Fair and Optimal Cohort Selection for Linear Utilities

Konstantina Bairaktari[1] Huy Le Nguyen[1] Jonathan Ullman[1]

[1] *Khoury College of Computer Sciences, Northeastern University*

February 16, 2021

Abstract

The rise of algorithmic decision-making has created an explosion of research around the fairness of those algorithms. While there are many compelling notions of individual fairness, beginning with the work of Dwork et al. [DHP+12], these notions typically do not satisfy desirable composition properties. To this end, Dwork and Ilvento [DI19] introduced the *fair cohort selection problem*, which captures a specific application where a single fair classifier is composed with itself to pick a group of candidates of size exactly k. In this work we introduce a specific instance of cohort selection where the goal is to choose a cohort maximizing a linear utility function. We give approximately optimal polynomial-time algorithms for this problem in both an offline setting where the entire fair classifier is given at once, or an online setting where candidates arrive one at a time and are classified as they arrive.

1 Introduction

The rise of algorithmic decision-making has created an explosion of research around the fairness of those algorithms. Beginning with the seminal work of Dwork, Hardt, Pitassi, Reingold, and Zemel [DHP+12], there is now a large body of algorithms that preserve various notions of fairness for individuals. These notions of individual fairness capture the principle that similar people should be treated similarly, according to some task-specific measure of similarity.

While these algorithms satisfy compelling notions of individual fairness in isolation, they will not be used in isolation, and will actually be used as parts of larger systems. To address these broader context, Dwork and Ilvento [DI19] initiated the study of fairness under composition. Their work introduced a number of models where fair mechanisms must be composed, demonstrate that naïve methods of composing fair algorithms do not preserve fairness, and identify new strategies.

In this work we consider the *fair cohort selection problem* introduced in [DI19]. In this problem, we would like to select k candidates for a job from a universe of n possible candidates. We are given a classifier that assigns a score $s_u \in [0, 1]$ to each candidate,[1] with the guarantee that the scores are individually fair with respect to some similarity measure $\mathcal{D}$. We would like to select a set a cohort of k candidates such that the probability p_u of selecting any candidate should be fair with respect to the same metric.

We can trivially solve the fair cohort selection problem by ignoring the scores and selecting a uniformly random set of k candidates, so that every user is chosen with the same probability k/n. Thus Smedemark-Margulies, Langton, and Nguyen [SMLN20] proposed to consider fair algorithms for cohort selection that optimize some utility function related to the scores, and construct algorithms for a particular choice of utility function.

In this work we consider a setting where the utility function is known and simple, specifically we assume a *linear utility function* where the utility of a cohort is the sum of the scores of all candidates in the cohort: $\sum_{u=1}^{n} p_i s_i$. This type of utility implicitly assumes that the scores s_u are themselves doing a good job of approximating the utility of a candidate, and that there are no complements or supplements among participants in the cohort.

KB is supported by NSF award CCF-1750640. HLN is supported by NSF awards CCF-1909314 and CCF-1750716. JU is supported by NSF awards CCF-1750640, CNS-1816028, and CNS-1916020.

[1] In the model of Dwork et al. [DHP+12], fairness requires that the classifier be randomized, so we can think of the scores as the probabilities

Our technical contributions are polynomial-time algorithms for fair and useful cohort selection with linear utilities, in two different algorithmic models.

Offline Setting. In this model, we are given all the scores $s_1, \ldots, s_n$ at once, and must choose the optimal cohort. We present a polynomial-time algorithm that computes a fair cohort with optimal exdpected utility.

Online Setting. In this model we are given the scores $s_1, s_2, \ldots$ as a stream. Ideally, after receiving s_u, we would like to immediately accept candidate u to the cohort or reject them from the cohort. However, this goal is too strong, so we consider a relaxation where candidate u can be either accepted, rejected, or kept on hold until later, and we would like to output a fair cohort with optimal expected utility while minimizing the number of candidates who are kept on hold. We give an *approximately fair and optimal* algorithm in this online model. Specifically, we present a polynomial-time algorithm for this online setting where the fairness constraints are satisfied and utility is optimized up to an additive error of ε, for any desired $\varepsilon > 0$, while ensuring that the number of users on hold never exceeds $2k + \frac{1}{\varepsilon}$.

To interpret our additive error guarantee, since the fairness constraint applies to are the *probabilities* $p_1, \ldots, p_n$, allowing the fairness constraint to be violated by an additive error of, say, $\varepsilon = 0.01$ means that every candidate is selected with probability that is within ±1% of the probability that candidate would have been selected by some fair mechanism.

1.1 Techniques

We start with a brief overview of the key steps in our algorithms for each model.

Offline Setting. Our offline algorithm is based on two main properties of the fair cohort selection problem. First, we use a dependent-rounding procedure from [SMLN20] that takes a set of probabilities $p_1, \ldots, p_n$ such that $\sum_{i=1}^{n} p_i = k$ and outputs a random cohort C, represented by an indicator vector $\tilde{p}_1, \ldots, \tilde{p}_n$ with $\sum_{i=1}^{n} \tilde{p}_i = k$ such that $\mathbb{E}(\tilde{p}_u) = p_u$ for every candidate u. Thus, it is enough for our algorithm to come up with a set of marginal probabilities for each candidate to appear in the cohort, rather than directly finding the cohort.

To find the marginal probabilities that maximize utilities with respect to the scores $s_1, \ldots, s_n \in [0, 1]$, we would like to solve the linear program

$$\begin{aligned} \text{maximize} \quad & \textstyle\sum_{i=1}^{n} p_i s_i \\ \\ \text{s.t.} \quad & \textstyle\sum_{i=1}^{n} p_i \leq k \\ & \forall i, j\ |p_i - p_j| \leq |s_i - s_j| \\ & \forall i\ 0 \leq p_i \leq 1 \end{aligned}$$

In this LP, the first and third constraints ensure that the variables p_u representing the marginal probability of selecting candidate u in a cohort of size k. The second constraint ensures that these probabilities p are fair with respect to the same measure $\mathcal{D}$ as the original scores s. Specifically, $|p_u - p_v| \leq |s_u - s_v| \leq \mathcal{D}(u, v)$. Although we could have used the stronger constraint $|p_u - p_v| \leq \mathcal{D}(u, v)$, writing the LP this way means that our algorithm doesn't need to know the underlying measure, and means our solution will preserve any stronger fairness that the scores s happen to satisfy.

While we could solve this linear program explicitly, we can get a faster solution that is more useful for extending to the online setting by noting that this LP has a specific closed form based on "water filling." Specifically, if $\sum_{i=1}^{n} s_i \leq k$, then the optimal solution simply adds some number $c \geq 0$ from all scores and sets $p_u = \min\{s_u + c, 1\}$, and an analogous solution works when $\sum_{i=1}^{n} s_i > k$.

