Simultaneous Individual and Group Fairness in Set Covering Problems

Sharmila Duppala
Uber Technologies*
Sunnyvale, CA 94085
sharmi92736@gmail.com

Nathaniel Grammel
Swarthmore College*
Department of Computer Science
Swarthmore, PA 19081
ngrammel@cs.swarthmore.edu

Tyler He
Georgia Institute of Technology
Department of Computer Science
Atlanta, GA 30332
thetailai08@gmail.com

Aravind Srinivasan
University of Maryland
Department of Computer Science
College Park, MD 20740
asriniv1@umd.edu

Abstract

Covering problems are fundamental combinatorial optimization problems with broad connections to clustering, facility location, and various applications such as machine learning and controlling disease outbreaks. In this work we study a variant of the set cover problem that generalizes the partition set cover problem [Bera et al., 2014]. In partition set cover, given an instance $(\mathcal{P}, \mathcal{S})$ with a universe $\mathcal{P} = \cup_{c\in[\ell]} P_c$ (where $[\ell]$ denotes $\{1, 2, \ldots, \ell\}$), a collection of subsets $\mathcal{S}$ (each with associated costs), and coverage requirements $k_1, k_2, \ldots, k_\ell$, the objective is to find a subcollection with minimum cost that covers at least k_c elements from each group $c \in [\ell]$. We generalize this problem by allowing for the groups $\mathcal{P} = \cup_{c\in[\ell]} P_c$ to be overlapping thus capturing the notion of both *group fairness* and *multigroup fairness*. We further allow for probabilistic coverage requirements for each element in the ground set thus capturing the notion of *individual fairness*. Our main result is a randomized iterated rounding based algorithm with provable guarantees. We supplement our theoretical results by showing that our algorithm has significantly lower running times and outperforms the best known algorithm of [Inamdar and Varadarajan, 2018].

1 Introduction

Covering problems form a family of fundamental combinatorial-optimization problems that have connections to clustering and facility location, and that have applications in machine learning, containing disease outbreaks, determining where to place critical infrastructure, etc. For instance, consider the problem of providing coverage of some service (e.g., telecommunications, emergency services, transportation, or public services) to a community through the construction of new infrastructure (e.g., cellular towers, hospitals, fire departments, train stations, libraries, etc.). Each new potential location for such infrastructure will "cover" some subset of the population by providing service to those living near the newly built infrastructure. As a concrete example, consider the selection of vaccine distribution sites as part of an emergency response in the midst of a pandemic.

With machine learning systems being increasingly deployed for major real-world applications, such systems are responsible for making critical, sometimes life-changing, decisions and many models can potentially make decisions that discriminate against underrepresented groups [Buolamwini and Gebru,

*This work was done when this author was a PhD student at University of Maryland, College Park

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

2018, Angwin et al., 2022]. For instance, members of an ethnic, economic, gender, or other minority group may end up less likely to receive service coverage. As a result there has been a growing interest in exploring meaningful notions of fairness and developing fair machine learning algorithms. While much of the research in fair ML has traditionally focused on the supervised-learning setting [Dwork et al., 2012, Zafar et al., 2017, Menon and Williamson, 2018] and the examination of fairness notions within it [Kleinberg et al., 2017, Hebert-Johnson et al., 2018], there is a recent trend towards investigating various fairness notions for classical combinatorial problems, including covering [Bandyapadhyay et al., 2023a, Asudeh et al., 2023], clustering [Chierichetti et al., 2017, Bera et al., 2019, Backurs et al., 2019, Abbasi et al., 2021, Esmaeili et al., 2020, Bandyapadhyay et al., 2019], graph matching [García-Soriano and Bonchi, 2020, Esmaeili et al., 2023, Bandyapadhyay et al., 2023b], and ranking [Asudeh et al., 2019, Gorantla et al., 2020].

We consider a *probabilistic* notion of *individual fairness*, which has been studied previously by Harris et al. [2019b] for center-type problems and later by Anegg et al. [2022] for fair k-center and Dinitz et al. [2022] for graph-cut problems; roughly speaking, this notion requires a randomized algorithm to provide coverage to each individual with some specified minimum probability. Other variations and interpretations of individual fairness have also been studied in clustering [Mahabadi and Vakilian, 2020, Brubach et al., 2020], ranking [Bower et al., 2021], and matching [Esmaeili et al., 2023]. Several notions of group fairness have been extensively studied in both deterministic and randomized settings for various problems including *fair matching* [Sankar et al., 2021, Bandyapadhyay et al., 2023b], *fair clustering* [Chierichetti et al., 2017, Abbasi et al., 2021, Bandyapadhyay et al., 2019, Anegg et al., 2022], *fair ranking* [Zehlike et al., 2017], and *fair set packing* Duppala et al. [2023].

In PARTITIONSC, we are given a ground set $\mathcal{P} = \bigcup_{c\in[\ell]} P_c$ of elements to be covered, partitioned into ℓ groups $P_1, P_2, \ldots, P_\ell$; a collection $\mathcal{S} = \{S_1, S_2, \ldots, S_m\}$ of m subsets of $\mathcal{P}$ with costs w_i for each $S_i \in \mathcal{S}$; and coverage requirements $k_1, k_2, \ldots, k_\ell$ for each group. Let f denote the *frequency*, i.e., the maximum number of subsets to which any element belongs. The goal is to find a minimum cost collection $\mathcal{C} \subseteq \mathcal{S}$ of subsets that covers at least k_c elements from P_c for all $c \in [\ell]$. We introduce the following notation for brevity, for any collection of subsets $F \subseteq \mathcal{S}$ we define $w(S) = \sum_{S_i\in F} w_i$.

In this paper, we generalize the problem by (i) allowing elements to belong to multiple groups simultaneously. This generalizes the partition constraints which capture group-level fairness to multi-group fairness (overlapping groups). Since the latter is a generalization of the former, we hereafter refer to both as group fairness constraints and refer to the problem as FAIRSC and (ii) adding probabilistic individual fairness constraints to ensure minimum coverage probabilities for each individual (this is similar to what Anegg et al. [2022] do for k-center). We formally define the problem in the next section. While group fairness and individual fairness are typically seen as conflicting objectives, they can sometimes be achieved simultaneously [Gorantla et al., 2020, Anegg et al., 2022]; see also Binns [2020] for a discussion of these two fairness notions and their relation to each other. In this work, we present algorithms for achieving both simultaneously.

2 Preliminaries and Problem Formulation

Typical approaches to design approximation algorithms for covering problems involve LP-based techniques. The following is a natural LP relaxation for PARTITIONSC problem given by Inamdar and Varadarajan [2018].

$$\min \sum_{i\in[m]} w_i x_i \tag{1a}$$

$$\text{s.t. } y_j \le \sum_{i:j\in S_i} x_i, \quad \forall j \in \mathcal{P} \tag{1b}$$

$$\sum_{j\in\mathcal{P}_c} y_j \ge k_c, \quad \forall c \in [\ell] \tag{GF}$$

$$0 \le x_i \le 1, \qquad \forall i \in [m] \tag{1c}$$

$$0 \le y_j \le 1, \qquad \forall j \in \mathcal{P} \tag{1d}$$

In (1), the LP variables x_i correspond to each subset $S_i \in \mathcal{S}$ and y_j to each element $j \in \mathcal{P}$. The optimal solution to (1) is a lower bound on the optimal solution to the corresponding PARTITIONSC. Further, the result still holds when the groups are overlapping, thus extending the LP to FAIRSC.

Definition of Individual Fairness. In addition to ensuring group fairness via the constraints (**GF**), in this work we also consider *probabilistic individual fairness*. For simplicity, we will refer to this probabilistic notion simply as *individual fairness*. Given a FAIRSC instance along with probabilities $p_j \in [0, 1]$ for each element $j \in \mathcal{P}$, the FAIRSC-IF problem asks for a *randomized* algorithm that outputs a solution to the given FAIRSC instance such that each element $j \in \mathcal{P}$ is covered with probability at least p_j; we say that such a randomized algorithm is individually fair. To address these individual fairness guarantees in our methods, we add the following family of constraints (**IF**).

$$y_j \geq p_j \ \ \forall j \in \mathcal{P} \tag{IF}$$

These probabilities p_j correspond to minimum guarantees on the probability of j receiving coverage. A machine learning model may assign these minimum coverage probabilities based on the needs and/or merits of an individual. Determining coverage probabilities for individuals remains an active area of research and falls outside the scope of this paper; for our purposes, we treat these values as given.

Motivation for Individual Fairness Covering problems like ours are closely related to clustering and facility location problems, such as that studied in Harris et al. [2019a], which offers a nice motivating application to mobile healthcare: Consider a setting where healthcare services may be offered through mobile-health vans or other mobile resources (see, e.g., paragraph 2 of section 1.2 of Harris et al. [2019a] for some additional discussion on applications such as mobile mammography, reproductive care, and prenatal care). Proximity to healthcare facilities impacts health outcomes, and so deploying mobile health resources such as these can provide meaningful benefit to populations that may otherwise be underserved [Harris et al., 2019a].

To see how our notions of fairness might apply to such a setting, consider a healthcare provider seeking to deploy a collection of mobile health resources (e.g., mobile-health vans) to some subset of candidate locations. Once deployed, the mobile health service will remain there for the day, providing their services to anyone nearby. The provider has the goal of ensuring adequate access to healthcare for a sufficient portion of the population. Each candidate location serves a predefined cluster of individuals, corresponding to one of the subsets in our problem. The group fairness constraint allows the healthcare service to guarantee access for a sufficient number of individuals in each of a number of protected groups. Since these mobile services remain only for the day, the provider plans to repeat this periodically, selecting a new set of locations to deploy mobile-health vans to every so often (say, once a month). In this repeated setting, the individual fairness provides an additional guarantee on top of the group fairness, ensuring that each individual will, over time after several deployments, have one of these mobile health resources nearby with some minimum frequency, with high probability.

As an additional application, consider a targeted advertising campaign, where certain advertisements will target specific subsets of individuals, and our goal is to select a set of advertisements that sufficiently cover a variety of demographic groups (e.g, across different age groups). The values in our individual fairness can provide additional constraints at the individual-level, tailored to factors such as the predicted likelihood that an ad impression on that individual will convert to a sale, website visit, etc. Additional applications for similar problems can also be found in Asudeh et al. [2023]. To our knowledge, we are the first to consider both individual and group fairness simultaneously for set cover in this way, although this simultaneous notion has been studied in the context of other classical problems like ranking [Gorantla et al., 2020] and bipartite matching [Esmaeili et al., 2023].

