Fairness in Streaming Submodular Maximization: Algorithms and Hardness

Marwa El Halabi[*]
MIT CSAIL
marwash@mit.edu

Slobodan Mitrović[*]
MIT CSAIL
slobo@mit.edu

Ashkan Norouzi-Fard[*]
Google Zurich
ashkannorouzi@google.com

Jakab Tardos[*]
EPFL
jakab.tardos@epfl.ch

Jakub Tarnawski[*]
Microsoft Research
jatarnaw@microsoft.com

Abstract

Submodular maximization has become established as the method of choice for the task of selecting representative and *diverse* summaries of data. However, if datapoints have sensitive attributes such as gender or age, such machine learning algorithms, left unchecked, are known to exhibit *bias*: under- or over-representation of particular groups. This has made the design of *fair* machine learning algorithms increasingly important. In this work we address the question: *Is it possible to create fair summaries for massive datasets?* To this end, we develop the first streaming approximation algorithms for submodular maximization under fairness constraints, for both monotone and non-monotone functions. We validate our findings empirically on exemplar-based clustering, movie recommendation, DPP-based summarization, and maximum coverage in social networks, showing that fairness constraints do not significantly impact utility.

1 Introduction

Machine learning algorithms are increasingly being used to assist human decision making. This led to concerns about the potential for bias and discrimination in automated decisions, especially in sensitive domains such as voting, hiring, criminal justice, access to credit, and higher-education [50, 20, 54, 27]. To mitigate such issues, there has been a growing effort towards developing *fair* algorithms for several fundamental problems, such as classification [59], ranking [13], clustering [16, 2, 33, 1], bandit learning [34, 46], voting [12], matching [17], influence maximization [58], and diverse data summarization [11].

In this work, we address fairness in another important class of problems, that of *streaming submodular maximization* subject to a cardinality constraint. Submodular functions are set functions that satisfy a diminishing returns property, which naturally occurs in a variety of machine learning problems. In particular, streaming submodular maximization is a natural model for *data summarization*: the task of extracting a representative subset of moderate size from a large-scale dataset. Being able to generate summaries efficiently and on-the-fly is critical to cope with the massive volume of modern datasets, which is often produced so rapidly that it cannot even be stored in memory. In many applications, such as exemplar-based clustering [23], document [44, 21] and corpus summarization [55], and recommender systems [25, 26], this challenge can be formulated as a streaming submodular maximization problem subject to a cardinality constraint. An extensive line of research focused on developing efficient algorithms in this context [14, 15, 8, 3, 51, 28].

[*]Equal contribution.

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

For monotone objectives, a one pass streaming algorithm achieving $(1/2 - \epsilon)$-approximation was proposed in [3] and shown to be tight in [29]. For non-monotone objectives, the state-of-the-art approximation is $1/5.82$, achieved by a randomized algorithm proposed in [28]. To the best of our knowledge, submodular maximization under fairness constraints has only been considered, in the *offline* setting, for monotone objectives. Celis et al. [12] provide a $(1 - 1/e)$-approximation based on the continuous greedy algorithm [9]. In this paper, we provide the first approximation algorithms for submodular maximization under fairness constraints, in the streaming setting, for both monotone and non-monotone objectives.

Characterizing what it means for an algorithm to be fair is an active area of research. Several notions of fairness have been proposed in the literature, but no universal metric of fairness exists. We adopt here the common notion used in various previous works [11, 12, 13, 17, 16], where we ask that the solution obtained is *balanced* with respect to some sensitive attribute (e.g., race, gender). Formally, we are given a set V of n items (e.g., people), where each item is assigned a color c encoding a sensitive attribute. Let $V_1, \cdots, V_C$ be the corresponding C *disjoint* groups of items sharing the same color. We say that a selection of items $S \subseteq V$ is *fair* if it satisfies $\ell_c \leq |S \cap V_c| \leq u_c$ for a given choice of lower and upper bounds $\ell_c, u_c \in \mathbb{Z}_{\geq 0}$, often set to be proportional to the fraction of items of color c, i.e., $|V_c|/n$. This definition captures several other existing notions of fairness such as statistical parity [24], diversity rules (e.g., 80%-rule) [19, 4], and proportional representation rules [48, 6] (see [12, Sect. 4]).

1.1 Our contribution

In this work, we develop a new approach for fair submodular maximization. We show how to reduce this problem to submodular maximization subject to a matroid constraint. In the case of monotone functions, our reduction preserves the approximation ratio and the number of oracle calls of the corresponding algorithm for the matroid constraint. In the non-monotone case, this reduction does not hold anymore, but it still plays an important role in our approach.

The monotone case Here we achieve two results, with respect to the memory requirement. First, a $1/2$-approximate algorithm that uses an exponential in k memory. This result is known to be tight due to [29]. Second, we design a low-memory efficient algorithm, matching the state-of-the-art result of the partition matroid, a special case of our problem. Namely, our proposed algorithm achieves a $1/4$-approximation using only $O(k)$ memory, and processes each element of the stream in $O(\log k)$ time and 2 oracle calls. These results are discussed in Section 4.

The non-monotone case In this context, we introduce the notion of *excess ratio*, denoted by q and defined as $1 - \max_{c \in C} \ell_c/|V_c|$. This refers to the "freedom" that an algorithm has in omitting elements from the solution. If the excess ratio is close to 0, then for at least one of the colors, the total number of elements in the stream is close to the lower bound. In this case, an algorithm has little flexibility in terms of which elements it chooses from this color. Conversely, if the excess ratio is close to 1, then the total number of elements for every color is significantly higher than the corresponding lower bound.

We show that the excess ratio is closely tied to the hardness of fair non-monotone submodular maximization in the streaming setting. Indeed, we propose a $q/5.82$-approximation algorithm using $O(k)$ memory, and then show that any algorithm that achieves a better than q-approximation requires $\Omega(n)$ memory. These results are discussed in Section 5. Note that in practice, the size of the summary is expected to be significantly smaller than the size of the input. Hence, it is natural to expect that the excess ratio will be close to 1, and thus our algorithm will perform well on real-world applications.

Empirical evaluation We study the empirical performance of our algorithms on various real-life tasks where being fair is important. We observe that our algorithms allow us to enforce fairness constraints at the cost of a small loss in utility, while also matching the efficiency and number of oracle calls of "unfair" state-of-the-art algorithms.

1.2 Additional related work

Submodular maximization has been extensively studied. The setting most similar to ours is that of streaming submodular maximization under a matroid constraint. The first result in this setting, for

monotone functions, is by [14] which proposed a $1/4p$-approximation algorithm under p-matroid constraints using $O(k)$ memory, which was later extended to p-matchoid constraints in [15]. For a single matroid constraint, the best known approximation is achieved by [32], who proposed a $1/2$-approximation algorithm using $k^{O(k)}$ memory. This is essentially tight, as [29] shows that a $(1/2+\epsilon)$-approximation of monotone submodular maximization requires $\Omega(n)$ space, even for cardinality constraint, for any positive ϵ. For non-monotone functions, the first streaming algorithm for this problem appears in [15], which achieves an approximation ratio of $(1-\epsilon)(2-o(1))/(8+e)p$ with $O(k \log k)$ memory. This was improved in [28] to a $1/(2p+2\sqrt{p(p+1)}+1)$-approximation using $O(k)$ memory. The latter implies the best known result for non-monotone functions under a single matroid constraint, with $1/(3+2\sqrt{2}) \approx 1/5.82$-approximation.

In the sequential setting, [12] studied the fair multiwinner voting problem, which they cast as a fair submodular maximization problem, and presented a $(1-1/e)$-approximation algorithm for it. They also considered the setting in which color groups can overlap. In this setup, even checking feasibility is NP-hard, when elements can belong to 3 or more colors. Nevertheless, if fairness constraints are allowed to be *nearly* satisfied, [12] gives a $(1-1/e-o(1))$-approximation algorithm. [36] studied data summarization with privacy and fairness constraints, but adopted a different notion of fairness, where part of the data is deleted or masked due to fairness criteria.

2 Preliminaries

We consider a (potentially large) collection V of n items, also called the *ground set*. We study the problem of maximizing a *non-negative submodular function* $f : 2^V \to \mathbb{R}_{\geq 0}$. Given two sets $X, Y \subseteq V$, the *marginal gain* of X with respect to Y is defined as

$$f(X \mid Y) = f(X \cup Y) - f(Y),$$

which quantifies the change in value when adding X to Y. The function f is *submodular* if for any two sets X and Y such that $X \subseteq Y \subseteq V$ and any element $e \in V \setminus Y$ we have

$$f(e \mid X) \geq f(e \mid Y).$$

We say that f is *monotone* if for any element $e \in V$ and any set $Y \subseteq V$ it holds that $f(e \mid Y) \geq 0$; otherwise, if $f(e \mid Y) < 0$ for some $e \in V$ and $Y \subseteq V$, we say that f is *non-monotone*. Throughout the paper, we assume that f is given in terms of a value oracle that computes $f(S)$ for given $S \subseteq V$. We also assume that f is *normalized*, i.e., $f(\emptyset) = 0$.