Online Setting. In the online model we do not have all the scores in advance, thus we cannot determine the solution to the linear program, and do not even know the value of the constant c that determines the solution. We give two algorithms for addressing this problem. The basic algorithm begins by introducing some *approximation*, in which we group users into $1/\varepsilon$ groups based on their scores, where g contains users with scores in $[g\varepsilon, (g+1)\varepsilon]$. This grouping can only reduce utility by ε, and can only lead to violating the fairness constraint by ε. Since users in each bucket are treated identically, we know that when we reach the end of the algorithm, we can express the final cohort as containing a random set of n_g members from each group g. Thus, to run the algorithm we use *reservoir sampling* to maintain a random set of at most k candidates from each group, reject all other members, and then run the offline algorithm at the end to determine how many candidates to select from each group.

The drawback of this method is that it keeps as many as k/ε candidates on hold until the end. Thus, we develop an improved algorithm that solves the linear program in an online fashion, and uses the information it obtains along the way to more carefully choose how many candidates to keep from each group. This final algorithm reduces the number of candidates on hold by as much as a quadratic factor, down to $2k + \frac{1}{\varepsilon}$.

1.2 Related Work

Our work fits into the line of work initiated by Dwork et al. [DHP+12] on *individual fairness*, which imposes constraints on how algorithms may disthttps://www.overleaf.com/project/5fc65043b2793a1695729a9dinguish specific individuals. Within this framework, issues with composing fair algorithms were first explored by Dwork and Ilvento [DI19], and later studied by [DIJ20] and [SMLN20].

A complementary line of work, initiated by Hardt, Price, and Srebro [HPS16] considers notions of *group fairness*, which imposes constraints on how algorithms may distinguish in aggregate between individuals from different groups. While our work is technically distinct from work on group privacy, issues of composition also arise in that setting, as noted in several works [BKN+17, KRZ19, AKRZ21].

2 Preliminaries

2.1 Fair Cohort Selection

For our model, we consider a set of n individuals U and a binary classifier C, that chooses individual $u \in U$ with probability p_u. The classifier outputs 1 when the individual is chosen and 0 otherwise. If we restrict the definition of individual fairness from to this particular model, we obtain the following definition.

Definition 1 (Individual Fairness [DHP+12])**.** Given a metric $\mathcal{D}$ over the individuals of set U and a randomized binary classifier C with outputs in $\{0, 1\}$ that assigns selection probability p_u to any individual u in U, we say that the classifier is *individually fair* if and only if for all u, v in U,

$$\mathcal{D}(u, v) \geq |p_u - p_v|.$$

In this work, we want to select a cohort of exactly k individuals out of the set U, by consulting an individually fair classifier C that assigns individual selection probabilities $s_1, s_2, \dots, s_n$ to the n individuals. We are interested in maintaining the fairness property for the probabilities of selection for the cohort. The general setting of this problem was defined in [**?**]. At the same time, we attempt to eliminate trivial solutions that would be fair, such as a uniform probability distribution. To achieve this, we define a linear utility function of the probabilities of selection, which we want to maximize during the cohort selection.

Definition 2 (Utility)**.** Given a set of n candidates U, a randomized binary classifier C that assigns individual selection probabilities $s_1, s_2, \dots, s_n$ to the members of U and a cohort selection algorithm $\mathcal{S}$ which chooses individuals with probabilities $p_1, p_2, \dots, p_n$, we define the utility to be

$$\sum_{i=1}^{n} p_i s_i.$$

We consider two variations of the cohort selection problem. The first one is the offline cohort selection, where the algorithm has full access to the set U and makes decisions offline. The second one is the streaming version of the cohort selection problem, for which we use a relaxation of the individual fairness.

Definition 3 (ε-Individual Fairness)**.** Given a metric $\mathcal{D}$ over the individuals of set U, a randomized binary classifier C with outputs in $\{0, 1\}$ that assigns selection probabilities p_u to any individual in U and $0 \leq \varepsilon \leq 1$, we say that the classifier is *ε-individually fair* if and only if for all u, v in U,

$$\mathcal{D}(u, v) + \varepsilon \geq |p_u - p_v|.$$

Now, we have the tools we need to define the problem we study in the following sections.

Definition 4 (Fair and Useful Cohort Selection Problem)**.** Given a set of n individuals U and the probabilities of selection an individually fair classifier C assigns to the members of U with respect to a metric $\mathcal{D}$ (and a constant ε in $(0, 1]$), choose a cohort of k individuals such that:

1. the probability of selection for the cohort is individually fair (or ε-individually fair) with respect to $\mathcal{D}$, and
2. it achieves the optimal cohort selection utility.

Since the input and the output probabilities of the cohort selection problem satisfy fairness conditions for the same metric, it will be implied in the theorems of the following sections that the result of the cohort selection is fair with respect to the same metric as the input.

2.2 A Rounding Algorithm

Both the offline and the streaming solutions for the cohort selection problem will be based on a dependent rounding algorithm that solves the offline fair and useful cohort selection problem when the sum of the input scores of all the candidates is equal to the number of people we want to choose k. The output of this algorithm consists of indicators $\tilde{s}_i$ which are 1 if the i-th candidate is selected to be part of the cohort and 0 otherwise. This rounding procedure is a special case of rounding a fractional solution in a matroid polytope (in this case, we have a uniform matroid). This problem has been studied extensively with rounding procedures satisfying additional desirable properties (see e.g. [CVZ10]). Here we describe a simple and very efficient rounding algorithm for the special case of the problem arising in our work.

Lemma 1 (see e.g. [SMLN20])**.** Let $s_1, s_2, \ldots, s_n \in [0, 1]$ be a list of scores with $\sum_{i=1}^{n} s_i = k \in \mathbb{N}$. Algorithm 1 outputs randomized $\tilde{s}_1, \ldots, \tilde{s}_n \in \{0, 1\}$ such that k elements will be equal to 1, the rest will be equal to 0 and for all $i \in \{1, \ldots, n\}$, $\mathbb{E}[\tilde{s}_i] = s_i$.

2.3 Reservoir Sampling

For the streaming algorithm which solves the cohort selection problem we use a procedure called random reservoir sampling. In particular, we want to maintain an upper-bounded number of people in the memory. We do this by choosing a subset of people uniformly at random if the number of people exceeds a constant and reject the rest. However, since our algorithm works in the streaming setting, we need a sampling method which creates the sample on the fly.