Performance Criteria To analyze the performance of our algorithm, we compare the expected weight of our covering relative to the expected weight of the solution returned by an optimal (randomized) algorithm for FAIRSC-IF. An *optimal solution* is a distribution over all integral partial covers satisfying Eq. (**GF**) such that, if we sample a cover from this distribution, each element $j \in \mathcal{P}$ is covered with probability at least p_j. Among all such valid distributions, this optimal solution minimizes the expected weight. We abuse notation slightly and use OPT to denote both this optimal distribution and its expected weight; the intended meaning should be clear from context. We say that an algorithm is a c-approximation for some constant $c > 1$ if $\mathbb{E}[w(\mathrm{ALG})]/\mathrm{OPT} \leq c$ where ALG is the (possibly randomized) algorithm, $w(\mathrm{ALG})$ is the weight of the covering returned (with expectation taken over randomness in ALG), and OPT is the expected weight of optimal solution.

Lemma 1. LP-FAIR *is a relaxation of* FAIRSC-IF *i.e., for any given instance of* FAIRSC-IF*, optimal value of the LP is a lower bound on the optimal solution* (OPT).

$$\min \sum_{i \in [m]} w_i x_i \qquad \text{(LP-FAIR)}$$
$$\text{s.t. Eq. (1b)-(1d), (\textbf{GF}),(\textbf{IF})}$$

Proof. Let OPT be the optimal distribution over valid integral set covers (satisfying the (**GF**)). We define a fractional solution (x^*, y^*) based on this distribution:

1. Let x_i^* be the probability that subset S_i is included in the solution chosen by OPT.
2. Let y_j^* be the probability that element j is covered by the solution chosen by OPT.

By definition, the weight of the fractional solution (x^*, y^*) matches the expected weight of optimal solution:

$$\sum w_i x_i^* = \mathbb{E}[w(\text{OPT})].$$

To show that LP-FAIR is a relaxation of OPT, we need to show that (x^*, y^*) is a feasible solution to the LP i.e., satisfies all the constraints in LP.

Let Y_j denote the random variable that element j is selected and X_i denote the random variable that subset S_i is chosen.

1. **Covering Constraint**: The event *element j is covered* is the union of the events *subset S_i is chosen* for all subsets S_i containing j. Therefore, by union bound, we have:

   $$Pr[Y_j = 1] \leq \sum_{i: j \in S_i} Pr[X_i = 1] = \sum_{i: j \in S_i} x_i^*.$$

   Since, $\Pr[Y_j = 1] = y_j^*$, we have $y_j^* \leq \sum_{i: j \in S_i} x_i^*$, as desired.

2. **Group fairness Constraint (GF)**: Any integral solution in the support of OPT satisfies the group fairness constraint: the number of covered elements in group P_c is at least k_c. Hence, by Linearity of Expectation,

   $$\sum_{j \in P_c} y_j^* = \mathbb{E}[\sum_{j \in P_c} Y_j] \geq k_c$$

   as desired.

3. **Individual Fairness Constraint (IF)** By the definition, any valid randomized algorithm must cover element j with probability at least p_j. Therefore, $\Pr[Y_j = 1] = y_j^* \geq p_j$.

Since any optimal solution (x^*, y^*) is a feasible solution to LP-FAIR, we can say that 1 minimizes the objective function over all feasible solutions, its optimal value must be less than or equal to the weight of (x^*, y^*). Thus, we have

$$\mathbb{E}[w(\text{OPT})] \geq \text{LPOPT}$$

where LPOPT is the weight of optimal LP solution. □

Therefore, we can evaluate our algorithm with respect to the optimal solution of the LP-FAIR.

Definition of β-PROBABLYINDFAIR. In the aforementioned definition, we ensure that our randomized algorithms provide ex-ante guarantees for elements i.e., the promise that an element j will be covered with a probability p_j. While valuable, these guarantees don't ensure that the realized outcomes are *ex-post* fair. For instance, if a set of individuals are each promised a 50% chance of receiving a service, do the realized outcomes reflect this promise? That is, does approximately half of that set actually receive the service? We introduce a β-PROBABLYINDFAIR fairness criterion to address this gap. It formalizes the intuitive notion that an algorithm's promises should be reflected in its realized outcomes for groups of similar elements. This is not group fairness based on protected attributes, but rather a strict criterion promoting consistency for elements that are similar (i.e., have

similar coverage probabilities). To formalize the notion of *similar elements*, we partition the set of all elements $\mathcal{P}$ into groups based on their coverage probabilities p_j. Given a base $q \in [0.5, 1)$ and a positive integer k, we define $k+1$ geometric partitions $\mathcal{Q}_i$:

$$\mathcal{Q}_i = \begin{cases} \{j \in \mathcal{P} \mid p_j \in (q^{i+1}, q^i]\} & \text{if } i < k \\ \{j \in \mathcal{P} \mid p_j \in [0, q^k]\} & \text{if } i = k. \end{cases}$$

This geometric partitioning groups elements with comparable coverage probabilities: for any two elements j_1, j_2 in the same partition, their probabilities are close, i.e., $|p_{j_1} - p_{j_2}| \leq q^i(1-q)$. With these similarity groups established, we can now define our fairness criterion. The definition leverages the notion of β-concentration from [Bansal, 2019] described below.

Definition 1 (β-concentration [Bansal, 2019])**.** *Let $\beta \geq 1$, and let $X_1, \ldots, X_n$ be possibly dependent* 0-1 *random variables with means $\mathbb{E}[X_i] = x_i$. For any subset $K \subseteq [n]$, $S = \sum_{i \in [K]} X_i$ is* β-concentrated *if, for all $t > 0$,*

$$\Pr[S - \mathbb{E}[S] \geq t] \leq \exp\left(\frac{-t^2}{2\beta\left(\sum_{i \in [K]}(x_i - x_i^2) + t/3\right)}\right).$$

Let Y_j represent the indicator random variable that an element j is covered with $\mathbb{E}[Y_j] = y_j$. For each $\mathcal{Q}_i$ let $Y^i = \sum_{j \in \mathcal{Q}_i}(Y_j)$ be the total number of covered elements from that partition.

Definition 2 (β-PROBABLYINDFAIR)**.** *We say that the algorithm satisfies* β-PROBABLYINDFAIR *if, for every $\mathcal{Q}_i$, $Y^i = \sum_{j \in \mathcal{Q}_i} Y_j$ is β-concentrated around the mean $\sum_{j \in \mathcal{Q}_i} y_j$.*

In simple terms, this guarantee ensures that the actual fraction of covered elements in any similarity group $\mathcal{Q}_i$ is, with high probability, very close to the average promised probability for that group. The parameter β quantifies the strength of this guarantee: a smaller β implies a stronger concentration. This means the observed outcomes are highly likely to be extremely close to their expected values.

3 Related Work and Contributions

A naive greedy algorithm gives a $\mathcal{O}(\log n)$-approximation [Slavík, 1997] for SETCOVER problem, and this can be extended to PARTITIONSC to yield an approximation ratio of $\mathcal{O}(\sum_{c \in [\ell]} \log k_c)$, which is $\mathcal{O}(\log n)$ in the worst case [Bera et al., 2014]. In the most general case, f may be large (i.e., it may be as large as n) and the $O(\log n)$-approximation is the best possible approximation unless $\mathsf{P} = \mathsf{NP}$ [Dinur and Steurer, 2014]. However, for cases where f is small (such as the classic example of VERTEXCOVER, where $f = 2$), there exist simple f-approximation algorithms for many covering problems (see, e.g., Gandhi et al. [2004] for a primal-dual approach for PARTIALSC).

For PARTITIONVC, Bera et al. [2014] achieve a $\mathcal{O}(\log \ell)$-approximation via a novel LP relaxation of the problem. For the unweighted case of PARTITIONSC, Bandyapadhyay et al. [2023a] gives an LP-based $(2 + \ell/\mathrm{OPT})$-approximation and describe a technique for improving this ratio to $2 + \epsilon$ in time $\mathcal{O}(n^{\ell/\epsilon})$. In the general (i.e., weighted) case of PARTITIONSC, Hung and Kao [2022] give a primal-dual algorithm with an approximation ratio of $f \ln(\ell) + \ln(\ell)$. Inamdar and Varadarajan [2018] gave a randomized LP-rounding algorithm that is an $\mathcal{O}(\rho + \log \ell)$-approximation for PARTITIONSC where ρ denotes the approximation guarantee for a related Set Cover instance obtained by rounding the standard LP; this implies a $\mathcal{O}(f + \log \ell)$ approximation.

3.1 Limitations of prior techniques

Existing techniques for PARTIALSC and PARTITIONSC are not easily extended to our more general setting with individual fairness. Here, we examine some of the techniques from prior work in more detail, and explain the challenges in extending the techniques to our problem.

First, while the primal-dual technique of Gandhi et al. [2004] can be slightly modified to yield a $(f + \ell)$-approximation algorithm for PARTITIONSC with a runtime of $O(n^\ell)$ time, this only works for a constant number of colors. Moreover, the algorithm cannot directly be adapted to satisfy individual fairness guarantees akin to constraints in (**IF**).

The standard greedy algorithm [Slavík, 1997] is deterministic, and such deterministic algorithms cannot be extended to provide our individual fairness guarantee which is probabilistic in nature. Both Bera et al. [2014] and Inamdar and Varadarajan [2018] do provide a randomized rounding of a novel LP relaxation for PARTITIONVC and PARTITIONSC respectively; however, while the algorithm is randomized, we note that it does not ensure minimum coverage probabilities for individuals. A detailed discussion of this is given in the Section C.

3.2 Our Contributions

We carefully design a randomized iterated rounding algorithm, FAIRITERATEDROUNDING, which returns a set cover of weight at most $f\mathrm{OPT} + \ell$ for FAIRSC-IF, while simultaneously satisfying both individual and group fairness constraints. This result generalizes the work of Bandyapadhyay et al. [2023a] for PARTITIONVC-IF with uniform weights.

For certain sparse set families, such as, *bounded set cardinality* i.e., $\Delta = \max_{i\in[m]} |S_i|$ or *bounded degeneracy* κ (see Definition 3), we introduce an iterated rounding algorithm, ALG(α) in Section 5. When ALG is used with an appropriate choice of the α, ensures that the set of active tight constraints are sparse i.e., rank of active constraint matrix is only a constant fraction of the number of active variables in each iteration. This sparsity allows us to leverage a technique from Bansal [2019] to achieve a stronger fairness guarantee, β-PROBABLYINDFAIR, with $\beta = \mathcal{O}(\Delta)$ for Δ-bounded set families albeit with a slightly worse approximation factor of $(f + 2\ell/\mathrm{OPT})$. We further extend this result to κ-degenerate set families by reducing the $\beta = \Theta(\Delta)$ to $\beta = \Theta(\kappa)$. This is a slightly strengthened result since $\kappa \leq \Delta$.