Fair submodular maximization We assume that the ground set V is colored so that each element has exactly one color. We index the colors $c = 1, 2, ..., C$ and denote by V_c the set of elements of color c. Thus $V = V_1 \cup ... \cup V_C$ is a partition. For each color c we assume that we are given a lower and an upper bound on the number of elements of color c that a feasible solution must contain. These represent fairness constraints and are denoted by ℓ_c and u_c, respectively. Let $k \in \mathbb{Z}_{\geq 0}$ be a global cardinality constraint. We denote by $\mathcal{F}$ the set of solutions feasible under these fairness and cardinality constraints, i.e.,

$$\mathcal{F} = \{S \subseteq V : |S| \leq k, |S \cap V_c| \in [\ell_c, u_c] \text{ for all } c = 1, ..., C\}.$$

The problem of maximizing a function f under *cardinality and fairness constraints* is defined as selecting a set $S \subseteq V$ with $S \in \mathcal{F}$ so as to maximize $f(S)$. We use OPT to refer to a set maximizing f. We assume that there exists a feasible solution, i.e., $\mathcal{F} \neq \emptyset$. In particular, this implies that $\sum_{c=1}^{C} \ell_c \leq k$.

Matroids In our algorithms we often reduce to submodular maximization under a matroid constraint: the problem of selecting a set $S \subseteq V$ with $S \in \mathcal{M}$ so as to maximize $f(S)$, where $\mathcal{M}$ is a matroid. We call a family of sets $\mathcal{M} \subseteq 2^V$ a *matroid* if it satisfies the following properties: $\mathcal{M} \neq \emptyset$; *downward-closedness:* if $A \subseteq B$ and $B \in \mathcal{M}$, then $A \in \mathcal{M}$; *augmentation:* if $A, B \in \mathcal{M}$ with $|A| < |B|$, then there exists $e \in B$ such that $A + e \in \mathcal{M}$.

3 Warm-up: Monotone Sequential Algorithm

In this section, we consider the classic sequential setting and assume that the submodular function f is monotone. We present a natural greedy algorithm FAIR-GREEDY, and show that it achieves

a $1/2$-approximate solution. The advantage of this algorithm compared to the algorithm provided in [12] based on continuous greedy, is its simplicity and faster running time of $O(|V|k)$. Moreover, the algorithm and ideas introduced in this section serve as a warm-up for the streaming setting.

The greedy algorithm picks at each step the element that has the largest marginal gain while satisfying some constraint. We start by observing that if this element was only required to satisfy the upper-bound and cardinality constraints, the greedy algorithm might not return a feasible solution. It might reach the global cardinality constraint without satisfying the lower bounds. Therefore, a more careful selection of the elements is needed. To that end, we define the following concept.

Definition 3.1 *We call a set $S \subseteq V$* extendable *if it is a subset $S \subseteq S'$ of some feasible solution $S' \in \mathcal{F}$.*

For a set S to be extendable, it must satisfy the upper bounds: $|S| \leq k$ and $|S \cap V_c| \leq u_c$ for all $c = 1, ..., C$. If S also satisfies the lower bounds ($|S \cap V_c| \geq \ell_c$ for all c), then S is already feasible. Otherwise, it is necessary to add at least $\ell_c - |S \cap V_c|$ elements of every color c for which S does not yet satisfy the lower bound. This yields a feasible extension as long as it does not violate the global cardinality constraint k. In short, we have the following simple characterization:

Observation 3.2 *A set $S \subseteq V$ is extendable if and only if*

$$|S \cap V_c| \leq u_c \quad \textit{for all } c = 1, ..., C \qquad \textit{and} \qquad \sum_{c=1}^{C} \max(|S \cap V_c|, \ell_c) \leq k\,.$$

The FAIR-GREEDY algorithm starts with $S = \emptyset$ and in each step takes the element with highest marginal gain that keeps the solution extendable.

Fact 3.3 FAIR-GREEDY *is a $1/2$-approximate algorithm with $O(|V|k)$ running time for fair monotone submodular maximization.*

The analysis of FAIR-GREEDY is deferred to Appendix A.

Algorithm 1 FAIR-GREEDY

$S \leftarrow \emptyset$
while $|S| < k$ **do**
  $U \leftarrow \{e \in V \mid S + e \text{ is extendable}\}$
  $S \leftarrow S + \operatorname{argmax}_{e \in U} f(e \mid S)$
return S

4 Monotone Streaming Algorithm

In this section, we present our algorithm for fair *monotone* submodular maximization in the streaming setting, and we prove its approximation guarantee. We begin by explaining the intuition behind our algorithm. If we removed the lower-bound constraints $|S \cap V_c| \geq \ell_c$ in $\mathcal{F}$, then the remaining constraints would give rise to a matroid (a so-called laminar matroid). There exist efficient streaming algorithms for submodular maximization under matroid constraint (e.g. [14, 28]), which we could use in a black-box manner. A solution obtained from such an algorithm $\mathcal{A}$ may of course violate the lower-bound constraints. We could hope to augment our solution to a feasible one using "backup" elements gathered from the stream in parallel to $\mathcal{A}$. As we are dealing with a *monotone* submodular function, adding such elements would not hurt the approximation guarantee inherited from $\mathcal{A}$.

However, doing so might violate the global cardinality constraint $|S| \leq k$. Indeed, as we remarked in Section 3, not every set satisfying the upper-bound constraints can be extended to a feasible solution. Recall that the right constraint to place was for the solution to be *extendable* (Definition 3.1) to a feasible set. Crucially, we show that such a solution can be efficiently found, as extendable subsets of V form a matroid.

Lemma 4.1 *Let $\tilde{\mathcal{F}} \subseteq 2^V$ be the family of all extendable subsets of V. Then $\tilde{\mathcal{F}}$ is a matroid.*

The proof of Lemma 4.1 can be found in Appendix B.1. Algorithms for submodular maximization under a matroid constraint require access to a membership oracle for the matroid. For $\tilde{\mathcal{F}}$, membership is easy to verify, as follows from Observation 3.2.

Now we are ready to present our algorithm FAIR-STREAMING for fair monotone submodular maximization. Let $\mathcal{A}$ be a streaming algorithm for monotone submodular maximization under a matroid constraint. FAIR-STREAMING runs $\mathcal{A}$ to construct an extendable set $S_{\mathcal{A}}$ that approximately maximizes f. In parallel, for every color c we collect a backup set B_c of size $|B_c| = \ell_c$. At the end, the solution $S_{\mathcal{A}}$ is augmented to a feasible solution S using a simple procedure: for every color such that $|S_{\mathcal{A}} \cap V_c| < \ell_c$, add any $\ell_c - |S_{\mathcal{A}} \cap V_c|$ elements from B_c to satisfy the lower bound. The pseudocode of FAIR-STREAMING is given as Algorithm 2. Thus we get the following black-box reduction, proved in Appendix B.2.

Algorithm 2 FAIR-STREAMING

$S_{\mathcal{A}} \leftarrow \emptyset, B_c \leftarrow \emptyset$ for all $c = 1, ..., C$
for every arriving element e of color c **do**
  process e with algorithm $\mathcal{A}$
  if $|B_c| < \ell_c$ **then**
    $B_c \leftarrow B_c + e$
$S_{\mathcal{A}} \leftarrow$ solution of algorithm $\mathcal{A}$
$S \leftarrow S_{\mathcal{A}}$ augmented with elements in sets B_c
return S

Theorem 4.2 *Suppose $\mathcal{A}$ is a streaming algorithm for monotone submodular maximization under a matroid constraint. Then there exists a streaming algorithm for fair monotone submodular maximization with the same approximation ratio and memory usage as $\mathcal{A}$.*

Applying Theorem 4.2 to the algorithm of [32] we get the following result.

Theorem 4.3 (Streaming monotone) *There exists a streaming algorithm for fair monotone submodular maximization that attains $1/2$-approximation and uses $k^{O(k)}$ memory.*

We remark that the $1/2$ approximation ratio is tight even in the simpler setting of monotone streaming submodular maximization subject to a cardinality constraint [29].
A more practical algorithm to use as $\mathcal{A}$ in FAIR-STREAMING is the $1/4$-approximation algorithm of Chakrabarti and Kale [14]. It turns out that we can further adapt and optimize our implementation to make our algorithm extremely efficient and use only 2 oracle calls and $O(\log k)$ time per element. We prove Theorem 4.4 in Appendix C, where we also state the algorithm of [14] for completeness.