3 Algorithms for Offline Cohort Selection

In this version of the problem, the algorithm has full access to the set U and makes decisions offline. We can formalize it as the following constrained utility maximization problem where we want to compute the selection probabilities for all n individuals.

$$\begin{array}{rl} \text{Maximize} & \sum_{i=1}^{n} p_i s_i \\ \\ \text{s.t.} & \sum_{i=1}^{n} p_i \leq k \\ & \forall i, j \ |p_i - p_j| \leq |s_i - s_j| \\ & \forall i \ 0 \leq p_i \leq 1 \end{array}$$

where $\forall i \ 0 \leq s_i \leq 1$. By lemma 1, we have that algorithm 1 can receive as input the list of individual cohort selection probabilities and form a cohort that respects these probabilities. Nevertheless, the sum of the initial probabilities generated by classifier C might not be equal to k. If the sum is greater than k, then the new selection probabilities become $p_i = \min\{s_i + c, 1\}$, where the constant c is calculated so that $\sum_{i=1}^{n} p_i = k$. This adjustment maintains the probability differences between the pairs of candidates, unless one of them becomes 1, in which case the difference will become smaller. As a result, the differences of the new probabilities will remain bounded by the same metric as the initial probabilities. The case where the sum is less than k is treated similarly. More specifically, the new probabilities are $p_i = \min\{0, s_i - c\}$. After the adjustment, algorithm 1 can decide which people will constitute the cohort based on the probabilities of the input.

Algorithm 1: Rounding

```
Input: list of scores $s_1, s_2, \ldots, s_n \in [0, 1]$ for the $n$ candidates with $\sum_{i=1}^{n} s_i = k$
Output: list of selection indicators $\tilde{s}_1, \tilde{s}_2, \ldots, \tilde{s}_n \in \{0, 1\}$

pendingIndex ⟵ 2
for $i$ from 1 to $n$ do
    if $i$ = pendingIndex then
        continue to next $i$
    end
    $a \longleftarrow s_i$
    $b \longleftarrow s_{\text{pendingIndex}}$
    choose $u$ randomly from $unif(0, 1)$
    if $a + b \leq 1$ then
        if $u < \frac{a}{a+b}$ then
            $s_i \longleftarrow a + b$
            $s_{\text{pendingIndex}} \longleftarrow 0$
            pendingIndex ⟵ $i$
        else
            $s_i \longleftarrow 0$
            $s_{\text{pendingIndex}} \longleftarrow a + b$
        end
    else
        if $u < \frac{1-b}{2-a-b}$ then
            $s_i \longleftarrow 1$
            $s_{\text{pendingIndex}} \longleftarrow a + b - 1$
        else
            $s_i \longleftarrow a + b - 1$
            $s_{\text{pendingIndex}} \longleftarrow 1$
            pendingIndex ⟵ $i$
        end
    end
end
return $s_1, s_2, \ldots, s_n$
```

Algorithm 2: Offline Cohort Selection

```
Input: list of scores $s_1, s_2, \ldots, s_n \in [0,1]$ for the $n$ candidates,
       the number of individuals that must be selected $k$
Output: list of selection indicators $\tilde{s}_1, \tilde{s}_2, \ldots, \tilde{s}_n \in \{0,1\}$

sum $\longleftarrow \sum_{i=1}^{n} s_i$
if sum $< k$ then
    $c \longleftarrow \frac{k - \text{sum}}{n}$
    $p_i \longleftarrow s_i + c, \forall i \in [n]$
    while $\exists p_i > 1$ do
        set $n_{<1}$ equal to the number of individuals with $p_j < 1$
        $p_j \longleftarrow p_j + \frac{p_i - 1}{n_{<1}}, \forall j : p_j < 1$
        $p_i \longleftarrow 1$
    end
else
    $c \longleftarrow \frac{\text{sum} - k}{n}$
    $p_i \longleftarrow s_i - c, \forall i \in [n]$
    while $\exists p_i < 0$ in $P$ do
        set $n_{>0}$ equal to the number of individuals with $p_j > 0$
        $p_j \longleftarrow p_j + \frac{p_i}{n_{>0}}, \forall j : p_j > 0$
        $p_i \longleftarrow 0$
    end
end
return Rounding($p_1, \ldots, p_n$)
```

Theorem 1. Given individually fair scores $s_1, s_2, \ldots, s_n$ for a set of n candidates, Algorithm 2 solves the Offline Cohort Selection problem by selecting k individuals with marginal probabilities $p_1, p_2, \ldots, p_n$ that:

1. are individually fair, and
2. optimize the cohort selection utility function.

Proof. We can rewrite the procedure that Algorithm 2 runs before the rounding in a succinct way. In particular, if the sum is less than k then we have $p_i = \min\{s_i + b, 1\}$, for a real number b so that $\sum_{i=1}^{n} p_i = k$. The value of the sum remains the same after the initialization of the p_is because the algorithm only redistributes the mass between individuals. All p_is that are not set to 1 are equal to the sum of s_i plus the same constant that consists of the initial c and the fractions of the p_is that exceeded 1 and, therefore, were set to 1. In the end no p_i can be greater than 1 because then its value will be set to 1 and the remaining mass will be uniformly redistributed to all p_js that are less than 1. Similarly, if the sum is greater than k the formula is $p_i = \max\{s_i - b, 0\}$, for a real number b so that $\sum_{i=1}^{n} p_i = k$.

Individual fairness The algorithm adjusts the scores of the individuals according the value of the $\sum_{i=1}^{n} s_i$ before running the rounding algorithm. Hence, there are three distinct cases.

1. $\sum_{i=1}^{n} s_i = k$. The input scores are used unaltered as the marginal probabilities for the cohort selection and, thus, we have for any pair of indices different i, j

$$|p_i - p_j| = |s_i - s_j|.$$

2. $\sum_{i=1}^{n} s_i < k$. The cohort selection probabilities are of the form $p_i = \min\{s_i + b, 1\}$. For any pair of individuals i, j if both have probability of being selected 1 then $|p_i - p_j| = 0$.Else if without loss of generality $p_i = 1$ and $p_j = s_j + b$, we have $|p_i - p_j| < |s_i - s_j|$. Finally, if $p_i = s_i + b$ and $p_j = s_j + b$, then it holds that $|p_i - p_j| = |s_i - s_j|$. Combining these, we conclude that for any pair of individuals i, j we have

$$|p_i - p_j| \leq |s_i - s_j|.$$

3. $\sum_{i=1}^{n} s_i > k$. Now, the cohort selection probabilities are of the form $p_i = \max\{s_i - b, 0\}$. For any pair of individuals i, j if both have zero probability of being selected then $|p_i - p_j| = 0$.Else if without loss of generality $p_i = 0$ and $p_j = s_j - b$, we have $|p_i - p_j| < |s_i - s_j|$. Finally, if $p_i = s_i - b$ and $p_j = s_j - b$, then it holds that $|p_i - p_j| = |s_i - s_j|$. We conclude that for any pair of individuals i, j we have

$$|p_i - p_j| \le |s_i - s_j|.$$

Therefore, the individual fairness is ensured by the assumption of the theorem that the input scores satisfy the individual fairness condition. More specifically, if the scores are individually fair with respect to a metric $\mathcal{D}$, we obtain $\forall i, j \in \{1, \ldots, n\}$

$$|p_i - p_j| \le |s_i - s_j| \le \mathcal{D}(i, j).$$

Optimal utility. Let p_i for $i \in \{1, \ldots, n\}$ denote the individual selection probabilities P of the solution of Algorithm 2. For the optimal solution the first constraint that refers to the sum is satisfied with equality. This happens because otherwise we would be able to increase the p_is by setting $p_i' = \min\{p_i + d, 1\}$ so that $\sum_{i=1}^{n} p_i' = k$ and obtain greater utility while not violating any constraint. We assume that there exists a different solution P' which is also individually fair and it is the optimal one, i.e. $\sum_{i=1}^{n} p_i' s_i > \sum_{i=1}^{n} p_i s_i$. Without loss of generality we can assume that the original scores s_i are sorted in increasing order. In particular, we have

$$s_1 \le s_2 \le \ldots \le s_n.$$

The solution of the algorithm above maintains the same order.

$$p_1 \le p_2 \le \ldots \le p_n.$$

Let i be the individual with the smallest index for whom the two solutions differ. There are two possible cases.

