \\
&\le \exp\left(-\frac{\delta^2 n}{3}\right). && (\textstyle\sum_{i=1}^{m} x_i = n)
\end{aligned}$$

By a similar argument, we have

$$\Pr\left[\sum\nolimits_{i=1}^{m} x_i Z_i < \sum_{i=1}^{m} x_i q_{i\ell} - n\delta\right] \le \exp\left(-\frac{\delta^2 n}{2}\right).$$

We get the required result by taking a union bound over all $\ell \in [p]$ and using Equation (13). ☐

5.1.2 Proof of Lemma 3.5

Proof. Assume $p < m$, otherwise the result is trivial. Recall that LP-Denoised has m variables, p pairs of upper and lower bound constraints, one cardinality constraint ($\sum_{i=1}^{m} x_i = n$), and $2m$ bounding-box constraints of the form: $(x_i \ge 0)$ and $(x_i \le 1)$ for each $i \in [m]$. If for some $\ell \in [p]$, $L_\ell = U_\ell$, replace the two inequality constraints by the equivalent equality constraint.

For any linear program, there exists a basic feasible solution [58, Theorem 2.6]. If x is a basic feasible solution, it must satisfy m linearly-independent equalities. Notice from each of the at most p pairs of upper and lower bound constraints[18] at most one of the upper bound or the lower bound can be satisfied with equality x. Thus, including the cardinality constraint ($\sum_{i=1}^{m} x_i = n$) (but, excluding the bounding-box constraints, x satisfies at most $(p+1)$ constraints with equality. Further, note that if any of bounding-box constraints are satisfied with equality, then the corresponding index in x is integral. This suffices to show that x has at most $(p+1)$ fractional entries.

Assume that exactly $(p+1)$ non-bounding-box constraints are satisfied with equality. By the previous discussion, one of them is the cardinality constraint, and the others must correspond to distinct values of $\ell \in [p]$. However, for each $i \in [m]$ it holds that $\sum_{\ell \in [p]} q_{i\ell} = 1$. Further, in this case, the upper bounds or lower bound corresponding to the p tight constraints must sum to n (otherwise, the $(p+1)$ non-bounding-box constraints cannot hold together). But, in this case, all the p upper or lower bound constraints imply the cardinality constraint. Thus, at most p of the $(p+1)$ non-bounding-box constraints are linearly independent. It follows that any basic feasible solution has at most p fractional constraints. ☐

The next fact shows that Lemma 3.5 is tight.

Fact 5.1. *There exists an instance of Program Denoised such that the* unique *optimal solution to LP-Denoised has p fractional entries.*

Proof. Fix $\varepsilon > 0$ and $s = 1$. Let $p := p_1$ and drop the superscripts on the variables. Let $n = p$, $m = p + 1$, and for all $\ell \in [p]$, $L_\ell = 0$, and $U_\ell = 1$. Let e_i be the i-th standard basis vector on $\mathbb{R}^p$. Define

$$w_i = \begin{cases} 1 & \text{if } i < m, \\ 2 & \text{if } i = m \end{cases} \quad \text{and } q_i = \begin{cases} e_i & \text{if } i < m, \\ \frac{1}{p}\mathbb{1}_p & \text{if } i = m. \end{cases}$$

[18]If the lower bound and the upper bound are the same, we discard one of the (repeated) inequalities.

Notice that $q_i \in \Delta^p$ for each $i \in [m]$. It is easy to see that any optimal solution x has $x_m = 1$. Further, x must pick n items ($\sum_{i=1}^m x_i = n$) and $\sum_{\ell \in [p]} U_\ell = n$. So, each upper bound constraint must be tight. This suffice to show the unique optimal is:

$$\widehat{x}_i^\star = \begin{cases} 1 - \frac{1}{p} & \text{if } 1 \leq i < m, \\ 1 & \text{if } i = m. \end{cases}$$

Note that $\widehat{x}^\star$ has p fractional values. □

5.1.3 Proof of Theorem 3.1

Proof of Theorem 3.1. We claim that Algorithm 1 satisfies the claims in the theorem.

Run time. Step 1 finds a basic feasible solution for LP-Denoised. Since this is a linear program, one can do so in time polynomial in the bit-complexity of the input, say, using the ellipsoid method (see, e.g.,[34]). Step 2 is clearly linear time. Thus, the bound on the running time follows.

Correctness. Since x (in Step 1) is feasible for LP-Denoised, it satisfies

$$\text{For all } \ell \in [p], \qquad L_\ell - \delta n \leq \sum\nolimits_{i=1}^m q_{i\ell} x_i \leq U_\ell + \delta n,$$
$$\sum\nolimits_{i=1}^m x_i = n.$$

Further, since x is a basic feasible solution of LP-Denoised, x has at most p fractional values by Lemma 3.5. Thus, x' obtained by rounding up each nonzero coordinate to 1 (in Step 2), satisfies (Recall that $q_{i\ell} \in [0,1]$)

$$\text{For all } \ell \in [p], \qquad L_\ell - \delta n \leq \sum\nolimits_{i=1}^m q_{i\ell} x_i' \leq U_\ell + \delta n + p, \tag{14}$$
$$n \leq \sum\nolimits_{i=1}^m x_i' \leq n + p. \tag{15}$$

Lemma 3.4 says that with probability at least $1 - 2p \exp\left(-\delta^2 n/3\right)$ (over the noise in the protected attributes)

$$\text{For all } \ell \in [p], \quad \left| \sum\nolimits_{i \in G_\ell} x_i' - \sum\nolimits_{i=1}^m q_{i\ell} x_i' \right| \leq n\delta. \tag{16}$$

Now, combining Equation (14) and Equation (16), we get that

$$\text{For all } \ell \in [p], \qquad L_\ell - 2\delta n \leq \sum\nolimits_{i=1}^m q_{i\ell} x_i' \leq U_\ell + 2\delta n + p \tag{17}$$

holds with probability at least $1 - 2p \exp\left(-\delta^2 n/3\right)$ (over the noise in the protected attributes). Equation (17) implies that x' violates Equation (4) by at most $(p + 2\delta n)$ (additive) and Equation (15) implies x' violates Equation (5) by at most p (additive). Let $\mathcal{E}_1$ be the event that Equation (17) and Equation (15) hold. From the above discussion, we conclude that

$$\Pr[\mathcal{E}_1] \geq 1 - 2p \exp\left(-\delta^2 n/3\right).$$

Let $x^\star$ be an optimal solution of Program Target. It remains to show that x' has a value higher than $x^\star$. Let $\mathcal{E}_2$ be the event that $x^\star$ is feasible for Program Target. Then, conditioning on $\mathcal{E}_2$ we that

$$\sum\nolimits_{i=1}^m w_i x_i^\star \leq \sum\nolimits_{i=1}^m w_i x_i \qquad (x \text{ is optimal for Program Target})$$
$$\leq \sum\nolimits_{i=1}^m w_i x_i'. \qquad (x' \geq x \text{ and } w \geq 0)$$

Thus, conditioning on $\mathcal{E}_2$ gives us that x' has a value higher than $x^\star$. From Lemma 3.4 we get that

$$\Pr[\mathcal{E}_2] \geq 1 - 2p \exp\left(-\delta^2 n/3\right).$$

Taking the union bound over $\mathcal{E}_1$ and $\mathcal{E}_2$ we get the required result. □

5.2 Hardness results

In this section, we present the formal statements of our hardness results for Program Denoised and their proofs.

5.2.1 Proof of Theorem 3.6 (1)

Theorem 5.2. *For all $p \geq 2$, deciding if Program Denoised is feasible is* **NP***-hard.*

Proof. We present a reduction from the subset-sum problem which is known to be **NP**-hard [23].

Subset sum problem. Given a set of m integers $A = \{a_1, \ldots, a_n\} \in \mathbb{Z}^n_{\geq 0}$ in binary, and a target sum $t \in \mathbb{Z}_{\geq 0}$. The subset-sum problem is: Is there a subset of A such that the items of the subset sum to t?

Without loss of generality assume that $a_i \leq t$ for all $i \in [n]$ and $t > 0$. If not, we can remove items which are larger than t. Since $a_i \geq 0$, we can solve the subset-problem for $t = 0$ in linear-time.

Reduction. Given an instance of the subset sum problem corresponding instance Program Target as follows: Let the number of items to select be n and let there be $m := 2n$ items. Consider one attribute (i.e., $s := 1$) with two disjoint values $p := 2$. For each item $i \in [m]$, let

$$q_i = \begin{cases} \left[\frac{a_i}{t}, 1 - \frac{a_i}{t}\right] & \text{if } i \in [n], \\ [0, 1] & \text{if } i \in [n+1, m]. \end{cases}$$

Define the constraints to be $U_1 = 1$ and $U_2 = n - 1$. In this construction, for each $i \in [n]$, the integer $a_i \in A$ corresponds to the i-th item. Notice that the input to Program Target is polynomial-sized in the input of the subset-sum problem. Further, the values q_i, and constraints U can be calculated in polynomial time.

It remains to show that A has a subset $S \subseteq A$ which sums to U if and only if Program Target is feasible.

