Learning to Generate Fair Clusters from Demonstrations

Sainyam Galhotra
University of Massachusetts Amherst
sainyam@cs.umass.edu

Sandhya Saisubramanian
University of Massachusetts Amherst
saisubramanian@cs.umass.edu

Shlomo Zilberstein
University of Massachusetts Amherst
shlomo@cs.umass.edu

ABSTRACT

Fair clustering is the process of grouping similar entities together, while satisfying a mathematically well-defined fairness metric as a constraint. Due to the practical challenges in precise model specification, the prescribed fairness constraints are often incomplete and act as proxies to the intended fairness requirement. Clustering with proxies may lead to biased outcomes when the system is deployed. We examine how to identify the intended fairness constraint for a problem based on limited demonstrations from an expert. Each demonstration is a clustering over a subset of the data. We present an algorithm to identify the fairness metric from demonstrations and generate clusters using existing off-the-shelf clustering techniques, and analyze its theoretical properties. To extend our approach to novel fairness metrics for which clustering algorithms do not currently exist, we present a greedy method for clustering. Additionally, we investigate how to generate interpretable solutions using our approach. Empirical evaluation on three real-world datasets demonstrates the effectiveness of our approach in quickly identifying the underlying fairness and interpretability constraints, which are then used to generate fair and interpretable clusters.

CCS CONCEPTS

• **Computing methodologies** → **Cluster analysis**; • **Mathematics of computing** → Maximum likelihood estimation.

KEYWORDS

Clustering; Fairness; Interpretability; Maximum-likelihood estimation

ACM Reference Format:
Sainyam Galhotra, Sandhya Saisubramanian, and Shlomo Zilberstein. 2021. Learning to Generate Fair Clusters from Demonstrations. In *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society (AIES '21), May 19–21, 2021, Virtual Event, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3461702.3462558

1 INTRODUCTION

Graph clustering is increasingly used for decision making in high-impact applications such as infrastructure development [27], health care [25], and criminal justice [4]. These domains involve highly consequential decisions and it is important to ensure that the generated solutions are unbiased. Fair clustering is the process by which similar nodes are grouped together, while satisfying a given fairness constraint [14]. Prior works on fair clustering focus on designing efficient algorithms to satisfy a given fairness metric [3, 5, 14, 20, 30]. These approaches *assume* that the specified fairness metric is complete and accurate. With the increased growth in the number of ways to define and measure fairness, a key challenge for system designers is to accurately specify the fairness metric for a problem.

Due to the practical challenges in the precise specification of fairness metrics and the complexity of machine learning models, the system's objective function and constraints are often tweaked in the development phase until it produces the desired behavior on a small subset of the data. This approach may result in inadvertent, incomplete specification of fairness metric that acts as a *proxy* to the intended metric. Clustering with incompletely specified fairness metrics may lead to undesirable consequences when deployed. It is challenging to identify the proxies during system design due to the nuances in the fairness definitions and unstated assumptions. Two similar fairness metrics that produce similar solutions during the design and initial testing may generate different solutions that are unfair in different ways to different groups, when deployed.

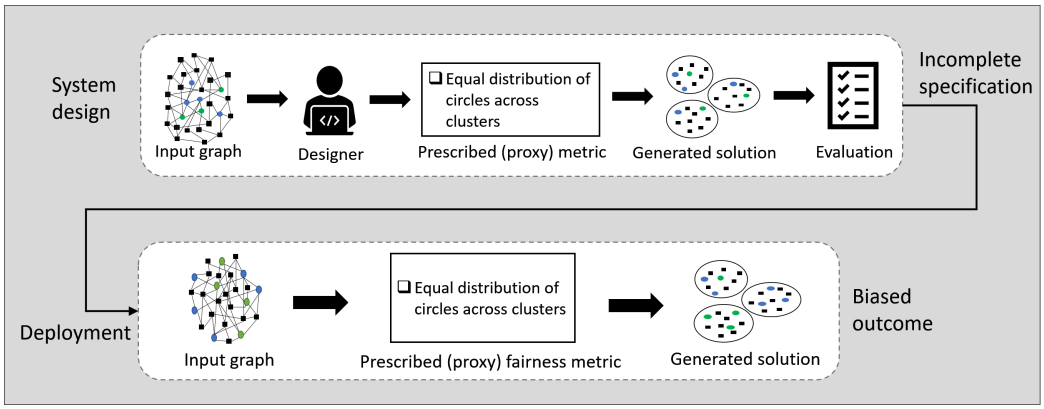
For example in Figure 1, the designer inadvertently specifies an incomplete fairness metric and assumes the system will behave as intended when deployed. This unintentional incomplete specification is not discovered during the initial testing since the generated results align with that of the intended metric on the training data, such as sample data from one city. Consequently, the system may generate biased solution when deployed in a different region, due to demographic shift. Thus, design decisions that seem innocuous during initial testing may have harmful impacts when the system is widely deployed. While the difficulty in selecting a fairness metric for a given problem is acknowledged [31], there exists no principled approach to address this meta-problem. *How to correctly identify the fairness metric that the designer intends to optimize for a problem?*

We present an approach that generates fair clusters by learning to identify the intended fairness metric using limited demonstrations from an oracle. It is assumed that there exists a true clustering with the intended fairness metrics, which are initially unknown. Each demonstration is a sample from the true clusters, providing information about a subset of the nodes in the dataset. Given a finite number of expert demonstrations, our solution approach first clusters the demonstrations to infer the likelihood of each candidate constraint and then generates clusters using the most likely constraint. By maintaining a distribution over the candidate metrics and updating it based on the demonstrations, the intended clusters can be recovered since demonstrations are i.i.d. The nodes in a demonstration are selected by the expert, abstracted as an oracle. This is in contrast to querying an oracle where the algorithm selects the nodes to query and the oracle responds if they belong to the same cluster or not. When the oracle is a human, demonstrations

AIES '21, May 19–21, 2021, Virtual Event, USA

ACM ISBN 978-1-4503-8473-5/21/05…$15.00
https://doi.org/10.1145/3461702.3462558

(a) Ideal setting



(b) Biased outcome due to incomplete specification

Figure 1: An illustration of incomplete specification of fairness metric leading to biased output—unequal distribution of green and blue nodes in each cluster—when deployed.

are easier to collect rather than querying for pairs of nodes, which require constant oversight.

While inferring the intended fairness metric is critical to minimize the undesirable behavior of the system, the ability of an end user to evaluate a deployed system for fairness and identify when to trust the system hinges on the interpretability of the results. Though clustering results are expected to be inherently interpretable, the end user may not be able to identify clear patterns when clustering with a large number of features [38]. While the existing literature has studied fair clustering and interpretable clustering independently [14, 38], to the best of our knowledge, there exists no approach to generate clusters that are both fair and interpretable. We show that our solution approach can generate *fair and interpretable* clusters by inferring both fairness and interpretability constraints, based on limited demonstrations.

Our primary contributions are as follows: (1) formalizing the problem of learning to generate fair clusters from demonstrations (Section 3); (2) presenting two algorithms to identify the fairness constraints for clustering, generate fair clusters, and analyzing their theoretical guarantees (Sections 4 and 5); and (3) empirically demonstrating the effectiveness of our approach in identifying the clustering constraints on three data sets, and using our approach to generate fair and interpretable clusters (Section 6).

2 BACKGROUND AND RELATED WORK

We present a brief overview of centroid-based clustering, fairness in machine learning, and learning from human demonstrations.

