APPENDIX

A ADDITIONAL PRELIMINARIES

Ensuring Pareto-optimality: A major challenge when formulating fair algorithms is that those may generate Pareto-dominated solutions [78]. That is, those may produce a fair solution that are worse for all groups, including minorities, compared to the unfair ones. In particular, in a utility assignment setting, let the utility assigned to each group $\mathbf{g}_i$ by the fair algorithm be u_i. There may exist another (unfair) assignment that assigns $u'_i > u_i$, $\forall \mathbf{g}_i \in \mathcal{G}$. This usually can happen when fairness is defined as the parity between different groups, without further specifications.

In this paper, the requirements 2 and 3 (single and pairwise fairness) have been specifically defined in a way to prevent generating Pareto-dominated fair solutions. To better explain the rational behind our definitions, let us consider pairwise fairness (the third requirement). Only requiring equal collision probability between various groups, the fairness constraint would translate to $Pr[h(p_i) = h(q_i)] = Pr[h(p_k) = h(q_k)]$, where p_i and q_i belong to group $\mathbf{g}_i$. Now let us consider a toy example with two groups $\{\mathbf{g}_1, \mathbf{g}_2\}$, two buckets $\{b_1, b_2\}$, and n points where half belong to $\mathbf{g}_1$. Let the hashmap $\mathcal{H}$ map each point $p \in \mathbf{g}_1$ to b_1 and each point $q \in \mathbf{g}_2$ to b_2. In this example, the collision probability between a random pair of points is 0.5, simply because each bucket contains half of the points. It also satisfies the collision probability equality between pairs of points from the same groups: $Pr[h(p_i) = h(q_i)] = Pr[h(p_k) = h(q_k)] = 1$.

This, however, is the worst assignment for both groups as their pairs *always collide*. In other words, it is fair, in a sense that it is equally bad for both groups. Any other hashmap would have a smaller collision probability for both groups and, even if not fair, would be more beneficial for both groups. In other words, this is a fair but Pareto-dominated solution.

In order to ensure Pareto-optimality while developing fair hashmaps, in requirements 2 and 3 (single and pairwise fairness), not only we require the probabilities to be equal, but also we require them to be equal to the *best case*, where the collision probability is $\frac{1}{m}$. As a result, we guarantee that (1) our hashmap is fair and (2) no other hashmap can do better for any of the groups.

B EXTENDED DETAILS OF DISCREPANCY-BASED HASHMAPS

B.1 Faster randomized algorithm for small m

The dynamic programming algorithm we designed depends on n^{d+2}, which is too large. Here we propose a faster randomized algorithm to return an $((1+\delta)\varepsilon_D + \gamma, 1)$-hashmap, with running time which is strictly linear on n. We use the notion of γ-approximation. Let w be a vector in $\mathbb{R}^d$, and let P_w be the projections of all points in P onto w. We also use P_w to denote the ordering of the points on w. Let $\mathcal{I}_w$ be the set of all possible intervals on the line supporting w. Let $(P_w, \mathcal{I}_w)$ be the range space with elements P_w and ranges $\mathcal{I}_w$. Let A_i be a random (multi-)subset of $\mathbf{g}_i$ of size $O(\gamma^{-2} \log \phi^{-1})$ where at each step a point from $\mathbf{g}_i$ is chosen uniformly at random (with replacement). It is well known [36, 56] that A_i is a γ-approximation of $(\mathbf{g}_i, \mathcal{I}_w)$ with probability at least $1 - \phi$, satisfying $|\frac{|\mathbf{g}_i \cap I|}{|\mathbf{g}_i|} - \frac{|A_i \cap I|}{|A_i|}| \leq \gamma$, for every $I \in \mathcal{I}_w$.

We discretize the range $[0, 1]$ creating the discrete values of discrepancy $E = \{0, \frac{1}{n}, (1+\delta)\frac{1}{n}, \ldots, 1\}$ and run a binary search on E. Let $\alpha \in E$ be the discrepancy we consider in the current iteration of the binary search. We design a randomized algorithm such that, if $\alpha > \varepsilon_D$ it returns a valid hashmap. Otherwise, it does not return a valid hashmap. The algorithm returns the correct answer with probability at least $1 - 1/n^{O(1)}$.

Algorithm. For each i, we get a sample set $A_i \subseteq \mathbf{g}_i$ of size $O(\frac{m^2}{\gamma^2} \log n^{d+2})$. Let $A = \cup_i A_i$. Then we run the binary search we described in the previous paragraph. Let α be the discrepancy we currently check in the binary search. Let $\mathcal{W}_A$ be the set of all combinatorially different vectors with respect to A as defined in Section 3. For each $w \in \mathcal{W}_A$ we define the ordering A_w. Using a straightforward dynamic programming algorithm we search for a partition of m buckets on A_w such that every bucket contains between $[(1 - \alpha - \gamma/2)\frac{|A_i|}{m}, (1 + \alpha + \gamma/2)\frac{|Ai|}{m}]$ points for each $\mathbf{g}_i$. If it returns one such partition, then we continue the binary search for smaller values of α. Otherwise, we continue the binary search for larger values of α. In the end, we return the partition/hashmap of the last vector w that the algorithm returned a valid hashmap.