($\implies$). If A has a subset $S \subseteq A$ such that $\sum_{a \in S} a = t$, define the solution x of Program Target by choosing $|S| \leq n$ items corresponding to integers in S, and any $n - |S| \leq n$ items from $[m] \backslash [n]$. It holds

$$\sum_{i \in [m]} x_i q_{i1} = \left(\sum_{i=1}^{n} x_i q_{i1} + \sum_{i \in [m] \backslash [n]} x_i q_{i1} \right) = \left(\sum_{a \in S} \frac{a}{t} + \sum_{i=1}^{n-|S|} 0 \right) = 1 \ \leq \ U_1,$$

$$\sum_{i \in [m]} x_i q_{i2} = \left(\sum_{i=1}^{n} x_i q_{i2} + \sum_{i \in [m] \backslash [n]} x_i q_{i2} \right) = \left(\sum_{a \in S} \left(1 - \frac{a}{t}\right) + \sum_{i=1}^{n-|S|} 1 \right) = n - 1 \ \leq \ U_2.$$

Thus, Program Target is feasible if the subset-sum instance is a 'YES' instance.

($\Longleftarrow$). For the opposite direction, notice that for any subset x, $\sum_{i \in [m]} x_i (q_{i1} + q_{i2}) = n$, i.e., the sum of $q_{i\ell}$ for all items and both properties is fixed. Since $U_1 + U_2 = n$, the constraints of both properties must hold with equality:

$$\sum_{i \in [m]} x_i q_{i1} = 1 \text{ and } \sum_{i \in [m]} x_i q_{i2} = n - 1. \tag{18}$$

Define a solution $S \subseteq A$ which has items of A corresponding to items $i \in [n]$ present in x. Since $q_{i1} = 0$ for any $i \in [m] \backslash [n]$, it follows from (18) that $\sum_{a \in S} a/t = 1$ or $\sum_{a \in S} a = t$. Thus, the subset sum instance is a 'YES' instance if Program Target is feasible.

Combining both cases gives us that the Program Target instance is feasible iff the subset sum instance is a 'YES' instance. Thus, we can use an algorithm to check the feasibility of Program Target to solve an arbitrary subset sum problem. □

5.2.2 Proof of Theorem 3.6 (2)

Theorem 5.3. *For all $p \geq 3$, Program Denoised is* **APX**-*hard.*

Proof. We present a reduction from the d-dimensional knapsack problem which is known to be **APX**-hard for $d \geq 2$ [47].

d-dimensional knapsack problem (d-KP). Given a vectors $v, c \in \mathbb{R}^k_{\geq 0}$, a matrix $w \in \mathbb{R}^{k\times d}_{\geq 0}$ the d-dimensional knapsack (d-KP) is the problem to solve the following integer linear program.

$$\max_{x\in\{0,1\}^k} \quad \sum_{i\in[k]} v_i x_i \tag{19}$$

$$\text{s.t.} \quad \forall\, \ell \in [d], \quad \sum_{i\in[k]} w_{i\ell} x_i \leq c_\ell, \tag{Capacity; 20}$$

$$x \in \{0,1\}^k. \tag{Integrality; 21}$$

Without loss of generality we assume $w_{i\ell} \leq c_\ell$ for all $i \in [k]$ and $\ell \in [d]$. If not, we can remove such items in linear time.

Reduction. Given an instance of d-KP define an instance of Program Target with $s \coloneqq 1$ as follows:

1. Set $p \coloneqq d+1$, $n \coloneqq k$, $m \coloneqq 2k$, and $U_p \coloneqq m$. For all $\ell \in [p-1]$ set
$$U_\ell \coloneqq \frac{c_\ell}{\sum_{k\in[d]} c_k}.$$
2. For each item $i \in [k]$ set $w_i \coloneqq v_i$, and for all $\ell \in [p-1]$ set
$$q_{i\ell} \coloneqq \frac{w_{i\ell}}{\sum_{j\in[d]} c_j}.$$
Finally set, $q_{ip} = 1 - \sum_{i\in[p-1]} q_{i\ell}$.
3. Add k additional *dummy* items $(k+1, k+2, \ldots, 2k)$. For all $i \in [m]\backslash[n]$, set $q_{ip} = 1$ and $w_i \coloneqq 0$. Also, for all $i \in [m]\backslash[n]$ and $\ell \in [p-1]$ set $q_{i\ell} = 0$.

Recall that Δ^p is the p-dimensional simplex. Note that for all $i \in [m]$, $q_i \in \Delta^p$. Now, it is easy to see that the construction is a valid instance of the Program Target.

Let $x^\star \in \{0,1\}^k$ be a solution d-KP, such that, $\sum_{i=1}^m x^\star = r \leq l$. Define a solution $y^\star \in \{0,1\}^m$ to Program Target by selecting any $(k-r)$ dummy items. Alternatively, given a solution given a $y^\star \in \{0,1\}^m$ to Program Target define a solution $x^\star \in \{0,1\}^k$ to d-KP, by omitting the dummy items.

($\Longleftrightarrow$). Consider a solution $x^\star$ to d-KP. Then for all $\ell \in [d]$, $x^\star$ satisfies

$$\begin{aligned}\sum_{i\in[k]} w_{i\ell} x_i^\star \leq c_\ell &\iff \sum_{i\in[k]} \left(\frac{w_{i\ell}}{\sum_{j\in[d]} c_j}\right) x_i^\star \leq \left(\frac{c_\ell}{\sum_{j\in[d]} c_j}\right) \\ &\iff \sum_{i\in[k]} q_{i\ell} x_i^\star \leq U_\ell \\ &\iff \sum_{i\in[m]} q_{i\ell} y_i^\star \leq U_\ell. \qquad \text{(Using } q_{i\ell} = 0 \text{ for } i \leq [m]\backslash[n], \ell \in [p-1])\end{aligned}$$

Further, by construction $\sum_{i=1}^m y_i^\star = n$. Thus $x^\star$ is feasible for d-KP iff $y^\star$ is feasible for the corresponding instance of Program Target. Finally, it holds

$$\sum_{i\in[k]} v_i x_i^\star = \sum_{i\in[k]} w_i x_i^\star = \sum_{i\in[m]} w_i y_i^\star. \qquad \text{(Using } w_i = 0 \text{ for } i \in [m]\backslash[n])$$

This shows that the reduction is approximation preserving, we conclude that Program Target is **APX**-hard for all $p = (d + 1) \geq 3$.

$\square$

5.2.3 Proof of Theorem 3.6 (3)

Theorem 5.4. *For every constant $c > 0$, the following* violation gap *variant of Program Denoised is* **NP**-*hard.*

- *Output YES if the input instance is satisfiable.*
- *Output NO if there is no solution which violates every upper bound constraint at most an additive factor of c.*

Our reduction below is similar to the one used in [17, Theorem 10.4].

Proof. We use the inapproximability of the independent set [40, 72] to prove the theorem. The inapproximability result states that: Given a graph $G(V, E)$ it is **NP**-hard to approximate the size of the maximum independent set to within a multiplicative factor of $|V|^{1-\varepsilon}$ for every constant $\varepsilon > 0$.

Fix any constant $c > 0$. Then our reduction is as follows:
Reduction. Consider an instance of the independent set problem, i.e., given a graph $G(V, E)$ and a number $n \in \mathbb{N}$ to check whether G has an independent set of size at least n. Construct an instance of Program Denoised with $m = |V|$ candidates and the same n. For each clique C in G of size at least $(c + 2)$ add a protected attribute C which takes two values, and set an upper bound on the number of items with value 1 for attribute C: $U_1^{(k)} = 1$. Formally, for each attribute C, set $L_1^{(C)} = 0$ and $U_1^{(C)} = 1$ and $L_0^{(C)} = 0$ and $U_0^{(C)} = n$. Further, for each vertex $v \in C$, set $q_{v,1}^{(C)} = 1$ and $q_{v,0}^{(C)} = 0$. Note that there are at most $\binom{m}{c+2} = m^{c+2} = \text{poly}(m)$ protected attributes.

We claim the following:

- If the Program Denoised is not feasible then G does not have an independent set of size n.
- If Program Denoised has a solution which violates the upper bound constraints by at most c (additive), then G has an independent set of size at least $n^{1/(c+1)}$.

These claims prove the theorem: if there was an algorithm $\mathcal{A}$ to solve Program Denoised in polynomial time, then we can use it to approximate independent set within $|V|^{1-1/(c+1)}$ factor in polynomial time,[19] which is **NP**-hard.

It remains to establish the claim. Notice that if G has an independent set S of size n, then this gives a feasible solution to Program Denoised by picking the items corresponding to elements in S. This establishes the first claim.

Given a subset S which violates the constraints of Program Denoised by at most c (additive), consider the subgraph G_S of G induced by S. G_S does not contain any $(c + 2)$-cliques. By [18, Lemma 4.3], G_S has an independent set of size at least $n^{1/(c+1)}$, and so also G. This establishes the second claim. $\square$

6 Theoretical results with multiple protected attributes

In this section, we extend our theoretical results to multiple nonbinary (and intersectional) protected attributes (i.e., $s \geq 1$ and $p \geq 1$). Like Section 3, our main algorithmic result is an approximation algorithm for Program Target.

Theorem 6.1 (**An approximation algorithm for Program Target when $s \geq 1$**). *There is an algorithm (Algorithm 2) that given an instance of Program Target and q from Definition 2.1, outputs a selection $x \in \{0, 1\}^m$, such that, with probability at least $1 - 4ps \exp\left(-\delta^2 n/3\right)$ over the noise in the protected attributes of each item, the selection x*

1. *has a value at least as* high *as the optimal value of Program Target,*

[19]The approximation algorithm returns the same answer as the $\mathcal{A}$.

2. violates the cardinality constraint (5) *(additively) by at most*

$$1+\sum\nolimits_{k=1}^{s}(p_k-1),$$

3. and violates the fairness constraints (4) *(additively) by at most*

$$1+2\delta n+\sum\nolimits_{k=1}^{s}(p_k-1).$$

The algorithms runs in polynomial time in the bit complexity of the inputs.