Theorem 4.4 (Streaming monotone) *There exists a streaming algorithm for fair monotone submodular maximization that attains $1/4$-approximation, using $O(k)$ memory. This algorithm uses $O(\log k)$ time and 2 oracle calls per element.*

5 Non-monotone Streaming Case

We now focus on non-monotone functions. One might consider applying the approach from the previous section, i.e., use a known algorithm for non-monotone submodular maximization under a matroid constraint to find a high quality extendable solution, and then add backup elements to satisfy the lower-bound constraints. However, this approach is more challenging now, as adding backup elements to a solution could drastically decrease its value.

For example, consider the following instance with two colors. Let $V = A \cup B \cup \{x\}$ where $A = \{a_i | i \in [m_1]\}$, $B = \{b_i | i \in [m_2]\}$, each $e \in A \cup B$ is *blue*, and x is *red*. Let $f(S) = |S|$ for each $S \subseteq A \cup B$, and let x "nullify" the contributions of B but not the contributions of A. Formally,

$$f(S) = \begin{cases} |S| & \text{if } x \notin S, \\ |S \cap A| & \text{if } x \in S. \end{cases}$$

It is easy to verify that f is submodular (a formal proof is given in the Appendix). Suppose that we have to pick exactly one red element, i.e., $\ell_{\text{red}} = u_{\text{red}} = 1$. This renders all elements in B useless, and the optimal solution takes only elements in A. However, before x appears, elements in A and B are *indistinguishable*, since $f(S) = |S|$ for any $S \subseteq A \cup B$. Therefore, if $m_1 \ll m_2$, and x is last in the stream, any algorithm that does not store the entirety of V will pick only a few elements from A, thus achieving almost zero objective value once x is included in the solution.

The core difficulty here, and in general, is that ℓ_c is nearly as large as $n_c = |V_c|$ for some color c, like for red in our example. To quantify this we introduce the *excess ratio*

$$q = 1 - \max_{c \in [C]} \ell_c / n_c.$$

We show that this quantity is inherent to the difficulty of the problem. Indeed, it is impossible to achieve an approximation ratio better that q with sublinear space.

Theorem 5.1 (Hardness non-monotone) *For any constant $\epsilon > 0$ and $q \in [0, 1]$, any algorithm for fair non-monotone submodular maximization that outputs a $(q + \epsilon)$-approximation for inputs with excess ratio above q, with probability at least $2/3$, requires $\Omega(n)$ memory.*

The proof of Theorem 5.1 is deferred to Appendix D.2. Note that in practice q is nearly always large, as the size of the data is significantly larger than the size of the summary. In what follows, we present a streaming algorithm for fair non-monotone submodular maximization, that nearly matches the above approximation lower-bound, using only $O(k)$ memory.

5.1 Non-monotone algorithm

Our non-monotone algorithm FAIR-SAMPLE-STREAMING is a variant of FAIR-STREAMING, where we modify the way backup elements are collected. Let $\mathcal{A}$ be an α-approximation algorithm for non-monotone submodular maximization under a matroid constraint. FAIR-SAMPLE-STREAMING runs algorithm $\mathcal{A}$ to construct an extendable set $S_{\mathcal{A}}$ that approximately maximizes f. In parallel, our algorithm collects for every color c a backup set B_c of size $|B_c| = \ell_c$, by sampling without replacement ℓ_c elements in V_c using reservoir sampling [43]. Note that we do not need to know the value of n_c to execute reservoir sampling. At the end, the solution $S_{\mathcal{A}}$ is augmented to a feasible solution S using the same simple procedure as in Section 4. The pseudocode of FAIR-SAMPLE-STREAMING is given as Algorithm 3. We show that adding elements from the back-up set reduces the objective value by a factor of at most q.

Algorithm 3 FAIR-SAMPLE-STREAMING

$S_{\mathcal{A}} \leftarrow \emptyset, B_c \leftarrow \emptyset$ for all $c = 1, ..., C$
for every arriving element e **do**
  process e with algorithm $\mathcal{A}$
  if $e \in V_c$ **then**
    $B_c \leftarrow \text{Reservoir-Sample}(B_c, e)$
$S \leftarrow S_{\mathcal{A}}$ augmented with elements in sets B_c
return S

Theorem 5.2 *Suppose $\mathcal{A}$ is a streaming α-approximate algorithm for non-monotone submodular maximization under a matroid constraint. Then, there exists a streaming algorithm for fair non-monotone submodular maximization with expected $q\alpha$ approximation ratio, and the same memory usage, oracle calls, and running time as $\mathcal{A}$.*

The proof is provided in Appendix D.1. Combining this with the state of the art $1/5.82$-approximation algorithm of Feldman, et al. [28] (restated in Appendix C for completeness) yields the following.

Theorem 5.3 (Streaming non-monotone) *There exists a streaming algorithm for fair non-monotone submodular maximization that achieves $q/5.82$-approximation in expectation, using $O(k)$ memory. This algorithm uses $O(k)$ time and $O(k)$ oracle calls per element.*

6 Empirical Evaluation

In this section, we empirically validate our results and address the question: *What is the price of fairness?* To this end, we compare our approach against several baselines on four datasets. We measure: (1) Objective values. (2) Violation of fairness constraints: Given a set S, we define $\text{err}(S) = \sum_{c \in [C]} \max\{|S \cap V_c| - u_c, \ell_c - |S \cap V_c|, 0\}$. A single term in this sum quantifies by how many elements S violates the lower or upper bound. Note that $\text{err}(S)$ is in the range $[0, 2k]$. (3) Number of oracle calls, as is standard in the field to measure the efficiency of algorithms.

We compare the following algorithms:

- **FAIR-STREAMING-CK**: monotone, $\mathcal{A}$ = Chakrabarti-Kale [14] (Theorem 4.4 and Appendix C.1),
- **FAIR-STREAMING-FKK**: monotone, $\mathcal{A}$ = Feldman et al. [28] (Appendix C.2),
- **FAIR-SAMPLE-STREAMING-FKK**: non-monotone, $\mathcal{A}$ = Feldman et al. [28] (Theorem 5.3),

- **UPPERBOUNDS**: [28] (Appendix C.2) applied to matroid defining upper bounds only (u_c and k),
- **FAIR-GREEDY**: monotone; where data size allows; see Section 3,
- **GREEDY**: monotone; when data size allows; no fairness constraints, only k,
- **SIEVESTREAMING**: Badanidiyuru et al. [3]; monotone; no fairness constraints, only k,
- **RANDOM**: maintain random sample of k elements; no fairness constraints,
- **FAIR-RANDOM**: maintain random feasible (fair) solution.

We now describe our experiments. We report the results in Fig. 1, and discuss them in Section 6.5. The code is available at `https://github.com/google-research/google-research/tree/master/fair_submodular_maximization_2020`.

6.1 Maximum coverage

Social influence maximization [37] and network marketing [42] are some of the prominent applications of the *maximum coverage* problem. The goal of this problem is to select a fixed number of nodes that maximize the coverage of a given network. Given a graph $G = (V, E)$, let $N(v) = \{u : (v, u) \in E\}$ denote the neighbors of v. Then the coverage of $S \subseteq V$, denoted by $f(S)$, is defined as the monotone submodular function $f(S) = \left|\bigcup_{v \in S} N(v)\right|$. We perform experiments on the Pokec social network [41]. This network consists of $1\,632\,803$ nodes, representing users, and $30\,622\,564$ edges, representing friendships. Each user profile contains attributes such as age, height and weight; these can take value "null". We impose fairness constraints with respect to (i) age and (ii) body mass index (BMI).

(i) We split ages into ranges $[1, 10], [11, 17], [18, 25], [26, 35], [36, 45], [46+]$ and consider each range as one color. We create another color for records with "null" age (around 30%). Then for every color c we set $\ell_c = \max\{0, |V_c|/n - 0.05\} \cdot k$ and $u_c = \min\{1, |V_c|/n + 0.05\} \cdot k$, except for the null color, where we set $\ell_c = 0$. The results are shown in Fig. 1a, 1b, and 1c.

(ii) BMI is computed as the ratio between weight (in kg) and height (in m) squared. Around 60% of profiles do not have set height or weight. We discard all such profiles, as well as profiles with clearly fake data (less than 2% of profiles). The resulting graph consists of $582\,319$ nodes and $5\,834\,695$ edges. The profiles are colored with respect to four standard BMI categories (underweight, normal weight, overweight and obese). Lower- and upper-bound fairness constraints are set again to be within 5% of their respective frequencies. The results are shown in Fig. 1d, 1e, and 1f.