If $p_i' < p_i$, then because $\sum_{i=1}^{k} p_i = \sum_{i=1}^{k} p_i' = k$ there exists an individual with $j > i$ such that $p_j' > p_j$. In addition $p_i > 0$ and $p_j < 1$. The values of the probabilities depend on the sum of the scores in comparison to k. If the sum is greater than k, all the p_is that are not equal to 0 are of the form $p_i = s_i - b$. Hence, considering that $p_i' < p_i \le p_j < p_j'$ we obtain

$$|p_i' - p_j'| = p_j' - p_i' > s_j - b - (s_i - b) = s_j - s_i = |s_i - s_j|.$$

Similarly, if the sum is less than k, the p_is that are not equal to 1 are of the form $p_i = s_i + b$, therefore giving

$$|p_i' - p_j'| = p_j' - p_i' > s_j + b - (s_i + b) = s_j - s_i = |s_i - s_j|.$$

If the sum is equal to k, we have

$$|p_i' - p_j'| = p_j' - p_i' > p_i - p_j = s_j - s_i = |s_i - s_j|.$$

In all three cases one of the constraints of the optimization is violated.

If $p_i' > p_i$, then there exists an individual with $j > i$ such that $p_j' < p_j$. Let j be the smallest such index. If there exists another index $l > j$ such that $p_l' \ge p_l$, then by the previous argument the constraints are not satisfied. Therefore, for all individuals l after j it must hold that $p_l' < p_l$. We can now separate the individuals in three groups:

$$G_1(P') = \{i : p_i = p_i'\}$$
$$G_2(P') = \{i : p_i' > p_i\}$$
$$G_3(P') = \{i : p_i' < p_i\}.$$

All individuals in G_3 have greater indices than those in G_1 and G_2. We can now take mass uniformly from the individuals in G_2 and distribute it uniformly to individuals in G_3 in a way that all individuals remain in the same group as before. Let p_i'' be the new score of individual i. The value of the objective function will increase because individuals in G_3, who have higher s_is than those in G_2, will get higher new scores. The constraints are still satisfied because the total sum remains the same and for any pair of individuals i, j we have $|p_i'' - p_j''| \le |p_i' - p_j'|$. By constructing a solution P'' which gives a greater objective than P', we arrive at a contradiction. As a result, P is the optimal solution. □

4 Algorithms for Streaming Cohort Selection

In this section, we consider the streaming setting of the cohort selection problem. Specifically, we propose algorithms 3 and 5 that read the initial scores from a stream and solve the cohort selection problem while achieving high utility and keeping a small number of people in the memory. In particular, due to the result in [DI19], which states that if the number of individuals in the input is unknown the online version of the cohort selection problem has no solution, a streaming algorithm cannot choose a cohort without having seen the entire stream. However, it can make the process more efficient for the candidates by rejecting some candidates throughout the process. We say that a candidate is pending if they have not yet been rejected. It is important to note though that no person is accepted before the algorithm reaches the end of the stream.

Both algorithms described in this section divide the people into groups with similar initial selection scores and treat any member of a group as equivalent. Hence, the final probability of being selected for the cohort is equal for any person assigned to the same group. This leads to a relaxation of the individual fairness property achieved, which is the reason why we defined ε-individual fairness. Even though for each group we keep a number of people pending and reject everyone else, we maintain the information of the rejected candidates by considering that each person pending represents themselves as well as a fraction of the rejected people from their group. The fraction of people represented by the i-th candidate is denoted by $n_i \geq 0$. Algorithm 3 maintains at most k people pending per group uniformly at random and every representative within a certain group represents the same fraction of people. Algorithm 5 reduces the memory required by allowing pending candidates of the same group to represent different numbers of people. Algorithm 4 is used to eliminate people and determine how many people each candidate represents.

4.1 Grouping by Scores

We split the people we see into groups according to their initial score. For example we can split the interval $[0, 1]$ into m intervals of size ε each, where $m = \lceil \frac{1}{\varepsilon} \rceil$. Person i is assigned to group $g \in \{1, \dots, m\}$, if $s_i \in [(g-1)\varepsilon, g\varepsilon]$. Once person i is in the group, they get a new score $\hat{s}_i$, equal to the score of all other members in the group $\hat{s}^g = g\varepsilon$. The use of the groups affects the performance of the cohort selection algorithm in terms of individual fairness and utility. Lemma 2 shows that the performance compromise of the streaming algorithms 3 and 5 is caused by the use of groups, as it is also observed in the offline algorithm 2 when the input scores are grouped in multiples of ε.

Lemma 2. Given individually fair scores $s_1, s_2, \dots, s_n$ for a set of n candidates and $\varepsilon \in (0, 1]$, we split the interval $[0, 1]$ into $m = \lceil \frac{1}{\varepsilon} \rceil$ intervals of length ε and for all $i \in \{1, \dots, n\}$ we set $\hat{s}_g = g\varepsilon$ if $s_i \in [(g-1)\varepsilon, g\varepsilon]$. Algorithm 2 with input the modified scores $\hat{s}_1, \dots, \hat{s}_n$ solves the Cohort Selection problem by choosing a cohort of size $k \in \mathbb{N}$ with individual selection probabilities $p_1, p_2, \dots, p_n$ that:

1. are ε-individually fair
2. achieve cohort selection utility $OPT - k\varepsilon \leq \sum_{i=1}^{n} p_i s_i \leq OPT$, where the optimal utility is with respect to the original scores $s_1, s_2, \dots, s_n$.

Proof. We have that for all individuals i in $\{1, \dots, n\}$ $0 \leq \hat{s}_i - s_i \leq \varepsilon$. Therefore, we obtain that for any pair of individuals i, j $|\hat{s}_i - \hat{s}_j| \leq |s_i - s_j| + \varepsilon$. By Theorem 1 we have that $|p_i - p_j| \leq |\hat{s}_i - \hat{s}_j|$. Combining the two inequalities, we obtain that the selection process is ε-individually fair.