Remark 6.2. *Note that in the special case that* $s=1$, $1+\sum_{k=1}^{s}(p_k-1)=p_1$ *and* $1+2\delta n+\sum_{k=1}^{s}(p_k-1)=1+2\delta n+p_1$*, and we recover Theorem 3.1 from Theorem 6.1.*

Compared to Theorem 3.1, the high-probability bound and approximation factor are weaker by roughly a factor of s. Note that, in many real-world applications, p and s are small constants compared to the number of items selected n. Here, Theorem 6.1 implies that x violates the fairness constraints (Equation (4)) by a multiplicative factor of at most $(1+2\delta+o(1))$ and the constraint Equation (5) by a multiplicative factor of at most $(1+o(1))$ with high probability.

Algorithm 2 is equivalent to Algorithm 1; the only difference is that Algorithm 2 solves the Program Denoised problem for $s\geq 1$.

Remark 6.3 (Analogous to Remark 3.3). *This remark is analogous to We can strengthen Theorem 6.1 to guarantee that Algorithm 2 finds an* $x\in\{0,1\}^m$ *which does not violate the lower bound fairness constraint (left inequality in Equation* (4)*) and violates the upper bound fairness constraints by at most* $(s\cdot(p-1)+2\delta n)$ *(without changing other conditions). In particular, this shows that, if one places only lower bound fairness constraints, then subset found by Algorithm 2 would never violate the fairness constraints.*

Proof of Theorem 6.1

The proof of Theorem 6.1 follows a similar template as the proof of Theorem 3.1. Instead of repeating the entire proof, we highlight how the proof of Theorem 6.1 differs from the proof of Theorem 3.1.

We first present a generalization of Lemma 3.4.

Lemma 6.4. *For all* $\delta\in(0,1)$ *and* $x\in[0,1]^m$*, s.t.,* $\sum_{i=1}^{m}x_i=n$*:*

$$\forall\ k\in[s],\ \ell\in[p_k],\qquad \left|\sum\nolimits_{i\in G_\ell^{(k)}}x_i-\sum\nolimits_{i\in[m]}q_{i\ell}^{(k)}x_i\right|\leq n\delta$$

holds with probability at least $1-2ps\exp\left(-\delta^2 n/3\right)$ *over the noise in the protected attributes of each item.*

Proof. The result follows by applying Lemma 3.4 for each attribute $k\in[s]$ and taking the union bound over the events. $\square$

Remark 6.5 (Hardness results). *Note that the hardness results from Theorem 5.2 and Theorem 5.3 also apply when* $s\geq 1$*. Like in Section 3, we overcome this hardness by allowing solutions to violate the constraints of Program Denoised by an additive amount.*

Consider the following linear-programming relaxation of Program Denoised

$$\begin{aligned}
\max_{x\in[0,1]^m}\quad & \sum\nolimits_{i=1}^{m}w_ix_i && \text{(Relaxed-Denoised; 22)}\\
\text{s.t.,}\quad & L_\ell^{(k)}-\delta n\leq\sum\nolimits_{i=1}^{m}q_{i\ell}^{(k)}x_i\leq U_\ell^{(k)}+\delta n,\quad \forall\ k\in[s],\ \ell\in[p_k], &&\\
& \sum\nolimits_{i=1}^{m}x_i=n. &&
\end{aligned}$$

We can prove the following lemma.

Lemma 6.6 (An optimal solution with p fractional entries). *Any basic feasible solution $x \in [0,1]^m$ of LP-Denoised has at most*

$$\min\left(m, 1+\sum\nolimits_{k=1}^{s}(p_k-1)\right)$$

fractional values, i.e.,

$$\sum\nolimits_{i=1}^{m} \mathbb{I}[x_i \in (0,1)] \leq \min\left(m, 1+\sum\nolimits_{k=1}^{s}(p_k-1)\right).$$

Proof. In the proof of Lemma 3.5, we show that any basic feasible solution satisfies at most p_1 linearly-independent non-bounding-box constraints with equality. Toward this, we show that any set of p_1 linear-independent lower bound or upper bound constraints which are satisfied with equality, must contain the cardinality constraint ($\sum_{i=1}^{m} x_i = n$) in their span.

Lemma 6.6 follows by using the same argument for all protected attributes and noticing that for each protected attribute $k \in [s]$, any set of p_k linear-independent lower bound or upper bound constraints which are satisfied with equality, must contain the cardinality constraint ($\sum_{i=1}^{m} x_i = n$) in their span. □

Remark 6.7. *Note that in the special case that $s = 1$, $1 + \sum_{k=1}^{s}(p_k - 1) = p_1$, and we recover Lemma 3.5 from Lemma 6.6. Further, using Fact 5.1, it follows that Lemma 3.5 is tight.*

Proof of Theorem 6.1. The run time follows since there are polynomial time algorithms to find a basic feasible solution of a linear program.

The rest of proof follows a similar template to that of Theorem 3.1, and follows by using the generalizations of the lemmas proved above and replacing the additive error of p_1 (in x' due to rounding) with

$$1+\sum\nolimits_{k=1}^{s}(p_k-1).$$

Let $x^\star$ be an optimal solution for Program Target. Using Lemma 6.4 it can be shown that $x^\star$ is feasible for Program Denoised with probability at least $1 - 2ps\exp\left(-\delta^2 n/3\right)$. Assuming this event happens, it holds that the rounded solution x' (from Step 2 of Algorithm 2) has a value at least as large as $x^\star$. (This follows from the argument as in the proof of Theorem 3.1).

Further, from Lemma 3.5 it follows that x' picks at most

$$1+\sum\nolimits_{k=1}^{s}(p_k-1)$$

more elements than x. Thus, x' violates Equation (7), and so, Equation (5) by at most $1 + \sum_{k=1}^{s}(p_k - 1)$ (additively). Further, x' violates the fairness constraints of Program Denoised by at most $1 + \sum_{k=1}^{s}(p_k - 1)$ (additively). Combining this with Lemma 6.4, we get that with probability at least $1 - 2ps\exp\left(-\delta^2 n/3\right)$, x' violates the constraints of Program Target by at most the bound claimed in Theorem 6.1.

Finally, taking a union bound over above two events completes the proof. □

Algorithm 2: Algorithm for Program Target

```
Input: Numbers $n, s \in \mathbb{N}$, utility vector $w \in \mathbb{R}^m$, and for each $k \in [s]$: a probability matrix
       $q^{(k)} \in [0,1]^{m \times p_k}$ and constraint vectors $L^{(k)}$, $U^{(k)} \in \mathbb{R}^{p_k}_{\geq 0}$.
1. Solve $x \leftarrow$ Find a basic feasible solution to linear-programming relaxation
                of Program Denoised with inputs $(n, s, q, w, L, U)$.
2. Set $x'_i := \lceil x_i \rceil$ for all $i \in [m]$.   //Round solution
3. Return $x'$.
```

7 Limitations and future work

We consider the natural setting where the utility of a subset is the sum of utilities of its items. A useful extension to this work could consider submodular objectives, which are relevant when the goal is to select a subset which summarizes a collection of items [53].

Apart from this, some works also study other variants of subset selection, for example, diverse (and fair) subset selection in the online settings [64]. Studying and mitigating bias in the presence of noise under this variant is an interesting direction for future work.

Further, our approach assumes access to probabilistic information about the true protected attributes. If this information is itself is skewed or incorrect, then our approach can have a poor performance. Although, our empirical results on real-world data suggest that our approach can improve fairness even when there is some skew in the noise information (see, e.g., Remark 4.4). Still, determining probabilistic information more reliably is an important problem, and recent works have made some progress toward this goal [43, 41].

Furthermore, while we focus on the subset selection problem, our results can also extend to the ranking problem (where after selecting a subset, it must be ordered) by satisfying the fairness constraints in the top-k positions, for a small number choices of k, say $k_1 \leq k_2 \leq \cdots \leq k_g$;[20] this reduces the high-probability guarantee from $1 - 4p \exp\left(-\delta^2 n/3\right)$ to $1 - 4gp \exp\left(-\delta^2 k_1/3\right)$. This is particularly relevant in the setting where results are displayed one page at a time. Satisfying the constraints for a larger number of positions with high probability might require stronger information about noise, and is an interesting direction for future work.

Empirically, we could report fairness on several other metrics, e.g., selection lift or extended difference [38, 37, 11]. We focus on risk difference as it is closer to our approach. Nevertheless, Program Target can mitigate discrimination with respect to selection lift (and other metrics) as well (e.g., see [16, 13]). Empirically evaluating this would be an important direction for future work.

Finally, we note that bias is a systematic issue and this work only addresses one aspect of it. Indeed, any such approach is limited by how people and the broader system uses the subset presented to them; e.g., a recruiter on an online hiring platform might deliberately reject minority candidates even when presented with a representative candidate subset. Thus, it is important to complement our approach with other necessary tools to mitigate bias and counter discrimination.

8 Conclusion

We consider the problem of mitigating bias in subset selection when the protected attributes are noisy, or are missing for some or all of the entries and must be predicted using proxy variables. We note that accounting for real-world noise in algorithms is important in mitigating bias, and not accounting for noise can have unintended adverse effects, e.g., adding fairness constraints in a noise oblivious fashion can even decrease fairness when the protected attributes are noisy (Section 4.2).

We consider a model of noise where we have probabilistic information about the protected attributes (Definition 2.1), and develop a framework to mitigate bias, which given this information, can satisfy one from a large class of fairness constraints with at most a small multiplicative error with high probability (Section 3). In our empirical study, we observe that this approach achieves a high level of fairness on standard fairness metrics (e.g., risk difference), even when the probabilistic information about protected attributes is skewed (see Remark 4.4 and Section 7), and has a better tradeoff between utility and fairness compared to several prior approaches (Section 4).