6.2 Movie recommendation

We use the Movielens 1M dataset [31], which contains $\sim$1M ratings for $3\,900$ movies by $6\,040$ users, to develop a movie recommendation system. We follow the experimental setup of prior work [47, 51]: we compute a low-rank completion of the user-movie rating matrix [57], which gives rise to feature vectors $w_u \in \mathbb{R}^{20}$ for each user u and $v_m \in \mathbb{R}^{20}$ for each movie m. Then $w_u^\top v_m$ approximates the rating of m by u. The (monotone submodular) utility function for a collection S of movies personalized for user u is defined as

$$f_u(S) = \alpha \cdot \sum_{m' \in M} \max\left(\max_{m \in S}\left(v_m^\top v_{m'}\right), 0\right) + (1 - \alpha) \cdot \sum_{m \in S} w_u^\top v_m.$$

The first term optimizes coverage of the space of all movies (enhancing diversity) [45], and the second term sums up user-dependent movie scores; α controls the trade-off between the two terms. In our experiment we recommend a collection of movies for $\alpha = 0.85$ and k-values up to 100. Each movie in the database is assigned to one of 18 genres c; our fairness constraints mandate a representation of genres in S similar to that in the entire dataset. More precisely, we set $\ell_c = \left\lfloor 0.8 \frac{|V_c|}{|V|} k \right\rfloor$ and $u_c = \left\lceil 1.4 \frac{|V_c|}{|V|} k \right\rceil$. The results are shown in Fig. 1g, 1h, and 1i.

6.3 Census DPP-based summarization

A common reliable method for data summarization is to use a Determinantal Point Process (DPP) to assign a diversity score to each subset, and choose the subset that maximizes this score. A DPP is

a probability measure over subsets, defined for every $S \subseteq V$, as $P(S) = \frac{\det(L_S)}{\det(I+L)}$, where L is an $n \times n$ positive semi-definite kernel matrix, L_S is the $|S| \times |S|$ principal submatrix of L indexed by S, and I is the identity matrix. To find the most diverse representative subset, we need to maximize the non-montone submodular function $f(S) = \log\det(L_S)$ [40]. To ensure non-negativity (on non-empty sets), we normalize f by a constant.
We use the Census Income dataset [22] which consists of $45\,222$ records extracted from the 1994 Census database, with 14 attributes such as age, race, gender, education, and whether the income is above or below 50K USD.We follow the experimental setup of [11] to generate feature vectors of dimension 992 for 5000 randomly chosen records.[2] We select fair representative summaries with respect to race, requiring that each of the race categories provided in the dataset (White, Black, Asian-Pac-Islander, Amer-Indian-Eskimo, and Other) have a similar representation in S as in the entire dataset. Accordingly, we set $\ell_c = \left\lfloor 0.9\frac{T}{n}k \right\rfloor$ and $u_c = \left\lceil 1.1\frac{T}{n}k \right\rceil$, and vary k between $50 - 600$. The results are shown in Fig. 1j, 1k, and 1l.

6.4 Exemplar-based clustering

We consider a dataset containing one record for each phone call in a marketing campaign ran by a Portuguese banking institution [49]. We aim to find a representative subset of calls in order to assess the quality of service. We choose numeric attributes such as client age, gender, account balance, call date, and duration, to represent each record in the Euclidean space. We require the chosen subset to have clients in a wide range of ages. We divide the records into six groups according to age: $[0, 29], [30, 39], [40, 49], [50, 59], [60, 69], [70+]$; the numbers of records in each range are respectively: $5\,273, 18\,089, 11\,655, 8\,410, 1\,230, 554$. We set our bounds so as to ensure that each group comprises $10 - 20\%$ of the subset. Then we maximize the following monotone submodular function [38], where R denotes all records:

$$f(S) = C - \sum_{r \in R} \min_{e \in S} d(r, e) \quad \text{where} \quad d(x, y) = \|x - y\|_2^2 .$$

We let $f(\emptyset) = 0$ and C be $|V|$ times the maximum distance.[3] The results are shown in Fig. 1m, 1n, and 1o, where the clustering cost refers to $C - f(S)$.

6.5 Results

We observe that in all the experiments our algorithms make smaller or similar number of oracle calls compared to the baselines in corresponding settings (streaming or sequential). Moreover, the objective value of the fair solutions obtained by our algorithms is similar to the unfair baseline solutions, with less than 15% difference.

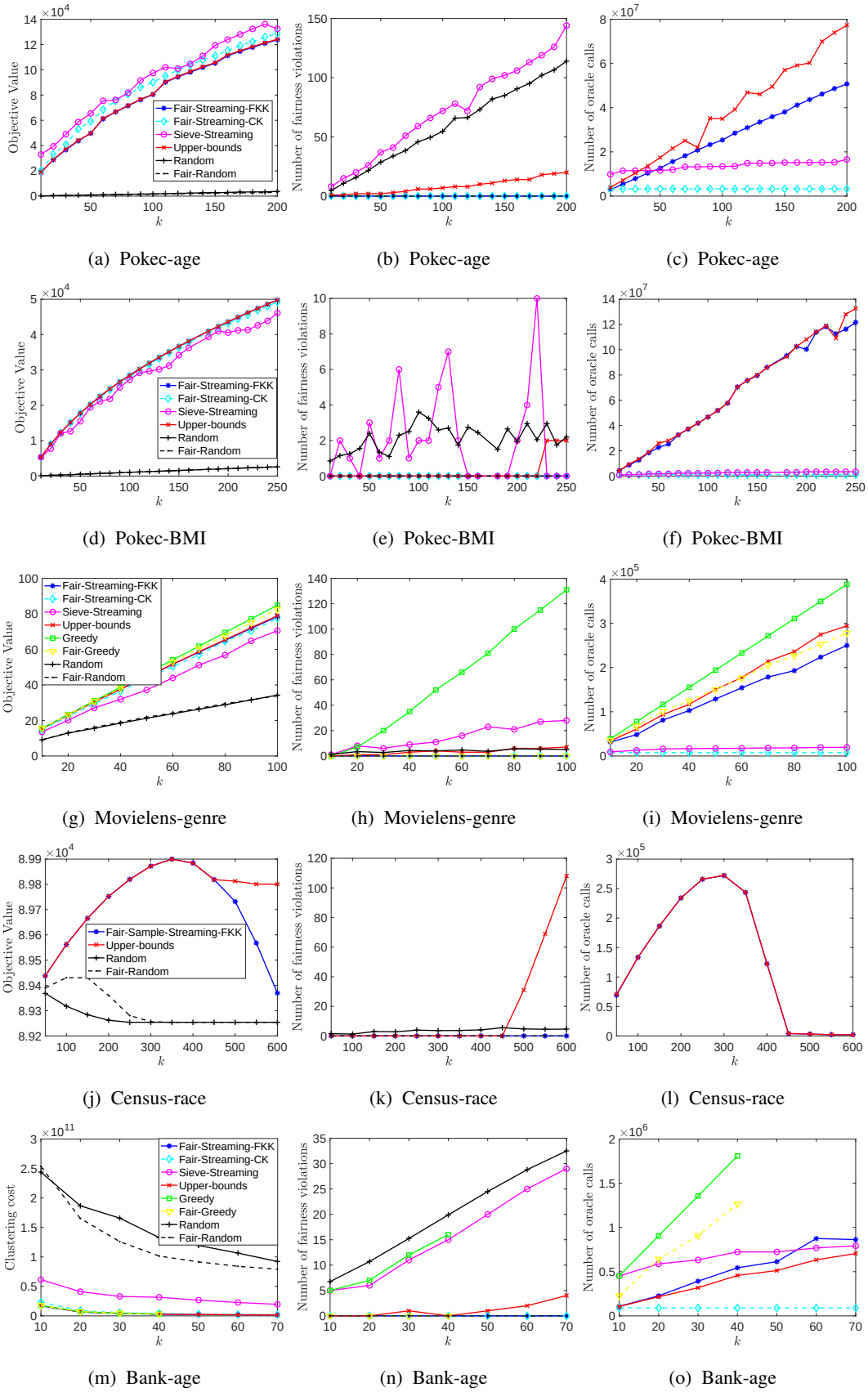
We also observe that the algorithms that do not impose fairness constraints introduce significant bias. For example, SIEVESTREAMING makes 150 errors in the maximum coverage experiment for $k = 200$ (see Fig. 1b), and 30 errors for $k = 70$ in the exemplar-based clustering experiment (see Fig. 1n). Moreover, even though UPPERBOUNDS satisfies the upper-bounds constraints, it still makes a noticeable amount of errors. For instance, it makes 20 errors in the maximum coverage experiment for $k = 200$ (see Fig. 1b), and 100 errors in the DPP-based summarization experiment for $k = 600$ (see Fig. 1k).