We supplement our theoretical analysis with experimental results. Using synthetically generated datasets, we compare the performance of our algorithm from Section 4 to the PARTITIONSC algorithm of Inamdar and Varadarajan [2018]. We find that the running time of our algorithm is faster, and our algorithm returns a solution of lower weight across all instances we tested.

4 Algorithmic Results

In Section 3.1, we discussed that existing methods fail to guarantee both group and individual fairness simultaneously. This section introduces our randomized iterated rounding algorithm, FAIRITERATEDROUNDING, which is specifically designed to overcome this challenge for the FAIRSC-IF problem. We will show that given instance of FAIRSC-IF, our algorithm FAIRITERATEDROUNDING returns a set cover of weight at most $(f\mathrm{OPT} + \ell)$ while satisfying both group and individual fairness. Our approach carefully adapts the randomized iterated rounding framework of Saha and Srinivasan [2018] to FAIRSC-IF. The process begins by solving the LP relaxation to find an optimal fractional solution at an extreme point of the polytope described by LP constraints. The algorithm then performs a guided random walk in a slightly relaxed polytope obtained by strategically dropping the individual fairness constraints (**IF**) to iteratively round the fractional vector to an integral solution. This allows us to provably satisfy the *hard* group fairness constraints while ensuring the individual fairness constraints are met in expectation.

FAIRITERATEDROUNDING Algorithm. The algorithm proceeds as follows: Given an instance of FAIRITERATEDROUNDING, we start by solving the LP in LP-FAIR to obtain an optimal fractional solution (x, y). Let $z^h \equiv (x^{(h)}, y^{(h)})$ and initialize $(x^{(0)}, y^{(0)}) = (x, y)$.

Initially, the algorithm relaxes the individual fairness constraints (from (**IF**)) and focuses on satisfying the covering and group fairness constraints. The rounding process is carefully designed so that the relaxed individual fairness constraints are still satisfied in expectation in the final solution.

More concretely, before the main loop of our algorithm begins, we drop the (**IF**) constraints and we run COVERROUND procedure described below on the optimal solution $z^{(0)}$ and update the solution to $z^{(1)} = (x^{(1)}, y^{(1)})$

FAIRITERATEDROUNDING is a iterated rounding procedure to compute an integral solution given a fractional vector. It proceeds in a sequence of iterations and progressively rounds variables to 0 or 1, and once a variable is rounded, its value is fixed for the remainder of the algorithm.

At the start of any iteration h, the algorithm operates on the set of *floating variables*, whose values have not been fixed to 0 or 1 yet. (i) *Floating Vector*: The fractional values of these variables are denoted by floating vector $z^h = (x^{(h)}, y^{(h)})$ and (ii) *Floating Subsets and Elements* ($\mathcal{S}^h$): The set of subsets S_i (elements j) where the variables $x_i^{(h)}$ ($y_j^{(h)}$) are still fractional (i.e., $x_i^{(h)}, y_j^{(h)} \in (0,1)$).

In each iteration, the algorithm operates based on the *active* constraints, those that contain at least one floating variable. Here R_h denote the set of active covering constraints (from Eq. (1b)) and G_h denotes the set of active group fairness constraints (from (**GF**)).

The main loop of the algorithm executes as long as the number of floating subsets, $m_h = |\mathcal{S}^h|$ is at least $\ell + 1$. Each iteration consists of two main steps:

1. RANDMOVE: The procedure begins with the current fractional solution z^h and performs a randomized update to produce an intermediate solution z'. This random step ensures that at least one additional variable is rounded in z' or an active, non-tight constraint becomes tight.
2. COVERROUND: This procedure takes the output z' from RANDMOVE and applies a deterministic, threshold-based rounding, thus rounding one or more variables, producing the floating vector z^{h+1} for the next iteration.

The loop terminates when $m_h \leq \ell$. At this point, any remaining floating subset variables in $\mathcal{S}^h$ are rounded to 1. This step completes the process, and the algorithm returns the final integral solution (X, Y). Next, we describe the core subroutines RANDMOVE and COVERROUND.

RANDMOVE. This procedure generates a new candidate solution by moving in the feasible solution space of *active* constraints from the current point z^h. The idea is to move along a random direction that does not violate the set of active constraints, which we can represent as $A^h z^h \leq b^h$.

1. First, the algorithm finds an arbitrary non-zero vector, $\vec{r}$ within the null space of the matrix representing the *active tight* constraints. Moving along this vector (or in the opposite direction) ensures that the solution remains feasible with respect to these specific constraints.
2. Next, the algorithm determines the maximum distance it can travel from z^h along the direction $\vec{r}$ (the step size $\alpha \geq 0$) and the opposite direction $-\vec{r}$ (the step size $\beta \geq 0$). We stop the step as soon as the solution hits a boundary, which occurs when either: (i) an additional, previously non-tight constraint becomes tight or (ii) a variable z_i^h becomes 0 or 1. Concretely, α is the smallest non-negative step t that satisfies any of these conditions, computed as following,
$$\alpha = \min \left\{ \begin{array}{l} \min\limits_{A_i \in A} \left\{ t \geq 0 \mid A_i(z^h + t\vec{r}) = b_i \text{ and } A_i \vec{r} > 0 \right\}, \\ \min\limits_{z_i^h \in z^h} \left\{ t \geq 0 \mid z_i^h + t r_i \in \{0,1\} \right\} \end{array} \right\}. \tag{3}$$
The step size β is calculated using the same formula, but with $-\vec{r}$ as the direction vector.
3. Finally, the floating vector is updated to $z' = z^h + \alpha\vec{r}$, with probability $\frac{\beta}{\alpha+\beta}$ and $z' = z^h - \beta\vec{r}$, with complementary probability $\frac{\alpha}{\alpha+\beta}$.

By construction, the RANDMOVE step size ensures that the solution moves exactly to the point where either a floating variable is rounded to 0 or 1 or a new active constraint becomes tight, while maintaining feasibility by keeping variables within $[0, 1]$ and satisfying the active tight constraints.

COVERROUND. Following the RANDMOVE step in each iteration, this deterministic procedure rounds additional variables based on a fixed threshold.

1. Any floating subset variable x_i' with a value of $1/f$ or greater is rounded to 1.
2. If any x_i' is rounded to 1, all corresponding variables y_j' for $j \in S_i \cap \mathcal{P}^h$ are also rounded to 1 ensuring their coverage.

The set of floating variables is then updated to $(x^{(h+1)}, y^{(h+1)})$. This concludes an iteration of our algorithm. We defer the pseudocode of our algorithm to Appendix A due to space constraints.

4.1 Correctness and Fairness Guarantees

The correctness of FAIRITERATEDROUNDING relies on the following invariants maintained throughout the iterations. In any iteration h,

1. **Floating Variables:** All floating variables lie between 0 and 1 i.e., $x_i^{(h)}, y_j^{(h)} \in (0,1)$.
2. **Guaranteed Progress:** After each RANDMOVE and COVERROUND step, at least one floating variable is rounded or at least one active non-tight constraint becomes tight.
3. **Feasibility:** The updated vector $(x^{(h+1)}, y^{(h+1)})$ satisfies the active covering and group fairness constraints, and all variable values remain within the interval $[0,1]$.
4. **Marginal Preservation:** The marginals are preserved in expectation i.e., $\mathbb{E}[y_j^{(h+1)}] \geq y_j^h$.
5. **Bounded weight:** For any $S_i \in \mathcal{S}$, the rounding maintains that $\mathbb{E}[x_i^{(h+1)}] \leq f \cdot x_i^{(0)}$, where $x_i^{(0)}$ is the variable's value in optimal fractional solution.

These invariants ensure that the final integer solution (X, Y) has the desired properties. Invariant 1 ensures that the floating vector is always valid for the RANDMOVE procedure to be applied. Invariant 2 ensures that at least one variable is fixed in each iteration, guaranteeing the algorithm terminates in at most $(n+m+\ell)$ rounds. Invariant 3 ensures that the final integral solution satisfies the group fairness constraints ($\sum_{j \in \mathcal{P}_c} Y_j \geq k_c$) with probability 1. Invariant 4, through recursive application, ensures that the final probability of covering an element meets the required threshold: $\Pr[Y_j = 1] \geq y_j^{(0)} \geq p_j$. Finally, Invariant Item 5 ensures that our algorithm produces an integral solution of expected weight of at most $f \cdot \mathrm{OPT}$. We formally prove the correctness of our algorithm below.

4.2 Analysis of FAIRITERATEDROUNDING

The analysis of the algorithm begins with a key observation. By definition, Invariant 1 holds at the start of each iteration, as all floating variables lie strictly between 0 and 1. The execution of the RANDMOVE subroutine relies on the system of active tight constraints being underdetermined. The following lemma proves that this is always the case as long as the main loop continues.

Lemma 2. *For any iteration h where $m_h \geq \ell + 1$, the system of active tight constraints is underdetermined. That is, the number of linearly independent active tight constraints is strictly less than the number of floating variables, $m_h + n_h$.*

Proof. In any iteration h, the floating vector $z^h = (x^{(h)}, y^{(h)})$ is subject to the following active constraints: the covering constraints in R_h, the group fairness constraints in G_h, and the variable bounds $0 \leq z_i^{(h)} \leq 1$. By Invariant 1, all floating variables are in the open interval $(0,1)$. This means the variable bound constraints are never tight. Therefore, the only constraints that can be tight are the covering and group fairness constraints. The total number of these potentially tight constraints, denoted by t is at most: $|R_h| + |G_h| \leq n_h + \ell$. The total number of floating variables is $m_h + n_h$. Since the algorithm's main loop only runs when $m_h \geq \ell + 1$, we can establish that:

$$t \leq n_h + \ell < n_h + (\ell + 1) \leq n_h + m_h$$

Since the number of tight constraints t is strictly less than the number of variables, the system is underdetermined. This guarantees that the null space of the tight constraint matrix contains a non-zero vector. □

The fact that the system of tight constraints has non-empty null space is what allows the RANDMOVE procedure to be invoked. Next, we show that this procedure correctly maintains the other key invariants. Lemma 3 establishes that invariants 2 (guaranteed progress) and 3 (feasibility) hold in any iteration where RANDMOVE is invoked.

Lemma 3. *In any iteration $h \geq 1$, the following properties hold: (i) **Guaranteed Progress**: The vector $z' = \textsc{RandMove}(z^h)$ hits a new boundary i.e., at least one previously floating variable in z^h becomes 0 or 1, or a previously non-tight constraint in $G_h \cup R_h$ becomes tight. (ii) **Feasibility:** The intermediate vector z' and the final updated vector $z^{h+1} = \textsc{CoverRound}(z')$ both satisfy the set of active constraints, $A^h z \leq b^h$.*

Next, we prove the final two invariants, which establish the fairness and weight guarantees of our rounding algorithm. We will show that the expected value of each variable is cleverly bounded throughout the rounding process.

