Abstract

Streaming submodular maximization is a natural model for the task of selecting a representative subset from a large-scale dataset. If datapoints have sensitive attributes such as gender or race, it becomes important to enforce fairness to avoid bias and discrimination. This has spurred significant interest in developing fair machine learning algorithms. Recently, such algorithms have been developed for monotone submodular maximization under a cardinality constraint. In this paper, we study the natural generalization of this problem to a matroid constraint. We give streaming algorithms as well as impossibility results that provide trade-offs between efficiency, quality and fairness. We validate our findings empirically on a range of well-known real-world applications: exemplar-based clustering, movie recommendation, and maximum coverage in social networks.

1. Introduction

Recent years have seen a growing trend of utilizing machine learning algorithms to support or replace human decision-making. An undesirable effect of this phenomenon is the potential for bias and discrimination in automated decisions, especially in sensitive domains such as hiring, access to credit and education, bail decisions, and law enforcement (Munoz et al., 2016; White House OSTP, 2022; European Union FRA, 2022). In order to attenuate such risks, the computer science community has been working on developing fair algorithms for fundamental tasks such as classification (Zafar et al., 2017), ranking (Celis et al., 2018c; Singh & Joachims, 2019), clustering (Chierichetti et al., 2017; Backurs et al., 2019; Böhm et al., 2021; Jia et al., 2022; Anegg et al., 2022; Angelidakis et al., 2022), online learning (Joseph et al., 2016; Chzhen et al., 2021), voting (Celis et al., 2018a), matching (Chierichetti et al., 2019), influence maximization (Tsang et al., 2019; Rahmattalabi et al., 2021), data summarization (Celis et al., 2018b), online selection (Correa et al., 2021), and graph problems (Rahmattalabi et al., 2019; Anagnostopoulos et al., 2020).

In this paper, we study fairness in the fundamental problem of streaming monotone submodular maximization over a matroid constraint. Submodularity is a well-studied property of set functions that captures the natural notion of diminishing returns and has found vast applications in machine learning, including active learning (Golovin & Krause, 2011), data summarization (Lin & Bilmes, 2011), feature selection (Das & Kempe, 2011), and recommender systems (El-Arini & Guestrin, 2011). Matroids are a popular and powerful class of independence systems, capturing a wide range of useful constraints such as cardinality, block cardinality, linear independence, and connectivity constraints. In all of the above applications, it is crucial to have the capacity to handle the massive volume of modern datasets, which are often produced so rapidly that they cannot even be stored in memory. This has motivated a long line of work on the streaming setting.

Without considering fairness, maximizing a monotone submodular function over a matroid constraint is a well-studied problem. While a tight \((1 − 1/e)\)-approximation is known in the centralized setting (Călinescu et al., 2007; Feige, 1998) and in the multi-pass streaming setting (Feldman et al., 2022), the single-pass approximability of the problem is still not settled. Currently, the best one-pass algorithm yields a \(0.3178\)-approximation (Feldman et al., 2022), while the best known upper bound is \(\frac{e}{e-1}\) (Oveis Gharan & Vondrak, 2011).

Fairness in the context of submodular maximization problems has already been investigated under a cardinality constraint, in both centralized and streaming models (Celis et al., 2018a; El Halabi et al., 2020). Defining the notion of algorithmic fairness is an active line of research and, although several notions have been proposed, no universally accepted metric exists. Here, we follow the common notion used in many previous works (Celis et al., 2018a;b;c;...
Chierichetti et al., 2017; 2019), where we require the solution to be balanced with respect to some sensitive attribute (e.g., race, political affiliation). Formally, given a set \( V \) of items, each item is assigned a color \( c \) representing a sensitive attribute. Let \( V_1, \ldots, V_C \) be the corresponding \( C \) disjoint groups of items of the same color. A set \( S \subseteq V \) is fair if it satisfies \( \ell_e \leq |S \cap V_c| \leq u_e \) for a given choice of lower and upper bounds \( \ell_e, u_e \in \mathbb{N} \). This definition encompasses several other existing notions of fairness such as statistical parity (Dwork et al., 2012), diversity rules (Cohoon et al., 2013; Biddle, 2006), and proportional representation rules (Monroe, 1995; Brill et al., 2017). For a more extended overview we refer to Celis et al. (2018a, Section 4).