Analysis. Let w be a vector in $\mathbb{R}^d$ and the ordering P_w. Let $\mathcal{I}_w$ be the intervals defined on the line supporting the vector w. By definition, A_i is a $\gamma/(2m)$-approximation of $(P_w, \mathcal{I}_w)$ with probability at least $1 - 1/n^{d+2}$. Since there are $k < n$ groups and $O(n^d)$ combinatorially different vectors (with respect to P) using the union bound we get the next lemma.

Lemma 4. *A_i is a $\gamma/(2m)$-approximation in $(P_w, \mathcal{I}_w)$ for every $w \in \mathbb{R}^d$ and every $i \in [1, k]$ with probability at least $1 - 1/(2n)$.*

Let w^* be a vector in $\mathbb{R}^d$ such that there exists an ε_D-discrepancy partition of P_{w^*}. Let w' be the vector in $\mathcal{W}_A$ that belongs in the same cell of the arrangement of $\mathcal{A}(\Lambda_A)$ with w^*, where Λ_A is the set of dual hyperplanes of A. We notice that the ordering $A_{w'}$ is exactly the same with the ordering A_{w^*}.

By definition, there exists a partition of m buckets/intervals on P_{w^*} such that each interval contains $[(1 - \varepsilon_D)\frac{|\mathbf{g}_i|}{m}, (1 + \varepsilon_D)\frac{|\mathbf{g}_i|}{m}]$ items from group $\mathbf{g}_i$, for every group $\mathbf{g}_i$. Let $I_1, I_2, \ldots I_m$ be the m intervals defining the ε_D-discrepancy partition. By the definition of A, it holds that

$$|\frac{|\mathbf{g}_i \cap I_j|}{|\mathbf{g}_i|} - \frac{|A_i \cap I_j|}{|A_i|}| \leq \gamma/(2m) \Leftrightarrow$$

$$\frac{1 - \varepsilon_D - \gamma/2}{m} \leq \frac{|A_i \cap I_j|}{|A_i|}| \leq \frac{1 + \varepsilon_D + \gamma/2}{m}$$

for every $i \in [1, k]$ and $j \in [1, m]$ with probability at least $1 - 1/n^d$. Since, the ordering of A_{w*} is the same with the ordering of $A_{w'}$ it also holds that there are $I'_1, \ldots, I'_m$ on the line supporting w' such that

$$\frac{1 - \varepsilon_D - \gamma/2}{m} \leq \frac{|A_i \cap I'_j|}{|A_i|}| \leq \frac{1 + \varepsilon_D + \gamma/2}{m}$$

with probability at least $1 - 1/n^d$. Let $\alpha \in E$ be the discrepancy we currently check in the binary search such that $\alpha > \varepsilon_D$. Since there exists a $(\varepsilon_D + \gamma/2)$-discrepancy partition in $A_{w'}$ it also holds that the straightforward dynamic programming algorithm executed with respect to vector w' will return a partition satisfying $[(1 - \alpha - \gamma/2)\frac{|A_i|}{m}, (1 + \alpha + \gamma/2)\frac{|A_i|}{m}]$ for each $\mathbf{g}_i$ with probability at least $1 - 1/n^d$.

Next, we show that any hashmap returned by our algorithm satisfies $(1 + \delta)\varepsilon_D + \gamma$ discrepancy, with high probability. Let $\alpha \in E$ be the discrepancy in the current iteration of the binary search such that $\alpha \in [\varepsilon_D, (1 + \delta)\varepsilon_D]$ (there is always such α in E). Let $\bar{w} \in \mathcal{W}_A$ be the vector such that our algorithm returns a valid partition with respect to $A_{\bar{w}}$ (as shown previously). Let $\bar{I}_1, \ldots, \bar{I}_m$ be the intervals on the returned valid partition defined on the line supporting $\bar{w}$ such that

$$(1 - \alpha - \gamma/2)\frac{|A_i|}{m} \leq |\bar{I}_j \cap A_i| \leq (1 + \alpha + \gamma/2)\frac{|A_i|}{m}, \tag{2}$$

for each $\mathbf{g}_i$ and $j \in [1, m]$. Using Lemma 4, we get that

$$|\frac{|\mathbf{g}_i \cap \bar{I}_j|}{|\mathbf{g}_i|} - \frac{|A_i \cap \bar{I}_j|}{|A_i|}| \leq \gamma/(2m) \tag{3}$$

for every $i \in [1, k]$ and every $j \in [1, m]$ with probability at least $1 - 1/(2n)$. From Equation (2) and Equation (3) it follows that

$$\frac{1 - (1 + \delta)\varepsilon_D - \gamma}{m} \leq \frac{|\mathbf{g}_i \cap \bar{I}_j|}{|\mathbf{g}_i|} \leq \frac{1 + (1 + \delta)\varepsilon_D + \gamma}{m}.$$

for every $i \in [1, k]$ and $j \in [1, m]$ with probability at least $1 - 1/n$. The correctness of the algorithm follows.