K-center Clustering. It is one of the most widely studied objectives in the clustering literature [42]. Let $H = G(V, d)$ be a graph with $V = \{v_1, v_2, \ldots, v_n\}$ denoting a set of n nodes, along with a pairwise distance metric $d : V \times V \to \mathbb{R}$. The nodes are described by features values, F. Given a graph instance H and an integer k, the goal is to identify k nodes as cluster centers (say S, $|S| = k$) and assign each node to the cluster center such that the maximum distance of any node from its cluster center is minimized. The output is a set of clusters $C = \{C_1, C_2, \ldots, C_k\}$. The clustering assignment function is defined by $\gamma : V \to [k]$ and the nodes assigned to a cluster C_i are $\{v \in V | \gamma(v) = i\}$. The objective value is calculated as:

$$o_{kC}(H, C) = \max_{v \in V} \min_{s \in S} d(v, s).$$

A simple greedy algorithm provides a 2-approximation for the k-center problem and it is NP-hard to find a better approximation factor [42]. The greedy algorithm selects the first center randomly and iteratively chooses the farthest point as the subsequent center.

Fairness in Machine Learning. The existing literature on fairness in machine learning can be broadly categorized into two lines of work: defining notions of fairness and designing fair algorithms. Various notions of fairness have been studied by researchers in different fields such as AI, Economics, Law, Philosophy, and Public Policy [7–9, 14, 20, 36, 41, 43]. The two commonly studied fairness criteria are as follows.

- *Group fairness* ensures the outcome distribution is the same for all groups of interest [14, 20, 43]. This is measured using metrics such as disparate impact [18] and statistical parity [29, 43] including conditional statistical parity, predictive parity, false positive error rates, and false negative error rates.

- *Individual fairness* ensures that any two individuals with the same attributes are not discriminated [5, 17, 28].

There has been an increased focus on studying causal notions of interventional fairness [13, 23, 37, 40] that prohibit the sensitive attributes from affecting the outcome. Given a mathematically well-defined fairness criteria, a fair algorithm produces outputs that are aligned with the given fairness definition. Examples include fair clustering [3, 5, 14, 30], fair ranking [11], and fair voting [10]. Although these works have laid vital ground work to assure fairness in some settings, much of the efforts in designing fair algorithms have focused on the algorithm's performance—efficiency, scalability, and providing theoretical guarantees. There is very little effort, if any, at the *meta-level*: designing algorithms that can identify a suitable fairness metric for a clustering problem, given a set of candidate metrics. There has been recent focus on learning a metric [28] or a representation that ensures fairness with respect to classification tasks [24, 26]. It is not straightforward to extend these fair classification techniques to fair clustering because the setting and objectives are different. This is further complicated by the lack of ground truth and NP-hardness of clustering. Therefore, it is critical to develop techniques to infer metrics for fair clustering.

Fair Clustering. Fair clustering approaches generate clusters that maximize the clustering objective value, while satisfying the given fairness requirement [2, 3, 5, 7, 14, 30]. The commonly considered fairness metrics in clustering are group fairness [14], individual fairness [28, 33], and distributional fairness [5]. These approaches require *exact* specification of fairness metrics a priori and generate fair clusters either by modifying the input graph or use the fairness metrics as constraints and solve it as a linear optimization.

Interpretable Clustering. Interpretable clustering is the process of generating clusters such that it is easy to identify patterns in the data for the end user. A recent approach to generate interpretable clusters maximizes the homogeneity of the nodes in each cluster, with respect to predefined features of interest to the user [38]. The problem is solved as a multi-objective clustering problem where both interpretability and the k-center objective value are optimized. Other notions of interpretable clustering have been studied in [12, 15, 32]. While both fairness and interpretability are typically investigated independently, the ability to evaluate the system for fairness violations often relies on its interpretability.

Clustering with an Oracle. A common existing approach to use additional knowledge from an oracle for clustering involves queries of the form 'do nodes u and v belong to the same cluster?' [6, 19, 21, 34, 35, 44]. Our approach is different from the oracle-based clustering in the following manner. First, in our approach, the oracle selects the nodes and determines what information is revealed. Second, the oracle provides information potentially about a subset of nodes, instead of pairwise relationships.

Learning from Demonstration. Learning from demonstration is a type of apprenticeship learning, where the learner learns by observing an expert (often a human) performing the task [1]. The learner tries to mimic the expert's behavior by observing the demonstrations and generalizing it to unseen situations. Learning from demonstration is a popular approach used to teach robots to complete a task [1] or avoid the negative side effects of their actions [39].

Likelihood Estimation. Maximum likelihood estimation (MLE) is a statistical method to estimate the parameters of a probability distribution by maximizing the likelihood function, such that the observed data are most probable under the assumed model [45]. Intuitively, it is a search problem in the parameter space to identify a set of parameters, for the model, that best fit the observed data. The maximum likelihood estimate is the point in the parameter space that maximizes the likelihood function.

3 PROBLEM FORMULATION

Problem Statement: Let $G = \langle V, d \rangle$ be the input graph with nodes/vertices V and distance metric d, and let o denote the clustering objective. Given a finite set of candidate fairness metrics, denoted by Ω, and a finite set of clustering demonstrations, denoted by Λ, the goal is to identify a fairness metric $\omega_F \in \Omega$ required to be satisfied by the clusters when optimizing objective o.

We present *learning to cluster from demonstrations* (LCD), an approach to infer ω_F using Λ. LCD is introduced and discussed in the context of fair clustering but it is a generic approach that can be used to infer any clustering constraint. LCD can also handle the case of clustering with multiple fairness metrics by simply considering Ω to be the power set over possible candidate metrics.

Clustering demonstrations: LCD relies on the availability of clustering demonstrations by an expert. It is relatively easier to gather demonstrations from a human expert than querying for pairs of nodes, which requires constant oversight or availability to answer the queries.

Definition 1. *A **clustering demonstration** λ provides the inter-cluster and intra-cluster links for a subset of nodes from the dataset $T \subseteq V, |T| \geq 2$, by grouping them according to the underlying objective function and constraints, $\lambda = \{C_1, \ldots, C_t\}$ with each C_i denoting a cluster such that $\cup_i C_i = T$ and $t \leq k$.*

To generate a demonstration, the oracle selects a subset of nodes and then clusters it, in accordance with the true clusters. The following assumption ensures that demonstrations are i.i.d and the expert is not acting as an adversary.

Assumption 1. *The nodes in each demonstration are randomly selected and clustered according to the ground-truth fairness constraints.*

Therefore, a demonstration λ is a sample of the underlying clustering, revealing the relationship between a subset of the nodes. However the relationship between distinct nodes in successive demonstrations is *unknown*. We illustrate this with an example. Consider seven nodes $\{u_1, \ldots, u_7\}$ whose true but initially unknown clustering is $C_1^* = \{u_1, u_2, u_3\}$, $C_2^* = \{u_4, u_5\}$, and $C_3^* = \{u_6, u_7\}$. Let $\lambda_1 = \{(u_1, u_2), (u_4)\}$ and $\lambda_2 = \{(u_3), (u_5), (u_6)\}$ denote two successive demonstrations. Demonstration λ_1 shows that u_1, u_2 are in the same cluster, and u_4 is in a separate cluster. Demonstration λ_2 shows that u_3, u_5 and u_6 are in different clusters. At the end of λ_1 and λ_2, it is not clear whether u_1, u_2 and u_3 belong to the same cluster.

Definition 2. ***Globally informative demonstration** provides the true cluster affiliation of a subset of nodes, $T \subseteq V$, and is denoted*

by $\lambda_g = \{\langle u_1, \gamma(u_1)\rangle, \ldots, \langle u_t, \gamma(u_t)\rangle\}$, $\forall u_i \in T$ *with* $\gamma(u)$ *indicating the cluster affiliation of node* u.

Globally informative demonstration provides information about the true cluster affiliation (cluster ID) of the nodes, which is used to retrieve the inter-cluster and intra-cluster links between the nodes and form clusters $\{C_1, \ldots, C_t\}$ with $t \leq k$. The information provided by a *single* globally informative demonstration is the same as a regular clustering demonstration. However, globally informative demonstrations facilitate cross-referencing the cluster affiliations across demonstrations, overcoming the drawback of general clustering demonstration. Consider the example with global demonstrations $\lambda_1 = \{\langle u_1, 1\rangle, \langle u_2, 1\rangle, \langle u_4, 2\rangle\}$ and $\lambda_2 = \{\langle u_3, 1\rangle, \langle u_5, 2\rangle, \langle u_6, 3\rangle\}$. Then we know that $C_1^* = \{u_1, u_2, u_3\}$. This subtle but important distinction accelerates the identification of fairness constraints.