7 Conclusion

We presented the first streaming approximation algorithms for fair submodular maximization, for both monotone and non-monotone objectives. Our algorithms efficiently generate balanced solutions with respect to a sensitive attribute, while using asymptotically optimal memory. We empirically demonstrate that fair solutions are often nearly optimal, and that explicitly imposing fairness constraints is necessary to ensure balanced solutions.

[2] Code available at: `https://github.com/DamianStraszak/FairDiverseDPPSampling`.

[3] Note that C is added to ensure that all values are non-negative. Any C with this property would be suitable.



(a) Pokec-age (b) Pokec-age (c) Pokec-age

(d) Pokec-BMI (e) Pokec-BMI (f) Pokec-BMI

(g) Movielens-genre (h) Movielens-genre (i) Movielens-genre

(j) Census-race (k) Census-race (l) Census-race

(m) Bank-age (n) Bank-age (o) Bank-age

Figure 1: Performance of FAIR-STREAMING-CK, FAIR-STREAMING-FKK and FAIR-SAMPLE-STREAMING-FKK compared to other baselines, in terms of objective value, violation of fairness constraints, and running time, on Movielens, Pokec, Census, and Bank-Marketing datasets.

Broader Impact

Several recent studies have shown that automated data-driven methods can unintentionally lead to bias and discrimination [35, 56, 5, 10, 52]. Our proposed algorithms will help guard against these issues in data summarization tasks arising in various settings – from electing a parliament, over selecting individuals to influence for an outreach program, to selecting content in search engines and news feeds. As expected, fairness does come at the cost of a small loss in utility value, as observed in Section 6. It is worth noting that this "price of fairness" (i.e., the decrease in optimal objective value when fairness constraints are added) should not be interpreted as fairness leading to a less desirable outcome, but rather as a trade-off between two valuable metrics: the original application-dependent utility, and the fairness utility. Our algorithms ensure solutions achieving a close to optimal trade-off.

Finally, despite the generality of the fairness notion we consider, it does not capture certain other notions of fairness considered in the literature (see e.g., [18, 58]). No universal metric of fairness exists. The question of which fairness notion to employ is an active area of research, and will be application dependent.

Acknowledgments and Disclosure of Funding

Marwa El Halabi was supported by a DARPA D3M award, NSF CAREER award 1553284, NSF award 1717610, and by an ONR MURI award. The views, opinions, and/or findings contained in this article are those of the authors and should not be interpreted as representing the official views or policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the Department of Defense. Slobodan Mitrović was supported by the Swiss NSF grant No. P400P2_191122/1, MIT-IBM Watson AI Lab and Research Collaboration Agreement No. W1771646, and FinTech@CSAIL. Jakab Tardos has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 759471).

References

[1] Georg Anegg, Haris Angelidakis, Adam Kurpisz, and Rico Zenklusen. A technique for obtaining true approximations for k-center with covering constraints. In Daniel Bienstock and Giacomo Zambelli, editors, *Integer Programming and Combinatorial Optimization*, pages 52–65, Cham, 2020. Springer International Publishing.

[2] Arturs Backurs, Piotr Indyk, Krzysztof Onak, Baruch Schieber, Ali Vakilian, and Tal Wagner. Scalable fair clustering. *arXiv preprint arXiv:1902.03519*, 2019.

[3] Ashwinkumar Badanidiyuru, Baharan Mirzasoleiman, Amin Karbasi, and Andreas Krause. Streaming submodular maximization: Massive data summarization on the fly. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 671–680, 2014.

[4] Dan Biddle. *Adverse impact and test validation: A practitioner's guide to valid and defensible employment testing*. Gower Publishing, Ltd., 2006.

[5] Tolga Bolukbasi, Kai-Wei Chang, James Y Zou, Venkatesh Saligrama, and Adam T Kalai. Man is to computer programmer as woman is to homemaker? debiasing word embeddings. In *Advances in neural information processing systems*, pages 4349–4357, 2016.

[6] Markus Brill, Jean-Francois Laslier, and Piotr Skowron. Multiwinner approval rules as apportionment methods. 2017.

[7] N. Buchbinder, M. Feldman, J. Naor, and R. Schwartz. Submodular maximization with cardinality constraints. SIAM, 2014.

[8] Niv Buchbinder, Moran Feldman, and Roy Schwartz. Online submodular maximization with preemption. In *Proceedings of the twenty-sixth annual ACM-SIAM symposium on Discrete algorithms*, pages 1202–1216. SIAM, 2014.

[9] Gruia Calinescu, Chandra Chekuri, Martin Pal, and Jan Vondrák. Maximizing a monotone submodular function subject to a matroid constraint. *SIAM Journal on Computing*, 40(6):1740–1766, 2011.

[10] Aylin Caliskan, Joanna J Bryson, and Arvind Narayanan. Semantics derived automatically from language corpora contain human-like biases. *Science*, 356(6334):183–186, 2017.

[11] Elisa Celis, Vijay Keswani, Damian Straszak, Amit Deshpande, Tarun Kathuria, and Nisheeth Vishnoi. Fair and diverse DPP-based data summarization. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 716–725, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

[12] L. Elisa Celis, Lingxiao Huang, and Nisheeth K. Vishnoi. Multiwinner voting with fairness constraints. In Jérôme Lang, editor, *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, pages 144–151. ijcai.org, 2018.

[13] L. Elisa Celis, Damian Straszak, and Nisheeth K. Vishnoi. Ranking with fairness constraints. In Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella, editors, *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic*, volume 107 of *LIPIcs*, pages 28:1–28:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.

[14] Amit Chakrabarti and Sagar Kale. Submodular maximization meets streaming: Matchings, matroids, and more. In Jon Lee and Jens Vygen, editors, *Integer Programming and Combinatorial Optimization*, pages 210–221, Cham, 2014. Springer International Publishing.

[15] Chandra Chekuri, Shalmoli Gupta, and Kent Quanrud. Streaming algorithms for submodular function maximization. In *International Colloquium on Automata, Languages, and Programming*, pages 318–330. Springer, 2015.

[16] Flavio Chierichetti, Ravi Kumar, Silvio Lattanzi, and Sergei Vassilvitskii. Fair clustering through fairlets. In *Advances in Neural Information Processing Systems*, pages 5029–5037, 2017.

[17] Flavio Chierichetti, Ravi Kumar, Silvio Lattanzi, and Sergei Vassilvtiskii. Matroids, matchings, and fairness. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 2212–2220, 2019.

[18] Alexandra Chouldechova and Aaron Roth. The frontiers of fairness in machine learning. *arXiv preprint arXiv:1810.08810*, 2018.

[19] Joanne McGrath Cohoon, James P. Cohoon, Seth Reichelson, and Selwyn Lawrence. Effective recruiting for diversity. In Randa L. Shehab, James J. Sluss, and Deborah Anne Trytten, editors, *IEEE Frontiers in Education Conference, FIE 2013, Oklahoma City, Oklahoma, USA, October 23-26, 2013*, pages 1123–1124. IEEE Computer Society, 2013.

[20] Sam Corbett-Davies, Emma Pierson, Avi Feller, Sharad Goel, and Aziz Huq. Algorithmic decision making and the cost of fairness. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 797–806, 2017.

[21] Anirban Dasgupta, Ravi Kumar, and Sujith Ravi. Summarization through submodularity and dispersion. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1014–1022, Sofia, Bulgaria, August 2013. Association for Computational Linguistics.

[22] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[23] Delbert Dueck and Brendan J. Frey. Non-metric affinity propagation for unsupervised image categorization. In *IEEE 11th International Conference on Computer Vision, ICCV 2007, Rio de Janeiro, Brazil, October 14-20, 2007*, pages 1–8. IEEE Computer Society, 2007.

[24] Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard Zemel. Fairness through awareness. In *Proceedings of the 3rd innovations in theoretical computer science conference*, pages 214–226, 2012.

[25] Khalid El-Arini and Carlos Guestrin. Beyond keyword search: discovering relevant scientific literature. In Chid Apté, Joydeep Ghosh, and Padhraic Smyth, editors, *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, CA, USA, August 21-24, 2011*, pages 439–447. ACM, 2011.

[26] Khalid El-Arini, Gaurav Veda, Dafna Shahaf, and Carlos Guestrin. Turning down the noise in the blogosphere. In John F. Elder IV, Françoise Fogelman-Soulié, Peter A. Flach, and Mohammed Javeed Zaki, editors, *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Paris, France, June 28 - July 1, 2009*, pages 289–298. ACM, 2009.

[27] Yuri Faenza, Swati Gupta, and Xuan Zhang. Impact of bias on school admissions and targeted interventions. *arXiv preprint arXiv:2004.10846*, 2020.

[28] Moran Feldman, Amin Karbasi, and Ehsan Kazemi. Do less, get more: streaming submodular maximization with subsampling. In *Advances in Neural Information Processing Systems*, pages 732–742, 2018.