We construct A in $O(\frac{km^2}{\gamma^2} \log n)$ time. Then the binary search runs for $O(\log \frac{\log n}{\delta})$ rounds. In each round of the binary search we spend $O(\frac{k^d m^{2d}}{\gamma^{2d}} \log^{d+1} n)$ time to construct $\mathcal{A}(\Lambda_A)$. The straightforward dynamic programming algorithm takes $O(\frac{k^2 m^5}{\gamma^4} \log^2 n)$ time. Overall, the algorithm runs in $O(n + \log(\frac{\log n}{\delta})\frac{k^{d+2} m^{2d+5}}{\gamma^{2d+4}} \log^{d+2} n)$ time.

Theorem 5. *Let P be a set of n tuples in $\mathbb{R}^d$ and parameters δ, γ. There exists an $O(n + \log(\frac{\log n}{\delta})\frac{k^{d+2} m^{2d+5}}{\gamma^{2d+4}} \log^{d+2} n)$ time algorithm such that, with probability at least $1 - \frac{1}{n}$ it returns an $((1+\delta)\varepsilon_D + \gamma, 1)$-hashmap $\mathcal{H}$ with collision probability at most $\frac{1+(1+\delta)\varepsilon_D+\gamma}{m}$ and single fairness in the range $[\frac{1-(1+\delta)\varepsilon_D-\gamma}{m}, \frac{1+(1+\delta)\varepsilon_D+\gamma}{m}]$.*

B.2 Cut-based

We use a direct implementation of the necklace splitting algorithm proposed in [10] for $k > 2$. Even though the authors argue that their algorithm runs in polynomial time, they do not provide an exact running time analysis. We slightly modify their technique to work in our setting providing theoretical guarantees on the collision probability, single fairness, pairwise fairness, and pre-processing construction time. Skipping the details we give the next result.

Theorem 6. *In the non-binary demographic group cases, there exists an algorithm that finds a $\left(\varepsilon, k(4 + \log \frac{1}{\varepsilon})\right)$-hashmap with collision probability within $[\frac{1}{m}, \frac{1+\varepsilon}{m}]$ and single fairness within $[\frac{1-\varepsilon}{m}, \frac{1+\varepsilon}{m}]$ in $O(mk^3 \log \frac{1}{\varepsilon} + knm(n + m))$ time. If $\varepsilon = \frac{1}{3nm}$, the algorithm finds a $\left(0, k(4 + \log n)\right)$-hashmap satisfying the collision probability and the single fairness in $O\left(mk^3 \log n + knm(n + m)\right)$ time.*

C MISSING PROOFS

Lemma 1. *While CDF-based hashmap satisfies collision probability, hence single fairness, it may not satisfy pairwise fairness.*

Proof: Let n_j be the number of tuples in the j-th bucket, and let $\alpha_{i,j}$ be the number of tuples of group $\mathbf{g}_i$ in the j-th bucket for every $i \leq k$ and $j \leq m$. From [66], it is always the case that each bucket contains the same number of tuples, i.e., $\frac{n_j}{n} = \frac{1}{m}$ for every $j = 1, \ldots, m$. Hence, the collision probability is $\sum_{j=1}^{m} \left(\frac{n_j}{n}\right)^2 = \sum_{j=1}^{m} \frac{1}{m^2} = \frac{1}{m}$. Furthermore, the single fairness for each group $\mathbf{g}_i$ is $\sum_{j=1}^{m} \frac{\alpha_{i,j}}{|\mathbf{g}_i|} \cdot \frac{n_j}{n} = \frac{1}{m \cdot |\mathbf{g}_i|} \sum_{j=1}^{m} \alpha_{i,j} = \frac{1}{m}$. Hence, the CDF-based hashmap satisfies both the collision probability and the single fairness.

Next, we show that CDF-based hashmap does not always satisfy pairwise fairness using a counter-example. Let $k = 2$, $m = 2$, $|\mathbf{g}_1| = 3$, $|\mathbf{g}_2| = 3|$, and $n = 6$. Assume the 1-dimensional tuples $\{1, 2, 3, 4, 5, 6\}$ where the first 3 of them belong to $\mathbf{g}_1$ and the last three belong to $\mathbf{g}_2$. The CDF-based hashmap will place the first three tuples into the first bucket and the last three tuples into the second bucket. By definition, the pairwise fairness is 1 (instead of 1/2) for both groups. □

Lemma 2. *In the binary demographic groups cases, where $\mathcal{G} = \{\mathbf{g}_1, \mathbf{g}_2\}$ and $r = |\mathbf{g}_1|$, the expected number of bins added by Sweep&Cut is bounded above by $2\left(\frac{r(n-r)}{n} + m\right)$.*

Proof: Sweep&Cut adds at most $m - 1$ boundaries between the neighboring pairs that both belong to the same group, simply because a boundary between neighboring pair can only happen when moving from one bucket to the next while there are m buckets.