1.1. Our contribution

We present streaming algorithms as well as impossibility results for the problem of fair matroid monotone submodular maximization (which we abbreviate to FMMSM), that provide trade-offs between memory and computation efficiency, quality and fairness.

We start by extending the result of Huang et al. (2022) to present a \( 1/2\)-approximation algorithm that uses memory exponential with respect to \( C \) and \( k \), where \( k \) is the rank of the input matroid. From the solution quality point of view, this result is tight: the approximation factor of \( 1/2 \) cannot be improved in the streaming setting using memory that is less than linear with respect to \( |V| \) (Feldman et al., 2020).

**Theorem 1.1.** For any constant \( \eta \in (0, 1/2) \), there exists a one-pass streaming \((1/2 - \eta)\)-approximation algorithm for FMMSM that uses \( 2^{O(k^2 + k \log C)} \cdot \log \Delta \) memory, where

\[
\Delta = \min_{v \in V} \max_{c \leq |f(v)| \leq \eta} J_c(v).
\]

The algorithm and its analysis are presented in Appendix E. Motivated by this result, we focus on memory-efficient algorithms, which are referred to as semi-streaming algorithms in the literature. An algorithm is semi-streaming if the memory that it uses is \( \tilde{O}(m) \), where \( m \) is the size of the output. We prove that, unlike the cardinality constraint case, it is impossible to design a multi-pass semi-streaming algorithm that even finds a feasible solution for a matroid constraint.

**Theorem 1.2.** Any (randomized) \( O(\sqrt{\log C}) \)-pass streaming algorithm that determines the existence of a feasible solution for FMMSM with probability at least \( 2/3 \) requires \( \max(k, C)^2 - o(1) \) memory.

Motivated by Theorem 1.2, we relax the constraints by allowing the fairness lower bounds to be violated by a factor 2. More precisely, the goal is to find a solution \( S \), feasible with respect to the matroid constraint, that maximizes the value of the submodular function while satisfying \( |\ell_e/2| \leq |S \cap V_c| \leq u_e \) for any color \( c = 1, \ldots, C \). We present a two-pass 1/11.656-approximation algorithm in this case.

**Theorem 1.3.** There exists a two-pass streaming algorithm for FMMSM that runs in polynomial time, uses \( O(k \cdot C) \) memory, and outputs a set \( S \) such that (i) \( S \) is independent, (ii) it holds that \( |\ell_e/2| \leq |V_c \cap S| \leq u_e \) for any color \( c = 1, \ldots, C \), and (iii) \( f(S) \geq \text{OPT}/11.656 \).

Note that although our algorithm is relatively memory-efficient, it is not a semi-streaming algorithm. Another limitation of our algorithm is that it operates in two passes, instead of a single pass over the stream. We show that at least one of these limitations is necessary, by proving that any one-pass semi-streaming algorithm, which violates the fairness bounds even further than our algorithm, still cannot find a feasible solution.

**Theorem 1.4.** There is no one-pass semi-streaming algorithm that determines the existence of a feasible solution for FMMSM with probability at least \( 2/3 \), even if it is allowed to violate the fairness lower bounds by a factor of 2 and completely ignore the fairness upper bounds.

In Appendices B and C, we investigate the special case of modular objectives. There, we present efficient exact algorithms for both the streaming and the centralized versions of the problem.

Finally, we study the performance of our algorithm in multiple real-world experiments: exemplar-based clustering, movie recommendation, and maximum coverage in social networks. We introduce two heuristics that improve the quality of the solution of our two-pass algorithm empirically. Moreover, we present a one-pass heuristic algorithm, based on the ideas of our two-pass algorithm, which is guaranteed to satisfy both the matroid and fairness constraints, but has no worst-case guarantee on the objective value. We observe that our two-pass algorithm achieves similar quality to “unfair” baselines, while incurring significantly fewer violations of the fairness constraint. Interestingly, our one-pass heuristic algorithm achieves quality that is not too far from our two-pass algorithm, without violating the fairness constraint.

1.2. Related work

The problem of fair submodular maximization has already been studied under a cardinality constraint. Celis et al. (2018a) provide a tight \((1 - 1/e)\)-approximation to the problem in the centralized setting using a continuous greedy algorithm. The streaming setting has been investigated by El Halabi et al. (2020), who show (i) a \( 1/2 \)-approximation one pass algorithm that uses memory exponential in \( k \) (this result is tight, see Feldman et al. (2020)) and (ii) a \( 1/4 \)-approximation one pass algorithm, which uses only \( O(k) \) memory and processes each element of the stream.
in $O(\log k)$ time and 2 oracle calls.