We denote by $p_1^*, \dots, p_n^*$ the optimal solution that offers utility $\sum_{i=1}^{n} p_i^* s_i \geq \sum_{i=1}^{n} p_i s_i$. The utility achieved by the algorithm is

$$\begin{aligned}\sum_{i=1}^{n} p_i s_i &= \sum_{i=1}^{n} p_i \hat{s}_i + \sum_{i=1}^{n} p_i (s_i - \hat{s}_i) \\ &\geq \sum_{i=1}^{n} p_i \hat{s}_i - \sum_{i=1}^{n} p_i \varepsilon \qquad (\hat{s}_i - s_i \leq \varepsilon, \forall i \in \{1, \dots, n\}) \\ &= \sum_{i=1}^{n} p_i \hat{s}_i - k\varepsilon \qquad (\sum_{i=1}^{n} p_i = k) \\ &\geq \sum_{i=1}^{n} p_i^* \hat{s}_i - k\varepsilon \qquad (\sum_{i=1}^{n} p_i \hat{s}_i \text{ is the optimal utility for scores } \hat{s}_i).\end{aligned}$$

Since we want a final result that involves $\sum_{i=1} p_i^* s_i$, we can rewrite $\sum_{i=1}^{n} p_i^* \hat{s}_i$ as

$$\begin{aligned}\sum_{i=1}^{n} p_i^* \hat{s}_i &= \sum_{i=1}^{n} p_i^* s_i + \sum_{i=1}^{n} p_i^* (\hat{s}_i - s_i) \\ &\geq \sum_{i=1}^{n} p_i^* s_i \qquad (\hat{s}_i - s_i \geq 0, \forall i \in \{1, \dots, n\}).\end{aligned}$$

Thus, we see that $\sum_{i=1}^{n} p_i s_i \geq \sum_{i=1}^{n} p_i^* s_i - k\varepsilon$. □

4.2 Basic algorithm

This algorithm adds each new person to the appropriate group and maintains at most k people from each group using reservoir sampling if the size of the group exceeds k. If there are more than k people in one group, each of the people from this group who are not rejected represent $\frac{1}{k}$ of the size of the group. For all people that are waiting for the decision, the algorithm keeps the score and the number of people they represent. When the stream ends, the algorithm adjusts the scores of the remaining people so that the sum of their scores is equal to k. The process followed is equivalent to that of the offline algorithm. To be more precise, it is modified to work with the limited information kept by the streaming algorithm so that the expected value of any score is equal to the value computed by the corresponding offline version.

Theorem 2. Given individually fair scores $s_1, s_2, \dots, s_n$ for a set of n candidates, the streaming algorithm 3 solves the cohort selection problem for any $\varepsilon \in (0, 1]$ by choosing a cohort with individual selection probabilities $p_1, p_2, \dots, p_n$ that:

1. are ε-individually fair
2. achieve cohort selection utility $OPT - k\varepsilon \leq \sum_{i=1}^{n} p_i s_i \leq OPT$.

In addition, it keeps at most $O(\frac{k}{\varepsilon})$ candidates pending.

Proof. We want to show that algorithm 3 gives the same marginal probabilities of selection as the offline algorithm 2 whose input is rounded into multiples of ε and the scores have become $\hat{s}_1, \dots, \hat{s}_n$. In other words, our goal is to show that for any candidate $i \in [n]$, $p_i = q_i$, where q_i is the probability that the candidate is selected by the offline algorithm. To prove this, we can show that two properties hold for scores $\tilde{s}_1, \dots, \tilde{s}_n$ that the candidates have right before the final rounding:

1. $\sum_{i=1}^{n} \tilde{s}_i = k$, and
2. $\forall i \in [n], \mathbb{E}[\tilde{s}_i] = q_i$.

In the offline algorithm, all members of a group g have the same score denoted by s^g, which, as seen in the proof of theorem 1, means that the individual selection probabilities are $q_i = q_j$ for any pair of candidates i, j from group g. The basic streaming version redistributes the score mass of the group among its members so that if the candidates are more than k, each of the pending candidates of the group represents $\frac{n^g}{k}$ of s^g and the rejected

Algorithm 3: Basic streaming cohort selection

```
Input: list of scores $s_1, s_2, \ldots, s_n \in [0, 1]$ for the $n$ candidates,
       the number of individuals that must be selected $k$,
       constant $\varepsilon \in (0, 1]$
Output: list of selection indicators $\tilde{s}_1, \tilde{s}_2, \ldots, \tilde{s}_n \in \{0, 1\}$

while stream has individuals do
    add the new person $e$ to group $g$ for which $s_e \in [(g-1)\varepsilon, g\varepsilon]$
    $n_e \leftarrow 1$ // e represent only themselves
    $\hat{s}_e \leftarrow g\varepsilon$
    if group $g$ has $n^g > k$ people then
        keep $k$ people from group $g$ with uniform random reservoir sampling of size $k$
        for any rejected person $j$ set $n_j \leftarrow 0$ and $\tilde{s}_j \leftarrow 0$
        for any pending person $i$ in group $g$ set $n_i \leftarrow \frac{n^g}{k}$
    end
end
sum $\leftarrow \sum_{i=1}^{n} \hat{s}_i$
if sum $< k$ then
    $c \leftarrow \frac{k - \text{sum}}{n}$
    for any person $i$ pending set $\tilde{s}_i \leftarrow n_i \hat{s}_i + n_i c$
    while $\exists \tilde{s}_i > 1$ do
        set $n_{<1}$ equal to the number of individuals with $\tilde{s}_j < 1$ from those pending
        for any pending person $j$ with $\tilde{s}_j < 1$ set $\tilde{s}_j \leftarrow \tilde{s}_j + \frac{n_j(\tilde{s}_i - 1)}{n_{<1}}$
        $\tilde{s}_i \leftarrow 1$
    end
else if sum $> k$ then
    $c \leftarrow \frac{\text{sum} - k}{n}$
    for any person $i$ pending set $\tilde{s}_i \leftarrow n_j \hat{s}_i - n_j c$
    while $\exists \tilde{s}_i < 0$ do
        set $n_{>0}$ equal to the number of individuals with $\tilde{s}_j > 0$ from those pending
        for any pending person $j$ with $\tilde{s}_j > 0$ set $\tilde{s}_j \leftarrow \tilde{s}_j + \frac{n_j \tilde{s}_i}{n_{>0}}$
        $\tilde{s}_i \leftarrow 0$
    end
else
    for any pending person $i$ set $\tilde{s}_i \leftarrow n_i \hat{s}_i$
end
return Rounding($\tilde{s}_1, \tilde{s}_2, \ldots, \tilde{s}_n$)
```

get score $\tilde{s} = 0$. We will show that $\tilde{s}_i = n_i q_i$, for any individual i, where $\tilde{s}_i$ is the score of i before the final rounding and n_i signifies the number of people represented by i. There are three ways the new scores $\tilde{s}_1, \ldots, \tilde{s}_n$ are calculated depending on the value of the sum of scores $\hat{s}_1, \ldots, \hat{s}_n$ when the stream has ended.