Next we show that given an LP solution (x, y), if $m_h \geq \ell + 1$ in any iteration, the updated solution $(x^{(h+1)}, y^{(h+1)})$ satisfies $\mathbb{E}[y_j^{(h+1)}] \geq p_j$, and $\mathbb{E}[x^{(h+1)}] \leq f x_i$ for all i, j.

Lemma 4. *For any iteration h, the two steps* RANDMOVE *and* COVERROUND *procedures satisfies the following properties: (i)* ***Marginal Preservation:*** *$\mathbb{E}[y_j^{(h+1)}] \geq y_j^{(h)}$ for all floating elements j and (ii)* ***Bounded weight:*** *$\mathbb{E}[x_i^{(h+1)}] \leq f \cdot x_i^{(0)}$ for all subsets i.*

Proof. The proof relies on analyzing the expected value of the floating vector z^h after each of the two subroutines.

Step-1: The RANDMOVE subroutine updates $z^{(h)}$ to z'. Its expected value is unchanged because by definition of the random step:

$$\mathbb{E}[z'] = \frac{\beta(z^{(h)} + \alpha r)}{\alpha + \beta} + \frac{\alpha(z^{(h)} - \beta r)}{\alpha + \beta} = \frac{(\alpha + \beta)z^{(h)} + (\alpha\beta - \alpha\beta)r}{\alpha + \beta} = z^{(h)}. \tag{4}$$

Thus, we have $\mathbb{E}[x_i'] = x_i^{(h)}$ and $\mathbb{E}[y_j'] = y_j^{(h)}$.

Step-2: The COVERROUND subroutine only increases or maintains the values of element variables. A variable y_j' is either left unchanged ($y_j^{(h+1)} = y_j'$) or rounded up to 1 ($y_j^{(h+1)} = 1$). In all cases, $y_j^{(h+1)} \geq y_j'$. Taking the expectation over the randomness of RANDMOVE: $\mathbb{E}[y_j^{(h+1)}] \geq \mathbb{E}[y_j'] = y_j^{(h)}$. This proves the invariant for a single iteration. Applying this argument recursively from the start of the algorithm, we get:

$$\mathbb{E}[y_j^{(h)}] \geq \mathbb{E}[y_j^{(h-1)}] \geq \cdots \geq y_j^{(0)}$$

Since the initial LP solution satisfied $y_j^{(0)} \geq p_j$, the individual fairness guarantee is maintained in expectation throughout, thus proving the invariant 4.

For any subset variable x_i, the COVERROUND procedure can round this variable at most once. For all iterations $k < h$ before x_i is rounded, its value is passed through unchanged, so $\mathbb{E}[x_i^{(k)}] = x_i^{(k-1)}$. At the iteration h where $x_i' \geq 1/f$ and it is rounded to 1, we use the following property of threshold rounding: $\mathbb{E}[x_i^{(h+1)}] \leq f \cdot \mathbb{E}[x_i']$.

Combining this with the marginal preservation property of RANDMOVE and the induction from previous steps:

$$\mathbb{E}[x_i^{(h+1)}] \leq f \cdot \mathbb{E}[x_i'] = f \cdot x_i^{(h)} = f \cdot x_i^{(h-1)} = \cdots = f \cdot x_i^{(0)}$$

For any variable that is never rounded by COVERROUND and is instead set to 1 in the final step, the same bound holds trivially. This proves the bounded weight invariant (invariant 5). □

With all the necessary invariants and properties of the subroutines established, we can now state the main theorem that establishes the guarantees of our FAIRITERATEDROUNDING algorithm.

Theorem 1. *For any instance of the problem, the* FAIRITERATEDROUNDING *algorithm returns a solution (X, Y) with the following guarantees: (i) The expected weight of the final solution is at most $f\mathrm{OPT} + \ell$ where OPT is the weight of optimal solution. (ii) Each element $j \in \mathcal{P}$ is covered with probability of at least p_j and (iii) With probability 1, at least k_c elements are covered from each group $c \in [\ell]$.*

Running time of our algorithm. Akin to other works in the area of LP based rounding methods, we do not give an explicit asymptotic running time analysis. However, its worth noting that algorithm runs in strongly polynomial time: (i) linear programs are known to be solvable in polynomial time, and (ii) our algorithm runs for at most $n + m$ iterations and in each iteration, our subroutine RANDMOVE runs in polynomial time since the matrix operations such as multiplication, finding

nullspace vectors or computing a basis of the null space (via, e.g., Gaussian elimination or Singular Value Decomposition) can easily be achieved in polynomial time. We further note for most real-world applications, our constraint matrix is highly sparse and for such sparse matrices there are faster algorithms for the aforementioned matrix operations.

5 β-PROBABLYINDFAIR guarantees for sparse set families or graphs

To achieve stronger individual fairness guarantees, we can incorporate a powerful rounding technique from [Bansal, 2019] called *subisotropic rounding*. This method can be applied to iterative algorithms like FAIRITERATEDROUNDING provided the algorithm maintains a 'sparse' constraint structure throughout each iteration in the execution. Specifically, if at any iteration h with r_h fractional variables, the active constraint matrix A^h satisfies a slack condition $\dim(A^h) \leq (1-\delta)r_h$ for some constant $\delta > 0$, then subisotropic rounding can be used. This rounding method satisfies some 'nice' properties which as described in the theorem below.

Theorem 2 (Bansal [2019])**.** *Subisotropic rounding takes a fractional solution x and produces an integral solution X with the following guarantees: (i)* ***Feasibility****: With probability 1, X satisfies all the constraints of original problem. (ii)* ***Marginal Preservation****: For every variable $\mathbb{E}[X_i] = x_i$. (iii)* ***β-concentration****: For any subset $K \subseteq [r_h]$ of variables $X = \sum_{i\in[K]} X_i$ is β-concentrated around $\sum_{i\in[K]} x_i$ with $\beta = 10/\delta$.*

5.1 Iterated Rounding $\mathrm{ALG}(\alpha)$.

To ensure the slack condition is met, we introduce a parameterized version of our FAIRITERATEDROUNDING algorithm, denoted $\mathrm{ALG}(\alpha)$. This is identical to the FAIRITERATEDROUNDING algorithm, except the main loop continues as long as $m_h \geq \alpha\ell$, where m_h is the number of floating subset variables. Our original algorithm corresponds to $\alpha = 1$. By increasing α, we force the algorithm to maintain a larger number of fractional variables relative to the number of constraints, which helps inducing the necessary slack. However, this also increases the cost as the final step rounds as large as $\alpha\ell$ number of subset variables.

Corollary 1. *The ALG(α) algorithm returns a solution with an expected cost of at most $f \cdot \mathrm{OPT} + \alpha\ell$, while satisfying all group and individual fairness constraints*

Thus, we can make use of subisotropic rounding if we can ensure that the number of active tight constraints in any iteration h, $\dim(A^h) < (1-\delta)r_h$ for some $\delta > 0$. Note that, the failure probability of our algorithm to return a β-PROBABLYINDFAIR solution increases (decreases) as β (δ) grows. Hence, higher δ has better success probability. We show that for instances where the maximum set size is bounded by $\Delta = \max_i |S_i|$, we can choose an α that guarantees the desired slack condition.

Lemma 5. *For instances with maximum set size Δ, the ALG(α) algorithm with $\alpha = \frac{1}{1-\delta(\Delta+1)}$ ensures the slack condition $\dim(A^h) \leq (1-\delta)(n_h + m_h)$ holds in every iteration.*

The proof can be found in Appendix Section E. For example, to achieve concentration with $\delta = \frac{1}{2(\Delta+1)}$, we must set $\alpha = 2$. We further explore the tradeoff between the parameter α and the concentration in Section E. Finally, we state our result for sparse instances below.

Theorem 3. *For any instance of the problem where the maximum set size is bounded by Δ, there exists a randomized algorithm that returns a solution with the following guarantees: (i) The expected cost is at most $f\mathrm{OPT} + 2\ell$. (ii) All group fairness and expected individual fairness constraints are satisfied and (iii) The solution is β-PROBABLYINDFAIR with $\beta = O(\Delta)$.*

Acknowledgments

We thank the anonymous reviewers for their helpful feedback. Sharmila Duppala, Nathaniel Grammel, and Aravind Srinivasan were supported in part by NSF award number CCF-1918749.

References

M. Abbasi, A. Bhaskara, and S. Venkatasubramanian. Fair clustering via equitable group representations. In *Proceedings of the 2021 ACM conference on fairness, accountability, and transparency*, pages 504–514, 2021.

G. Anegg, H. Angelidakis, A. Kurpisz, and R. Zenklusen. A technique for obtaining true approximations for k-center with covering constraints. *Mathematical Programming*, pages 1–25, 2022.

J. Angwin, J. Larson, S. Mattu, and L. Kirchner. Machine bias. In *Ethics of data and analytics*, pages 254–264. Auerbach Publications, 2022.

A. Asudeh, H. Jagadish, J. Stoyanovich, and G. Das. Designing fair ranking schemes. In *Proceedings of the 2019 international conference on management of data*, pages 1259–1276, 2019.

A. Asudeh, T. Berger-Wolf, B. DasGupta, and A. Sidiropoulos. Maximizing coverage while ensuring fairness: A tale of conflicting objectives. *Algorithmica*, 85(5):1287–1331, 2023.

A. Backurs, P. Indyk, K. Onak, B. Schieber, A. Vakilian, and T. Wagner. Scalable fair clustering. In *International Conference on Machine Learning*, pages 405–413. PMLR, 2019.

S. Bandyapadhyay, T. Inamdar, S. Pai, and K. Varadarajan. A Constant Approximation for Colorful k-Center. In M. A. Bender, O. Svensson, and G. Herman, editors, *27th Annual European Symposium on Algorithms (ESA 2019)*, volume 144 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 12:1–12:14, Dagstuhl, Germany, 2019. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. ISBN 978-3-95977-124-5. doi: 10.4230/LIPIcs.ESA.2019.12. URL https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.ESA.2019.12.

S. Bandyapadhyay, A. Banik, and S. Bhore. On colorful vertex and edge cover problems, 2023a.

S. Bandyapadhyay, F. V. Fomin, T. Inamdar, and K. Simonov. Proportionally fair matching with multiple groups. *arXiv preprint arXiv:2301.03862*, 2023b.

N. Bansal. On a generalization of iterated and randomized rounding. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 1125–1135, 2019.

S. Bera, D. Chakrabarty, N. Flores, and M. Negahbani. Fair algorithms for clustering. *Advances in Neural Information Processing Systems*, 32, 2019.

S. K. Bera, S. Gupta, A. Kumar, and S. Roy. Approximation algorithms for the partition vertex cover problem. *Theoretical Computer Science*, 555:2–8, 2014.