Acknowledgements

This research was supported in part by a J.P. Morgan Faculty Award. We would like to thank Nisheeth K. Vishnoi for several useful discussions on the problem.

[20]To do so: first, pick k_1 items and place them top-k_1 positions, then from those remaining pick $(k_2 - k_1)$ items and place them in the next $(k_2 - k_1)$ positions, and so on.

References

[1] OpenCV: Open Source Computer Vision Library. https://github.com/opencv/opencv_3rdparty/raw/dnn_samples_face_detector_20170830/res10_300x300_ssd_iter_140000.caffemodel.

[2] Akshay Agrawal, Robin Verschueren, Steven Diamond, and Stephen Boyd. A rewriting system for convex optimization problems. *Journal of Control and Decision*, 5(1):42–60, 2018.

[3] Dana Angluin and Philip D. Laird. Learning from noisy examples. *Mach. Learn.*, 2(4):343–370, 1987.

[4] Pranjal Awasthi, Matthäus Kleindessner, and Jamie Morgenstern. Equalized odds postprocessing under imperfect group information. In *International Conference on Artificial Intelligence and Statistics*, pages 1770–1780. PMLR, 2020.

[5] Tony Barboza and Joseph Serna. As coronavirus deaths surge, missing racial data worry l.a. county officials. *Los Angeles Times*, April 2020. https://www.latimes.com/california/story/2020-04-06/missing-racial-data-coronavirus-deaths-worries-los-angeles-county-officials.

[6] Marianne Bertrand and Sendhil Mullainathan. Are emily and greg more employable than lakisha and jamal? a field experiment on labor market discrimination. *American economic review*, 94(4):991–1013, 2004.

[7] Joy Buolamwini and Timnit Gebru. Gender shades: Intersectional accuracy disparities in commercial gender classification. In *FAT*, volume 81 of *Proceedings of Machine Learning Research*, pages 77–91. PMLR, 2018.

[8] Consumer Financial Protection Bureau. *Using publicly available information to proxy for unidentified race and ethnicity.* 2014. https://files.consumerfinance.gov/f/201409_cfpb_report_proxy-methodology.pdf.

[9] United States Census Bureau. Current Population Survey (CPS). https://www.census.gov/programs-surveys/cps.html.

[10] United States Census Bureau. FINC-02. Age of Reference Person, by Total Money Income, Type of Family, Race and Hispanic Origin of Reference Person. https://www.census.gov/data/tables/time-series/demo/income-poverty/cps-finc/finc-02.html.

[11] Toon Calders and Sicco Verwer. Three naive bayes approaches for discrimination-free classification. *Data Min. Knowl. Discov.*, 21(2):277–292, 2010.

[12] Carlos Castillo. Fairness and transparency in ranking. *SIGIR Forum*, 52(2):64–71, January 2019.

[13] L. Elisa Celis, Lingxiao Huang, Vijay Keswani, and Nisheeth K. Vishnoi. Classification with Fairness Constraints: A Meta-Algorithm with Provable Guarantees. In *FAT*, pages 319–328. ACM, 2019.

[14] L. Elisa Celis, Lingxiao Huang, Vijay Keswani, and Nisheeth K. Vishnoi. Fair classification with noisy protected attributes: A framework with provable guarantees. *CoRR*, abs/2006.04778, 2020.

[15] L. Elisa Celis and Vijay Keswani. Implicit Diversity in Image Summarization. *ACM Conference on Computer-Supported Cooperative Work and Social Computing*, 2020.

[16] L. Elisa Celis, Vijay Keswani, Damian Straszak, Amit Deshpande, Tarun Kathuria, and Nisheeth K. Vishnoi. Fair and Diverse DPP-Based Data Summarization. In *ICML*, volume 80 of *Proceedings of Machine Learning Research*, pages 715–724. PMLR, 2018.

[17] L. Elisa Celis, Damian Straszak, and Nisheeth K. Vishnoi. Ranking with Fairness Constraints. In *ICALP*, volume 107 of *LIPIcs*, pages 28:1–28:15. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018.

[18] Chandra Chekuri and Sanjeev Khanna. On multidimensional packing problems. *SIAM journal on computing*, 33(4):837–851, 2004.

[19] Jiahao Chen, Nathan Kallus, Xiaojie Mao, Geoffry Svacha, and Madeleine Udell. Fairness under unawareness: Assessing disparity when protected class is unobserved. In *FAT*, pages 339–348. ACM, 2019.

[20] Flavio Chierichetti, Ravi Kumar, Silvio Lattanzi, and Sergei Vassilvitskii. Fair clustering through fairlets. In *NIPS*, pages 5036–5044, 2017.

[21] Flavio Chierichetti, Ravi Kumar, Silvio Lattanzi, and Sergei Vassilvitskii. Matroids, matchings, and fairness. In *AISTATS*, volume 89 of *Proceedings of Machine Learning Research*, pages 2212–2220. PMLR, 2019.

[22] Andrew J Coldman, Terry Braun, and Richard P Gallagher. The classification of ethnic status using name information. *Journal of Epidemiology & Community Health*, 42(4):390–395, 1988.

[23] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms.* MIT press, 2009.

[24] Matthew Costello, James Hawdon, Thomas Ratliff, and Tyler Grantham. Who views online extremism? individual attributes leading to exposure. *Computers in Human Behavior*, 63:311–320, 2016.

[25] N.R. Council, D.B.S.S. Education, C.N. Statistics, P.D.C.R.E. Data, E. Perrin, and M.V. Ploeg. *Eliminating Health Disparities: Measurement and Data Needs.* National Academies Press, 2004.

[26] Marina Drosou, H. V. Jagadish, Evaggelia Pitoura, and Julia Stoyanovich. Diversity in big data: A review. *Big Data*, 5(2):73–84, 2017.

[27] Marc Elliott, Peter Morrison, Allen Fremont, Daniel Mccaffrey, Philip Pantoja, and Nicole Lurie. Using the census bureau's surname list to improve estimates of race/ethnicity and associated disparities. *Health Services and Outcomes Research Methodology*, 9:252–253, 06 2009.

[28] Marc N Elliott, Allen Fremont, Peter A Morrison, Philip Pantoja, and Nicole Lurie. A new method for estimating race/ethnicity and associated disparities where administrative records lack self-reported race/ethnicity. *Health services research*, 43(5 Pt 1):1722—1736, October 2008.

[29] Erin L Faught, Patty L Williams, Noreen D Willows, Mark Asbridge, and Paul J Veugelers. The association between food insecurity and academic achievement in canadian school-aged children. *Public health nutrition*, 20(15):2778–2785, 2017.

[30] Kevin Fiscella and Allen Fremont. Use of geocoding and surname analysis to estimate race and ethnicity. *Health services research*, 41:1482–500, 09 2006.

[31] Kevin Fiscella and Allen M Fremont. Use of geocoding and surname analysis to estimate race and ethnicity. *Health services research*, 41(4p1):1482–1500, 2006.

[32] Benoît Frénay and Michel Verleysen. Classification in the presence of label noise: A survey. *IEEE Trans. Neural Networks Learn. Syst.*, 25(5):845–869, 2014.

[33] Sahin Cem Geyik, Stuart Ambler, and Krishnaram Kenthapadi. Fairness-aware ranking in search & recommendation systems with application to linkedin talent search. In *KDD*, pages 2221–2231. ACM, 2019.

[34] Martin Grötschel, László Lovász, and Alexander Schrijver. *Geometric algorithms and combinatorial optimization*, volume 2. Springer Science & Business Media, 2012.

[35] Maya R. Gupta, Andrew Cotter, Mahdi Milani Fard, and Serena Wang. Proxy fairness. *CoRR*, abs/1806.11212, 2018.

[36] LLC Gurobi Optimization. Gurobi optimizer reference manual, 2020.

[37] Sara Hajian, Josep Domingo-Ferrer, and Oriol Farràs. Generalization-based privacy preservation and discrimination prevention in data publishing and mining. *Data Mining and Knowledge Discovery*, 28(5):1158–1188, Sep 2014.

[38] Sara Hajian, Josep Domingo-Ferrer, Anna Monreale, Dino Pedreschi, and Fosca Giannotti. Discrimination- and privacy-aware patterns. *Data Mining and Knowledge Discovery*, 29(6):1733–1782, Nov 2015.

[39] Moritz Hardt, Eric Price, and Nati Srebro. Equality of opportunity in supervised learning. In Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 3315–3323, 2016.

[40] Johan Hastad. Clique is hard to approximate within n/sup 1-/spl epsiv. In *Proceedings of 37th Conference on Foundations of Computer Science*, pages 627–636. IEEE, 1996.

[41] Úrsula Hébert-Johnson, Michael P. Kim, Omer Reingold, and Guy N. Rothblum. Multicalibration: Calibration for the (computationally-identifiable) masses. In *ICML*, volume 80 of *Proceedings of Machine Learning Research*, pages 1944–1953. PMLR, 2018.

[42] Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of IR techniques. *ACM Trans. Inf. Syst.*, 20(4):422–446, 2002.

[43] Christopher Jung, Changhwa Lee, Mallesh M. Pai, Aaron Roth, and Rakesh Vohra. Moment multicalibration for uncertainty estimation. *CoRR*, abs/2008.08037, 2020.