3.1 Fairness and Interpretability Constraints

In the rest of the paper, we focus on inferring the following constraints, with constraint thresholds defined below.

Disparate impact or group fairness (ω_{GF}). This commonly studied fairness metric requires the fraction of nodes belonging to all groups, characterized by a sensitive feature, to have a fair representation in each cluster. Let the sensitive feature takes two values—Red or Blue, with each node assigned one of the two colors. This constraint requires the fraction of red and blue nodes in a cluster to be within $[\alpha, \beta]$ where $\alpha, \beta \in [0, 1]$ are called *constraint thresholds* [7, 14].

Equal representation (ω_{EQ}). This fairness constraint enforces equal distribution of nodes with a specific feature value, across clusters. An example is requiring all clusters to have equal number of nodes with the feature value 'Red'. This clustering constraint has been particularly useful in team formation settings, where the resources are fixed and certain colored nodes need to be distributed equally among teams (clusters). More formally, let α_i denote the number of nodes with feature value α in cluster C_i. Constraint ω_{EQ} requires $\alpha_i = \alpha_j$. Restricting all nodes of feature value α to be distributed equally may be very strict for some applications. A generalization of this constraint requires the distribution ratio to be greater than a pre-defined threshold β, $\frac{\alpha_i}{\alpha_j} > \beta$, for every pair of clusters [16, 22]. This ratio captures the relative distribution of α-valued nodes across the clusters.

Interpretability (ω_{IC}). This constraint considers a specific feature of interest (say 'Color') and requires that all clusters are homogenized according to the considered feature. The homogeneity of a cluster with respect to a feature f is characterized by the fraction of nodes of a cluster that have same feature value for the input feature. For example, consider a cluster with 7 blue nodes, 2 red nodes and 1 green colored node. Then the homogeneity of the cluster with respect to the feature 'color' and feature value 'blue' is 0.7. Generating interpretable clusters requires satisfying a homogeneity threshold β— each cluster is required to have at least β fraction of nodes with respect to f [38].

Symbol	Formula	Parameter	Reference
ω_{GF}	Ratio of each feature value $\in [\alpha, \beta]$	α, β	[7, 14]
ω_{EQ}	Relative distribution of a specific feature value	β	[16, 22]
ω_{IC}	Homogeneity of clusters	β	[38]

Table 1: Candidate fairness and interpretable constraints (Ω).

These constraints, described by a feature f and a threshold β, are summarized in Table 1. Given the set of candidate constraints Ω and demonstrations Λ, LCD aims to identify the constraint, along with its feature and corresponding threshold, that has the maximum likelihood.

4 SOLUTION APPROACH

We begin by describing a naive approach to infer the constraint thresholds and discussing its limitations. We then propose an algorithm that infers the constraint threshold and generates clusters using existing clustering algorithms. To extend our approach to handle fairness metrics that are not currently supported by the existing algorithms, we present a greedy clustering approach.

4.1 Naive algorithm

A naive approach to infer the clustering constraint from a given set of demonstrations Λ is to exhaustively generate all possible clusterings for each type of constraint, its corresponding feature, and threshold. Among these clusterings, the most likely set of clusters correspond to the one having maximum conformance with the demonstrations Λ. This approach is highly effective in identifying the desired set of clusters but does not scale, given that the fairness constraint threshold can take infinite values. For example, the disparate impact constraint ω_{GF} take two parameters α, β as input, which can take any value in the range $[0, 1]$. To efficiently infer the constraint, we build on the following observations.

- k-center clustering (and centroid-based clustering in general) aims to minimize the maximum distance of any node from the cluster center. Therefore, it is very unlikely that a particular node is assigned to the farthest center.
- Our problem can be modeled as a likelihood estimation problem, where the most likely constraint is expected to correspond to the ground truth constraint.

Given a cluster C, we can estimate the most likely threshold of C with respect to a constraint, by following the procedure discussed in the previous section. For example, if a cluster has 3 red nodes and 5 blue nodes, we can infer that the fraction of nodes of each color is at least min(3/8, 5/8). Using this constraint threshold estimation, a simple approach is to estimate the likelihood of different clustering constraints by considering each demonstration as an independent set of clusters and calculate threshold with respect to each constraint over these clusters. A major drawback of this approach is that a single clustering demonstration generally does not contain representation from all k clusters and feature values for the considered feature. This may mislead the likelihood estimation when a demonstration considered in isolation.

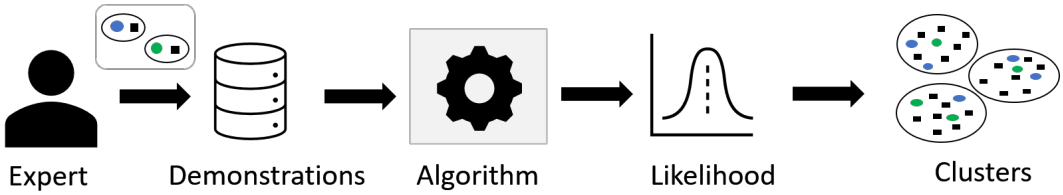


Figure 2: Overview of solution approach.

Algorithm 1 Maximum Likelihood Constraint

Input: Demos Λ, Nodes V, Features of interest F
Output: Clusters C

1: **for** $v \in \Lambda$ **do**
2: $\quad C \leftarrow C \cup \{v\}$
3: $C \leftarrow$ `ConstructClusters`(Λ)
4: **while** $|C| > k$ **do**
5: $\quad C \leftarrow$ `MergeClosest`(C)
6: $T(\omega, f) \leftarrow 0, \forall \omega \in \Omega, f \in F$
7: **for** $\omega \in \Omega, f \in F$ **do**
8: $\quad T(\omega, f) \leftarrow$ `CalculateThreshold`(C, ω, f)
9: **for** $(\omega, f) \in T$ **do**
10: $\quad C_{\omega,f} \leftarrow$ Cluster(ω, f, V)
11: $\quad \mathcal{L}_{\omega,f} \leftarrow$ `Likelihood`$(C_{\omega,f}, \Lambda)$
12: $(\omega, f) \leftarrow \arg\max(\mathcal{L}_{\omega,f})$
13: Return the clustering corresponding (ω, f)

Example 1. *Consider an optimal clustering for $\boldsymbol{\omega_{GF}}$, denoted by $C_1 = \{r_1, r_2, b_1, b_2\}$ and $C_2 = \{r_3, b_3\}$, where r_1, r_2, r_3 are the red nodes and b_1, b_2, b_3 are blue colored nodes. Suppose one of the demonstration is $\lambda = \{(r_1, r_2), (r_3)\}$. Based on this demonstration, the inferred constraint is $\boldsymbol{\omega_{IC}}$ with $\beta = 1$, which incorrectly indicates that all the nodes in a cluster have the same color.*

4.2 Our Algorithm

We present Algorithm 1 that clusters the given demonstrations and processes these clusters to infer the most likely constraint and its parameter values (feature and threshold). Figure 2 presents the high level architecture of our proposed technique. Given a collection of demonstrations generated by an expert, our algorithm greedily merges them to generate k clusters. These clusters are then used to calculate the likelihood of each fairness constraint and infers the clustering with maximum likelihood.

Algorithm 1 proceeds in two phases. In the *first phase* (Lines 1-5), the algorithm forms k clusters of the demonstrations Λ. This phase initializes a clustering C over the set of nodes in demonstrations Λ (`ConstructClusters`(Λ)) which correspond to the different clusters identified by the expert. Note that the set C may contain more than k clusters. In that case, we greedily merge the closest pair of clusters until k clusters have been identified. The distance between any pair of clusters $C_i, C_j \in C$ is measured as the maximum distance between any pair of nodes in C_i and C_j:

$$d(C_1, C_2) = \max_{u \in C_1, v \in C_2} d(u, v).$$

In the *second phase* (Lines 6-12), the identified clusters C are processed to calculate the most likely threshold with respect to each feature and constraint (denoted by T). The identified threshold is used to generate a set of k clusters on the original dataset V for each ⟨constraint, feature⟩ pair. At the end of this step, there are $|F| \times |\Omega|$ clusterings, with one of them corresponding to the intended set of clusters.