R. Binns. On the apparent conflict between individual and group fairness. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, FAT* '20, page 514–524, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450369367. doi: 10.1145/3351095.3372864. URL https://doi.org/10.1145/3351095.3372864.

A. Bower, H. Eftekhari, M. Yurochkin, and Y. Sun. Individually fair ranking. *arXiv preprint arXiv:2103.11023*, 2021.

B. Brubach, D. Chakrabarti, J. Dickerson, S. Khuller, A. Srinivasan, and L. Tsepenekas. A pairwise fair and community-preserving approach to k-center clustering. In *International conference on machine learning*, pages 1178–1189. PMLR, 2020.

J. Buolamwini and T. Gebru. Gender shades: Intersectional accuracy disparities in commercial gender classification. In *Conference on fairness, accountability and transparency*, pages 77–91. PMLR, 2018.

F. Chierichetti, R. Kumar, S. Lattanzi, and S. Vassilvitskii. Fair clustering through fairlets. *Advances in neural information processing systems*, 30, 2017.

M. Dinitz, A. Srinivasan, L. Tsepenekas, and A. Vullikanti. Fair disaster containment via graph-cut problems. In *International Conference on Artificial Intelligence and Statistics*, pages 6321–6333. PMLR, 2022.

I. Dinur and D. Steurer. Analytical approach to parallel repetition. In *Proceedings of the Forty-Sixth Annual ACM Symposium on Theory of Computing*, STOC '14, page 624–633, New York, NY, USA, 2014. Association for Computing Machinery. ISBN 9781450327107. doi: 10.1145/2591796.2591884. URL https://doi.org/10.1145/2591796.2591884.

S. Duppala, J. Luque, J. Dickerson, and A. Srinivasan. Group fairness in set packing problems. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*, pages 391–399, 2023.

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

S. Esmaeili, B. Brubach, L. Tsepenekas, and J. Dickerson. Probabilistic fair clustering. *Advances in Neural Information Processing Systems*, 33:12743–12755, 2020.

S. Esmaeili, S. Duppala, D. Cheng, V. Nanda, A. Srinivasan, and J. P. Dickerson. Rawlsian fairness in online bipartite matching: Two-sided, group, and individual. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(5):5624–5632, 2023.

R. Gandhi, S. Khuller, and A. Srinivasan. Approximation algorithms for partial covering problems. *Journal of Algorithms*, 53(1):55–84, 2004.

D. García-Soriano and F. Bonchi. Fair-by-design matching. *Data Mining and Knowledge Discovery*, 34:1291–1335, 2020.

S. Gorantla, A. Deshpande, and A. Louis. Ranking for individual and group fairness simultaneously. *arXiv preprint arXiv:2010.06986*, 2020.

D. G. Harris, S. Li, T. Pensyl, A. Srinivasan, and K. Trinh. Approximation algorithms for stochastic clustering. *Journal of Machine Learning Research*, 20(153):1–33, 2019a.

D. G. Harris, T. Pensyl, A. Srinivasan, and K. Trinh. A lottery model for center-type problems with outliers. *ACM Transactions on Algorithms (TALG)*, 15(3):1–25, 2019b.

U. Hebert-Johnson, M. Kim, O. Reingold, and G. Rothblum. Multicalibration: Calibration for the (Computationally-identifiable) masses. In J. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1939–1948. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/hebert-johnson18a.html.

E. Hung and M.-J. Kao. Approximation algorithm for vertex cover with multiple covering constraints. *Algorithmica*, pages 1–12, 2022.

T. Inamdar and K. R. Varadarajan. On the partition set cover problem. *CoRR*, abs/1809.06506, 2018. URL http://arxiv.org/abs/1809.06506.

J. Kleinberg, S. Mullainathan, and M. Raghavan. Inherent Trade-Offs in the Fair Determination of Risk Scores. In C. H. Papadimitriou, editor, *8th Innovations in Theoretical Computer Science Conference (ITCS 2017)*, volume 67 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 43:1–43:23, Dagstuhl, Germany, 2017. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. ISBN 978-3-95977-029-3. doi: 10.4230/LIPIcs.ITCS.2017.43. URL https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.ITCS.2017.43.

S. Mahabadi and A. Vakilian. Individual fairness for k-clustering. In *International conference on machine learning*, pages 6586–6596. PMLR, 2020.

A. K. Menon and R. C. Williamson. The cost of fairness in binary classification. In *Conference on Fairness, accountability and transparency*, pages 107–118. PMLR, 2018.

B. Saha and A. Srinivasan. A new approximation technique for resource-allocation problems. *Random Structures & Algorithms*, 52(4):680–715, 2018.

G. S. Sankar, A. Louis, M. Nasre, and P. Nimbhorkar. Matchings with group fairness constraints: Online and offline algorithms. *arXiv preprint arXiv:2105.09522*, 2021.

P. Slavík. Improved performance of the greedy algorithm for partial cover. *Information Processing Letters*, 64(5):251–254, 1997.

M. B. Zafar, I. Valera, M. G. Rogriguez, and K. P. Gummadi. Fairness constraints: Mechanisms for fair classification. In *Artificial intelligence and statistics*, pages 962–970. PMLR, 2017.

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

A Additional Definitions and Pseudocode

All the code to our experiments can be found in the anonymized repository: https://anonymous.4open.science/r/SetCoverSubmission-NeurIPS2026/

A.1 Pseudocode for our Algorithm FAIRITERATEDROUNDING

Algorithm 1 Algorithm FAIRITERATEDROUNDING

Require: A basic feasible solution $(x, y) \in [0,1]^{m+n}$ to an instance of PARTITIONSC
Ensure: A random integral solution $X \in \{0,1\}^m$

For $h \in \mathbb{Z}$, let $z^h \equiv (x^{(h)}, y^{(h)})$
Initialize $z^{(0)} = (x, y)$
Let $z^{(1)} = \text{COVERROUND}(z^{(0)})$
Let A^h, b^h denote the matrix and vector representing the active constraints in any iteration h, $A^h z^h \leq b^h$
Drop the constraints (**IF**) from the system (A_0, b_0)

Phase 1:
for $h = 1, 2, \ldots$ **do**
  if $|\mathcal{S}^h| \leq \alpha\ell$ **then**
    Exit the loop and enter **Phase 2**
  else
    $z' \leftarrow \text{RANDMOVE}(A^h, b^h, z^h)$
    $z^{h+1} \leftarrow \text{COVERROUND}(z')$
  end if
end for

Phase 2:
Set $X_i = \lceil x_i \rceil$ for each u
Set $Y_j = 1$ for any j such that $j \in S_i$ for some i with $X_i = 1$
return X

A.2 Pseudocode for the Algorithm of Bera et al. [2014]

Algorithm 2 Algorithm RANDOMIZEDROUNDING [Bera et al., 2014]

Require: A feasible solution $x \in [0,1]^m$ to an instance of PARTITIONSC
Ensure: A random integral solution $X \in \{0,1\}^m$

for Each S_i in $\mathcal{S}$ **do**
  if $x_i \neq 0$ & $x_i \neq 1$ **then**
    $X_i \leftarrow 0$
    Set $x_i \leftarrow \min\{6x_i, 1\}$
    if $x_i < 1$ **then**
      Set $X_i \leftarrow 1$ with probability $6x_i$
    end if
  end if
end for
return X

B Omitted Proof from Section 2

C Omitted Proofs from Section 3

The randomized algorithm of Bera et al. [2014] is given in Algorithm 2. Here, we construct an example where this algorithm fails to satisfy the desired individual fairness constraints.

Example 1. *Consider an instance* $(\mathcal{P}, \mathcal{S})$ *of* PARTITIONSC *where* $\mathcal{P} = \{1, 2, \cdots, 8\}$, $S_1 = \{1, 2\}, S_2 = \{1, 3\}, \ldots, S_7 = \{1, 8\}$, *and the probabilities* $p_1 = 1$, *and* $p_j = \frac{1}{7}$ *for* $j \in \{2, 3, \cdots, 8\}$.

The optimal LP solution to the instance is given by $x_i = 1/7$ for each $i = 1, 2, \ldots, 7$ and $y_1 = 1, y_j = 1/7$ for each $j = 2, 3, \ldots, 8$.

Lemma 6. *Algorithm 2 is not* INDIVIDUALFAIR *for* FAIRSC-IF *problem.*

Proof. Let X denote the integral solution returned by Algorithm 2. The probability that an element $j = 1$ is chosen by the algorithm is given by,

$$\begin{aligned}\Pr[j \text{ is selected}] &= 1 - \Pr[\bigcap_{i:j \in S_i} S_i \text{ is not picked}] \\ &= 1 - \prod_{i:j \in S_i} (1 - 6x_i) \\ &= 1 - \frac{1}{7^7} < 1\end{aligned}$$

Therefore, it is evident that the Algorithm 2 fails to achieve the individual fairness guarantees for element for $j = 1$ since $p_1 = 1$. □

D Omitted Proofs from Section 4

Proof of Lemma 3. We begin by proving the first part. This property follows directly from the definition of the RANDMOVE step. The step sizes α and β are explicitly calculated as the minimum non-negative distance from z^h along directions $\vec{r}$ and $-\vec{r}$ required to hit a new boundary (as defined in Eq. (3)). Since the algorithm takes a full step of size α or β, the resulting point z' must lie on at least one such new boundary by construction.

Let $z' = z^h + \eta\vec{r}$, where $\eta \in \{\alpha, -\beta\}$. We must show that z' satisfies all active constraints in $G_h \cup R_h$, which we partition into the set of tight constraints ($A'z^h = b'$) and non-tight constraints ($A''z^h < b''$). (i) For the tight constraints ($A'z^h \leq b'$): The direction vector $\vec{r}$ is chosen from the null space of A'. Therefore,

$$A'z' = A'\left(z^h + \eta\vec{r}\right) = A'z^h + \eta A'\vec{r} = A'z^h + \eta(\vec{0}) = b' \tag{5}$$

This shows the tight constraints remain satisfied (and are still tight). (ii) For the non-tight constraints ($A''z^h \leq b''$): The step size η is chosen to be the smallest distance to any boundary. This guarantees that for any non-tight constraint $A''_i z^h < b''_i$, the step does not violate it. Formally:

$$A''_i z' = A'_i(z^h + \eta\vec{r}) \leq b''_i$$

This inequality holds because if $A''_i\vec{r} > 0$, η would have been capped at $(b''_i - A''_i z^h)/(A''_i\vec{r})$ (see Eq. (3)) or smaller to prevent violation. Thus, z' is a feasible solution.