[44] Nathan Kallus, Xiaojie Mao, and Angela Zhou. Assessing algorithmic fairness with unobserved protected class using data combination. In *FAT**, page 110. ACM, 2020.

[45] Matthew Kay, Cynthia Matuszek, and Sean A. Munson. Unequal representation and gender stereotypes in image search results for occupations. In *CHI*, pages 3819–3828. ACM, 2015.

[46] Michael Kearns, Aaron Roth, and Zhiwei Steven Wu. Meritocratic fairness for cross-population selection. In *International Conference on Machine Learning*, pages 1828–1836, 2017.

[47] Hans Kellerer, Ulrich Pferschy, and David Pisinger. *Knapsack problems*. Springer, 2004.

[48] Jon M. Kleinberg and Manish Raghavan. Selection problems in the presence of implicit bias. In *ITCS*, volume 94 of *LIPIcs*, pages 33:1–33:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.

[49] Howard K Koh, Garth Graham, and Sherry A Glied. Reducing racial and ethnic disparities: the action plan from the department of health and human services. *Health Affairs*, 30(10):1822–1829, 2011.

[50] Gueorgi Kossinets. Effects of missing data in social networks. *Social Networks*, 28(3):247–268, 2006.

[51] Matt J. Kusner, Joshua R. Loftus, Chris Russell, and Ricardo Silva. Counterfactual fairness. In *NIPS*, pages 4066–4076, 2017.

[52] Alexandre Louis Lamy and Ziyuan Zhong. Noise-tolerant fair classification. In *NeurIPS*, pages 294–305, 2019.

[53] Hui Lin and Jeff Bilmes. A class of submodular functions for document summarization. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 510–520, 2011.

[54] LinkedIn. Inferred Age or Gender on LinkedIn, February 2018. `https://www.linkedin.com/help/linkedin/answer/3566/inferred-age-or-gender-on-linkedin?lang=en`.

[55] Tongliang Liu and Dacheng Tao. Classification with noisy labels by importance reweighting. *IEEE Trans. Pattern Anal. Mach. Intell.*, 38(3):447–461, 2016.

[56] Naresh Manwani and P. S. Sastry. Noise tolerance under risk minimization. *IEEE Trans. Cybern.*, 43(3):1146–1151, 2013.

[57] Rajeev Motwani and Prabhakar Raghavan. *Randomized algorithms.* Cambridge university press, 1995.

[58] Christos H Papadimitriou and Kenneth Steiglitz. *Combinatorial optimization: algorithms and complexity.* Courier Corporation, 1998.

[59] Adrian Rosebrock. Face detection with OpenCV and deep learning, February 2018. `https://www.pyimagesearch.com/2018/02/26/face-detection-with-opencv-and-deep-learning/`.

[60] Rasmus Rothe, Radu Timofte, and Luc Van Gool. IMDB-WIKI – 500k+ face images with age and gender labels. `https://data.vision.ee.ethz.ch/cvl/rrothe/imdb-wiki/`.

[61] Salvatore Ruggieri. Using t-closeness anonymity to control for non-discrimination. *Trans. Data Priv.*, 7(2):99–129, 2014.

[62] Catherine Saunders, Gary Abel, Anas El Turabi, Faraz Ahmed, and Georgios Lyratzopoulos. Accuracy of routinely recorded ethnic group information compared with self-reported ethnicity: Evidence from the english cancer patient experience survey. *BMJ open*, 3, 06 2013.

[63] Ashudeep Singh and Thorsten Joachims. Fairness of Exposure in Rankings. In *KDD*, pages 2219–2228. ACM, 2018.

[64] Julia Stoyanovich, Ke Yang, and H. V. Jagadish. Online set selection with fairness and diversity constraints. In *EDBT*, pages 241–252. OpenProceedings.org, 2018.

[65] USA The Census Bureau. Frequently Occurring Surnames from the Census 2010, April 2020. `https://www.census.gov/topics/population/genealogy/data/2010_surnames.html`.

[66] Eric Luis Uhlmann and Geoffrey L Cohen. Constructed criteria: Redefining merit to justify discrimination. *Psychological Science*, 16(6):474–480, 2005.

[67] Ke Yang, Vasilis Gkatzelis, and Julia Stoyanovich. Balanced ranking with diversity constraints. In *IJCAI*, pages 6035–6042. ijcai.org, 2019.

[68] Ke Yang, Joshua R. Loftus, and Julia Stoyanovich. Causal intersectionality for fair ranking. *CoRR*, abs/2006.08688, 2020.

[69] Ke Yang and Julia Stoyanovich. Measuring fairness in ranked outputs. In *SSDBM*, pages 22:1–22:6. ACM, 2017.

[70] Meike Zehlike, Francesco Bonchi, Carlos Castillo, Sara Hajian, Mohamed Megahed, and Ricardo A. Baeza-Yates. FA*IR: A Fair Top-k Ranking Algorithm. In *CIKM*, pages 1569–1578. ACM, 2017.

[71] Meike Zehlike and Carlos Castillo. Reducing disparate exposure in ranking: A learning to rank approach. In *WWW*, pages 2849–2855. ACM / IW3C2, 2020.

[72] David Zuckerman. Linear degree extractors and the inapproximability of max clique and chromatic number. In *STOC 2006*, pages 681–690, 2006.

A Extended empirical results

A.1 Implementation details

Optimization libraries. We used CVXPY [2] and Gurobi [36] to implement the algorithms.

Rounding scheme. The rounding scheme from our theoretical results (Step 2 of Algorithm 1) crucially uses the fact that the intermediate solution found has a small number of fractional entries. Since the solutions of MultObj can have a large number of fractional entries, the same rounding scheme is not useful. Instead, our can use randomized rounding [57], which, given a solution $x \in [0,1]^m$, outputs a subset $S \subseteq [m]$ of size n with probability $\prod_{i \in S} x_i$.

In our simulations, we use the same rounding scheme — randomized rounding [57] — for MultObj, FairExpec, and FairExpecGrp. We also ran a version of our experiments where, we used the rounding scheme from our theoretical results (Step 2 of Algorithm 1) for FairExpec, and FairExpecGrp, and randomized rounding for MultObj. We observed similar results in these simulations.

Choice of m and n. We set $m = 500$ and $n = 100$ in all experiments, except in Section 4.3 — where we set $m = 2000$ following [68] — and in Section 4.4 — where increase m to 1000. We increase m to 1000 in Section 4.4 to ensure that the sample of candidates has at least $n/4$ candidates from each of the four races. (Any algorithm requires this to be able to satisfy the fairness constraints.)

Specifics of cropping. For each image where the face-detector found a face, we cropped the image to the region "around" the detected face. More precisely, we expand the bounding box returned by the face-detector by 40% (on each side) and then crop the image to this expanded bounding box. If the expanded bounding box exceeded the image dimensions, we filled in the empty region by copying the last pixel layer.

Face-detector. In our implementation, we referenced the implementation by [59].

Specifics of computing $q_{i\ell}$. We ran the image classifier on images to get a set of predictions $F := \{f_i\}_i$. Then, we binned the values in F into $b = 20$ equally sized bins (over $[0,1]$). For all $j \in [b]$, let $B_j \subseteq [m]$ be the set of images in the j-th bin. Further, let G_m and G_f be the set of images containing men and women, respectively. Then, taking an uncalibrated f_i as input, we output calibrated $q_{i,\ell}$ as follows: Let f_i fall in the j-th bin, then for all $\ell \in \{m, f\}$ output

$$q_{i,\ell} := \frac{|B_j \cap G_\ell|}{|B_j|}. \tag{23}$$

A.2 Optimizing for Selection Lift

Given subset $S \in [m]$ and target $t \in [0,1]^p$, define the selection lift $\mathcal{F}_L(S,t) \in [0,1]$ as

$$\mathcal{F}_L(S,t) := \min_{\ell,k \in [p]} \left(\frac{|S \cap G_\ell|}{n \cdot t_\ell} \cdot \frac{n \cdot t_k}{|S \cap G_k|} \right). \tag{Selection lift; 24}$$

Note that when the target is proportional representation, then the above definition reduces to the usual definition of selection lift [38, 37, 11]. Let $\mathcal{A}(w,q) \subseteq [m]$ be the subset outputted by algorithm $\mathcal{A}$ on input (w,q). We report

$$\mathcal{F}_{L,\mathcal{A}} := \mathbb{E}\left[\mathcal{F}_L(\mathcal{A}(w,q),t)\right],$$

where the expectation is over the choices of (w,q). We drop the subscript $\mathcal{A}$ when the algorithm is not important or clear from context.

Discussion. Let selection lift of a selection $x \in \{0,1\}^m$, be $\mathcal{F}_L(x) \in [0,1]$. Fix the desired level $\sigma \in [0,1]$ of selection lift, and let the set of selections $x \in \{0,1\}^m$ which have $\mathcal{F}_L(x) \geq \sigma$ selection lift be $R(\sigma)$. Ideally, we would like to find

$$\operatorname{argmax}_{x \in R(\sigma)} w^\top x. \tag{25}$$

We believe FairExpec is not Pareto-optimal for $\mathcal{F}_L(\cdot)$ because its feasible, $R'(\sigma)$, is a strict subset of $R(\sigma)$, i.e., it outputs

$$\operatorname{argmax}_{x \in R'(\sigma)} w^\top x. \tag{26}$$

However, a reduction from Program (25) to (multiple instances of) Program (26) should ensure that FairExpec is Pareto-optimal for $\mathcal{F}_L(\cdot)$ (see, e.g., [13, Theorem 3.1]).