Case 1: $\sum_{i=1}^{n} \hat{s}_i = k$. The streaming algorithm sets $\tilde{s}_i = n_i \hat{s}_i$ in line 29 for the people who are pending, which also holds for those rejected because their n_i and $\tilde{s}_i$ are zero. The offline algorithm makes no modifications in the scores and, hence, every person i keeps their score $\hat{s}_i$. By lemma 1, the probability of candidate i being selected by the offline algorithm is $q_i = \hat{s}_i$. Thus, we obtain that for any person i has score $\tilde{s}_i = n_i q_i$ before the final rounding.

Case 2: $\sum_{i=1}^{n} \hat{s}_i < k$. First, we see that if the size of a group g is $n^g > k$, then for any member i of group g the probability of selection by the offline algorithm q_i is at most $\frac{k}{n^g}$. This holds because if there exists a person i in group g with $q_i > \frac{k}{n^g}$, then any other member j of this group has selection probability $q_j = q_i > \frac{k}{n^g}$. By adding together the probabilities of the candidates in this group, we obtain probability mass greater than k, which leads to a contradiction because the offline algorithm gives a solution with $\sum_{i=1}^{n} q_i = k$. Therefore, only the probabilities of people in groups of size smaller than k (who have $n_i = 1$) can reach 1. This observation is useful for the analysis of lines 13-19 of the streaming algorithm that are executed in this case. In line 14 all scores of pending candidates are initialized with $n_i(\hat{s}_i + c)$. The offline algorithm initializes the scores of all candidates with $\hat{s}_i + c$. The total amount of mass added to the candidates in the streaming is equal to that in the offline algorithm, but instead of distributing it uniformly, it is added according the fraction represented by each individual. If no score exceeds 1 in both the streaming and the offline, we have the property we want, because the offline will select i with probability $q_i = \hat{s}_i + c$. Otherwise, the loop is executed. Due to our observation, only people with $n_i = 1$ can have score greater than 1 at any point of the adjustment. The same people will have score greater than 1 in the offline too, because $n_i(\hat{s}_i + c) = \hat{s}_i + c$. As a consequence, the term $\frac{\hat{s}_i - 1}{n_{<1}}$ of line 17 involves only the original scores, which are the same as in the offline. Since any term added to any candidate i is multiplied by n_i, we have that if the offline algorithm has computed that the score of the i-th candidate is $\hat{s}_i + b$, then the streaming has computed score $\tilde{s}_i = n_i(\hat{s}_i + b)$ and if the offline has assigned score 1, then the streaming has also assigned score 1. Therefore, any person i is assigned score $\tilde{s}_i = n_i q_i$ by the streaming algorithm before the final rounding.

Case 3: $\sum_{i=1}^{n} \hat{s}_i < k$. The adjusting process is analogous to that of case 2 and, hence, it can be similarly shown that for any person i we have $\tilde{s}_i = n_i q_i$.

Additionally, the new sum of scores is

$$\begin{aligned}\sum_{i=1}^{n} \tilde{s}_i = \sum_{i=1}^{n} n_i q_i &= \sum_{g=1}^{m} \sum_{i \text{ in group } g} n_i q_i \\ &= \sum_{g=1}^{m} \sum_{i \text{ in group } g} \frac{n^g}{k} s^g \\ &= \sum_{g=1}^{m} n^g s^g = k.\end{aligned}$$

The computation of n_is is randomized because of the random reservoir sampling. The value of n_i for candidate i who is in a group with at most k members is equal to 1, which gives $\mathbb{E}[n_i] = 1$. If candidate i is in a group g with $n^g > k$ people, then $\mathbb{E}[n_i] = \frac{n^g}{k} \frac{k}{n^g} = 1$. Combining, the expected value of n_i with the fact that $\tilde{s}_i = n_i q_i$, we can compute the expected value of $\tilde{s}_i$

$$\mathbb{E}[\tilde{s}_i] = \mathbb{E}[n_i q_i] = q_i \mathbb{E}[n_i] = q_i.$$

Since the indicators $\tilde{s}_1, \ldots, \tilde{s}_n$ have values in $\{0, 1\}$, the probability p_i of person i getting selected by the streaming algorithm is $p_i = q_i$. Thus, we see that algorithm 3 is equivalent to the algorithm of lemma 2 and properties 1 and 2 of the lemma are satisfied.

The number of groups is $m = \lceil \frac{1}{\varepsilon} \rceil$ and each group keeps at most k candidates. As a result, the algorithm keeps at most $mk = \lceil \frac{1}{\varepsilon} \rceil k \leq \left(\frac{1}{\varepsilon} + 1\right)k$ candidates pending in total. □

4.3 Improved algorithm

The size of memory needed by the basic streaming algorithm is determined by the way the representatives are defined within the groups. By keeping a subset of k people for each group that has size greater than k and distributing the numbers of people they represent uniformly, algorithm 3 keeps at most $O(\frac{k}{\varepsilon})$ candidates pending. We can improve this by optimizing the process that determines who gets rejected and who represents which subgroup of rejected people. Instead of distributing the mass of a group uniformly to the candidates who have not been rejected, we perform a rounding process presented in algorithm 4. Its input consists of the representation numbers $n_1, \dots, n_l$, where n_i is the number of people the i-th candidate represents, and a value v, which determines the maximum number of people a pending candidate can represent. At every iteration, it rounds two entries by randomly setting one equal to zero and one equal to their sum, if their sum is less than v else it sets one entry equal to v and the other one equal to the remainder so that their sum is maintained. Its goal is to maximize the number of people it can reject while having each pending candidate represent at most v people.

```
Algorithm 4: IRounding
    Input: list of representation numbers $n_1, n_2, \dots, n_\ell \in \mathbb{N}$ for $\ell$ candidates and maximum number of
           people one person can represent $v \in \mathbb{N}$
    Output: list of adjusted representation numbers $\tilde{n}_1, \tilde{n}_2, \dots, \tilde{n}_\ell \in \mathbb{N}$

pendingIndex $\leftarrow$ 2
for $i$ from 1 to $\ell$ do
    if $i$ = pendingIndex then
        continue to next $i$
    end
    $a \leftarrow n_i$
    $b \leftarrow n_{\text{pendingIndex}}$
    choose $u$ randomly from $unif(0, 1)$
    if $a + b \leq v$ then
        if $u < \frac{a}{a+b}$ then
            $n_i \leftarrow a + b$
            $n_{\text{pendingIndex}} \leftarrow 0$
            pendingIndex $\leftarrow i$
        else
            $n_i \leftarrow 0$
            $n_{\text{pendingIndex}} \leftarrow a + b$
        end
    else
        if $u < \frac{v-b}{2v-a-b}$ then
            $n_i \leftarrow v$
            $n_{\text{pendingIndex}} \leftarrow a + b - v$
        else
            $n_i \leftarrow a + b - v$
            $n_{\text{pendingIndex}} \leftarrow v$
            pendingIndex $\leftarrow i$
        end
    end
end
return $n_1, n_2, \dots, n_\ell$
```

Lemma 3. Let $n_1, n_2, \dots, n_{n^g} \in \{0, 1, \dots, v\}$ be a list of the number of people each candidate of the same group g represents. Algorithm 4 rounds each n_i up to value v so that $\lfloor \frac{n^g}{v} \rfloor$ elements will be equal to v, one element will be equal to $n^g - \lfloor \frac{n^g}{v} \rfloor v$ and the rest will be equal to 0. Moreover, for all $i \in [n^g]$ $\mathbb{E}[\tilde{n}_i] = n_i$.