[29] Moran Feldman, Ashkan Norouzi-Fard, Ola Svensson, and Rico Zenklusen. The one-way communication complexity of submodular maximization with applications to streaming and robustness. *STOC*, 2020.

[30] Marshall L Fisher, George L Nemhauser, and Laurence A Wolsey. An analysis of approximations for maximizing submodular set functions—ii. In *Polyhedral combinatorics*, pages 73–87. Springer, 1978.

[31] F Maxwell Harper and Joseph A Konstan. The MovieLens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 5(4):19, 2016.

[32] Chien-Chung Huang, Naonori Kakimura, Simon Mauras, and Yuichi Yoshida. Approximability of monotone submodular function maximization under cardinality and matroid constraints in the streaming model, 2020.

[33] Xinrui Jia, Kshiteej Sheth, and Ola Svensson. Fair colorful k-center clustering. In Daniel Bienstock and Giacomo Zambelli, editors, *Integer Programming and Combinatorial Optimization*, pages 209–222, Cham, 2020. Springer International Publishing.

[34] Matthew Joseph, Michael Kearns, Jamie H Morgenstern, and Aaron Roth. Fairness in learning: Classic and contextual bandits. In *Advances in Neural Information Processing Systems*, pages 325–333, 2016.

[35] Matthew Kay, Cynthia Matuszek, and Sean A Munson. Unequal representation and gender stereotypes in image search results for occupations. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pages 3819–3828, 2015.

[36] Ehsan Kazemi, Morteza Zadimoghaddam, and Amin Karbasi. Scalable deletion-robust submodular maximization: Data summarization with privacy and fairness constraints. volume 80 of *Proceedings of Machine Learning Research*, pages 2544–2553, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

[37] David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 137–146, 2003.

[38] Andreas Krause and Ryan G Gomes. Budgeted nonparametric learning from data streams. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 391–398, 2010.

[39] Ilan Kremer, Noam Nisan, and Dana Ron. On randomized one-round communication complexity. *Comput. Complex.*, 8(1):21–49, 1999.

[40] Alex Kulesza, Ben Taskar, et al. Determinantal point processes for machine learning. *Foundations and Trends® in Machine Learning*, 5(2–3):123–286, 2012.

[41] Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`, June 2014.

[42] Fa-Hsien Li, Cheng-Te Li, and Man-Kwan Shan. Labeled influence maximization in social networks for target marketing. In *2011 IEEE third international conference on privacy, security, risk and trust and 2011 IEEE third international conference on social computing*, pages 560–563. IEEE, 2011.

[43] Kim-Hung Li. Reservoir-sampling algorithms of time complexity o (n (1+ log (n/n))). *ACM Transactions on Mathematical Software (TOMS)*, 20(4):481–493, 1994.

[44] Hui Lin and Jeff Bilmes. A class of submodular functions for document summarization. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 510–520, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

[45] Erik Lindgren, Shanshan Wu, and Alexandros G Dimakis. Leveraging sparsity for efficient submodular data summarization. In *Advances in Neural Information Processing Systems*, pages 3414–3422, 2016.

[46] Yang Liu, Goran Radanovic, Christos Dimitrakakis, Debmalya Mandal, and David C Parkes. Calibrated fairness in bandits. 2017.

[47] Slobodan Mitrović, Ilija Bogunović, Ashkan Norouzi-Fard, Jakub Tarnawski, and Volkan Cevher. Streaming robust submodular maximization: A partitioned thresholding approach. In *Advances in Neural Information Processing Systems*, 2017.

[48] Burt L Monroe. Fully proportional representation. *American Political Science Review*, 89(4):925–940, 1995.

[49] Sérgio Moro, Paulo Cortez, and Paulo Rita. A data-driven approach to predict the success of bank telemarketing. *Decis. Support Syst.*, 62:22–31, 2014.

[50] Cecilia Munoz, Smith Megan, and DJ Patil. *Big data: A report on algorithmic systems, opportunity, and civil rights*. Executive Office of the President. The White House, 2016.

[51] Ashkan Norouzi-Fard, Jakub Tarnawski, Slobodan Mitrović, Amir Zandieh, Aida Mousavifar, and Ola Svensson. Beyond $1/2$-approximation for submodular maximization on massive data streams. *ICML*, 2018.

[52] Cathy O'neil. *Weapons of math destruction: How big data increases inequality and threatens democracy*. Broadway Books, 2016.

[53] J.G. Oxley. *Matroid Theory*. Oxford graduate texts in mathematics. Oxford University Press, 2006.

[54] Jad Salem and Swati Gupta. Closing the gap: Group-aware parallelization for online selection of candidates with biased evaluations. *Available at SSRN 3444283*, 2019.

[55] Ruben Sipos, Adith Swaminathan, Pannaga Shivaswamy, and Thorsten Joachims. Temporal corpus summarization using submodular word coverage. In Xue-wen Chen, Guy Lebanon, Haixun Wang, and Mohammed J. Zaki, editors, *21st ACM International Conference on Information and Knowledge Management, CIKM'12, Maui, HI, USA, October 29 - November 02, 2012*, pages 754–763. ACM, 2012.

[56] Latanya Sweeney. Discrimination in online ad delivery. *Queue*, 11(3):10–29, 2013.

[57] Olga Troyanskaya, Michael Cantor, Gavin Sherlock, Pat Brown, Trevor Hastie, Robert Tibshirani, David Botstein, and Russ B Altman. Missing value estimation methods for DNA microarrays. *Bioinformatics*, 17(6):520–525, 2001.

[58] Alan Tsang, Bryan Wilder, Eric Rice, Milind Tambe, and Yair Zick. Group-fairness in influence maximization. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 5997–6005. International Joint Conferences on Artificial Intelligence Organization, 7 2019.

[59] Muhammad Bilal Zafar, Isabel Valera, Manuel Gomez Rodriguez, and Krishna P Gummadi. Fairness constraints: Mechanisms for fair classification. 2017.

A Details of FAIR-GREEDY

A.1 Proof of Fact 3.3

We remark that, once we know that extendable sets form a matroid (Lemma 4.1), the approximation ratio of FAIR-GREEDY can be seen to follow from the fact that the greedy algorithm for submodular maximization under matroid constraints achieves a $1/2$-approximation guarantee [30]. For completeness, below we also give a self-contained proof.

Proof. Let an optimal solution be $O \subseteq V$ and let the output of greedy be $G = \{g_1, \ldots, g_k\}$, where the elements where chosen in the order $g_1, \ldots g_k$ by the algorithm. To prove the lemma, we will show that $f(G) \geq \frac{1}{2} \cdot f(G \cup O)$.

Let us order the elements of O as $o_1, \ldots, o_k$ in a way that the colors of g_j and o_j coincide as much as possible. Specifically, we want an ordering such that for all j

- either g_j and o_j are the same color
- or, if $c(g_j) = c_1$ and $c(o_j) = c_2$ are different, then $|G \cap V_{c_1}| > |O \cap V_{c_1}|$ and $|G \cap V_{c_2}| < |O \cap V_{c_2}|$,

where $c(v)$ denotes the color of element $v \in V$. Such a matching between elements of G and O can be easily constructed recursively. Indeed, as long as there remain elements of G and O that are the same color match them together; once all remaining elements are of different color match them arbitrarily.

Claim A.1 *For any j, $G \backslash \{g_j\} \cup \{o_j\}$ is a feasible solution.*

Indeed, if g_j and o_j are the same color, exchanging them does not change the color profile of G and it remains feasible. On the other hand, if $c(g_j) = c_1$ and $c(o_j) = c_2$ are different, then $|G \cap V_{c_1}| > |O \cap V_{c_1}| \geq \ell_{c_1}$, and removing g_j from G does not violate any conditions. Similarly, $|G \cap V_{c_2}| < |O \cap V_{c_2}| \geq u_{c_2}$ and adding o_j to G does not violate any conditions either.

From Claim A.1 it follows that $g_1, \ldots, g_{j-1}, o_j$ is a feasible partial solution (see Definition 3.1). Therefore, by the definition of FAIRGREEDY, $f(g_j | g_1, \ldots, g_{j-1}) \geq f(o_j | g_1, \ldots, g_{j-1})$.

Therefore,

$$\begin{aligned} f(G) &= \sum_{j=1}^{k} f(g_j | g_1, \ldots, g_{j-1}) \\ &\geq \sum_{j=1}^{k} f(o_j | g_1, \ldots, g_{j-1}) \\ &\geq \sum_{j=1}^{k} f(o_j | g_1, \ldots, g_k, o_1, \ldots, o_{j-1}) \\ &= f(O|G), \end{aligned}$$

and so

$$2f(G) \geq f(O \cup G) \geq f(O).$$

□

A.2 Checking extendability

In order for Algorithm 1 to run in time $O(|V|k)$ we must solve the problem of generating the set $U = \{e \in V \mid S + e \text{ is extendable}\}$ in $O(|V|)$ time. That is, we must be able to check if adding a single element e to our set S would maintain extendability in $O(1)$ time.