A.3 Additional simulations varying $^n/_m$

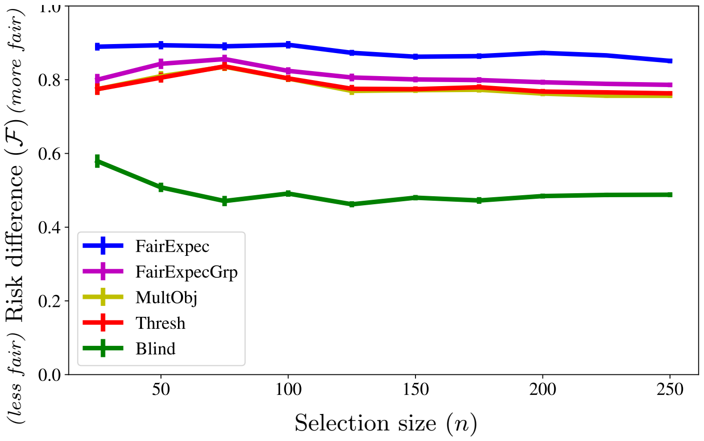


Figure 8: ***Risk difference on varying $^n/_m$ (Section 4.5):*** Towards analyzing the robustness of our approach to the fraction of items selected ($^n/_m$), we fix $m = 1000$ and vary n from 25 to 250 in Simulation 4.5. The y-axis shows the risk different $\mathcal{F}$ of different algorithms, and the x-axis shows n; $\mathcal{F}$ values are averaged over 200 trials, and the error bars represent the std. error of the mean. We find that on increasing n the difference between the $\mathcal{F}$ of different algorithms remains roughly the same. We refer the reader to Remark 4.6 for the details.

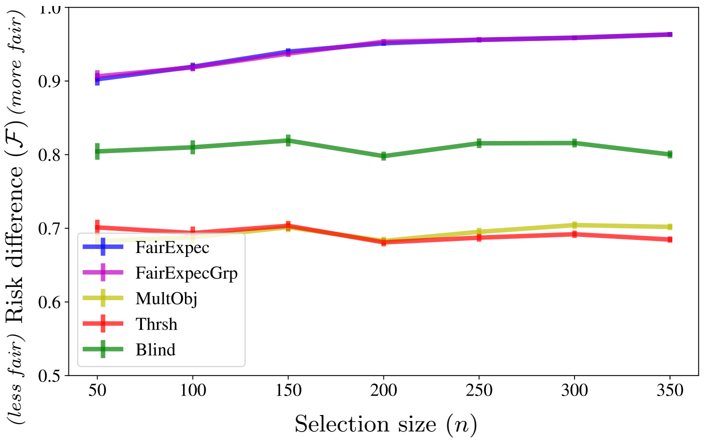


Figure 9: ***Risk difference on varying $^n/_m$ (Section 4.2):*** Towards analyzing the robustness of our approach to the fraction of items selected ($^n/_m$), we fix $m = 1000$ and vary n from 50 to 350 in Simulation 4.2. The y-axis shows the risk different $\mathcal{F}$ of different algorithms, and the x-axis shows n; $\mathcal{F}$ values are averaged over 100 trials, and the error bars represent the std. error of the mean. We find that on increasing n FairExpec and FairExpecGrp become slightly fairer compared to others. We refer the reader to Remark 4.6 for the details.

A.4 Additional plots for Section 4.2

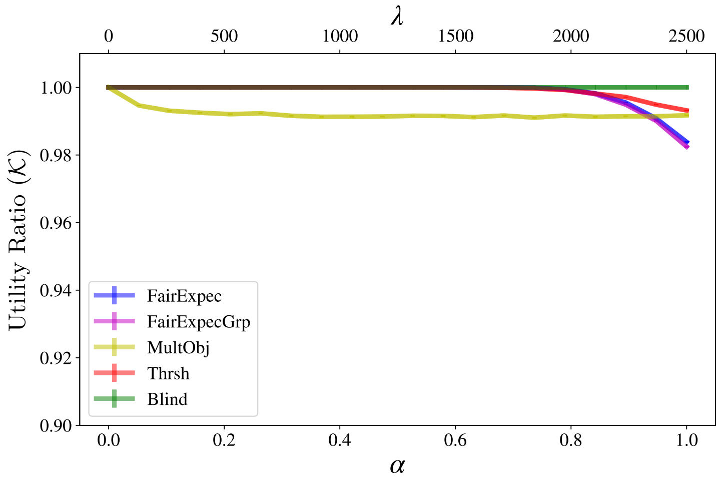


Figure 10: *Additional plot for Section 4.2:* This simulation considers the setting where the minority group (40% of total) has a higher FDR (30%) compared to the majority group, whose FDR is 10%, and the utilities of all candidates are iid from the uniform distribution. We refer the reader to Section 4.2 for the details of the simulation. The y-axis shows the utility ratio $\mathcal{K}$ of different algorithms, and the x-axis shows the constraint parameters (α for FairExpec, FairExpecGrp, and Thrsh, and λ for MultObj); $\mathcal{K}$ averaged over 500 trials, and the error bars represent the standard error of the mean. We observe that FairExpec and FairExpecGrp lose a larger fraction of the utility. Although, note that this because they have a higher level of fairness; see Figure 1.

Remark A.1 (Reading the double x-axis). *We note that both subfigures in Figure 10 and Figure 11 have a double x-axis: one for α and one for λ. Therefore, one should note compared the $\mathcal{F}$ or $\mathcal{K}$ of* MultObj *with other algorithms at a particular x-coordinate. It could be useful to consider the limiting their values at the largest α and λ, respectively.*

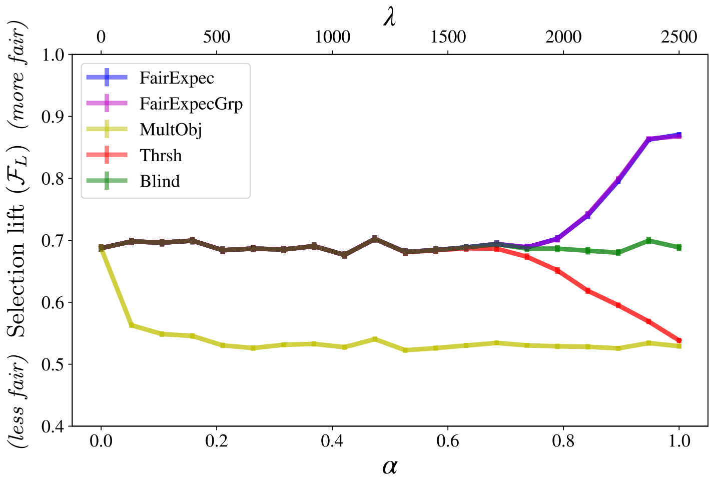


Figure 11: *Additional plot for Section 4.2 with selection lift:* This simulation considers the setting where the minority group (40% of total) has a higher FDR (30%) compared to the majority group, whose FDR is 10%, and the utilities of all candidates are iid from the uniform distribution. We refer the reader to Section 4.2 for the details of the simulation. The y-axis shows the selection lift $\mathcal{F}_L$ of different algorithms, and the x-axis shows the constraint parameters (α for FairExpec, FairExpecGrp, and Thrsh, and λ for MultObj); $\mathcal{F}_L$ values are averaged over 500 trials, and the error bars represent the standard error of the mean. We observe similar results as with risk difference ($\mathcal{F}$). For a definition of selection lift see Equation (24).

A.5 Additional plots for Section 4.3

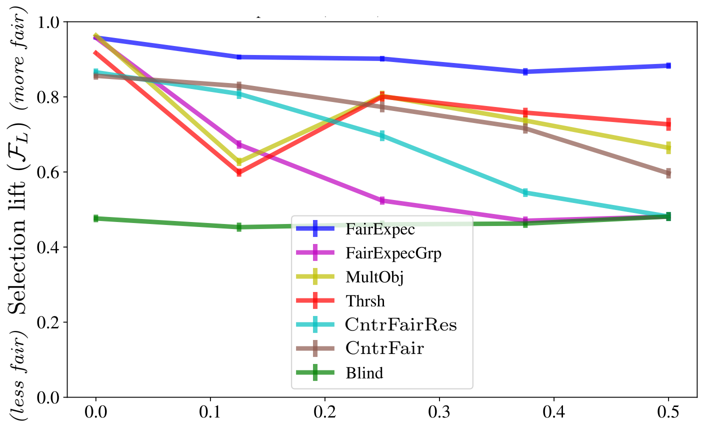


Figure 12: *Additional results for Section 4.3 with selection lift:* The y-axis shows the selection lift $\mathcal{F}_L$ of different algorithms, and the x-axis shows the amount of noise added $\tau \in [0, 1/2]$; $\mathcal{F}_L$ values are averaged over 100 trials, and the error bars represent the standard error of the mean. We refer the reader to Section 4.3 for the details of the simulation. We observe similar results as with risk difference ($\mathcal{F}$). For a definition of selection lift see Equation (24).