To identify the set of clusters with maximum likelihood ($\mathcal{L}$), we calculate the accuracy of each clustering with respect to the input demonstrations and return the set of clusters that have the highest accuracy. The accuracy of a set of clusters C is calculated by labeling each pair of nodes as intra-cluster or inter-cluster, and then measuring the fraction of pairs that have same labels according to C and Λ. The accuracy estimate of the clusters C captures the likelihood of a particular constraint.

Complexity. The first phase of Algorithm 1 is initialized with $O(|\Lambda|)$ demonstrations and iteratively reduced to k clusters. In each iteration, it calculates the distance between pairs of clusters, resulting in $O(|\Lambda^2|)$ run time. The second phase considers all combinations of constraint and features, thereby performing clustering $|F| \times |\Omega|$ times where F denotes the set of features for each node. Therefore, the run time complexity of Algorithm 1 to calculate clusters over the demonstrations is $O(\log^3 n)$ and it takes $O(n|F||\Omega|)$ to construct clusters and calculate likelihood.

Algorithm 1 identifies the optimal set of clusters and the maximum likelihood constraints for a given set of demonstrations, assuming that a clustering technique *exists* for an input constraint. We now present a greedy algorithm that does not rely on the clustering technique and greedily generates the set of clusters with maximum likelihood.

4.3 Greedy Algorithm for Novel Metrics

To handle the fairness objectives for which fair clustering algorithms do not currently exist, we present a greedy algorithm that generates k clusters without assuming any knowledge about the clustering algorithm for the input constraints.

Our approach is outlined in Algorithm 2. Given a collection of demonstrations Λ and vertices V, the algorithm proceeds in two phases. The first phase of Algorithm 2 (Lines 1-5) is similar to that of Algorithm 1, where all nodes are initialized as singleton clusters and all nodes that are grouped together in Λ are merged. The closest pair of clusters in C are sequentially merged until k clusters have been identified. Let C denote the final set of k clusters.

The second phase (Lines 6-10) begins with estimating the constraint threshold (T) over the demonstrations, as in Algorithm 1.

Algorithm 2 Greedy Algorithm for Novel Metrics

Input: Demos Λ, Nodes V, Features of interest F
Output: Clusters C

```
for v ∈ V do
    C ← C ∪ {v}
C ← ConstructClusters(Λ)
while |C| > k do
    C ← MergeClosest(C)
T ← Calculate constraint threshold of each constraint over the demonstrations Λ
for (ω, f) ∈ T do
    Modify C greedily to satisfy the constraint ω with respect to feature f
    L_{ω,f} ← Likelihood(C_{ω,f}, Λ)
Return the clustering corresponding arg max(L_{ω,f})
```

The estimated threshold is used to greedily post-process the clusters C according to each constraint. This greedy processing transfers the nodes from one cluster to another, following the constraint requirements and is similar to local search techniques that move nodes between clusters to satisfy a constraint. At the end of this phase, there are $|F| \times |\Omega|$ different sets of clusters, with each optimizing a different fairness constraint. The clustering that has the highest likelihood with the input demonstrations is returned as the final set of clusters. The likelihood is estimated in terms of the accuracy of pairwise intra-cluster and inter-cluster labels.

5 THEORETICAL ANALYSIS

In this section, we analyze the effectiveness of Algorithm 1 to identify the constraints even when the oracle presents $\Theta(\log n)$ demonstrations, where $n = |V|$. We first show that the estimated constraint is accurate with a high probability under the assumption that the oracle chooses nodes uniformly at random. We then extend the analysis to settings where the presented demonstrations are biased towards specific clusters. This analysis assumes that each demonstration $\lambda \in \Lambda$ has constant size[1].

Let $\tilde{V}$ denote the set of nodes that have been clustered in atleast one of the demonstrations. Lemma 3 shows that the sample $\tilde{V}$ contains $\Theta(\log n)$ nodes from a cluster C^* whenever $|C^*| \geq \frac{n}{k}$.

Lemma 3. *Consider a random sample $\tilde{V} \subseteq V$ such that $|\tilde{V}| \geq 16\, k \log n$ and each node in $\tilde{V}$ is chosen uniformly at random, then $|\tilde{V} \cap C^*| > 8 \log n$, $\forall |C^*| \geq \frac{n}{k}$ with a probability of $1 - 1/n$.*

Proof. Let X_v be an indicator variable such that $X_v = 1$ if $v \in \tilde{V}$ and 0 otherwise. Since, each node v is chosen uniformly at random, $Pr[v \text{ is chosen}] = \frac{|\tilde{V}|}{n}$. Let C^* denote a cluster such that $|C^*| \geq n/k$. Therefore,

$$E\left[|\tilde{V} \cap C^*|\right] = \frac{|\tilde{V}|}{n}|C^*| \geq 16 \log n.$$

Using Chernoff bound,
$Pr[|\tilde{V} \cap C^*| \leq \frac{1}{2}\mu] < e^{-\mu/8} = \frac{1}{n^2}$ where $\mu = E[|\tilde{V} \cap C^*|]$
Therefore,
$|\tilde{V} \cap C^*| > 8 \log n$ with a probability of $1 - \frac{1}{n^2}$. Performing a union bound over all clusters, we get that $|\tilde{V} \cap C^*| > 8 \log n$ holds for all C^* such that $|C^*| \geq n/k$ with probability of atleast $1 - \frac{1}{n}$. □

Consider a set of ground truth clusters, $\mathcal{C}^* = \{C_1^*, \ldots, C_k^*\}$, such that $\forall |C_i^*|$ satisfy one of the clustering constraint $\omega \in \Omega$. This means that $\exists i$ such that $|C_i^*| \geq \frac{n}{k}$. For the next part of the proof, we will consider this C_i^* to analyze the quality of estimated constraint threshold.

Lemma 4. *Suppose the optimal cluster C_i^* satisfies the constraint, ω_{GF} with parameters $[\alpha, \beta]$ and $|\tilde{V} \cap C_i^*| = \Theta(\log n)$, then the estimated threshold on processing $|\tilde{V} \cap C_i^*|$ is $[\alpha(1-\epsilon), \beta(1+\epsilon)]$ with a high probability.*

Proof. Suppose the optimal fairness constraint ω_{GF} considers a feature f with parameters $[\alpha, \beta]$. Let $A = \{a_1, \ldots, a_t\}$ denote the domain of values for the feature f. According to the fairness constraint, the subset of C_i^* that has feature value $a_j, \forall j$ is within a fraction $[\alpha, \beta]$. Suppose the fraction of nodes with feature value a_i be α_i.

We claim that the fraction of nodes with feature α_i in the sample $\tilde{V} \cap C^*$ is within $[\alpha_i(1-\epsilon), \alpha_i(1+\epsilon)]$ with a high probability, where ϵ is a small constant. Let X_v denote a binary random variable such that X_v is one if v is present in the sample $\tilde{V}$ and 0 otherwise. The expected number of nodes that have feature α_i and belong to the set $\tilde{V} \cap C_i^*$ is $\frac{\alpha_i |C_i^*||\tilde{V}|}{n} = \Theta(\log n)$. Following the proof of Lemma 3 and using Chernoff bound, we get that the number of nodes with value a_j is within a factor of $[(1-\epsilon/2), (1+\epsilon/2)]$ of the expected value with a high probability. Additionally, the expected number of nodes that belong to the sample $|C_i^* \cap \tilde{V}| = \frac{|C_i^*||\tilde{V}|}{n}$ and the number of nodes is within a factor of $[(1-\frac{\epsilon}{2}), (1+\frac{\epsilon}{2})]$ with a high probability.