In order to find the upper-bound on the number of bins added, in the following we compute the expected number of neighboring pairs from different groups. Consider the in the sorted list of points $\langle p_1, \cdots, p_n \rangle$ based on the attribute x. The probability that a point p_i belongs to group g_1 is

$$Pr_1 = Pr(\mathbf{g}(p_i) = \mathbf{g}_1) = \frac{r}{n}$$

Now consider two consecutive points p_i, p_{i+1}, in the list. The probability that these two belong to different groups is $2Pr_1(1 - Pr_1)$. Hence, the expected number of such pairs is

$$E[\text{pairs from diff. groups}] = \sum_{i=1}^{n-1} 2Pr_1(1-Pr_1) = 2(n-1)\frac{r}{n}(1-\frac{r}{n}) < \frac{2r(n-r)}{n}$$

Therefore,

$$E[\text{No. bins}] \leq E[\text{pairs from diff. groups}] + 2m < 2\left(\frac{r(n-r)}{n} + m\right)$$

□

Lemma 3. *Let $\mathcal{H}$ be a hashmap satisfying γ-discrepancy. Then $Cp \leq \frac{1+\gamma}{m}$, and $\frac{1-\gamma}{m} \leq Sp_i \leq \frac{1+\gamma}{m}$ and $Pr_i \leq \frac{1+\gamma}{m}$ for each group $\mathbf{g}_i$.*

Proof: Let n_j be the number of items in bucket j and let $\alpha_{i,j}$ be the number of items from group i in bucket j. Notice that $\sum_{j=1}^{m} \alpha_{i,j} = |\mathrm{g}_i|$ and $\sum_{j=1}^{m} n_j = n$. We have

$$Pr_i = \sum_{j=1}^{m} \left(\frac{\alpha_{i,j}}{|\mathrm{g}_i|}\right)^2 = \sum_{j=1}^{m} \frac{\alpha_{i,j}}{|\mathrm{g}_i|} \cdot \frac{\alpha_{i,j}}{|\mathrm{g}_i|} \leq \sum_{j=1}^{m} \frac{(1+\gamma)\frac{\mathrm{g}_i}{m}}{|\mathrm{g}_i|} \cdot \frac{\alpha_{i,j}}{|\mathrm{g}_i|}$$
$$= \frac{1+\gamma}{m} \sum_{j=1}^{m} \frac{\alpha_{i,j}}{|\mathrm{g}_i|} = \frac{1+\gamma}{m}.$$

Similarly we show that

$$Cp = \sum_{j=1}^{m} \left(\frac{n_j}{n}\right)^2 \leq \frac{1+\gamma}{m}.$$

Using the same arguments it also holds that

$$Sp_i = \sum_{j=1}^{m} \frac{\alpha_{i,j}}{|\mathrm{g}_i|} \frac{n_j}{n} \leq \frac{1+\gamma}{m}$$

and

$$Sp_i \geq \frac{1-\gamma}{m}.$$

□

D EXTENDED EXPERIMENT RESULTS

D.1 Datasets

Adult [23] is a commonly used dataset in the fairness literature [24], with column `sex={male, female}` as the sensitive attribute and `fnlwgt` and `education-num` columns with continuous real and discrete ordinal values for ordering the tuples. Adult dataset includes 15 census data related attributes with ~49000 records describing people and is primarily used to predict whether an individual's income exceeds $50K per year.

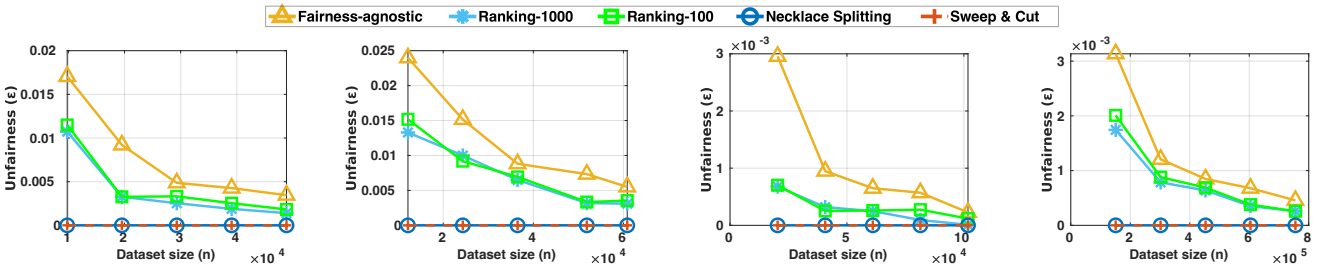
Compas [1] is another benchmark dataset widely used in the fairness literature. This dataset includes ~61000 records of defendants from Broward County from 2013 and 2014 with 29 attributes including demographics and criminal history. Compas was originally used to predict the likelihood of re-offense by the criminal defendants. In our experiments, we picked `sex={male, female}` and `race={White, Black, Hispanic, other}` as the sensitive attributes and `Person_ID` and `Raw_Score` with discrete ordinal and continuous real values as the columns used for building the hashmap.

Diabetes [96] is a dataset of ~102K records of patients with diabetes collected over 10 years from 130 US hospitals and integrated delivery networks. Diabetes has 49 attributes from which we chose `sex={male, female}` as the sensitive attribute and `encounter_id` and `patient_nbr` columns with discrete ordinal values for constructing the hashmap.

ChicagoPop [77] is a semi-synthetic dataset used to simulate individual-level data with demographic information for the city of Chicago. This dataset is available in various sizes and in our experiments we use the variant with 1M entries. It has 5 attributes among which we use `race={White, Black}` as the sensitive attribute and `latitude` and `longitude` columns with continuous real values for building the hashmap.