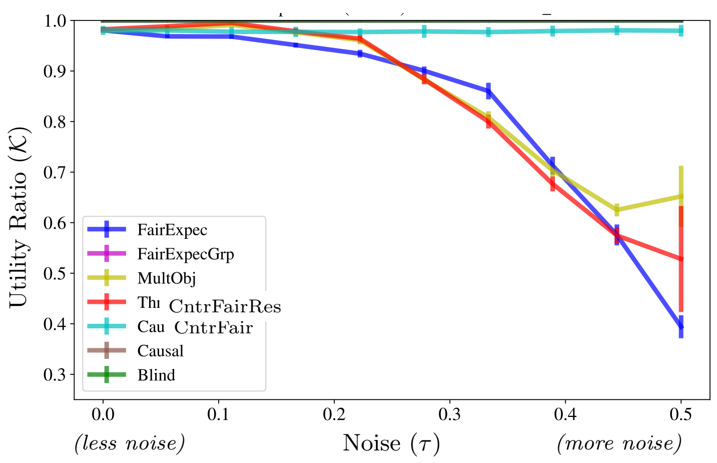


Figure 13: *Additional results for Section 4.3:* This simulation considers the setting where the utilities of a minority group have a lower average than the majority group, and both groups have identical noise. The y-axis shows the utility ratio $\mathcal{K}$ of different algorithms, and the x-axis shows the amount of noise added $\tau \in [0, 1/2]$; $\mathcal{K}$ values are averaged over 20 trials, and the error bars represent the standard error of the mean. We refer the reader to Section 4.3 for details and discussion. We note that the utility ratio of all algorithms decreases on adding noise. Finally, we note that utilities in this experiment can be negative; we observe that CntrFair has a negative utility ratio. The caveat is that CntrFair and CntrFairResolving, try to maximize the counterfactual utilities.

Given subset $S \in [m]$ and a $\ell \in [p]$, we define the selection rate of S with respect to group G_ℓ as

$$SR(S, \ell) \coloneqq \frac{|S \cap G_\ell|}{n} \cdot \frac{m}{|G_\ell|} \qquad \text{(Selection rate; 27)}$$

Let $\mathcal{A}(w, q) \subseteq [m]$ be the subset outputted by algorithm $\mathcal{A}$ on input (w, q). We report

$$SR_{\mathcal{A},\ell} \coloneqq \mathbb{E}\left[SR(\mathcal{A}(w, q), \ell)\right],$$

where the expectation is over the choices of (w, q). We drop the subscript $\mathcal{A}$ when the algorithm is not important or clear from context.

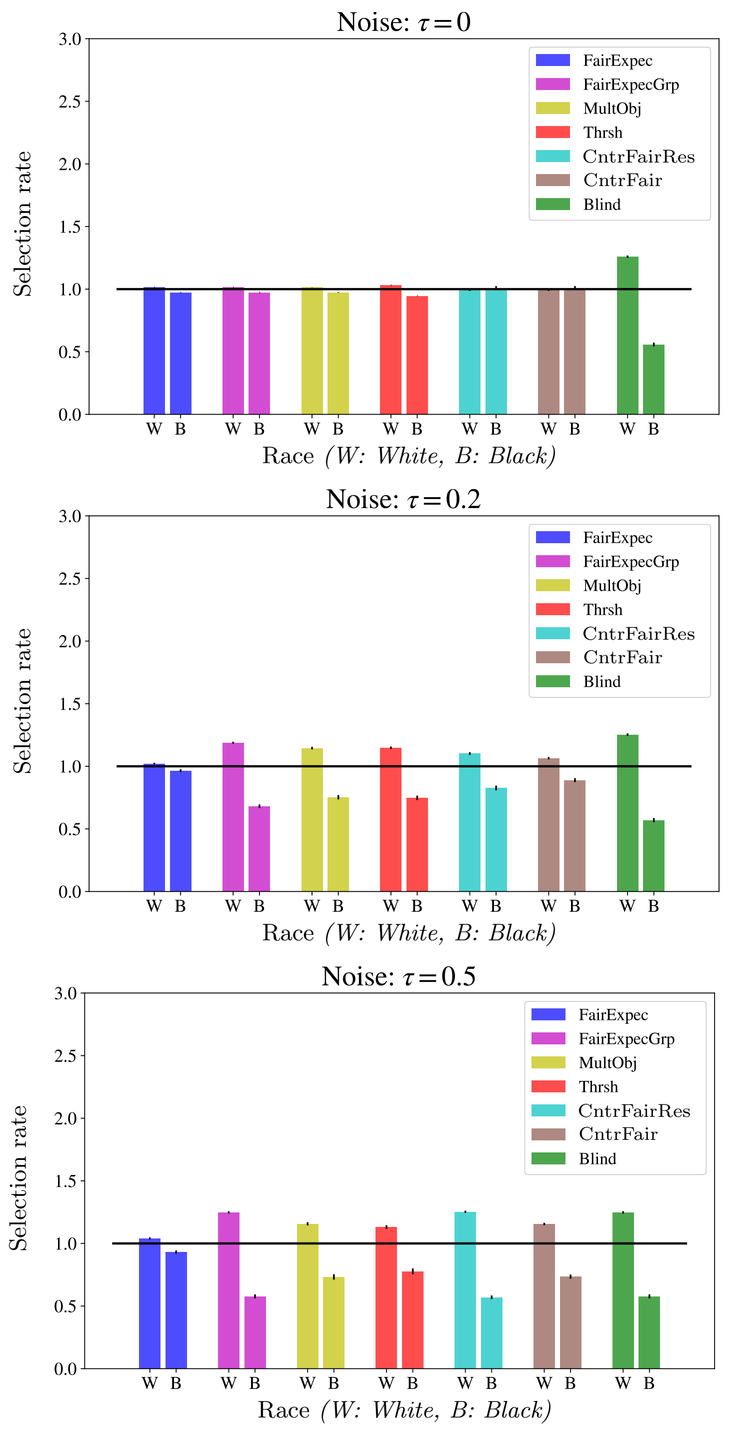
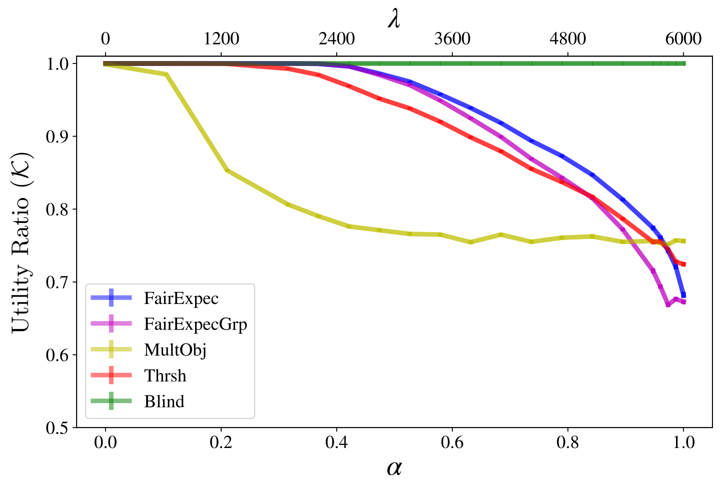
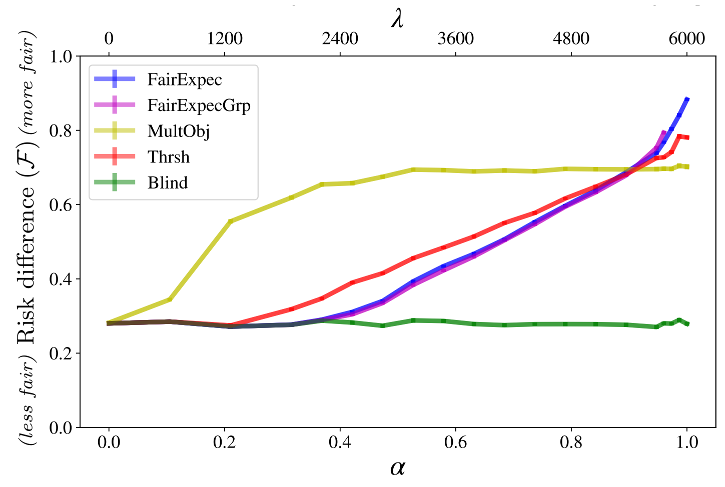


Figure 14: *Additional plots for Section 4.3:* This simulation considers the setting where the utilities of a minority group have a lower average than the majority group, and both groups have identical noise. The three plots show three values of noise $\tau \in \{0, 0.2, 0.5\}$ The y-axis shows the selection rate (see Equation (27)) of different algorithms; Selection rate values are averaged over 50 trials, and the error bars represent the standard error of the mean. We refer the reader to Section 4.3 for details and discussion. We observe that FairExpec is the closest having a selection rate of 1 across noise levels. We extended the code by [68] to generate this plot.

A.6 Additional results for Section 4.4



(a) The y-axis shows the utility ratio $\mathcal{K}$ of different algorithms, and the x-axis shows the constraint parameters (α for FairExpec, FairExpecGrp, and Thrsh, and λ for MultObj); $\mathcal{K}$ values are averaged over 100 trials, and the error bars represent the standard error of the mean.



(b) The y-axis shows the risk difference $\mathcal{F}$ of different algorithms, and the x-axis shows the constraint parameters (α for FairExpec, FairExpecGrp, and Thrsh, and λ for MultObj); $\mathcal{F}$ values are averaged over 100 trials, and the error bars represent the standard error of the mean.

Figure 15: *Additional plot for candidate selection (Section 4.4):* This simulation considers race as the protected attribute which takes $p = 4$ values, where an individual's utility is drawn from different distributions depending on their race. The simulation uses last name as a proxy to derive noisy information of race. We refer the reader to Section 4.4 for the details of the simulation.