This can be done by simply maintaining the counts $t_c = |S \cap V_c|$ of elements of each color in S, as well as the sum $Q = \sum_{c=1}^{C} \max(t_c, \ell_c)$. Recall Observation 3.2 which states that S is extendable if $t_c \leq u_c$ for each c and $Q \leq k$.

At the beginning of our algorithm we initialize these variables. Then, whenever a potential extension e of color c is considered, we call CANDIDATE(c) to determine whether adding it would maintain extendability. Once we augment S with an element e of color c, we update the stored variables using UPDATE(c).

Algorithm 4 Checking extendability

```
procedure INITIALIZE
    for c ∈ [C] do
        t_c ← 0
    Q ← Σ_{c=1}^{C} ℓ_c
procedure UPDATE(c)
    t_c ← t_c + 1
    if t_c > ℓ_c then
        Q ← Q + 1
procedure CANDIDATE(c)
    if t_c = u_c then
        return false
    if t_c < ℓ_c then
        return true
    if ℓ_c ≤ t_c < u_c then
        return Q < k
```

To implement the non-monotone submodular maximization algorithm of [28] which we use in Section 5.1, it is also useful to be able to verify whether a pair of elements can be swapped in the current solution. Suppose we are trying to add element e_1 of color c_1 to S, while removing element e_2 of color c_2. To verify if this is legal, we call SWAP(c_1, c_2).

Algorithm 5 Checking extendability

```
procedure SWAP(c_1, c_2)
    if c_1 = c_2 then
        return true
    if t_{c_1} = u_{c_1} then
        return false
    if Q = k and t_{c_1} ≥ ℓ_{c_1} and t_{c_2} ≤ ℓ_{c_2} then
        return false
    else
        return true
```

B Monotone Streaming – Proofs

B.1 Proof of Lemma 4.1

Proof. Let $\mathcal{B}$ consist of all maximal sets in $\mathcal{F}$. We will show that $\mathcal{B}$ satisfies the following two axioms.

(B1) $\mathcal{B} \neq \emptyset$.

(B2) If $B_1, B_2 \in \mathcal{B}$ and $x \in B_1 \setminus B_2$, then there exists $y \in B_2 \setminus B_1$ such that $B_1 - x + y \in \mathcal{B}$.

These axioms imply (see e.g. [53, Theorem 1.2.3]) that the downward closure (collection of all subsets) of $\mathcal{B}$ is a matroid (having $\mathcal{B}$ as its set of bases). However, the downward closure of $\mathcal{B}$ is equal to $\tilde{\mathcal{F}}$, as any subset of V that can be extended to a feasible solution can also be extended to a maximal feasible solution. Therefore we are left with proving (B1-B2). As we had assumed that $\mathcal{F} \neq \emptyset$, we also have $\mathcal{B} \neq \emptyset$, which establishes (B1).

For (B2), let $B_1, B_2 \in \mathcal{B}$ and $x \in B_1 \setminus B_2$. Let c be the color of x. A simple case is when $B_2 \setminus B_1$ contains some element $y \in V_c$. Then $B_1 - x + y$ has the same number of elements of each color as B_1, thus it is also in $\mathcal{B}$. Now consider the other case, i.e., that $V_c \cap B_2 \subseteq B_1 - x$. Then we have

$$u_c - 1 \geq |V_c \cap (B_1 - x)| \geq |V_c \cap B_2| \geq \ell_c \,. \tag{1}$$

There must be another color d where B_2 has more elements than B_1, for otherwise $B_2 + x$ would be feasible, contradicting the maximality of B_2. We claim that picking any element $y \in V_d \cap (B_2 \setminus B_1)$ yields a maximal feasible solution $B_1 - x + y \in \mathcal{B}$. The lower bounds are clearly satisfied already for $B_1 - x$ (for color c, this follows by (1)). The upper bound for color c is satisfied by (1), and for color d since $|V_d \cap (B_1 - x + y)| = |V_d \cap B_1| + 1 \leq |V_d \cap B_2| \leq u_d$. The global upper bound is satisfied as $|B_1 - x + y| = |B_1| \leq k$. To show maximality of $B_1 - x + y$, we note that any maximal set in $\mathcal{F}$ has the same size, namely $\min(k, \sum_c \min(u_c, |V_c|))$, and that $B_1 - x + y$ is already of the same size as B_1, which is maximal. □

B.2 Proof of Theorem 4.2

Proof. The feasibility of S follows as $S_{\mathcal{A}}$ is extendable and by Observation 3.2. If $\mathcal{A}$ is an α-approximation algorithm, then it returns a solution $S_{\mathcal{A}}$ of value at least α times that of the best extendable set, and every feasible set is extendable. Adding elements does not decrease the value, as f is monotone.

Our extra memory usage is $|\bigcup_c B_c| = \sum_c \ell_c \leq k$. □

C Algorithms for Matroid-Constrained Submodular Maximization

In this section we describe the streaming algorithms for submodular maximization under a matroid constraint of Chakrabarti and Kale [14] (monotone $1/4$-approximation) and Feldman, Karbasi and Kazemi [28] (non-monotone $1/5.82$-approximation). We also describe how to implement FAIR-STREAMING, together with the former algorithm, so as to obtain nearly-linear runtime and oracle complexity.

Both algorithms are given access to a matroid $\mathcal{M} \subseteq 2^V$ in the form of an independence oracle. To differentiate between querying f and $\mathcal{M}$, we refer to the former as oracle calls and to the latter as matroid queries.

C.1 The monotone case

Algorithm 6 Chakrabarti-Kale [14] (monotone)

$S \leftarrow \emptyset$
for every arriving element e **do**
  $w(e) \leftarrow f(e \mid S)$
  if $S + e \in \mathcal{M}$ **then**
    $S \leftarrow S + e$
  else
    $U \leftarrow \{e' \in S : S + e - e' \in \mathcal{M}\}$
    $e' \leftarrow \operatorname{argmin}_{e' \in U} w(e')$
    if $w(e) \geq 2w(e')$ **then**
      $S \leftarrow S + e - e'$
return S

Let us look at the per-element oracle complexity and runtime. Algorithm 6 clearly makes only two oracle calls (to compute $f(e \mid S)$). As for the runtime, it is dominated by Lines 4, 7 and 8. Clearly, these can be implemented naively using $O(k)$ time and matroid queries, where k is the rank of matroid $\mathcal{M}$ (we have $|S| \leq k$). The runtimes of these queries would further depend on the matroid in question.

However, for special matroids $\mathcal{M}$ the implementation can be optimized. Let us first consider the special case of $\mathcal{M}$ being the k-uniform matroid ($S \in \mathcal{M} \Leftrightarrow |S| \leq k$): in other words, the setting of cardinality-constrained submodular maximization. In that case, Line 4 takes $O(1)$ time, and Line 7 becomes just $U \leftarrow S$. The runtime then becomes dominated by finding the element $e' \in S$ with the lowest w-weight. If we maintain a priority queue P containing S sorted by w, then this can be done in $O(\log k)$ time.

Now we can extend this idea to $\mathcal{M}$ being the extendability matroid (see Definition 3.1 and Lemma 4.1) used by FAIR-STREAMING. That is, we prove Theorem 4.4. Let us restate it again for convenience.

Theorem 4.4 (Streaming monotone) *There exists a streaming algorithm for fair monotone submodular maximization that attains $1/4$-approximation, using $O(k)$ memory. This algorithm uses $O(\log k)$ time and 2 oracle calls per element.*

Proof. Recall that FAIR-STREAMING (Algorithm 2) uses Algorithm 6 as $\mathcal{A}$. By Theorem 4.2, FAIR-STREAMING returns a feasible solution that is $1/4$-approximate. It makes 2 oracle calls per element (these are made by Algorithm 6, see above). We are left with the runtime.

We maintain the extendability data structure from Appendix A.2. This allows us to implement Line 4 in constant time. Now let us consider the problem of finding the minimal $w(e')$ among $e' \in U$, i.e., among those elements $e' \in S$ that have $S + e - e' \in \mathcal{M}$. Clearly, whether an element $e' \in S$ is in U or not depends only on its color c'. We will say that color c' is *good* if elements $e' \in S$ of color c' are in U. Let c be the color of e. Following Algorithm 5, we have the following logic:

- if $t_c = u_c$, then only c is good,
- otherwise, if $Q < k$ or $t_c < \ell_c$, then every color is good,
- otherwise, the good colors are c and those colors c' that have $t_{c'} > \ell_{c'}$.