A closely related problem to FMMSM is monotone submodular maximization over two matroid constraints; FMMSM reduces to this problem when $\ell_c = 0$ for all $c$. Chakrabarti & Kale (2015) gave a 1/8-approximation one-pass streaming algorithm for this problem. The current state-of-the-art (Garg et al., 2021) is a 1/5.828-approximation one-pass streaming algorithm.

2. Preliminaries

We consider a ground set $V$ of $n$ items and a non-negative monotone submodular function $f : 2^V \to \mathbb{R}_+$. Given any two sets $X, Y \subseteq V$, the marginal gain of $X$ with respect to $Y$ quantifies the change in value of $f$ when adding $X$ to $Y$ and is defined as

$$f(X | Y) = f(X \cup Y) - f(Y).$$

We use the shorthand $f(x | Y)$ for $f(\{x\} | Y)$. The function $f$ is submodular if for any two sets $Y \subseteq X \subseteq V$, and any element $e \in V \setminus X$, it holds that $f(e | Y) \geq f(e | X)$. We say that $f$ is monotone if for any element $e \in V$ and any set $Y \subseteq V$ if holds that $f(e | Y) \geq 0$. 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., $f(\emptyset) = 0$.

**Matroids.** A non-empty family of sets $I \subseteq 2^V$ is called a *matroid* if it satisfies the following properties:

- **Downward-closedness:** if $A \subseteq B$ and $B \in I$, then $A \in I$;
- **Augmentation:** if $A, B \in I$ with $|A| < |B|$, then there exists $e \in B$ such that $A + e \in I$.

We write $A + e$ for $A \cup \{e\}$ and $A - e$ for $A \setminus \{e\}$. We call a set $A \subseteq V$ independent if $A \in I$. We assume that the matroid is available to the algorithm in the form of an independence oracle. An independent set that is maximal with respect to inclusion is called a *base*; all the bases of a matroid share the same cardinality $k$, referred to as the *rank* of the matroid. An important class of matroids are *partition matroids*, where the universe is partitioned into blocks $V = \bigcup_i V_i$, each with an upper bound $k_i$, and a set $A$ is independent if $|A \cap V_i| \leq k_i$ for all $i$.

A crucial property that follows directly from the definition of a matroid is that given two bases $B_1$ and $B_2$, one can find two elements $b_1 \in B_1$ and $b_2 \in B_2$ that can be swapped while maintaining independence, i.e., such that both $B_1 - b_1 + b_2$ and $B_2 - b_2 + b_1$ are independent. This property can be generalized to subsets, see e.g., (Schrijver, 2003, Statement 42.31 in Chapter 42):

**Lemma 2.1** (Exchange property of bases). In any matroid, for any two bases $B_1$ and $B_2$ and for any partition of $B_1$ into $X_1$ and $Y_1$, there is a partition of $B_2$ into $X_2$ and $Y_2$ such that both $X_1 \cup Y_2$ and $X_2 \cup Y_1$ are bases.

When two matroids $I_1$ and $I_2$ are defined on the same set $V$, it is possible to define their intersection $I_1 \cap I_2$ as the family of the subsets of $V$ that are independent for both matroids. Although the intersection of two matroids is generally not a matroid itself, it is still possible to efficiently compute a maximum-cardinality subset in it, or one of maximum weight when there are weights associated to elements.

**Lemma 2.2** (Theorem 41.7 in (Schrijver, 2003)). A maximum-weight common independent set in two matroids can be found in strongly polynomial time.

**Fair Matroid Monotone Submodular Maximization (FMMSM) problem.** Recall that each element of $V$ is assigned exactly one color from the set $\{1, \ldots, C\}$; $V_c$ is the set of elements of color $c$. We are given fairness bounds $(\ell_c, u_c)_{c=1,\ldots,C}$ and a matroid $I$ on $V$ of rank $k$. We denote by $F$ the collection of solutions feasible under the fairness and matroid constraints, i.e.,

$$F = \{ S \subseteq V | S \in I, \ell_c \leq |S \cap V_c| \leq u_c \ \forall c = 1, \ldots, C \}.$$