Remark A.2 (**Reading the double x-axis).** *We note that both subfigures in Figure 15 have a double x-axis: one for α and one for λ. Therefore, one should note compared the $\mathcal{F}$ or $\mathcal{K}$ of* MultObj *with other algorithms at a particular x-coordinate. It could be useful to consider the limiting their values at the largest α and λ, respectively.*

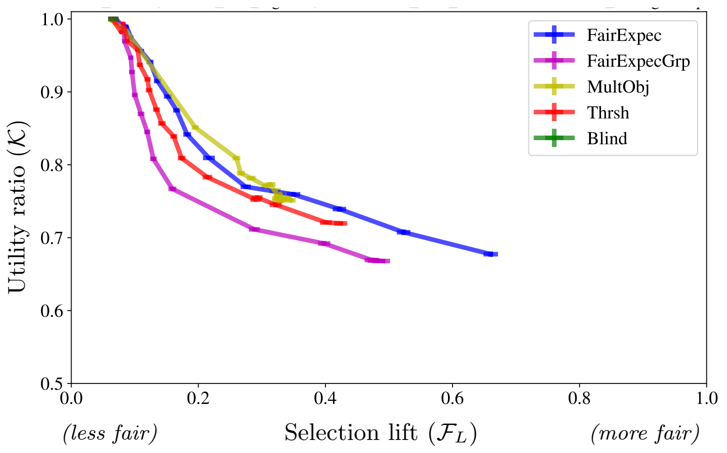


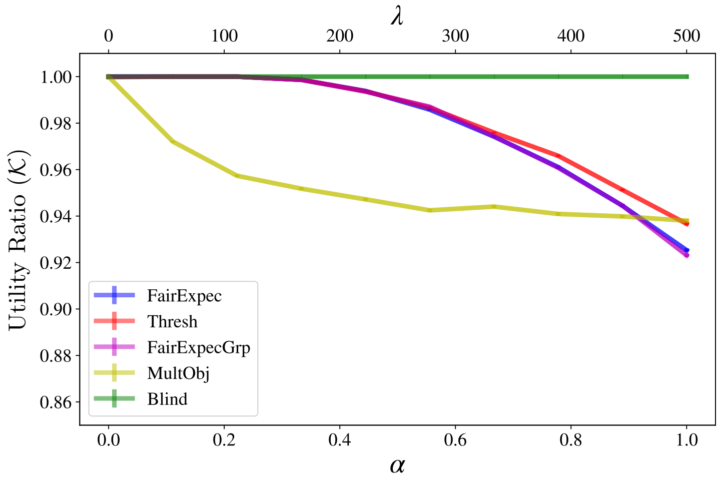
Figure 16: *Additional results for Section 4.4 with selection lift:* The y-axis shows the selection lift $\mathcal{F}_L$ of different algorithms, and the x-axis shows the risk difference of different algorithms; both $\mathcal{K}$ and $\mathcal{F}_L$ values are averaged over 200 trials, and the error bars represent the standard error of the mean. We refer the reader to Section 4.4 for the details of the simulation. We observe that FairExpec has lower utility ratio than MultObj for some values of selection lift; see Section 4.6 for a discussion. For a definition of selection lift see Equation (24).

A.7 Additional results for Section 4.5

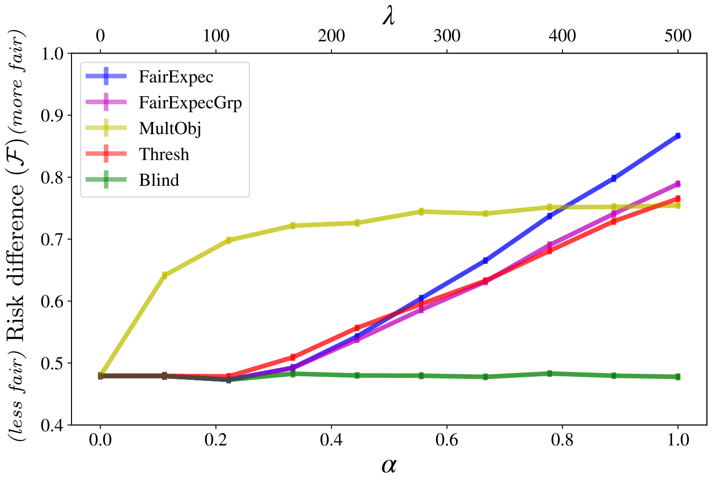
	Occupation name
$\textsc{Sty}_f(0.8)$ Num: 12	administrative assistant; counselor; dental hygienist; flight attendant; hairdresser; housekeeper; massage therapist; nurse; nurse practitioner; receptionist; social worker; special ed teacher
$\textsc{Sty}_m(0.8)$ Num: 29	announcer; barber; bill collector; building inspector; building painter; butcher; chief executive officer clergy member; construction worker; courier; crane operator; detective; dishwasher; electrician exterminator; garbage collector; groundskeeper; logistician; machinist; parking attendant; plumber police officer; private investigator; roofer; security guard; taxi driver; teller; web developer; welder

Figure 17: ***Occupations with at least 80% of images from one gender (from the occupations dataset [15])***. We refer the reader to Section 4.5 for more details, and a definition of $\textsc{Sty}_m(0.8)$ and $\textsc{Sty}_f(0.8)$.

Remark A.3 (**Reading the double x-axis**)**.** *We note that both subfigures in Figure 18 have a double x-axis: one for α and one for λ. Therefore, one should note compared the $\mathcal{F}$ or $\mathcal{K}$ of* MultObj *with other algorithms at a particular x-coordinate. It could be useful to consider the limiting their values at the largest α and λ, respectively.*

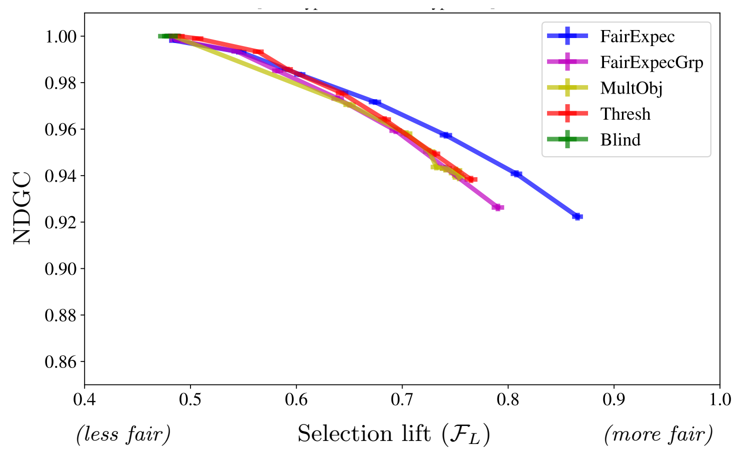


(a) The y-axis shows the utility ratio $\mathcal{K}$ of different algorithms, and the x-axis shows the constraint parameters (α for FairExpec, FairExpecGrp, and Thrsh, and λ for MultObj); $\mathcal{K}$ values are averaged over 200 trials, and the error bars represent the standard error of the mean.



(b) The y-axis shows the risk difference $\mathcal{F}$ of different algorithms, and the x-axis shows the constraint parameters (α for FairExpec, FairExpecGrp, and Thrsh, and λ for MultObj); $\mathcal{F}$ values are averaged over 200 trials, and the error bars represent the standard error of the mean.

Figure 18: *Additional for image selection (Section 4.5):* This simulation considers gender as the protected attribute and uses a CNN-based classifier to derive noisy information about the gender of the person in the image. The utility of a image at the r-th position in the dataset is $(\log(1+r))^{-1}$. We refer the reader to Section 4.5 for the details of the simulation.



(a) The y-axis shows the NDCG value of different algorithms, and the x-axis shows selection lift ($\mathcal{F}_L$); both values are averaged over 100 trials.

Figure 19: *Additional figure for Section 4.5:* This simulation considers gender as the protected attribute and uses a CNN-based classifier to derive noisy information about the gender of the person in the image. The utility of a image at the r-th position in the dataset is $(\log(1+r))^{-1}$. The y-axis shows the NDCG value of different algorithms, and the x-axis shows selection lift ($\mathcal{F}_L$); both values are averaged over 100 trials. We refer the reader to Section 4.5 for details of the simulation. We observe similar results to those with utility ratio ($\mathcal{K}$) (see Figure 5).

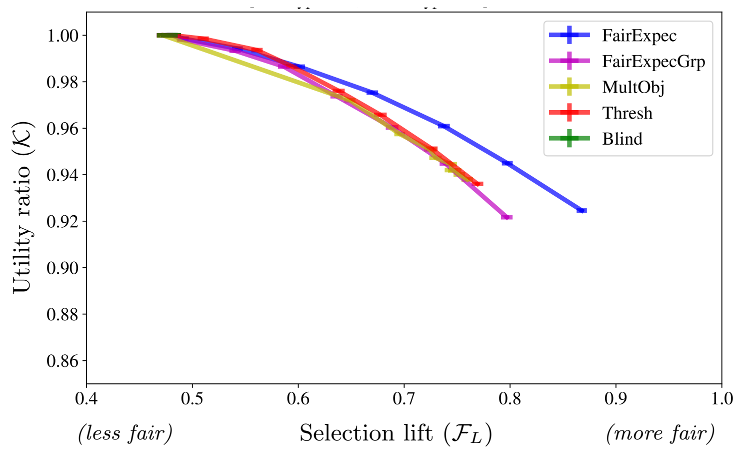


Figure 20: *Additional results for Section 4.5 with selection lift:* The y-axis shows the utility ratio ($\mathcal{K}$) of different algorithms, and the x-axis shows the selection lift ($\mathcal{F}_L$) of different algorithms; both $\mathcal{K}$ and $\mathcal{F}_L$ values are averaged over 200 trials, and the error bars represent the standard error of the mean. We refer the reader to Section 4.5 for the details of the simulation. We observe similar results as with risk difference ($\mathcal{F}$). See Equation (24) for a definition of selection lift ($\mathcal{F}_L$).