Proof. We begin by showing that for all $i \in [n^g]$ $\mathbb{E}[\tilde{n}_i] = n_i$. At step i, we have two candidates a and b who get rounded and obtain new values denoted by $\tilde{a}$ and $\tilde{b}$, respectively. The rounding depends on the value of $a + b$. If $a + b$ is at most equal to v, then $\mathbb{E}[\tilde{a}] = (a + b)\frac{a}{a+b} = a$. If $a + b$ greater than v and at most $2v$, then $\mathbb{E}[\tilde{a}] = v\frac{v-b}{2v-a-b} + (a+b-v)\frac{v-a}{2v-a-b} = a$. Similarly, we obtain $\mathbb{E}[\tilde{b}] = b$. Since, the expected values remain constant throughout the process, we conclude that $\mathbb{E}[\tilde{n}_i] = n_i$.

We notice that during the procedure $\forall j : j < i$ and $j \neq$ pendingIndex $n_j = 0$ or $n_j = v$. At the end all the candidates with index $j : j < i$and $j \neq$ pendingIndex and one of the candidates with index n^g or pendingIndex have value 0 or v. Since $\sum_{i=1}^{n^g} n_i = n^g$, we obtain that $\lfloor \frac{n^g}{v} \rfloor$ of the people with $j : j < i$and $j \neq$ pendingIndex have value v and the rest of them have value 0. The remaining mass is assigned to either the n^g-th candidate or the pendingIndex. □

The structure of algorithm 5 is similar to that of algorithm 3. The most important change is the rounding process which determines who represents whom and who gets rejected before the stream ends. Since all the members of one group have the same score $\hat{s}$, we adjust the scores, as new people are considered, on the level of groups. Specifically, for every iteration we calculate the scores of the groups so that $\sum_{g=1}^{m} n^g s^g = k$, where n^g is the size of group g and s^g is its adjusted score. Then, we perform the rounding process while making sure that for all groups g no candidate has $n_i s^g > 1$, where n_i is the number of people candidate i represents. This is ensured by setting the argument $v = \lfloor \frac{1}{s^g} \rfloor$. In the end, every person who has not been rejected while the algorithm reads from the stream is assigned score $\tilde{s}_i = n_i s^g$. The final step calls the algorithm 1 with input $\hat{s}_1, \ldots, \hat{s}_n$.

Theorem 3. Given individually fair scores $s_1, s_2, \ldots, s_n$ for a set of n candidates, the streaming algorithm 5 solves the cohort selection problem for any $\varepsilon \in (0, 1]$ by choosing a cohort with individual selection probabilities $p_1, p_2, \ldots, p_n$ that:

1. are ε-individually fair
2. achieve cohort selection utility $OPT - k\varepsilon \leq \sum_{i=1}^{n} p_i s_i \leq OPT$.

In addition, it keeps at most $O(k + \frac{1}{\varepsilon})$ candidates pending.

Proof. By comparing algorithm 5 to the algorithm described in lemma 2 we want to prove that they are equivalent. In particular, we show that the probability p_i that individual i is selected by algorithm 5 is equal to the probability q_i that i is selected by the offline algorithm with input the modified scores $\hat{s}_1, \ldots, \hat{s}_n$, where $\hat{s}_i = g\varepsilon$ if $s_i \in [(g-1)\varepsilon, g\varepsilon]$. The first step is to prove that the scores $\tilde{s}_1, \tilde{s}_2, \ldots, \tilde{s}_n$ which are the input of the final rounding in line 31 satisfy the following properties:

1. $\sum_{i=1}^{n} \tilde{s}_i = k$
2. $\mathbb{E}[\tilde{s}_i] = q_i, \forall i \in [n]$.

We saw in the proof of theorem 1 that for the offline algorithm all candidates of the same group have the same selection probability q_i. We will show that for any person i we have $q_i = s^g$, where s^g is the final calculated score of the group g to which i belongs. The s^g we are referring to is calculated by the final execution of lines 5-24. Lines 6-24 of algorithm 5 describe the same procedure as lines 2-18 of algorithm 2, but from the point of view of groups instead of individuals. We consider three cases that depend on the value of the sum of the initial scores.

Case 1: $\sum_{i=1}^{n} \hat{s}_i = k$. The offline algorithm does not change the scores of the individuals and, thus, assigns probability $q_i = \hat{s}_i = g\varepsilon$ to candidate i of group g. Similarly, algorithm 5 assigns score $s^g = g\varepsilon$ to group g and in line 30 sets $\tilde{s}_i = n_i s^g = n_i g\varepsilon = n_i q_i$ for the people who have not been rejected.

Case 2: $\sum_{i=1}^{n} \hat{s}_i < k$. The process that calculates s^g starts by setting $s^g = g\varepsilon + c$. The offline algorithm initializes the probability of i who is a member of g as $q_i = \hat{s}_i + c = g\varepsilon + c$. If for all groups g, $g\varepsilon + c \leq 1$, then the adjustment stops for both algorithms and we have that for any group g and any member i of g has $q_i = s^g$. If there exists g such that $q_i > 1$, then the corresponding s^g exceeds 1 by the same amount. Additionally, at this point the probabilities of all people in the same group as i will exceed 1 by this amount. Therefore, the $n_{<1}$ of the two algorithms is the same. The offline algorithm runs the loop for all members of all groups that have individuals with $q_i > 1$. The streaming version aggregates the excess mass from all n^g members of group g and redistributes