Note that we use *independent* to denote a set that respects the matroid constraint and *feasible* for a set in $F$. Clearly, feasibility implies independence, but not vice versa. The problem of maximizing a monotone submodular function $f$ under matroid and fairness constraints, which we abbreviate FMMSM, is defined as selecting a set $S \subseteq V$ with $S \in F$ to maximize $f(S)$. We use OPT to refer to a feasible set maximizing $f$. We assume a feasible solution exists, i.e., $F \neq \emptyset$. In this paper, we study approximations to this problem; in particular, we say that an algorithm is an $\alpha$-approximation to the problem when its output $\text{ALG}$ is in $F$ (or possibly, in some relaxed version of $F$) and has $f(\text{ALG}) \geq \alpha \cdot f(\text{OPT})$.

3. Semi-streaming Impossibility Results

In this section we present our impossibility results. We start by showing that even with $O(1)$ passes, finding a feasible solution requires more memory than the semi-streaming setting allows.

**Theorem 1.2.** Any (randomized) $O(\sqrt{\log C})$-pass streaming algorithm that determines the existence of a feasible solution for FMMSM with probability at least $2/3$ requires $\max(k, C)^{2 - o(1)}$ memory.

To prove this result, we exploit the fact that it is possible to capture perfect bipartite matching as the intersection of a partition matroid and a fairness constraint. We use the
following result for streaming bipartite matching, where, given a stream of edges $E$ that belong to a $2n$-vertex bipartite graph $G = (P \cup Q, E)$, the goal is to find a perfect matching. If a perfect matching exists, we say that $G$ is perfectly-matchable.

**Theorem 3.1** (Theorem 5.3 in (Chen et al., 2021)). Any randomized $O(\sqrt{\log n})$-pass streaming algorithm that, given a $2n$-vertex bipartite graph $G(P \cup Q, E)$, determines whether $G$ is perfectly-matchable with probability at least $2/3$, requires $n^{2-o(1)}$ memory.

Theorem 1.2 follows from Theorem 3.1 by simply setting up a partition matroid to enforce that every vertex in $P$ has at most one adjacent edge in the solution, and fairness constraints to enforce that every vertex in $Q$ has exactly one adjacent edge in the solution (we assign color $q$ to every edge $(p, q)$). We then have $k = |P| = n$ and $C = |Q| = n$. A more detailed proof can be found in Appendix F.

We continue by presenting our second hardness result, which is based on a reduction to the hardness result of Kapralov (2021). Its proof is provided in Appendix D.

**Theorem 1.4.** There is no one-pass semi-streaming algorithm that, given as input the edges of a perfectly-matchable bipartite graph $G = (P \cup Q, E)$, with probability at least $2/3$ finds a matching of size at least $\frac{2}{3}|P|$.

The above theorem shows that it is impossible to approximate the matching problem better than a factor $\frac{2}{3}$ in the semi-streaming model. This result does not yet directly imply Theorem 1.4, as in Theorem 1.4 we allow the fairness bounds to be violated. To handle this, we use the following lemma, which is the key ingredient in our reduction. We use $\deg_X(p)$ to denote the degree of a vertex $p$ in the set of edges $X$.

**Lemma 3.3.** There is no one-pass semi-streaming algorithm that, given as input the edges of a perfectly-matchable bipartite graph $G = (P \cup Q, E)$, with probability at least $2/3$ finds a set $X \subseteq E$ such that $\deg_X(p) \leq 2$ for all $p \in P$ and $\deg_X(q) = 1$ for all $q \in Q$.

**Proof.** Suppose towards a contradiction that such an algorithm $A$ exists. We use it to design a semi-streaming algorithm for the maximum matching problem as follows:

1. Initialize two copies of the algorithm $A, A'$ (where $A'$ will operate on a "flipped" graph whose edges come from $Q \times P$).
2. When an edge $(p, q)$ arrives:
   - pass $(p, q)$ to $A$
   - pass $(q, p)$ to $A'$
3. Let $X$ and $X'$ be the solutions returned by $A$ and $A'$, respectively
4. Let $X'' = \{(p, q) : (q, p) \in X'\}$
5. Return a maximum matching in $X \cup X''$

Note that the above algorithm uses $O(n)$ memory (where $n = |P|$). We show that it returns a matching of size at least $\frac{2}{3}n$, which contradicts Theorem 3.2. To see this, assume otherwise. Then, by König’s theorem, the graph $(P \cup Q, X \cup X'')$ contains an independent set $I$ of size larger than $2n - \frac{2}{3}n = \frac{4}{3}n$. It follows that either $|P \cap I| > \frac{2}{3}n$ or $|Q \cap I| > \frac{2}{3}n$.