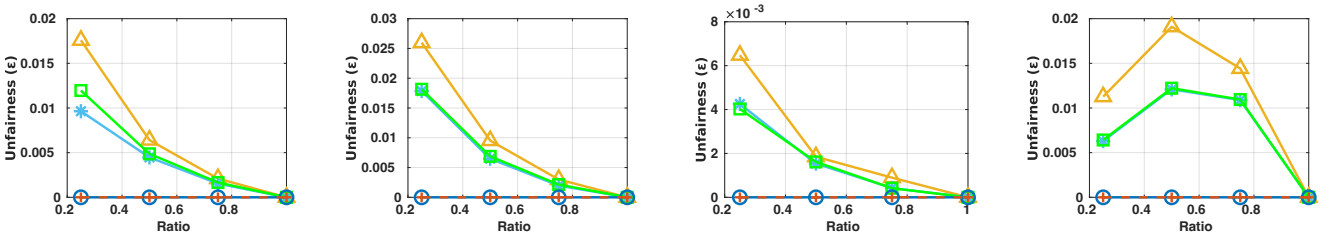
D.2 Non-binary Sensitive Attribute Evaluation

We repeated our experiments for Ranking and Sweep&Cut algorithms using the Compas dataset, with the non-binary sensitive attribute chosen as `race`. The results, as illustrated in Figures 45-51, align with the findings from our earlier experiments. Consistent with our expectations, our algorithms exhibit independence from the number of demographic groups and seamlessly extend to non-binary sensitive attributes without compromising efficiency or memory requirements.



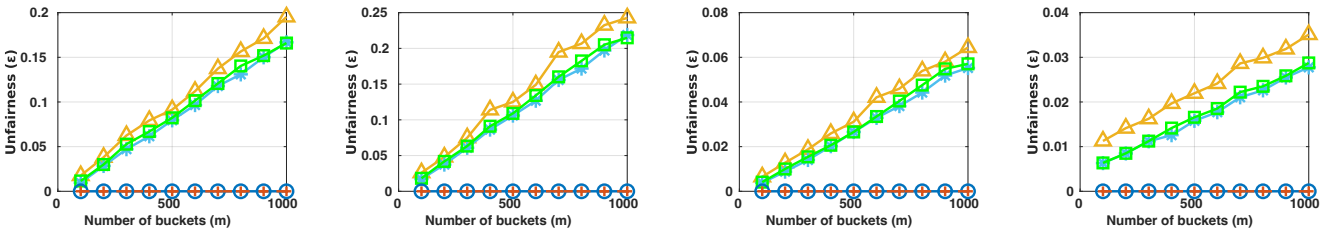
(a) Adult (b) Compas (c) Diabetes (d) ChicagoPop

Figure 33: Effect of varying dataset size n on unfairness



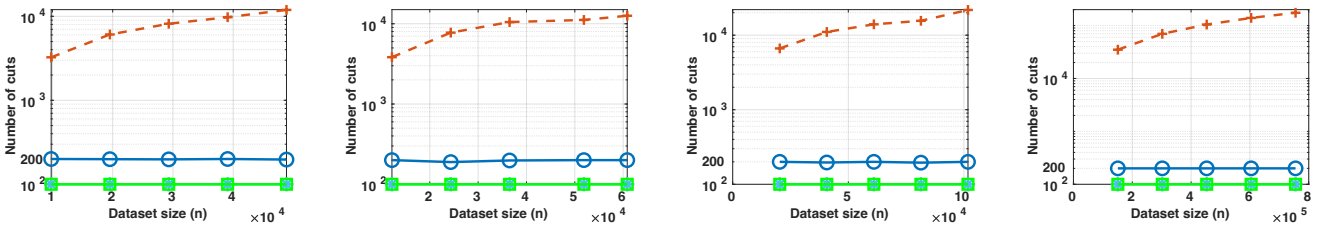
(a) Adult (b) Compas (c) Diabetes (d) ChicagoPop

Figure 34: Effect of varying majority to minority ratio on unfairness



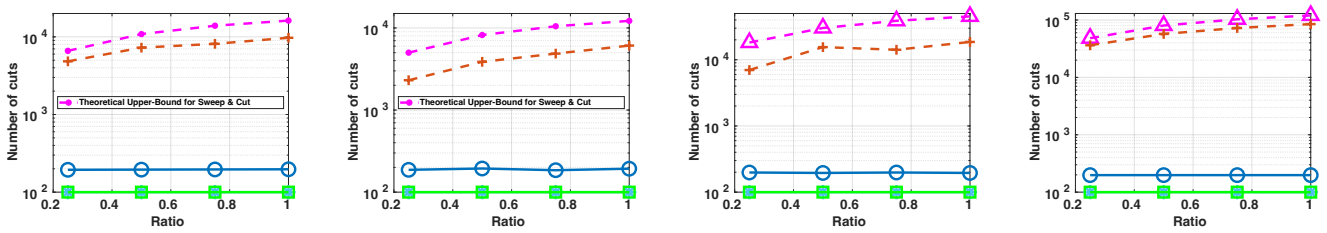
(a) Adult (b) Compas (c) Diabetes (d) ChicagoPop