Therefore, the ratio of node that have feature value a_i and belong to the sample $\tilde{V} \cap C_i^*$ is always within a factor of $\left[\frac{1-\epsilon/2}{1+\epsilon/2}, \frac{1+\epsilon/2}{1-\epsilon/2}\right] \sim [1-\epsilon, 1+\epsilon]$ for small values of ϵ. Taking a union bound over all feature values, we guarantee that the estimated parameter is within a factor of $[1-\epsilon, 1+\epsilon]$ with a high probability. □

Lemma 5. *Suppose the optimal cluster C_i^* satisfies the constraint, ω_{IC} with parameter β (some constant) and $|\tilde{V} \cap C_i^*| = \Theta(\log n)$, then the estimated threshold on processing $|\tilde{V} \cap C_i^*|$ is $[\beta(1-\epsilon), \beta(1+\epsilon)]$ with a high probability.*

Proof. Suppose the optimal cluster C_i^* satisfies ω_{IC} with parameter β with respect to a feature value α. Therefore, β fraction of the nodes in C_i^* have the feature value α. To analyze the fraction of nodes of feature value α, we define binary random variable X_v for each v such that $X_v = 1$ if $v \in \tilde{V}$ and 0 otherwise. The expected number of nodes with feature value α in the sample $\tilde{V} \cap C_i^*$ is $\frac{\beta |C_i^*||\tilde{V}|}{n}$. Following the analysis of Lemma 4, we get that the fraction of nodes of color α is within a factor of $[\beta(1-\epsilon), \beta(1+\epsilon)]$ with a high probability. □

Lemma 6. *Suppose the optimal cluster C_i^* satisfies the constraint, ω_{EQ} with parameter β and $|\tilde{V} \cap C_i^*| = \Theta(\log n)$, then the estimated threshold on processing $|\tilde{V} \cap C_i^*|$ is $[\beta(1-\epsilon), \beta(1+\epsilon)]$ with a high probability.*

[1] Our proofs extend to the setting where demonstration size is $\Omega(1)$ too.

Proof. This analysis is similar to that of Lemma 5. □

Lemmas 4, 5 and 6 show that the estimated parameter from a cluster C_i^* with respect to the considered fairness constraints is within a factor of $[(1-\epsilon),(1+\epsilon)]$ of the true constraint threshold with a high probability. Using these results, we prove the following theorem.

Theorem 7. *Given a collection of nodes V and randomly chosen globally informative demonstrations $\Lambda = \Theta(\log n)$ such that each demonstration reveals the true cluster affiliation of a constant number of records, then the optimal cluster constraint is identified within a multiplicative factor of $[(1-\epsilon),(1+\epsilon)]$ with a high probability.*

Proof. Let Λ denote a collection of globally informative demonstrations such that $|\Lambda| = \Theta(\log n)$ and let $\tilde{V} = \cup_{\lambda_g \in \Lambda} \lambda_g$. Using Lemma 3, we know that $\tilde{V} \cap C_i^* = \Theta(\log n)$ for all C_i^* containing $\Theta(n)$ nodes and therefore, using Lemmas 4, 5 and 6 we are guaranteed to estimate the correct threshold for the cluster C_i^*. Hence, Algorithm 1 correctly estimates the constraint with maximum likelihood with $\Theta(\log n)$ globally informative demonstrations. □

Remark 8. *In this section, the constants in Θ notation would depend on the number of features $|F|$ and number of fairness constraints $|\Omega|$. We do not optimize for these constants because Algorithm 1 empirically converges in less than $2 \log n$ demonstrations.*

We extend the proof of Theorem 7 to the setting where the demonstrations are not globally informative but the ground truth clusters satisfy an interesting property, similar to the γ-margin property studied in prior work [6]. We first define the margin property. Let $\tilde{V}$ denote a subset of nodes and C^* denote the set of clusters corresponding optimal constraint. The set $\tilde{V}$ is considered to satisfy margin property if $d(u,x) > d(u,v)$ where $u, v \in C_i^* \cap \tilde{V}$ and $x \in \tilde{V} \setminus C_i^*$.

Theorem 9. *Given a collection of nodes V and randomly chosen demonstrations $\Lambda = \Theta(\log n)$ such that each demonstration reveals the clustering over a subset of nodes, then the optimal cluster constraint is identified within a multiplicative factor of $[(1-\epsilon),(1+\epsilon)]$ with a high probability if the sampled nodes $\cup_{\lambda \in \Lambda} \lambda$ satisfy the margin property.*

Proof. Let Λ denote a collection of demonstrations such that $|\Lambda| = \Theta(\log n)$ and let $\tilde{V} = \cup_{\lambda \in \Lambda} \lambda$. Using Lemma 3, we know that $\tilde{V} \cap C_i^* = \Theta(\log n)$ for all C_i^* containing $\Theta(n)$ nodes. This guarantees that we have $\Theta(\log n)$ nodes sampled from C_i^* but we may not have merged all these nodes to form a single cluster. In order to show that the nodes present in merged cluster (after Line 5 of Algorithm 1) belong to the same cluster, we use the margin property. The margin property assumes that all nodes that belong to same cluster are closer to each other as compared to nodes of other clusters. Therefore, `MergeClosest` always merges a pair of clusters that belong to same optimal cluster C_i^*, thereby guaranteeing its correctness. Since C_i^* has been constructed correctly, the rest of the proof is same as that of Theorem 7. □

Theorem 10. *Given a collection of nodes V and randomly chosen demonstrations $\Lambda = \Theta(\log n)$ such that each demonstration reveals the clustering over a subset of nodes, then Algorithm 2 recovers ground truth clusters with a high probability if the nodes V satisfy the margin property.*

Proof. This analysis is similar to that of Theorem 9. □

Discussion. The analysis of Theorem 9 assumed margin property over the sampled nodes. In many real world datasets, clusters are often well separated, thereby automatically implying the margin property. Additionally, even if the margin property does not hold on overall clusters, expert can choose samples for the demonstration such that the samples of different clusters are present sufficiently away. The proof of Theorem 9 can be extended to settings where a constant fraction of sampled nodes do not obey the margin property.

Another important assumption that is crucial in the analysis presented above is the randomness of sampled nodes. Theorem 7 and 9 assume that every node is chosen uniformly at random. Note that these assumptions can be relaxed and our proofs extend to settings when the *samples are biased towards a specific cluster*. For example, the number of samples a specific cluster (say C_i^*) is much higher than $\Theta(\log n)$ but the samples from other clusters are much fewer. In this case, Algorithm 1 will correctly estimate the threshold from C_i^* with fewer demonstrations but it may require more number of demonstrations to achieve accurate estimate from other clusters.

6 EXPERIMENT SETUP

In this section, we evaluate the effectiveness of `LCD` on three real world datasets. We show that our techniques efficiently calculate the true likelihood of each constraint and the generated set of clusters are closer to the desired output, compared to other baselines.

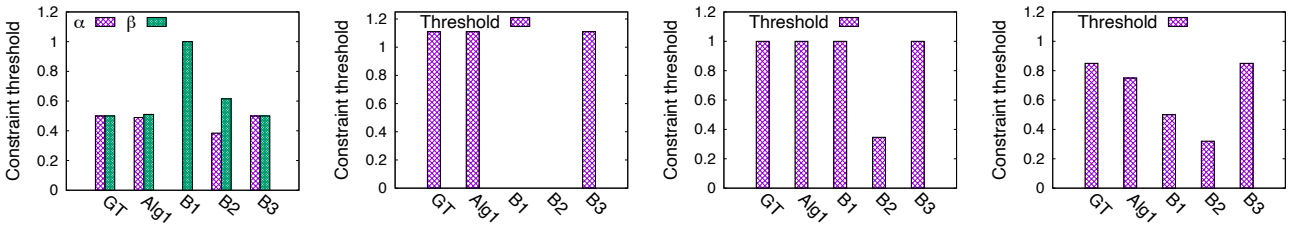
Datasets. We evaluate our approach on three datasets, which are borrowed from the prior work that experiment with the metrics of interest.