We first consider the case $|P \cap I| > \frac{2}{3}n$. Focus on the edges in $X$ incident to $Q \cap I$. There are $|Q \cap I|$ many, because $\deg_X(q) = 1$ for all $q \in Q$. As $I$ is an independent set also in the graph $(P \cup Q, X)$, all these edges must have their other endpoints in $P \setminus I$. We have

$$2|P \setminus I| = 2n - 2|P \cap I| < \frac{4}{3}n - |P \cap I| < |Q \cap I|,$$

which means that some vertex in $P \setminus I$ must have degree larger than 2 in $X$, contradicting that $\deg_X(p) \leq 2$ for all $p \in P$.

For the case $|Q \cap I| > \frac{2}{3}n$ we proceed similarly, swapping the role of $P$ with $Q$ and $X$ with $X''$.

We are now ready to prove Theorem 1.4.

**Proof of Theorem 1.4.** We show that if such an algorithm $A$ exists, then it can be used to solve the problem from the statement of Lemma 3.3. Given any bipartite graph $G = (P \cup Q, E)$, let us define an instance of FMMSM on the edges $E$ as follows: the matroid constraint is given by a partition matroid that requires that for a solution $X \subseteq E$ we have $\deg_X(p) \leq 2$ for each $p \in P$; and the color constraints dictate that $\deg_X(q) = 2$ for each $q \in Q$ (that is, an edge $(p, q)$ has color $q$ and we set $\ell_F = w = 2$ for all colors $q$).

For each edge arriving on the stream, we pass two copies of it to $A$. Then, if we have a feasible instance of the problem from the statement of Lemma 3.3 (i.e., a perfectly-matchable graph), it gives rise to a feasible instance of FMMSM as in the paragraph above (taking two copies of the perfect matching gives a solution with all vertex degrees equal to
2). Now, if \( A \) is an algorithm as in the statement of this theorem, then it returns a solution \( X' \) with \( \deg_X'(p) \leq 2 \) for each \( p \in P \) and \( \deg_X(q) \geq 1 \) for each \( q \in Q \). We obtain \( X \) from \( X' \) by simply removing, for each \( q \in Q \), any \( \deg_X(q) - 1 \) edges incident to \( q \). Then we have \( \deg_X(q) \leq 2 \) for all \( p \in P \) and \( \deg_X(q) = 1 \) for all \( q \in Q \), as required by Lemma 3.3.

Note that Theorem 1.4 does not rule out the existence of a two-pass semi-streaming algorithm with otherwise the same properties as in its statement. However, such an algorithm would give rise to a two-pass semi-streaming 2/3-approximation for maximum matching, for perfectly-matchable bipartite graphs (using the same arguments as in Theorem 1.4). This would significantly improve over the current state of the art, which is a \((2 - \sqrt{2}) \approx 0.585\)-approximation (Konrad, 2018).

4. Streaming Algorithm

In this section, we present a two-pass algorithm for FMMSM. In particular, we show how to transform any \( \alpha \)-approximation for streaming submodular maximization over the intersection of two matroids into an \( \alpha/2 \)-approximation for FMMSM, at the cost of a factor-2 violation of the fairness lower bound constraints. Finally, we show that the problem can be solved exactly in one-pass in the special case of modular objectives.

4.1. First pass: finding a feasible set

The algorithm for the first pass, FAIR-RESERVOIR, is simple: it collects a maximal independent set \( I_c \) (with respect to \( \mathcal{I} \)) for each color independently. The number of kept elements is at most \( C \cdot k \) (recall that \( k \) is the rank of the matroid \( \mathcal{I} \)). Then it computes a feasible solution in \( \bigcup_c I_c \) as follows. First, it defines the partition matroid \( \mathcal{I}_C \) on \( V \) as:

\[
\mathcal{I}_C = \{ S \subseteq V | |V_c \cap S| \leq \ell_c \ \forall c = 1, \ldots, C \}. \tag{1}
\]

Second, it uses any polynomial-time algorithm to find a maximum-size common independent set in \( \bigcup_c I_c \) with respect to the two matroids \( \mathcal{I} \) and \( \mathcal{I}_C \).