Figure 35: Effect of varying number of buckets m on unfairness



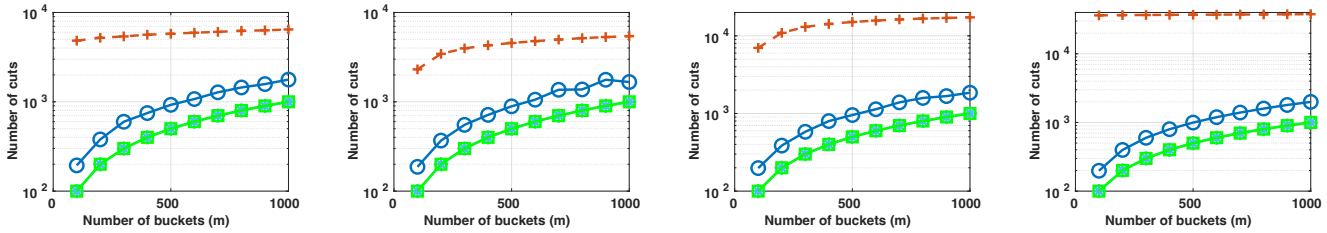
(a) Adult (b) Compas (c) Diabetes (d) ChicagoPop

Figure 36: Effect of varying dataset size n on space



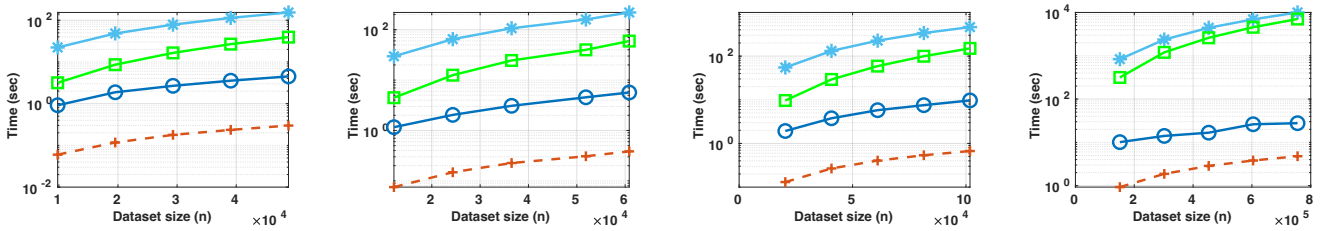
(a) Adult (b) Compas (c) Diabetes (d) ChicagoPop

Figure 37: Effect of varying majority to minority ratio on space



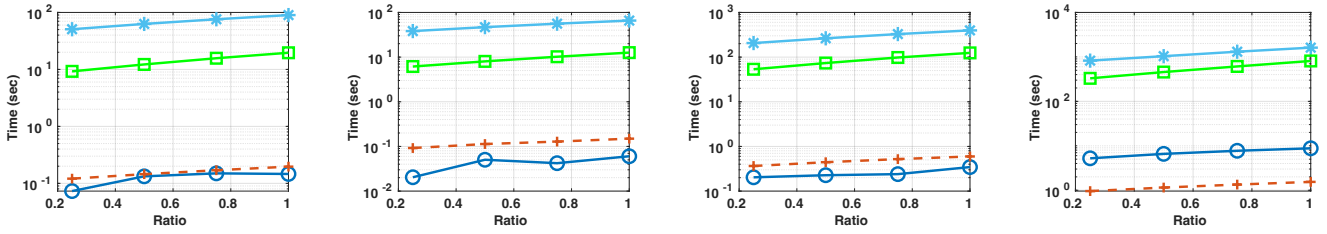
(a) Adult (b) Compas (c) Diabetes (d) ChicagoPop

Figure 38: Effect of varying number of buckets m on space



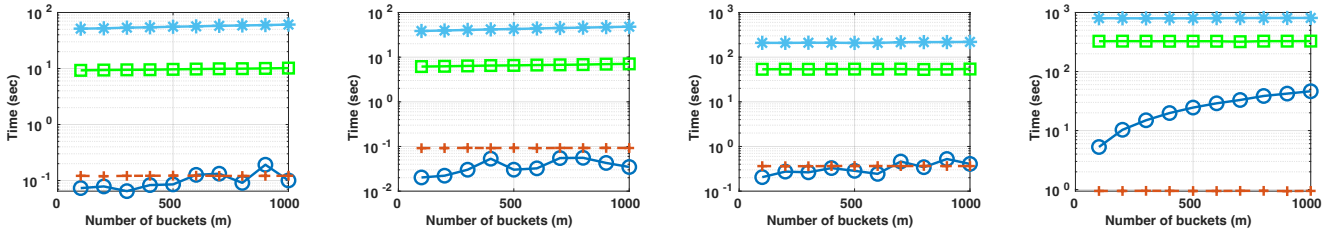
(a) Adult (b) Compas (c) Diabetes (d) ChicagoPop