- **Bank** dataset [7] containing 4521 data nodes corresponding to phone calls from a marketing campaign by a Portuguese banking institution. The marital status of the records is considered as the sensitive feature for ω_{GF} constraint, with parameters $[0.49, 0.51]$.
- **Adult** dataset [38] containing 1000 records with the income information of individuals along with their demographic attributes. 'Age', 'occupation', and 'income' features are considered as the features of interest. Fairness constraint ω_{EQ} is optimized with respect to 'occupation' and ω_{IC} with respect to 'age' and 'income'.
- **Crime** dataset [38] contains crime information about different 1994 communities in the United States, where 'number of crimes per 100K population' is used for ω_{IC} fairness constraint.

The features in these datasets are considered to calculate distance between every pair of nodes. Euclidean distance is calculated between numerical attributes and Jaccard distance between the categorical attributes. Please refer to [7, 38] for more details.

Baselines. We compare the results of our techniques with the following baselines:

- `B1` calculates the likelihood by considering each demonstration as a separate set of clusters;



(a) Bank, ω_{GF} (b) Adult, ω_{EQ} (c) Crime, ω_{IC} (d) Adult, ω_{IC}

Figure 3: Comparison of estimated constraints for different datasets. GT denotes the ground truth constraint threshold. Algorithm 1 and B3 identify the most accurate estimate of the constraint threshold.

- B2 merges the different clusters in the demonstration to identify k clusters and infers the likelihood over the identified clusters; and
- B3 performs a grid search over all possible fairness constraints and identifies the clustering that conforms with the generated demonstrations.

Algorithm 1 is referred as Alg1 and Algorithm 2 is labeled Alg2 in all the plots in this section. Unconstrained k-center clustering technique is labeled as kC. The clusters generated using ground truth fairness constraint is denoted by GT.

Setup. We use the original implementations of ω_{GF}, ω_{IC}, and ω_{EQ}. Their code base were used to generate ground truth clusters for an input constraint requirement. All algorithms were implemented in Python and tested on an Intel Core i5 computer with 16GB of RAM.

Our experiments compare the identified fairness parameter by our algorithm and each baseline. To compare the quality of identified clusters, we compute the F-score of the identified intra-cluster pairs of nodes. F-score denotes the harmonic mean of the precision and recall, where precision refers to the fraction of correctly identified intra-cluster pairs and recall refers to the fraction of intra-cluster pairs that are identified by our algorithm. In all experiments, we report results with $k = 5$. We execute the code of prior constrained clustering techniques with specified parameters to generate ground truth clustering. Each demonstration is generated by sampling a subset of *five* nodes randomly from these clusters. Unless otherwise specified, we consider $2 \log n$ demonstrations as input and these demonstrations do not reveal the true cluster affiliation of the considered nodes. In case there are multiple constraints that generate the same set of demonstrations, the algorithm output is considered correct if it correctly identifies any one of those constraints[2].

7 RESULTS AND DISCUSSION

We now present results of our approach on three datasets. Our results show that the proposed approach can effectively identify the intended metrics using demonstrations, and can generate fair and interpretable clusters.

[2]Among the considered constraints, this situation does not arise whenever $|\Lambda| > 5$

7.1 Effectiveness of Algorithm 1

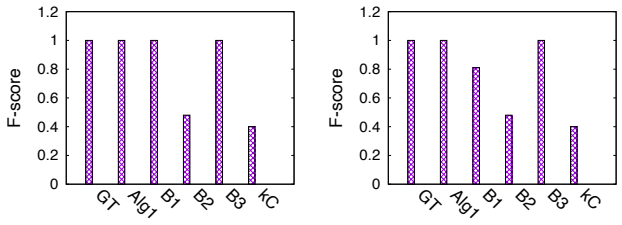
The effectiveness of Algorithm 1 is measured based on the constraint threshold and the quality of the generated clusters. Figure 3 compares the estimated threshold of the most-likely constraint, calculated by Algorithm 1 with the ground truth and other baselines. Across all datasets, Algorithm 1 estimates the optimal threshold for every considered constraint, matching the performance of ground truth. This validates the effectiveness of Algorithm 1 to correctly estimate the most likely constraint and its corresponding threshold.

Among the baselines, B3 achieves a similar performance. This is an expected behavior since B3 performs naive grid search to explore all threshold values. Although it is effective in inferring the threshold, this technique is orders of magnitude inefficient due to the exhaustive enumeration of clusters using the different sets of constraints, features and their respective thresholds. It is therefore practically infeasible to implement this for problems with large input graphs and large Ω.

The other baselines B1 and B2 consistently show poor performance. Baseline B1 does not identify any fairness constraint in settings where the demonstrations obey ω_{GF} and ω_{EQ} (Figure 3(a) and 3(b) respectively). However, it identifies the optimal clustering constraint only in case of ω_{IC}. Given that each demonstration has fewer than 5 nodes, the information available in a single demonstration is not sufficient for B1 to infer the true fairness constraint. On the other hand, B2 overcomes the limitations of B1 by merging the demonstrations randomly in order to capture constraint information over all demonstrations collectively. This approach has better performance than B1 but does not identify the true clustering constraint in majority of the cases. It does not identify the fairness constraint ω_{EQ} (Figure 3(b)) and the identified constraint threshold in all other cases are sub-optimal.

Figure 4 compares the quality of the returned clusters, by comparing the F-score of the clustering output of each technique with the ground truth clusters. In this experiment, Algorithm 1 and B3 achieve optimal performance as they identify the true ground truth clusters across all parameter settings. All other baselines did not identify the clusters correctly and achieved low F-score. Particularly, in case of ω_{EQ} and ω_{GF}, the baselines B1 and B2 did not identify the optimal constraint threshold and generated biased clusters.

Table 2 compares the running time of Alg1 and other baselines for different datasets and clustering constraints. Among all datasets, Alg1 is orders of magnitude faster than B3. In the worst case, Alg1

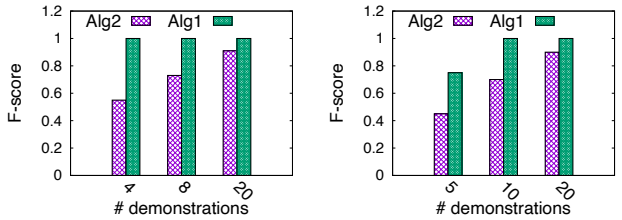


(a) Adult ω_{IC}, $\beta = 1$ (b) Adult ω_{IC}, $\beta = 0.9$

Figure 4: F-score comparison of the generated clusters for different techniques along with GT denoting the ground truth set of clusters.

Dataset	Alg1	B1	B2	B3
Bank	0.57	0.49	0.52	100
Adult	1.14	1.01	1.1	117
Crime	1.02	0.9	0.97	104

Table 2: Running time results (in minutes).



(a) Bank, ω_{GF} (b) Adult, ω_{IC}

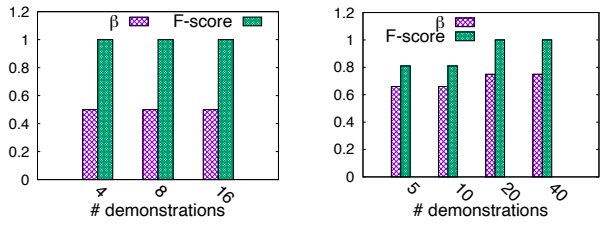
Figure 5: Effect of number of demonstrations on our algorithms' performance.

generates $O(|\Omega| \times |F|)$ sets of clusters whereas B3 generates clusters exhaustively for every value of constraint threshold. The running time of Alg1 is comparable with B1 and B2.