To analyze FAIR-RESERVOIR we need two ingredients: that a feasible solution is always contained in \( \bigcup_c I_c \), and that our algorithm finds one.

**Lemma 4.1.** For each color \( c \), let \( I_c \subseteq V_c \) be any maximal subset that is independent with respect to \( \mathcal{I} \). Then, as long as \( \mathcal{F} \neq \emptyset \), there exists a feasible set \( R \subseteq \bigcup_c I_c \).

**Proof.** Let \( R \) be any set in \( \mathcal{F} \) such that \( |R \setminus \bigcup_c I_c| \) is minimal; note that such \( R \) exists as we are assuming that \( \mathcal{F} \neq \emptyset \). To prove the lemma it is enough to show that \( |R \setminus \bigcup_c I_c| \) is actually 0.

**Algorithm 1 FAIR-RESERVOIR**

1. \( I_c \leftarrow \emptyset \) for all \( c = 1, \ldots, C \)
2. for each element \( e \) on the stream do
3. \( \quad \) Let \( c \) be the color of \( e \)
4. \( \quad \) if \( I_c + e \in \mathcal{I} \) then \( I_c \leftarrow I_c + e \)
5. \( \quad \) Consider the partition matroid \( \mathcal{I}_C \) on \( V \) defined in (1)
6. \( \quad \) \( S \leftarrow \) a max-cardinality subset of \( \bigcup_c I_c \) in \( \mathcal{I} \cap \mathcal{I}_C \)
(Lemma 2.2)
7. \( \quad \) Return \( S \)

Assume towards a contradiction that \( |R \setminus \bigcup_c I_c| > 0 \). We show how to exchange an element \( x \in R \setminus \bigcup_c I_c \) for an element \( y \in \bigcup_c I_c \setminus R \) of the same color as \( x \), such that \( R - x + y \in \mathcal{I} \). This contradicts the choice of \( R \), as \(|(R - x + y) \setminus \bigcup_c I_c| = |R \setminus \bigcup_c I_c| - 1|.

Without loss of generality, assume that \((R \cap V_1) \setminus I_1 \neq \emptyset\), and let \( x \) be any of its elements. Extend \( I_1 \) to any maximal independent set \( I_1' \) in \( I_1 \cup R \) containing \( I_1 \). By maximality of \( I_1' \), and since \( R, I_1' \in \mathcal{I} \) we have

\[
|R - x| < |R| \leq |I_1'|.
\]

By the matroid augmentation property, there exists \( y \in I_1' \setminus (R - x) \) such that \( R - x + y \in \mathcal{I} \). Because

\[
I_1' \setminus (R - x) \subseteq (I_1 \cup R) \setminus (R - x) \subseteq I_1 \setminus R + x,
\]

we must have \( y \in I_1 \setminus R \) or \( y = x \). The latter is impossible; if \( y = x \), then \( x \in I_1' \) and thus \( I_1 + x \subseteq I_1' \); as the latter is independent, so is the former. However, as \( x \in V_1 \), this contradicts the maximality of \( I_1 \) (as an independent subset of \( V_1 \)). Thus we must have \( y \in I_1 \setminus R \) (and recall that \( R - x + y \in \mathcal{I} \), as desired.

**Theorem 4.2.** There exists a one-pass streaming algorithm that runs in polynomial time, uses \( O(k \cdot C) \) memory, and outputs a feasible solution.

**Proof.** Any set \( I_c \) computed by FAIR-RESERVOIR is a maximal subset of \( V_c \) that is independent in \( \mathcal{I} \), thus Lemma 4.1 guarantees the existence of a feasible set \( R \subseteq \bigcup_c I_c \). By the downward-closeness property of \( \mathcal{I} \), we can further assume that \( R \) has exactly \( \ell_c \) elements of each color \( c \) (by removing any elements beyond that number), i.e., that \( |R| = \sum_c \ell_c \). Note that any set independent in \( \mathcal{I}_C \) has at most \( \sum_c \ell_c \) elements; therefore a maximum-cardinality set \( S \) as returned by FAIR-RESERVOIR will necessarily have \( |S| \geq \sum_c \ell_c \) and thus \( |S \cap V_c| = \ell_c \). Hence, \( S \) is feasible.