Figure 39: Effect of varying dataset size n on preprocessing time



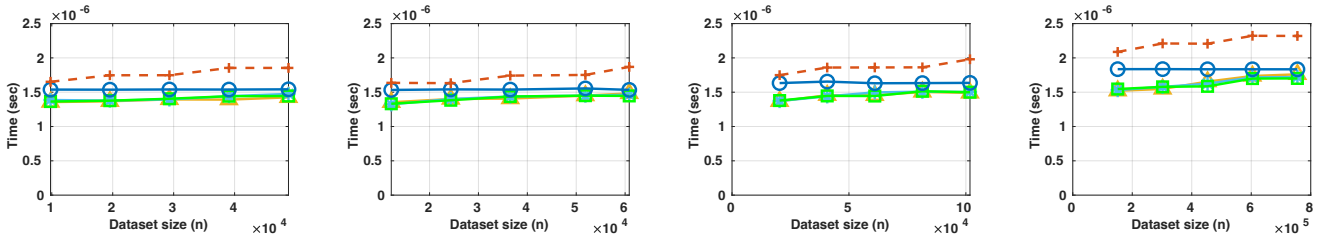
(a) Adult (b) Compas (c) Diabetes (d) ChicagoPop

Figure 40: Effect of varying majority to minority ratio on preprocessing time



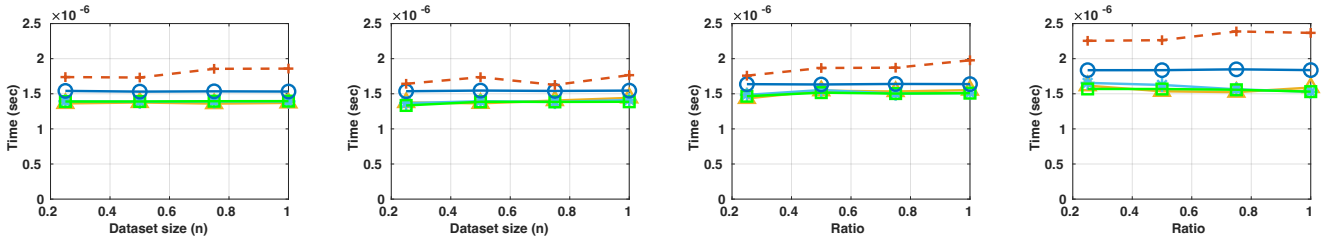
(a) Adult (b) Compas (c) Diabetes (d) ChicagoPop

Figure 41: Effect of varying number of buckets m on preprocessing time



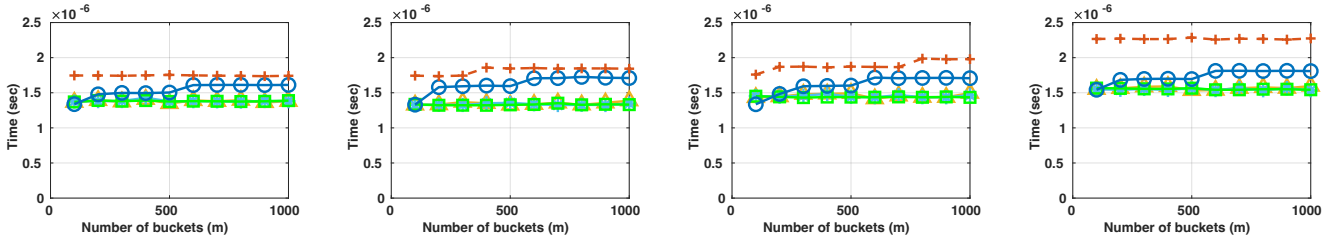
(a) Adult (b) Compas (c) Diabetes (d) ChicagoPop

Figure 42: Effect of varying dataset size n on query time



(a) Adult (b) Compas (c) Diabetes (d) ChicagoPop

Figure 43: Effect of varying majority to minority ratio on query time



(a) Adult (b) Compas (c) Diabetes (d) ChicagoPop

Figure 44: Effect of varying number of buckets m on query time

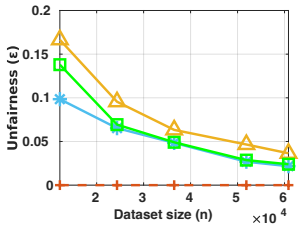


Figure 45: Effect of varying dataset size n on unfairness, COMPAS, race

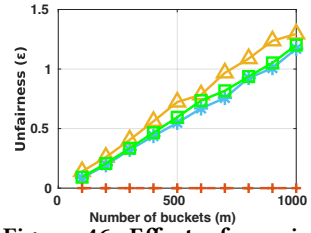


Figure 46: Effect of varying number of buckets m on unfairness, COMPAS, race

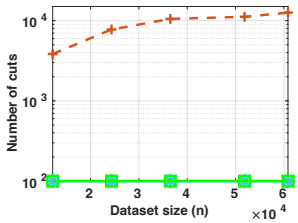


Figure 47: Effect of varying dataset size n on space, COMPAS, race

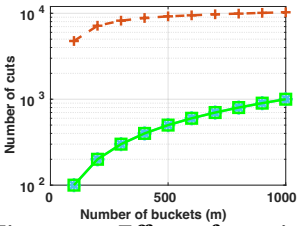


Figure 48: Effect of varying number of buckets m on space, COMPAS, race

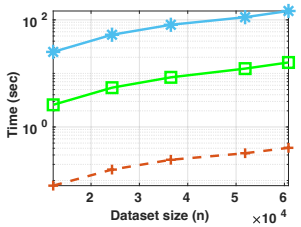


Figure 49: Effect of varying dataset size n on preprocessing time, COMPAS, race

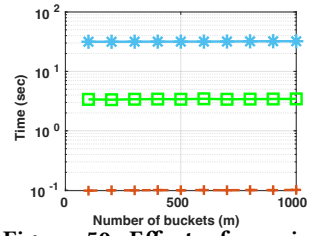


Figure 50: Effect of varying number of buckets m on preprocessing time, COMPAS, race