To be able to quickly find the minimum-weight good-colored element in S, we will maintain a number of priority queues:

- (as before) P containing S sorted by w,
- P_c for each color c, where we keep elements in $S \cap V_c$ sorted by w,
- P', containing colors rather than elements: in P' we keep those colors c' for which $t_{c'} > \ell_{c'}$, sorted by $\min_{e' \in S \cap V_{c'}} w(e')$.

It is not hard to see that this data structure can be maintained in $O(\log k)$ time per element, and that using it we can implement the logic above in the same time. □

Our implementation In the experimental evaluations, we use a variant of FAIR-STREAMING where the condition in Line 9 of Algorithm 6 is replaced by the more direct $f(S + e - e') \geq f(S)$. We find that this yields better solutions in practice. We still make only two oracle calls per element; this is made possible by storing the value $f(S)$ between calls. For simplicity, we also do not use the priority-queue-based data structure from the above proof of Theorem 4.4. This has no bearing on the reported experimental results, as we measure oracle calls rather than runtime.

C.2 The non-monotone case

The non-monotone algorithm of Feldman, Karbasi and Kazemi [28], which is used by FAIR-SAMPLE-STREAMING, is similar to Algorithm 6. The main differences are that the algorithm subsamples incoming elements, and that instead of caching the marginal contribution of every element at the time it is added (as $w(e)$), it always uses the contribution of an element e to the part of the current solution that arrived before e. For completeness, we give it as Algorithm 7.

Algorithm 7 Feldman, Karbasi and Kazemi [28] (non-monotone)

1: $S \leftarrow \emptyset$
2: **for** every arriving element e **do**
3: with probability $2/3$ **return**
4: **if** $S + e \in \mathcal{M}$ **then**
5: $S \leftarrow S + e$
6: **else**
7: $U \leftarrow \{e' \in S : S + e - e' \in \mathcal{M}\}$
8: $e' \leftarrow \operatorname{argmin}_{e' \in U} f(e' : S)$
9: **if** $f(e \mid S) \geq 2f(e' : S)$ **then**
10: $S \leftarrow S + e - e'$
11: **return** S

Here we use the notation $f(e' : S)$ to denote $f(e' \mid S')$, where S' consists of those elements of S that had arrived on the stream before e'. Note that this is different from $w(e')$ from Algorithm 6.

Algorithm 7 uses $O(k)$ oracle calls and $O(k)$ matroid queries per element.

Our implementation As previously, in the experimental evaluations, in FAIR-SAMPLE-STREAMING we use a variant of Algorithm 7 where the condition in Line 9 is replaced by the more direct $f(S + e - e') \geq f(S)$. We also use $f(e' \mid S)$ in lieu of $f(e' : S)$. Finally, whenever we apply Algorithm 7 in a monotone setting, we omit Line 3.

D Non-monotone Streaming

D.1 Non-monotone algorithm

We make use of the following known lemma to bound the loss in value resulting from the addition of backup elements.

Lemma D.1 ([7, Lemma 2.2]) *Let $g : 2^V \to \mathbb{R}_{\geq 0}$ be a non-negative submodular function, and let B be a random subset of V containing every element of V with probability at most p (not necessarily independently). Then $\mathbb{E}[g(B)] \geq (1-p)g(\emptyset)$.*

Theorem 5.2 *Suppose $\mathcal{A}$ is a streaming α-approximate algorithm for non-monotone submodular maximization under a matroid constraint. Then, there exists a streaming algorithm for fair non-monotone submodular maximization with expected $q\alpha$ approximation ratio, and the same memory usage, oracle calls, and running time as $\mathcal{A}$.*

Proof. By assumption, we have $\mathbb{E}[f(S_{\mathcal{A}})] \geq \alpha \max_{S \in \tilde{\mathcal{F}}} f(S)$, and since $\mathcal{F} \subseteq \tilde{\mathcal{F}}$, we have $\mathbb{E}[f(S_{\mathcal{A}})] \geq \alpha f(\mathrm{OPT})$. We define $g : 2^V \to \mathbb{R}_{\geq 0}$ to be the function $g(S) = f(S \cup S_{\mathcal{A}})$, and $B = S \setminus S_{\mathcal{A}}$ the set of backup elements added to $S_{\mathcal{A}}$. Since B contains every element in V with probability at most $1 - q = \max_c \frac{\ell_c}{n_c}$, then by Lemma D.1 $\mathbb{E}[g(B)] \geq q \cdot g(\emptyset)$. It follows then that

$$\mathbb{E}[f(S)] \geq q\,\mathbb{E}[f(S_{\mathcal{A}})] \geq q\alpha f(\mathrm{OPT}).$$

□

D.2 Non-monotone hardness

In this section we will show that our assumption that the dependence of our approximation ratio on $q = 1 - \max_{c \in [C]} \ell_c / n_c$ is necessary. Indeed, to get an approximation ratio better than q for fair non-monotone submodular maximization requires nearly linear space. We prove this by reduction to the INDEX problem which we define below.

Definition D.2 *The INDEX problem is a two party communication problem. In it we have two parties, Alice and Bob. Alice receives x, a bit string of length n, and Bob receives a single index i^* between 1 and n. The aim of problem is for bob to output x_{i^*}.*

Theorem D.3 *[39] The one way communication complexity of index, $R^{pub}_{2/3} \geq n/100$. That is any one way communication protocol that solves INDEX on any input with probability at least $2/3$ requires at least $n/100$ bits of communication.*

We use this to prove hardness of the approximate maximization of non-monotone submodular functions under fairness constraints. Specifically we will show a reduction from INDEX to this problem.

Theorem 5.1 (Hardness non-monotone) *For any constant $\epsilon > 0$ and $q \in [0, 1]$, any algorithm for fair non-monotone submodular maximization that outputs a $(q + \epsilon)$-approximation for inputs with excess ratio above q, with probability at least $2/3$, requires $\Omega(n)$ memory.*

Proof. Suppose such an algorithm exists. We will produce an instance of such submodular maximization that allows us to solve INDEX with the same space complexity and success probability.

The submodular function we define will be a cut function. That is, we define some directed graph $D = (V, A)$ on the universe V. The function evaluated at $S \subseteq V$ will be the size of the $(S, \overline{S})$ cut. That is

$$f(S) = |\{(v, w) \in A : v \in S \ \wedge \ w \notin S\}|.$$

It is easy to see that this is indeed a non-negative submodular function.

It remains to define V and D. Suppose Alice and Bob receive an input for INDEX for length n. Let the input of Alice be x and the input of Bob be i^*. We define V and A based on this input

Let a/b be a rational approximation of q in the sense that $a, b \in \mathbb{N}$ and $q \leq a/b < q + \epsilon$. Such a and b can always be chosen such that $b = O(1/\epsilon)$. Let V consist of three colors V_1, V_2, and V_3 where $V_1 = \{v_i : i \in [n], x_i = 1\} \cup \{w_i : i \in [n], x_i = 0\}$, $V_2 = \{y^j_{i^*} : j \in [b]\}$ and $V_3 = \{z^j : j \in [b]\}$. Let the color-wise constraints be $\ell_1 = u_1 = 1$, $\ell_2 = u_2 = b - a$, and $\ell_3 = u_3 = 0$, which satisfies $1 - \max_{c\in[3]} \ell_c/n_c = a/b \geq q$. If the element u_{i^*} appears (that is if $x_{i^*} = 1$), it is connected to V_3, that is A contains all edges in $\{v_{i^*}\} \times V_3$. All other elements of V_1 are connected to all elements of V_2, that is A contains all edges in $V_1 \backslash \{v_{i^*}\} \times V_2$.

Alice first runs the algorithm for submodular maximization on a stream consisting of V_1. Since $f|_{V_1}$ is simply cardinality times b, Alice can answer all oracle queries without knowing Bob's input. Alice then passes the state of the algorithm to Bob, who inputs the rest of the stream: V_2 and V_3. As we show below, if $x_{i^*} = 1$, the optimal solution is b, while if $x_{i^*} = 0$, the optimal solution is only a. Therefore, Bob can correctly solve INDEX by reading off the output of the $(q + \epsilon)$-approximation algorithm, since $q + \epsilon > a/b$.

Indeed, if $x_{i^*} = 0$ and $v_{i^*} \notin V$, then

$$f(S) = |S \cap V_1| \cdot (b - |S \cap V_2|).$$

Given the strict color-wise constraints this is always equal to $b - a$. On the other hand, if $x_{i^*} = 1$ and $v_{i^*} \in V$ then we have the optimal solution

$$S = \{v_{i^*}\} \cup \{y^j_{i^*} : j \in [a]\}$$

which has value b.

Since INDEX needs $\Omega(n)$ memory to solve, the algorithm for fair submodular maximization must have $\Omega(n)$ memory as well. □