4.2. Second pass: extending the feasible solution

Starting with the solution output by FAIR-RESERVOIR (which is feasible but has no guaranteed objective value), we show how to find in another pass a high-value independent set that also respects the fairness constraint, up to some
slack in the lower bounds. First, the feasible set $S$ is split into two sets $S_1$ and $S_2$ in a balanced way, i.e.,

$$||S_1 \cap V_c| - |S_2 \cap V_c|| \leq 1 \quad \forall c = 1, 2, \ldots, C.$$ 

Both $S_1$ and $S_2$ are independent in $I$ (as subsets of $S$). The goal of the second pass is to extend $S_1$ and $S_2$ by adding elements to them to maximize the submodular function. To that end, we construct two matroids for each of the sets $S_1$ and $S_2$ as follows. First, a partition matroid $I^C$ induced by the upper bounds on the colors (note the difference with $I_C$, where the partition was induced by the lower bounds):

$$I^C = \{X \subseteq V \mid |X \cap V_c| \leq u_c \quad \forall c = 1, 2, \ldots, C\}. \quad (2)$$

Second, two matroids $I_1$ and $I_2$ induced on $I$ by $S_1$ and $S_2$:

$$I_i = \{X \subseteq V \mid X \cup S_i \in I\}. \quad (3)$$

It is easy to verify that $I_i$ is indeed a matroid.

Let algorithm $A$ be any streaming algorithm that maximizes a monotone submodular function subject to two matroid constraints. We run two parallel independent copies of $A$: the first one with matroids $I^C, I_1$ and the second one with matroids $I^C, I_2$. Let $S'_1$ and $S'_2$ be the results of these two runs of the algorithm, respectively. We return the solution with the larger value, adding as many elements as necessary from $S_i$ to satisfy the relaxed lower bounds. The details of the algorithm are presented in FAIR-STREAMING.

**Algorithm 2 FAIR-STREAMING**

1: **Input**: Set $S$ from FAIR-RESERVOIR and routine $A$
2: $S_1 \leftarrow \emptyset, S_2 \leftarrow \emptyset$
3: for $e$ in $S$ do
4:   if $|S_1 \cap V_c| < |S_2 \cap V_c|$ then
5:     $S_1 \leftarrow S_1 + e$
6:   else
7:     $S_2 \leftarrow S_2 + e$
8: Define matroids $I^C, I_1, I_2$ as in Equations (2) and (3)
9: Run two copies of $A$, one for matroids $I^C, I_1$ and one for matroids $I^C, I_2$, and let $S'_1$ and $S'_2$ be their outputs
10: for $i = 1, 2$ do
11:   for $e$ in $S_i$ do
12:     Let $e$ be the color of $e$
13:     if $|S'_i \cap V_c| < u_c$ then $S'_i \leftarrow S'_i + e$
14: Return $S' = \arg \max \{f(S'_1), f(S'_2)\}$

We begin the analysis of FAIR-STREAMING by bounding the violation with respect to upper and lower bounds.

**Lemma 4.3.** The output $S'$ of FAIR-STREAMING is independent in $I$ and for any color $c$ it holds that $|c/2| \leq |V_c \cap S'| \leq u_c$.

**Proof.** Without loss of generality assume that $S' = S'_1$ and divide it into two parts: the elements $X$ that were added by $A$, and the elements $Y$ that were added from $S'_1$ in the for loop on Line 12. As $X$ is in $I_2$, we have $X \cup S_1 \in I$ and therefore also $S'_1 = X \cup Y \in I$ by downward-closedness.

Consider now the color matroid $I^C$ that models the upper bounds $u_c$. As $X$ is in $I^C$, and the elements added in the for loop on Line 12 never violate the upper bounds, we have $S'_1 \in I^C$.

Finally we consider the constraints $\ell_c$ and show that $|c/2| \leq |V_c \cap S'|$ for all colors $c$. The set $S$ output by FAIR-RESERVOIR is such that $|V_c \cap S| \geq \ell_c$ and is then divided into $S_1$ and $S_2$ in a balanced way, so that

$$|S_1 \cap V_c| \geq |\ell_c/2| \quad \forall c = 1, 2, \ldots, C. \quad (4)$$

For any color $c$ such that $|S'_1 \cap V_c| < u_c$, all the elements in $S_1 \cap V_c$ are added to $S'_1$, and thus the guarantees on the lower bounds in (4) are passed onto $S'_1$.