Next, we show that z^{h+1} remains feasible after the COVERROUND procedure since it can only increase the values of variables (i.e., x'_i and y'_j can only be rounded to 1). A constraint $A_j z \geq 0$ from this set R_h is of the form $\sum_{i:j \in S_i} x_i - y_j \geq 0$. If COVERROUND rounds $x'_k = 1$ for some k where $j \in S_k$, it also rounds $y'_j = 1$. The constraint remains satisfied as $\sum x_i^{(h+1)} - y_j^{(h+1)} \geq 1 - 1 = 0$. The constraints in G_h are of the form $\sum_{j \in \mathcal{P}_c} y_j \geq k_c$. Since COVERROUND only increases y_j values ($y_j^{(h+1)} \geq y'_j$), the sum can only increase or remain unchanged, ensuring the constraint is satisfied. Thus, both invariants 2 and 3 hold in every iteration of the main loop. □

Proof of Theorem 1. We prove each of the three guarantees by combining the invariants proven in the previous lemmas. Let (X, Y) be the final integral solution returned by the algorithm.

The final set of chosen subsets X can be partitioned into two groups: those rounded during the main loop (when $m_h \geq \ell + 1$) and those rounded deterministically in the final step. Let $R = \{i : X_i = 1, i \in [m]\}$

$$\mathbb{E}[w(R)] = \mathbb{E}\left[\sum_i w_i X_i\right] \leq f \sum_{i \in \text{Main Loop}} w_i x_i^{(0)} + \sum_{k \in \text{Final Step}} w_k$$

The expected weight is the sum of expectations over these two groups. By Invariant 5 (Bounded Weight), the expected weight of the subsets rounded during the main loop is bounded by f times their initial fractional LP value. The subsets in the final step whose variables are all set to 1.

Since $\sum w_i x_i^{(0)}$ is the weight of the initial LP solution, we have:

$$\begin{aligned}\mathbb{E}[w(R)] &\leq f \cdot \sum w_i x_i^{(0)} + \sum_{k \in \text{Final Step}} w_k \\ &\leq f \cdot \text{OPT} + \ell \max_{k \in \text{Final Step}} w_k\end{aligned}$$

The value of the initial LP solution, $\sum w_i x_i = \sum w_i x_i^{(0)}$, provides a lower bound on the true integer optimum ($\sum w_i x_i \leq \text{OPT}$). For the second term, which represents the weight of the sets rounded in the final step, we first note that this step involves at most ℓ subsets. We can assume, without loss of generality, that all weights $w_i \leq 1$, since the problem can be scaled by $\max_i w_i$ without affecting the optimal choice of subsets. Under this assumption, the weight from the final step, $\sum_{k \in \text{Final Step}} w_k$ is at most ℓ as desired.

The individual fairness guarantee follows directly from invariant 4 established in Lemma 4. During every iteration of the main loop, the expected value of each element variable never decreases ($\mathbb{E}[y_j^{(h+1)}] \geq y_j^{(h)}$). This property holds for the floating vector until the end of the main loop.

The algorithm then performs a final rounding step, where any remaining floating subset variables are deterministically set to 1. This step can only increase the values of their corresponding element variables (or leave them unchanged) but never decrease them. Therefore, the expectation guarantees are preserved in this final step as well.

By chaining the inductive argument from the main loop with this final step, we confirm that the final expected value is at least p_j i.e.,

$$\Pr[Y_j = 1] = \mathbb{E}[Y_j] \geq y_j^{(0)}$$

Since the initial LP solution satisfied the constraint $y_j^{(0)} \geq p_j$, the final probability of covering element j has to be at least p_j.

Finally, the group fairness guarantee is a direct consequence of invariant 3 (feasibility) established in Lemma 3. That is the group fairness constraints, $\sum_{j \in \mathcal{P}_c} y_j \geq k_c$, are satisfied by the floating vector at the end of every iteration of the main loop. The algorithm's final step only rounds the remaining variables up to 1, which can only increase these sums. Since the constraints already hold, they cannot be violated by this final step. This entire process is deterministic, so the final integral solution satisfies the group fairness constraints with probability 1.

□

E Omitted Details from Section 5

Proof of Lemma 7. From the counting argument in Lemma 2, we know the number of tight constraints, t, is bounded by $t \leq n_h + \ell$. To satisfy the slack condition, we need to show that this is less than $(1-\delta)(n_h + m_h)$.

$$n_h + \ell \leq (1-\delta)(n_h + m_h) = n_h + m_h - \delta(n_h + m_h)$$

This simplifies to $\ell \leq m_h - \delta(n_h + m_h)$. Next, we use the property that the number of floating elements is bounded by the number of floating subsets times their maximum size, i.e., $n_h \leq \Delta m_h$. With this, it is sufficient to satisfy the stricter inequality:

$$\ell \leq m_h - \delta(\Delta m_h + m_h) = m_h(1 - \delta(\Delta + 1)) \tag{6}$$

Rearranging this gives the required condition on m_h:

$$m_h \geq \frac{\ell}{1 - \delta(\Delta + 1)}$$

The ALG(α) loop runs as long as $m_h \geq \alpha\ell$. By setting $\alpha = \frac{1}{1-\delta(\Delta+1)}$, we ensure this condition is always satisfied. □

Tradeoffs between weight vs. concentration. This analysis reveals fundamental tradeoff. Achieving better concentration (a smaller β, which requires a larger δ) forces us to use a larger α. A larger α, in turn, increases the additive term $\alpha\ell$ in the overall weight of the solution.

This relationship between α and β is summarized in Table 1 below. Stakeholders must decide whether to prioritize a lower-weight solution or a solution with stronger, more concentrated fairness guarantees.

Slack δ	(α)	Set Cover weight	β
$\frac{1}{(\Delta+1)^2}$	$\frac{\Delta+1}{\Delta}$	$f\mathrm{OPT} + (1 + \frac{1}{\Delta})\ell$	$\mathcal{O}(\Delta^2)$
$\frac{1}{2(\Delta+1)}$	2	$f\mathrm{OPT} + 2\ell$	$\mathcal{O}(\Delta)$
$\frac{1}{(\Delta+2)}$	$(\Delta + 2)$	$f\mathrm{OPT} + (\Delta + 2)\ell$	$\mathcal{O}(\Delta)$

Table 1: Tradeoffs between the algorithm's weight and the concentration guarantee.

Proof of Theorem 3. The proof is a direct application of the subisotropic rounding framework (from the Theorem 2) to our ALG(α). This specific parameterized algorithm creates the necessary slack to apply the powerful subisotropic rounding in each iteration.

First, we consider the guarantees of the ALG(2) algorithm. As established previously in Corollary 1 and Lemma 7, this procedure ensures two key properties:

1. The algorithm's structure ensures the final expected weight is bounded at $f \cdot \mathrm{OPT} + 2\ell$.
2. Setting $\alpha = 2$ ensures that active constraint matrix maintains a slack of $\delta = \mathcal{O}(\frac{1}{\Delta})$ as long as $m_h \geq \alpha\ell$. This sparsity condition is a necessary for the next step.

Second, since the sparsity condition is met, we can apply the subisotropic rounding method to replace the RANDMOVE and COVERROUND subroutines in an iteration h if $m_h \geq \alpha\ell$. The properties of this rounding step provide the final guarantees:

1. The group and individual fairness guarantees that are maintained throughout the ALG(2) iterations are preserved because subisotropic rounding also maintains the invariants 4 and 3.
2. The weight of this algorithm is same as the weight of ALG(2) as the subisotropic rounding preserve all the variables in expectation.

Additionally, the subisotropic rounding framework used within ALG(2) also guarantees the β-PROBABLYINDFAIR guarantees with $\beta = \mathcal{O}(\Delta)$.

□

E.1 Extension to κ-degenerate Set Families

The concentration guarantees can be further strengthened for a broader class of set families that exhibit a property known as κ-degeneracy.

Definition 3. *A set family is κ-degenerate if its sets can be ordered such that each set introduces at most κ new elements not present in any of the preceding sets.*

This is a more general condition than having a maximum set size of Δ, since the number of new elements (κ) is always less than or equal to the total set size (Δ).

The key step of our previous analysis for Δ-bounded instances relied on the inequality $n_h \leq \Delta m_h$ to ensure the sparsity condition. For κ-degenerate families, we can use the much tighter bound $n_h \leq \Delta m_h$ in Eq. (6). By substituting the tighter bound, the parameter α needed to create a δ-slack now depends on κ instead of Δ. This directly improves the final concentration guarantee from $\beta = O(\Delta)$ to $\beta = O(\kappa)$. This extension provides significantly better concentration for instances where sets have high overlap (i.e., $\kappa \ll \Delta$) without affecting the cost of the solution.

E.2 Background on Sparse Set Families

In this section we introduce a set of sparse set families called κ-degenerate families for some $\kappa \geq 1$. We establish a useful structural lemma that is critical in introducing the δ-slack in any iteration of the *modified* version of ALG(α)

Lemma 7. *Let $\mathcal{S}$ be a κ-degenerate family of m subsets over a ground set with n elements. Then, $n \leq m\kappa$.*

Proof. By the definition of a κ-degenerate family (see Definition 3), there exists an ordering of the sets, $S_1, S_2, \ldots, S_m$, such that each set contains at most κ elements not present in any of the preceding sets. Formally, for every $i \in [m]$:

$$|S_i \setminus \bigcup_{j<i} S_j| \leq \kappa \tag{7}$$

The total number of elements, n, is the size of the union of all sets in the family. Alternatively, we can write the size of this union by summing the number of new elements introduced at each i in the ordering:

$$n = \left| \bigcup_{i \in [m]} S_i \right| = \sum_{i \in [m]} \left| S_i \setminus \bigcup_{j<i} S_j \right|$$

Applying the definition of κ-degenerate families in Eq. (7) overall all m subsets yields the desired result:

$$n = \sum_{i \in [m]} \left| S_i \setminus \bigcup_{j<i} S_j \right| \leq \sum_{i \in [m]} \kappa = m\kappa.$$

□

We modify the FAIRITERATEDROUNDING algorithm to enable the application of subisotropic rounding from [Bansal, 2019]. This is achieved by introducing a δ-slack into the active constraints during each rounding iteration.

For the algorithm to work on κ-degenerate set families, we must guarantee a slack of $\delta = 1/(\kappa + 2)$ in every iteration h. This requires maintaining the following condition between the number of floating elements ($n_h := \mathcal{P}^h$) and the number of active subsets ($m_h := \mathcal{S}^h$):

$$n_h \leq \kappa m_h. \tag{8}$$

This ensures that at the beginning of each iteration, we operate in a subspace with a dimension of at most $(1 - \delta)(n_h + m_h)$, following the same reasoning as in Theorem 3. Notice that Lemma 7 seems to establish this desired relationship; however, if at some iteration h, a subset S_i is rounded so that $x_i^{(h)} = 0$, then this relationship may not necessarily hold. To see this, note that rounding x_i to 0 is analogous to removing it from the set family without removing any of its elements (since this corresponds to choosing not to include S_i in the cover). In this case, the argument in Lemma 7 might no longer hold: the set family resulting from the removal of S_i now has that the elements of S_{i+1} that are not contained in a set $S_{i'}$ for $i' < i + 1$ may no longer be bounded by κ and so the key inequality of Lemma 7 may no longer hold.

$$|S_{i+1} \setminus \bigcup_{i'<i} S_{i'}| \not\leq \kappa$$

Note that this may happen even before the first iteration of our rounding algorithm, since we at the outset remove sets S_i for which $x_i = 0$.