Algorithm 5: Improved streaming cohort selection

```
Input: list of scores $s_1, s_2, \dots, s_n \in [0, 1]$ for the $n$ candidates,
       the number of individuals that must be selected $k$,
       constant $\varepsilon \in (0, 1]$
Output: list of selection indicators $\tilde{s}_1, \tilde{s}_2, \dots, \tilde{s}_n \in \{0, 1\}$

while stream has individuals do
    add the new person $e$ to group $g$ for which $s_e \in [(g-1)\varepsilon, g\varepsilon]$
    $\hat{s}_e \leftarrow g\varepsilon$
    $n_e \leftarrow 1$
    sum $\leftarrow \sum_{i=1}^{e} \hat{s}_i$
    if sum $< k$ then
        $c \leftarrow \frac{k - sum}{n}$
        for any group $g$ set $s^g \leftarrow g\varepsilon + c$
        while $\exists$ group $g$ with $s^g > 1$ do
            set $n_{<1}$ equal to the number of individuals from groups with $s^f < 1$
            for any group $f$ with $s^f < 1$ set $s^f \leftarrow s^f + n^g \frac{(s^g - 1)}{n_{<1}}$
            $s^g \leftarrow 1$
        end
    else if sum $> k$ then
        $c \leftarrow \frac{sum - k}{n}$
        for any group $g$ set $s^g \leftarrow g\varepsilon - c$
        while $\exists$ group $g$ with $s^g < 0$ do
            set $n_{>0}$ equal to the number of individuals from groups with $s^f > 0$
            for any group $f$ with $s^f > 0$ set $s^f \leftarrow s^f + n^g \frac{s^g}{n_{>0}}$
            $s^g \leftarrow 0$
        end
    else
        for any group $g$ set $s^g \leftarrow g\varepsilon$
    end
    for group $g$ do
        if $s^g > 0$ then
            /* $v$ is the max number of people a person $i$ can represent s.t. $n_i v \leq 1$ */
            $(\{n_i\}_{i \text{ in group } g})$ = IRounding($\{n_i\}_{i \text{ in group } g}$, $v = \lfloor \frac{1}{s^g} \rfloor$)
            for any person $i$ in group $g$ with $n_i = 0$ set $\tilde{s}_i \leftarrow 0$
        end
    end
end
for any group $g$ and any person $i$ in $g$ set $\tilde{s}_i \leftarrow n_i s^g$
return Rounding($\tilde{s}_1, \tilde{s}_2, \dots, \tilde{s}_n$)
```

it all at once instead of running separate iterations for every member of the group as the offline does. No extra mass is added after the initialization but it is only moved from group to group. Hence, we obtain that at the end of this process $q_i = s^g$.

Case 3:$\sum_{i=1}^n \hat{s}_i > k$. Analogously to case 2, if i is a member of group g, then $q_i = s^g$ and for all .

Those who were rejected have $\tilde{s}_i = 0$ and $n_i = 0$. As a result, we see that for any person i, $\tilde{s}_i = n_i q_i$. From this we can infer that

$$\sum_{i=1}^{n} \tilde{s}_i = \sum_{i=1}^{n} n_i q_i = \sum_{g=1}^{m} \sum_{i \in g} n_i s^g = \sum_{g=1}^{m} n^g s^g = \sum_{g=1}^{m} \sum_{i \in g} q_i = k.$$

As new people are added to the groups, the group scores s^g become smaller in order for the sum of scores $n^g s^g$ to be equal to k. Therefore, the maximum number v of people that can be represented by a candidate in a given group either stays the same or increases after every iteration. By lemma 3, we obtain that the rounding process maintains the expected value of the number of people each candidate represents equal to their initial value. Since every person begins by representing only themselves, we have that for the i-th candidate $\mathbb{E}[n_i] = 1$. Finally, we obtain

$$\mathbb{E}[\hat{s}_i] = \mathbb{E}[n_i q_i] = \mathbb{E}[n_i] q_i = q_i,$$

because the calculation of s^gs is deterministic. The final rounding process makes the final decisions and outputs 0 if the candidate is rejected and 1 if the candidate is selected. Due to properties 1 and 2, the probability of candidate i being selected by the streaming algorithm is q_i. Thus, algorithm 5 and the offline algorithm with scores rounded to multiples of ε have the same selection probabilities. The theorem follows by the application of lemma 2.

Because of the online score adjustments, we have that for $n \geq k$ the sum of all the scores is equal to k at the end of each loop. Therefore, we have $\sum_{g=1}^m n_g s^g = k$. If $s^f > 0$, each person can represent at most $\lfloor \frac{1}{s^g} \rfloor$ candidates. By lemma 3, the number of representatives per group is at most

$$\left\lceil \frac{n^g}{\lfloor \frac{1}{s^g} \rfloor} \right\rceil \leq \frac{n^g}{\lfloor \frac{1}{s^g} \rfloor} + 1 \leq 2n^g s^g + 1,$$

since $\frac{n_g}{\lfloor \frac{1}{s^g} \rfloor} \leq 2n_g s^g$. If we sum the number of representatives for all groups we obtain

$$\sum_{g=1}^{m} \left\lceil \frac{n^g}{\lfloor \frac{1}{s^g} \rfloor} \right\rceil \leq \sum_{g=1}^{m} (2n^g s^g + 1) = 2k + \frac{1}{\varepsilon}.$$

This completes the proof. □

References

[AKRZ21] Eshwar Ram Arunachaleswaran, Sampath Kannan, Aaron Roth, and Juba Ziani. Pipeline Interventions. In *Innovations in Theoretical Computer Science Conference*, ITCS '21, 2021. https://arxiv.org/abs/2002.06592.

[BKN+17] Amanda Bower, Sarah N Kitchen, Laura Niss, Martin J Strauss, Alexander Vargas, and Suresh Venkatasubramanian. Fair pipelines. *arXiv preprint arXiv:1707.00391*, 2017.

[CVZ10] Chandra Chekuri, Jan Vondrák, and Rico Zenklusen. Dependent randomized rounding via exchange properties of combinatorial structures. In *IEEE Symposium on Foundations of Computer Science*, FOCS '10. IEEE, 2010. https://arxiv.org/abs/0909.4348.

[DHP+12] Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard Zemel. Fairness through awareness. In *Innovations in Theoretical Computer Science*, ITCS '12. ACM, 2012.

[DI19] Cynthia Dwork and Christina Ilvento. Fairness under composition. *Innovations in Theoretical Computer Science*, 2019. http://arxiv.org/abs/1806.06122.

[DIJ20] Cynthia Dwork, Christina Ilvento, and Meena Jagadeesan. Individual Fairness in Pipelines. In *Symposium on Foundations of Responsible Computing*, FORC '20, 2020. https://arxiv.org/abs/2004.05167.

[HPS16] Moritz Hardt, Eric Price, and Nathan Srebro. Equality of opportunity in supervised learning. In *Conference on Neural Information Processing Systems*, NIPS '16, 2016. https://arxiv.org/abs/1610.02413.

[KRZ19] Sampath Kannan, Aaron Roth, and Juba Ziani. Downstream effects of affirmative action. In *Conference on Fairness, Accountability, and Transparency*, FAT* '19. ACM, 2019. https://arxiv.org/abs/1808.09004.

[SMLN20] Niklas Smedemark-Margulies, Paul Langton, and Huy L Nguyen. Fair and useful cohort selection. *arXiv preprint arXiv:2009.02207*, 2020.