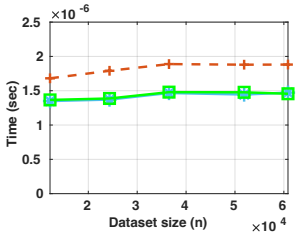


Figure 51: Effect of varying dataset size n on query time, COMPAS, race

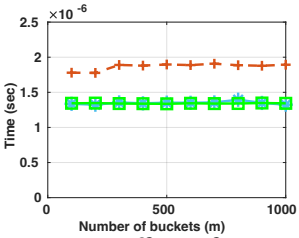


Figure 52: Effect of varying number of buckets m on query time, COMPAS, race

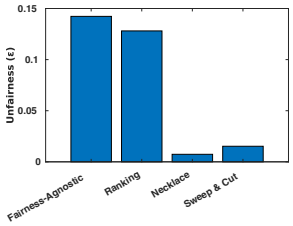


Figure 53: Learning setting: Unfairness evaluation over held out data, ADULT

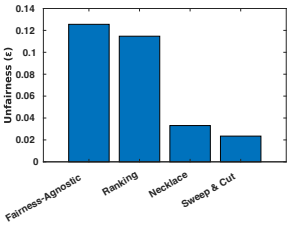


Figure 54: Learning setting: Unfairness evaluation over held out data, COMPAS

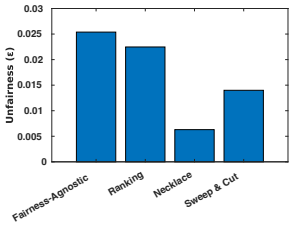


Figure 55: Learning setting: Unfairness evaluation over held out data, DIABETES

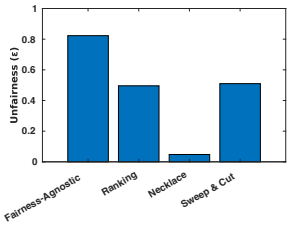


Figure 56: Learning setting: Unfairness evaluation over held out data, CHICAGOPOP

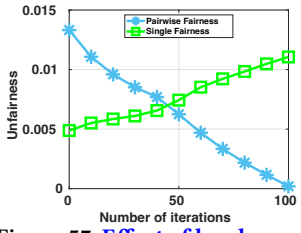


Figure 57: Effect of local search based algorithm on unfairness, ADULT

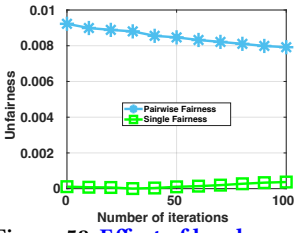


Figure 58: Effect of local search based algorithm on unfairness, COMPAS

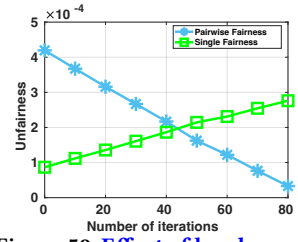


Figure 59: Effect of local search based algorithm on unfairness, DIABETES

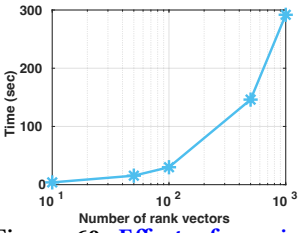


Figure 60: Effect of varying number of sampled vectors on preprocessing time, DIABETES

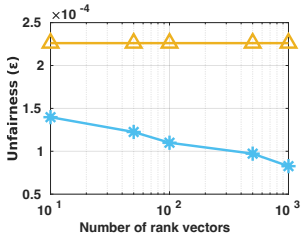


Figure 61: Effect of varying number of sampled vectors on unfairness, DIABETES

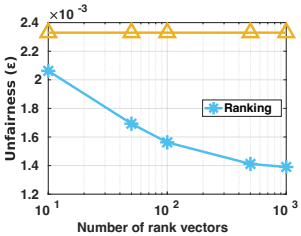


Figure 62: Effect of varying number of sampled vectors on unfairness, ADULT

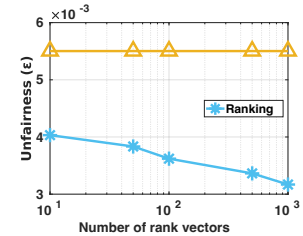


Figure 63: Effect of varying number of sampled vectors on unfairness, COMPAS, sex

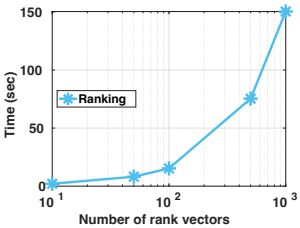


Figure 64: Effect of varying number of sampled vectors on preprocessing time, COMPAS