Therefore, at the beginning of any iteration $h \geq 1$ of the ALG, we invoke a pre-processing step PEELELEMENTS to introduce the desired δ-slack by ensuring that $n_h \leq \kappa m_h$.

PEELELEMENTS. In any iteration h, if a subset S_i containing element j was rounded to zero in the previous iteration, i.e., $x_i^{(h-1)} = 0$, then we must settle the elements $j \in S_i$ at the beginning of the current iteration to ensure that the subspace we operate in iteration h is sparse i.e., we satisfy the inequality Eq. (8). Specifically, we perform the following update:

1. For each $j \in \mathcal{P}^{(h)}$, if there exists a subset S_i such that $x_i^{(h-1)} = 0$ (i.e., it was rounded to zero in the previous iteration), then we set $y_j^{(h)} = \sum_{k \neq i : j \in S_k} x_k^{(h)}$.
2. We then drop the covering constraint in (1b) for element j, and substitute $y_j^{(h)}$ with $\sum_{k \neq i : j \in S_k} x_k^{(h)}$ in the coloring constraints (**GF**) for the partition $P_c \ni j$.
3. We also replace the element-variable constraint in (1d) with $\sum_{k \neq i : j \in S_k} x_k^{(h)}$.

Lemma 8. *The modified algorithm with* PEELELEMENTS *satisfies that* $\Pr[Y_j] \geq p_j$*, coloring constraints are all satisfied with probability* 1*, and the total weight of selected subsets is at most* $f \sum_i w_i x_i \leq f\mathrm{OPT}$*. Further, in each iteration, the dimension of the subspace of active constraints following the call to* PEELELEMENTS *is at most* $(1-\delta)(n_h + m_h)$*.*

Proof. First, notice that at the start of the PEELELEMENTS routine, we know that every set S_i satisfies $x_i^{(h)} < 1/f$ and so $\sum_{k : j \in S_k} x_k^{(h)} < \sum_{k : j \in S_k} 1/f \leq 1$ since j is contained in at most f sets. Thus, the constraint (1d) for element j is satisfied and *non-tight* when we substitute y_j for this sum. Similarly, due to constraint (1b) for j, the substitution can only improve (i.e., increase) the LHS of (**GF**), so it remains satisfied.

Next, notice that due to moving in the nullspace, RANDMOVE will not change the value of $\sum_{k : j \in S_k} x_k^{(h)}$ will not change, and so the constraint (1d) will remain non-tight. Moreover, treating $y_j^{(h)} := \sum_{k : j \in S_k} x_k$ for every iteration h after this substitution, the RANDMOVE ensures that $\mathbb{E}[y_j^{(h)}] \geq p_j$ will remain unchanged in every iteration following the substitution. If at any point some set $S_i \ni j$ has $x_i^{(h)} \geq 1/f$, then S_i gets selected and j becomes covered, and in such an iteration we get $\mathbb{E}[Y_j] = 1 \geq p_j$, where $Y_j \in \{0, 1\}$ is the final rounded value for element j (i.e., $Y_j = 1$ if and only if j is covered). Thus the individual fairness requirement is satisfied.

Also note that, as in Theorem 1, since each set S_i is selected only when $x_i^{(h)} \geq 1/f$, until we have $m_h \geq 2\ell$ when we select all remaining subsets, our algorithm incurs a weight of at most $f \sum_i w_i x_i + 2\ell \leq f\mathrm{OPT} + 2\ell$.

Finally, we argue that this ensures sufficient δ-slack at each step. Notice that, at the start, the set family consisting of the active sets (those with floating LP variables) and the elements to be covered is κ-degenerate. When any set S_i becomes rounded to 1, its elements are all covered and removed from the set family; if a set S_i has x_i rounded to 0, then any element $j \in S_i$ is removed from the set family by fixing its LP variable. Thus, whenever a variable corresponding to a set becomes integral, all of its elements are removed from the set family; this preserves the κ degeneracy of the induced sub-family. By an argument analogous to above lemma, the dimension is at most $(1-\delta)(n_h + m_h)$ for $\delta = 1/(\kappa + 2)$. □

We can now prove our main result, Theorem 4.

Theorem 4. *For any* κ*-degenerate instance of* FAIRSC-IF*, there is a rounding where the integral solution* (X, Y) *has a covering weight of at most* $f\mathit{OPT} + 2\ell$ *and satisfies* $\mathbb{E}[Y_j] \geq p_j$ *for all* $j \in \mathcal{P}$ *and is* β-PROBABLYINDFAIR *with* $\beta = \Theta(\kappa)$*.*

Proof. This follows from Lemma 8 and the β-concentration property of Theorem 2. □

F Algorithm from Indamdar et al.

Here, we describe the algorithm of Inamdar and Varadarajan [2018] for the partition set cover problem. They use an LP-based approach, that begins by solving a strengthened LP for the problem. Their LP consists of the constraints of (1) with the addition of exponentially-many *knapsack covering*

inequalities. To define these new inequalities, we first introduce some notation. Let $\text{cov}(A) = \bigcup_{S_i \in A} S_i$ denote the set of elements covered by the subsets in A. Then, let $\deg_c(S_i, A) = |(S_i \setminus \text{cov}(A)) \cap P_c|$ denote the number of elements of color c that are *newly covered* by adding set S_i to the partial cover A. We also define $k_c(A) = k_c - |\text{cov}(A) \cap P_c|$ to be the number of elements of color c that still need to be covered after selecting the subsets in A. Finally, we define $\deg'_c(S_i, A) = \min\{\deg_c(S_i, A), k_c(A)\}$. Then, the knapsack covering inequalities are:

$$\sum_{S_i \notin A} x_i \cdot \deg'_c(S_i, A) \geq k_t(A), \ \forall A \subseteq \mathcal{S}, c \in [\ell]. \tag{9}$$

We call the linear program resulting from the addition of these knapsack covering inequalities LP-PSC-KNAPSACK. Despite the fact that the LP with the knapsack covering inequalities has an exponential number of constraints, algorithmic techniques such as those employed in Inamdar and Varadarajan [2018] utilize a separation oracle that relies only on checking (9) for only a specific subset A. The separtion oracle also depends on "guessing" the optimal weight, which is found via binary search; the pseudocode for the separation oracle is given in Algorithm 3. Once given a fractional solution, they perform rounding as described in Algorithm 4.

Algorithm 3 Separation Oracle of Inamdar and Varadarajan [2018]

Require: Candidate Solution (x, y), Optimal weight Δ, Parameter $\alpha > 1$
Add $\sum_i w_i x_i \leq \Delta$ to LP-PSC-KNAPSACK
Let (x, y) be a candidate solution from the Ellipsoid Method
Check if $\sum_i w_i x_i \leq \Delta$ and return as violated constraint if not
Check constraints (1b), (**GF**), (1c), and (1d); return any violated constraint
Let $A = \{S_i : x_i \geq 1/6\alpha$
Check constraint (9) for set A and all colors $c \in [\ell]$; if any are violated, return them as a violated constraint.
If all tested constraints were satisfied, return `FEASIBLE`

Algorithm 4 Rounding Algorithm of Inamdar and Varadarajan [2018]

Let $l, h \leftarrow 0, \sum_i w_i$
while $l < 2h$ **do**
  Run the Ellipsoid Method using Algorithm 3 as the separation oracle, with $\Delta = (l + h)/2$, $\alpha = 6$
  if Ellipsoid returned `NOT FEASIBLE` **then**
    Let $l \leftarrow (l + h)/2$
  else
    Let (x, y) be the feasible solution found.
    $h \leftarrow (l + h)/2$
  end if
end while
Let (x, y) be the final feasible solution found.
Let $H := \{j : \sum_{S_i \ni j} x_i \geq 1/6\alpha\}$
Use a Set Cover algorithm to find an approximately optimal cover for $(H, \{S_i \cap H : i \in [m]\})$.
Let A be the cover obtained.
Let $A \leftarrow A \cup \{S_i : x_i \geq 1/6\alpha\}$
for Each set $S_i \notin A$ **do**
  for $r = 1, \ldots, c \log \ell$ **do**
    Let $X \sim \text{Ber}(\max\{6\alpha x_i, 1\})$
    if $X = 1$ **then**
      $A \leftarrow A \cup \{S_i\}$
    end if
  end for
end for
return A

G Experimental Results

Dataset Generation. We generate synthetic PARTITIONSC instances to evaluate the performance of our approximation algorithms. Each instance consists of a universe $\mathcal{P}$ of $n = 200$ elements, where each element is assigned one of three colors $\{1, 2, 3\}$ with color requirements $(50, 40, 40)$. The instance generation process creates a collection of 500 subsets, where each subset contains at most $d = 5$ elements and is assigned a random weight drawn uniformly from $[5.0, 10.0]$ which is later normalized such that the maximum weight of any subset is at most 1. Our algorithms are based on the parameter f, the maximum frequency of any element. Therefore, our experiments follow a similar setup where we consider the family of instances with frequency parameter where $f \in \{3, \ldots, 10\}$, ensuring that no element appears in more than f subsets. Color assignments are randomized after ensuring minimum color requirements are met (to ensure the instance feasibility), with remaining elements distributed according to a color distribution $(0.33, 0.33, 0.34)$. For each element j, p_j is drawn uniformly from the interval $[0, 1]$. For each frequency value, we generate 100 instances in our experimental results.

Experiment Setup. We evaluate our FAIRITERATEDROUNDING algorithm against the INAMDARROUNDING (pseudocode can be found in Section A) baseline across instances with varying maximum frequency $f \in \{3, 4, \ldots, 10\}$. Both algorithms are evaluated on 800 instances in total across all the f values. We compute the approximation ratios, number of elements that satisfy the individual fairness constraints (**IF**) (averaged over 100 runs of the rounding algorithm for each instance), and runtime performance to assess the trade-offs between cost of the covering, efficiency and fairness guarantees. We run the experiments using Python 3.12 and LPs are solved using Gurobi 12.