7.2 Effectiveness of Algorithm 2

Alg2 identifies k clusters such that the returned output obeys the fairness constraint reflected from the demonstrations Λ. Figure 5 plots the F-score of Alg2 for two data sets and the results are compared with that of Alg1. This allows us to compare the performance of our greedy Alg2 with that of an existing efficient solver. In Figure 5(a), we employed Bera et al. [7] to generate the ground truth clusters according to ω_{GF} and tested the effectiveness of Alg2 to recover ground truth clusters for varying number of demonstrations. Similarly in Figure 5(b), ground truth is generated using ω_{IC}.

When the number of demonstrations is less than 5, the F-score of the generated clusters is 0.55 for both domains. As we increase the number of demonstrations, we observe that the performance of Alg2 improves and is closer to that of Alg1. Alg2 achieves more than 0.9 F-score in less than 20 demonstrations. The continuous improvement in accuracy demonstrates the effectiveness of Alg2 in recovering clusters without relying on a clustering algorithm.



(a) Bank, ω_{GF} (b) Adult, ω_{IC}

Figure 6: Effect of # demonstrations on Alg1 performance.

7.3 Effect of Demonstrations

We now investigate the effect of number of demonstrations on the performance of our techniques in identifying the optimal constraint threshold. We varied the number of demonstrations in multiples of $\log n$: $0.5 \log n, \log n, 2 \log n$. Figure 6 compares the constraint threshold and the F-score of the identified clusters using Alg1, with varying number of demonstrations on the Bank and Adult dataset. In case of ω_{GF}, the ground truth constraint requires equal representation of the different groups in each cluster. Algorithm 1 correctly identifies the fairness constraint and achieves perfect F-score even when Λ contains as low as four demonstrations. Increasing the number of demonstrations does not improve its performance as the constraint likelihood has already converged. In ω_{IC}, the ground truth clusters are generated according to threshold $\beta = 0.85$. When the number of input demonstrations $|\Lambda|$ is low ($|\Lambda| = 4$), the estimated interpretability constraint threshold is inaccurate and the constraint estimation improves as the number of demonstrations are increased. Algorithm 1 is able to achieve an F-score more than 0.8 with just ten demonstrations and the quality of final clusters improves monotonically with increasing demonstrations. It converges to the accurate constraint threshold whenever $|\Lambda| \geq 20$ and therefore achieves perfect F-score.

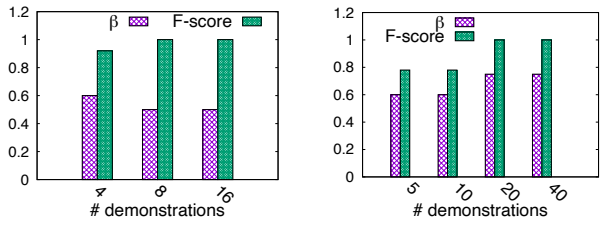
In Figure 3, the input demonstrations do not reveal the true cluster affiliation of any of the nodes. We ran an additional experiment with the globally informative demonstrations (Definition 2), which reveals the ground truth cluster affiliation of each node in the demonstration. With this additional information, we observe that Algorithm 1 converges to the optimal constraint threshold in less than ten demonstrations. This experiment validates that Algorithm 1 is able to leverage the extra information provided by globally informative demonstration to converge faster.

Next, we evaluate the effect of number of demonstrations on the performance of Algorithm 2. Figure 5 shows that as we increase the number of demonstrations, Alg2 matches the F-score of Alg1.

7.4 Ablation Study

To test the effectiveness of our constraint estimation techniques, we varied the size of demonstration from 4 to 10 for the different constraints. As expected, the number of required demonstrations reduces linearly with increase in demo size[3]. Therefore, an increase in demonstration size helps Alg1 converge faster.

[3]We do not consider smaller demonstrations because clustering fewer than 4 nodes do not reveal enough information about the underlying clusters.



(a) Bank, ω_{GF} (b) Adult, ω_{IC}

Figure 7: Effect of sampling bias on Alg1 performance.

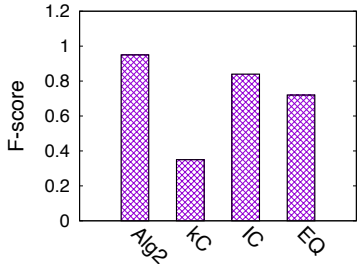


Figure 8: F-score of fair and interpretable clusters generated by different techniques.

We tested the robustness of our constraint threshold estimation techniques by generating demonstrations according to a biased distribution. In the first experiment (Figure 7), we employed a *biased* sampling procedure, where each demonstration is biased in favor of some specific clusters but all nodes within those chosen clusters are equally likely to be chosen for the demonstration. Specifically, we follow a two step procedure where we first sample the cluster C_i with probability p_i and the nodes from the sampled cluster are chosen randomly. This introduction of bias did not affect the quality of our techniques and Alg1 was able to recover ground truth clusters in $\Theta(\log n)$ demonstrations. The second experiment considered a biased sampling procedure where the expert samples fewer nodes from the marginalized groups. For example, a node having 'red' color is chosen with probability $\frac{1}{n}$ but a blue colored node is chosen with probability $\frac{4}{n}$. In such setting, the returned demonstrations are biased against the marginalized groups and the inferred clustering threshold is not accurate. We observe that this bias translates into the constraint threshold estimation procedure of Alg1. This experiment justifies the requirement of an unbiased expert annotator that chooses nodes randomly, without considering their sensitive attributes.

To further study the effect of k, we vary the number of clusters as $k = \{5, 10, 15, 20, 50\}$ for adult dataset and calculated the number of demonstrations required to identify the true clustering constraint. For all values of k, Alg1 identified the optimal set of clusters in less 20 (which is around $2 \log n$) demonstrations and the number of required demonstrations increases sub-linearly with k. For example, it required 20 demonstrations for $k = 5$ and 60 demonstrations were enough for $k = 50$. This increase in number of demonstrations is justified because Alg1 tries to merge presented demonstrations into k clusters. If the number of clusters in presented demonstrations is smaller than k, then it might end up partitioning some clusters which may introduce some noise in the likelihood estimation procedure. However, when the input demonstrations are globally informative, the number of required demonstrations do not increase with k and therefore we do not include the plots. Alg1 converges to the optimal clustering constraint as soon as there are $\Theta(\log n)$ nodes from any of the clusters.

7.5 Fair and Interpretable Clusters

To evaluate the effectiveness of generating fair and interpretable clusters, we ran interpretable clustering algorithm [38] with $\beta = 1$ for Adult dataset. The generated clusters were then post-processed to satisfy ω_{EQ}. Since *none* of the existing clustering algorithms optimize for fairness and interpretability, we implemented a greedy technique to process the output of interpretable clusters and satisfy fairness constraint. We considered this output as the ground truth to generate globally informative demonstration Λ and ran Alg2 to calculate the set of clusters with maximum likelihood. Alg2 achieved F-score of more than 0.9 (Figure 8) with less than 25 demonstrations, each with 5 nodes. Any baseline that optimizes ω_{IC} or ω_{EQ} alone achieved sub-optimal performance. This experiment demonstrated the ability of Alg2 to generate clusters even when the constraint optimization algorithm is not known. Additionally, Alg2 requires the expert to label less than 25% dataset to generate fair and interpretable clusters.

8 SUMMARY AND FUTURE WORK

With the availability of many nuanced fairness definitions, it is non-trivial to specify a fairness metric that captures what we intend. As a result, systems may be deployed with an incomplete specification of the fairness metric, which leads to biased outcomes. We formalize the problem of inferring the fairness metric that the designer intends to optimize for a given problem. Our solution approach combines graph clustering and learning from demonstrations to generate fair and interpretable clusters. We present an algorithm to generate fair clusters by inferring the fairness constraint using expert demonstration and analyze its theoretical guarantees. We also present a greedy approach to generate fair clusters for objectives which are not currently supported by the existing suite of fair clustering algorithms. We empirically demonstrate the effectiveness of our approach in inferring fairness and interpretability metrics, and then generate clusters that are fair and interpretable. Although we discuss the framework in the context of fair clustering, our proposed framework can be used to infer any clustering constraints, as shown in the experiments.