Experiment Results And Analysis. Across all values of f, our proposed algorithm, FAIRITERATEDROUNDING, consistently produces solutions with an average covering cost 50% lower than those from INAMDARROUNDING which is the best known algorithm of PARTITIONSC without the Eq. (**IF**) constraints. As expected from their theoretical guarantees, both algorithms successfully satisfy the group fairness constraints in all instances i.e., the violations in group fairness is 0 across the board for both algorithms. A key difference is that INAMDARROUNDING does not take the

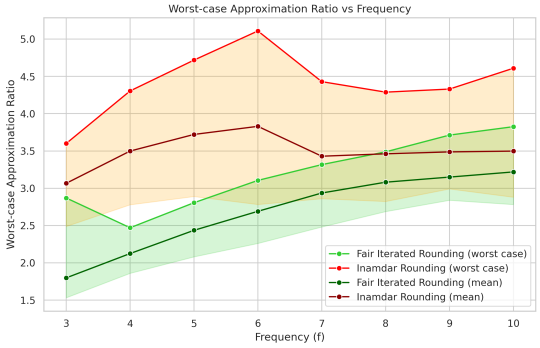

Figure 1: Comparing the approximation ratios of FAIRITERATEDROUNDING vs. INAMDARROUNDING

individual coverage requirements (**IF**) into consideration while selecting the subsets, but has a more conservative approach to selecting the subsets thereby satisfying coverage requirements for close to 100% of elements on an average, but this comes at a cost, as reflected in its higher objective values and approximation ratios. In contrast, FAIRITERATEDROUNDING achieves lower costs by satisfying the coverage requirements for close to 90–95% of elements on average, which improves with more rounds. Finally, our experiments show that FAIRITERATEDROUNDING also has a substantially lower running time, making it significantly better performing overall. Additional experiments can be found in Section G

COMPAS Dataset We generate the instances of FAIRSC-IF from the COMPAS dataset as following:

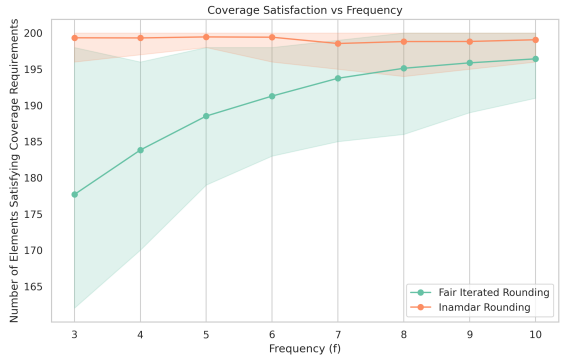

Figure 2: No. of elements satisfying coverage requirements for FAIRITERATEDROUNDING vs. INAMDARROUNDING.

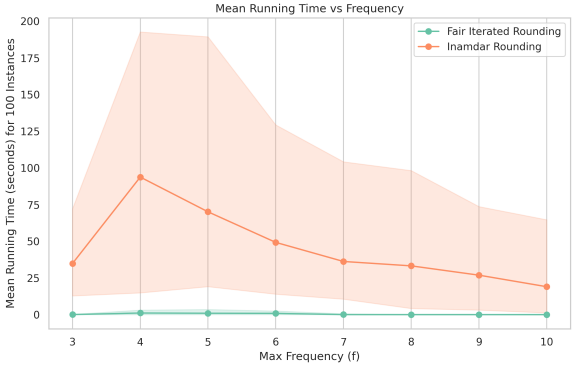

Figure 3: Running time for 100 runs across 100 instances FAIRITERATEDROUNDING vs. INAMDAR-ROUNDING.

1. **Data Cleaning**: First, we filter out any rows that have missing age values. We also restrict our attention to the following (`race`, `sex`) pairs: (`Black`, `Male`), (`Hispanic`, `Male`), (`Caucasian`, `Male`), (`Caucasian`, `Female`), (`Black`, `Female`), and (`Hispanic`, `Female`). We filter out any rows that do not have one of these (`race`, `sex`) pairs. The resulting dataset is the `cleaned-dataset`.

2. **Instance Creation (for a given frequency f)**:

   We consider three "groups" corresponding to the (`race`, `sex`) pairs: one group consists of individuals who are (`Black`, `Male`) or (`Hispanic`, `Male`); the second is (`Caucasian`, `Male`) or (`Caucasian`, `Female`); and the third is (`Black`, `Female`) or (`Hispanic`, `Female`).

   The motivating scenario for our instances is a situation where some subset of individuals from a given age group can be interviewed on any given day, and our goal is to interview a certain number of individuals from each of our groups. Thus, in our generated instances, the elements to be covered are individuals, and the subsets each correspond to some collection of individuals within the same age bracket. The individuals are grouped by age into one of four groups, corresponding to the ages 18–25, 26–35, 36–45, and over 46.

   To generate our instances, we first sample $n = 100$ rows at random from the `cleaned-dataset` to get the set of individuals to be covered. Then, individuals are grouped by age group, and we randomly generate 200 subsets of each age group, ensuring that no individual is sampled and included in more than f subsets. Each subset is given a weight between 5 and 10.

   We generate 30 instances in this way and run our code on these instances.

Experiment Results As with the synthetic datasets, we evaluate our algorithm against the Inamdar Rounding algorithm on the aforementioned FAIRSC-IF instances generated from the COMPAS dataset. We find that our algorithm significantly outperforms Inamdar's for

smaller values of f. However, the gap in approximation ratio narrows as f increases. This behavior is expected, since for larger f, our algorithm rounds a greater number of subsets in each iteration of CoverRound converging to the Inamdar's approach.

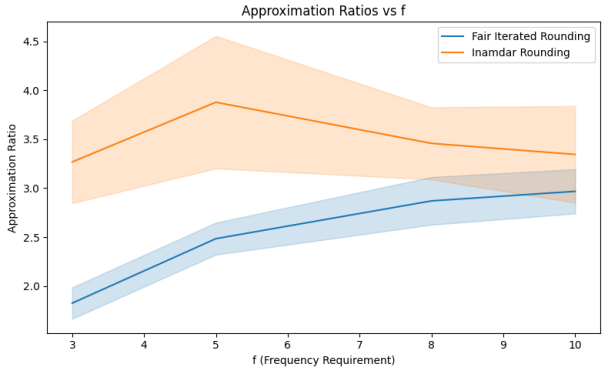

Figure 4: Approximation Ratio vs. Max. Frequency

The conservative approach of Inamdar selects all elements, thereby covering 100% of them, while our algorithm covers at least 90% of the elements on average. As expected, as the value of f increases, our algorithm gradually converges to Inamdar's conservative strategy, ultimately covering nearly 100% of the elements.

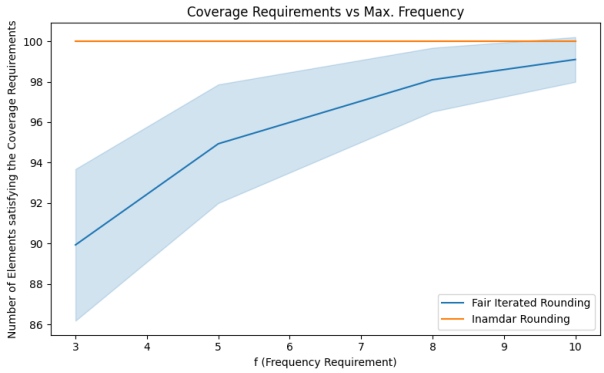

Figure 5: Coverage Satisfaction vs. Max. Frequency for Fair Iterated Rounding vs Inamdar Rounding

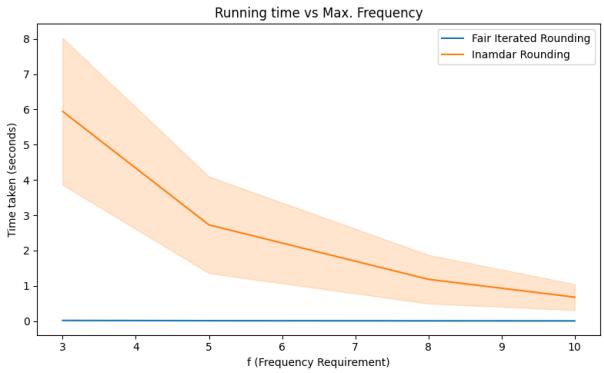

Figure 6: Approximation Ratios vs. Max. Frequency for Fair Iterated Rounding vs Inamdar Rounding

Finally, we highlight that our algorithm achieves significantly lower running time compared to the Inamdar Rounding algorithm, which becomes a bottleneck for the instance sizes we

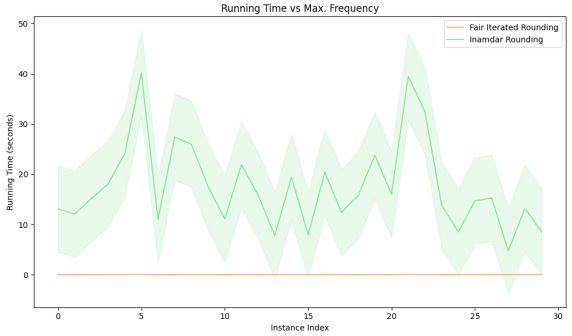

Figure 8: Running time for Fair Iterated Rounding vs Inamdar Rounding for $f = 3, n = 150, m = 250$.

consider (see Fig. 8). This efficiency allows us to run our algorithm on substantially larger instances, where it continues to demonstrate superior performance (see Fig. 7).



Figure 7: Approximation Ratios for Fair Iterated Rounding vs Inamdar Rounding for $f = 3, n = 150, m = 250$.

NeurIPS Paper Checklist

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While [Yes] is generally preferable to [No], it is perfectly acceptable to answer [No]

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We provide full proofs and experimental results to support our claims.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Learning the coverage requirements p_j (or general success/acceptance thresholds) for individuals $j \in \mathcal{P}$ is not within the scope our paper and we assume these quantities are known. We discuss this limitation along with some existing methods (e.g., ML predictions) to estimate these p_j values.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: A full set of proofs and assumptions are provided for all the claims made in the work.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide the data generation method for the synthetic data along with random seed. We provide code in the supplementary materials.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: Supplemtary materials are provided for code along with instructions

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [N/A]

Justification: No training, test or validation data and model hyperparameters.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [N/A]

   Justification: [N/A]

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification:We provide the system requirements.

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: This paper is submitted anonymously.

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The paper's main contribution is theoretical, although the problem itself is motivated from wide-spread applications such as facility location, vaccine distribution etc., The parameters used in the problem such as group coverage requirements $(k_c)_{c\in[\ell]}$ and individual coverage requirements $(p_j)_{j\in\mathcal{P}}$ are both assumed to be given. However, in practice, these values must be estimated to fit an underlying fairness policy - which itself is a challenging problem.

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: [N/A]

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All the authors names is provided accurately.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We do provide the details about the datasets used in the empirical analysis, and provide the full code to generate the experimental results along with comments.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: [N/A]

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: [N/A]

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: We use LLMs for improving the writing, editing the grammar and presentation purposes. We do not rely on LLMs for the original ideas or scientific contributions in the paper such as problem formulation, algorithm design, analysis, and extensions.