In the future, we plan to conduct a human subjects study to evaluate our approach and design robust algorithms to infer the intended metrics in the presence of noise. Developing robust techniques to handle bias in demonstrations is another interesting question for future work. Extending our algorithm to handle other fairness metrics and interpretability metrics will broaden the scope of problems that can be handled by our approach.

ACKNOWLEDGMENTS

We thank the reviewers for their valuable feedback.

REFERENCES

[1] Abbeel, P.; and Ng, A. Y. 2004. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the 21st International Conference on Machine Learning*.

[2] Ahmadi, S.; Galhotra, S.; Saha, B.; and Schwartz, R. 2020. Fair Correlation Clustering. *arXiv preprint arXiv:2002.03508* .

[3] Ahmadian, S.; Epasto, A.; Kumar, R.; and Mahdian, M. 2019. Clustering without over-representation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*.

[4] Aljrees, T.; Shi, D.; Windridge, D.; and Wong, W. 2016. Criminal Pattern Identification Based on Modified K-means Clustering. In *IEEE International Conference on Machine Learning and Cybernetics*, volume 2, 799–806.

[5] Anderson, N.; Bera, S. K.; Das, S.; and Liu, Y. 2020. Distributional Individual Fairness in Clustering. *arXiv preprint arXiv:2006.12589* .

[6] Ashtiani, H.; Kushagra, S.; and Ben-David, S. 2016. Clustering with same-cluster queries. In *Advances in Neural Information Processing Systems*, 3216–3224.

[7] Bera, S. K.; Chakrabarty, D.; Flores, N. J.; and Negahbani, M. 2019. Fair Algorithms for Clustering. *CoRR* abs/1901.02393. URL http://arxiv.org/abs/1901.02393.

[8] Binns, R. 2018. Fairness in machine learning: Lessons from political philosophy. In *Conference on Fairness, Accountability and Transparency*.

[9] Brams, S. J.; and Taylor, A. D. 1996. *Fair Division: From cake-cutting to dispute resolution*. Cambridge University Press.

[10] Celis, L. E.; Huang, L.; and Vishnoi, N. K. 2018. Multiwinner voting with fairness constraints. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*.

[11] Celis, L. E.; Straszak, D.; and Vishnoi, N. K. 2018. Ranking with Fairness Constraints. In *Proceedings of the 45th International Colloquium on Automata, Languages, and Programming*.

[12] Chen, J.; Chang, Y.; Hobbs, B.; Castaldi, P.; Cho, M.; Silverman, E.; and Dy, J. 2016. Interpretable Clustering Via Discriminative Rectangle Mixture Model. In *IEEE 16th International Conference on Data Mining (ICDM)*, 823–828.

[13] Chiappa, S. 2019. Path-specific counterfactual fairness. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 7801–7808.

[14] Chierichetti, F.; Kumar, R.; Lattanzi, S.; and Vassilvitskii, S. 2017. Fair clustering through fairlets. In *Advances in Neural Information Processing Systems*.

[15] Dasgupta, S.; Frost, N.; Moshkovitz, M.; and Rashtchian, C. 2020. Explainable k-Means Clustering: Theory and Practice. In *XXAI Workshop, ICML*.

[16] Ding, H. 2020. Faster balanced clusterings in high dimension. *Theoretical Computer Science* .

[17] Dwork, C.; Hardt, M.; Pitassi, T.; Reingold, O.; and Zemel, R. 2012. Fairness through awareness. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, 214–226.

[18] Feldman, M.; Friedler, S. A.; Moeller, J.; Scheidegger, C.; and Venkatasubramanian, S. 2015. Certifying and removing disparate impact. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

[19] Firmani, D.; Saha, B.; and Srivastava, D. 2016. Online entity resolution using an oracle. *Proceedings of the VLDB Endowment* 9(5): 384–395.

[20] Galhotra, S.; Brun, Y.; and Meliou, A. 2017. Fairness Testing: Testing Software for Discrimination. In *Proceedings of the 11th Joint Meeting on Foundations of Software Engineering*.

[21] Galhotra, S.; Firmani, D.; Saha, B.; and Srivastava, D. 2018. Robust entity resolution using random graphs. In *Proceedings of the International Conference on Management of Data*, 3–18.

[22] Galhotra, S.; Saisubramanian, S.; and Zilberstein, S. 2019. Lexicographically Ordered Multi-Objective Clustering. *CoRR arXiv preprint:1903.00750* .

[23] Galhotra, S.; Shanmugam, K.; Sattigeri, P.; and Varshney, K. R. 2020. Fair Data Integration. *arXiv preprint arXiv:2006.06053* .

[24] Gillen, S.; Jung, C.; Kearns, M.; and Roth, A. 2018. Online learning with an unknown fairness metric. In *Advances in neural information processing systems*, 2600–2609.

[25] Haraty, R. A.; Dimishkieh, M.; and Masud, M. 2015. An Enhanced K-means Clustering Algorithm for Pattern Discovery in Healthcare Data. *International Journal of Distributed Sensor Networks* .

[26] Hilgard, S.; Rosenfeld, N.; Banaji, M. R.; Cao, J.; and Parkes, D. C. 2019. Learning representations by humans, for humans. *arXiv preprint arXiv:1905.12686* .

[27] Hospers, G.-J.; Desrochers, P.; and Sautet, F. 2009. The Next Silicon Valley? On the Relationship Between Geographical Clustering and Public Policy. *International Entrepreneurship and Management Journal* 5(3): 285–299.

[28] Ilvento, C. 2019. Metric learning for individual fairness. *arXiv preprint arXiv:1906.00250* .

[29] Kamishima, T.; Akaho, S.; Asoh, H.; and Sakuma, J. 2012. Fairness-aware classifier with prejudice remover regularizer. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*.

[30] Kleindessner, M.; Awasthi, P.; and Morgenstern, J. 2019. Fair k-center clustering for data summarization. *International Conference of Machine Learning* .

[31] Knight, W. 2019. Researchers Want Guardrails to Help Prevent Bias in AI. URL https://www.wired.com/story/researchers-guardrails-prevent-bias-ai/.

[32] Laber, E.; and Murtinho, L. 2021. On the price of explainability for some clustering problems. *arXiv* .

[33] Mahabadi, S.; and Vakilian, A. 2020. Individual Fairness for k-Clustering. In *Proceedings of the 37th International Conference on Machine Learning*.

[34] Mazumdar, A.; and Saha, B. 2017. Clustering with noisy queries. In *Advances in Neural Information Processing Systems*, 5788–5799.

[35] Mazumdar, A.; and Saha, B. 2017. Query complexity of clustering with side information. In *Advances in Neural Information Processing Systems*, 4682–4693.

[36] Mehrabi, N.; Morstatter, F.; Saxena, N.; Lerman, K.; and Galstyan, A. 2019. A survey on bias and fairness in machine learning. *arXiv preprint arXiv:1908.09635* .

[37] Nabi, R.; and Shpitser, I. 2018. Fair inference on outcomes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

[38] Saisubramanian, S.; Galhotra, S.; and Zilberstein, S. 2020. Balancing the Tradeoff Between Clustering Value and Interpretability. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, 351–357.

[39] Saisubramanian, S.; Kamar, E.; and Zilberstein, S. 2020. A Multi-Objective Approach to Mitigate Negative Side Effects. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence*.

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

[41] Thomson, W. 1983. Problems of fair division and the egalitarian solution. *Journal of Economic Theory* 31(2): 211–226.

[42] Vazirani, V. V. 2013. *Approximation Algorithms*. Springer Science & Business Media.

[43] Verma, S.; and Rubin, J. 2018. Fairness definitions explained. In *IEEE/ACM International Workshop on Software Fairness*.

[44] Vesdapunt, N.; Bellare, K.; and Dalvi, N. 2014. Crowdsourcing algorithms for entity resolution. *Proceedings of the VLDB Endowment* 7(12): 1071–1082.

[45] White, H. 1982. Maximum likelihood estimation of misspecified models. *Econometrica: Journal of the Econometric Society* 1–25.