Let $S$ be the set output by FAIR-STREAMING that is then divided into $S_1$ and $S_2$, and let OPT be the optimal solution. We apply Lemma 2.1 on the partition $S_1, S_2$ of $S$ and $\text{OPT}$. Thus $\text{OPT}$ can be partitioned into two sets $O_1$ and $O_2$ such that $O_1 \cup S_1 \in I$ and $O_2 \cup S_2 \in I$ or, equivalently, $O_i \in I_i$ for $i = 1, 2$. Moreover, both $O_1$ and $O_2$ respect the color matroid $I^C$, thus the approximation guarantee of $A$ and the monotonicity of $f$ imply that

$$f(S'_i) \geq \alpha \cdot f(O_i) \quad \forall i = 1, 2. \quad (5)$$

Now, we are ready to prove the result:

$$f(S') \geq \frac{1}{2} (f(S'_1) + f(S'_2)) \geq \frac{\alpha}{2} (f(O_1) + f(O_2)) \geq \frac{\alpha}{2} f(\text{OPT})$$

where the first inequality follows by the definition of $S'$, the second by (5), and the last one by submodularity.

If we plug in the state-of-the-art $1/5.828$-approximation algorithm by Garg et al. (2021) as $A$, we get Theorem 1.3:

**Theorem 1.3.** There exists a two-pass streaming algorithm for FMMSM that runs in polynomial time, uses $O(k \cdot C)$ memory, and outputs a set $S$ such that (i) $S$ is independent, (ii) it holds that $|\ell_c/2| \leq |V_c \cap S| \leq u_c$ for any color $c = 1, 2, \ldots, C$, and (iii) $f(S) \geq \text{OPT}/11.656$. 

---

2Lemma 2.1 is stated for bases of the matroid $I$, but it clearly holds also for general independent sets.
Heuristics. Although in principle, the feasible solution chosen in the first pass does not need to have any value, empirically it helps to choose a feasible solution with good value. In our empirical evaluation (Section 5), we use an alternative algorithm, GREEDY-FAIR-RESERVOIR, in the first pass instead of FAIR-RESERVOIR. Rather than collecting a maximal independent set \( I_c \) of arbitrary elements for each color \( c \), GREEDY-FAIR-RESERVOIR picks elements greedily (see Appendix A.1 for details). Similarly, instead of adding arbitrary elements of \( S_1, S_2 \) at the end (lines 11-14 in FAIR-STREAMING), we can use \( A \) to select good elements of \( S_1, S_2 \) to add (see Appendix A.2 for details). We call the resulting algorithm FAIR-STREAMING+.

We also propose a simple one-pass heuristic streaming algorithm, GREEDY-FAIR-STREAMING, which runs GREEDY-FAIR-RESERVOIR to find a feasible solution, then greedily augments it with elements from \( \bigcup_c I_c \) (see Appendix A.3 for details).

### 4.3. The modular case

We can obtain better results in the special case of modular objectives. In particular, we present in Appendix B a one-pass algorithm which solves the fair matroid modular maximization (F3M) problem exactly. The algorithm greedily collects maximal independent sets \( I_c \) for each color \( c \) in the same way as in GREEDY-FAIR-RESERVOIR, then returns an optimal feasible solution in \( \bigcup_c I_c \). The second step can be done in polynomial time, as we show in Appendix C, where we present two polynomial time algorithms for the centralized version of F3M. For further details, we refer the reader to Appendices B and C.

**Theorem 4.5.** There exists a one-pass streaming algorithm for F3M, which finds an optimal solution, uses \( O(k \cdot C) \) memory, and runs in polynomial time.

### 5. Empirical Evaluation

In this section, we empirically evaluate the performance of our algorithms on three applications: maximum coverage, movie recommendation, and exemplar-based clustering, with various choices of fairness and matroid constraints. In comparing our algorithms against two baselines, we consider objective values, as well as violations of fairness constraints, which we define for a given set \( S \) as

\[
\text{err}(S) = \sum_c \max \{ |S \cap V_c| - u_c, \ell_c - |S \cap V_c|, 0 \}.
\]

Each term in this sum counts the number of elements by which \( S \) violates the lower or upper bound. Note that \( \text{err}(S) \) is in the range \([0, 2k]\). We compare the following algorithms:

- **TwoPass-Fair-Streaming**: using GREEDY-FAIR-RESERVOIR in the first pass, and FAIR-STREAMING+ in the second pass with \( A = \text{MATROID-INTERSECTION} \), explained below.
- **Greedy-Fair-Streaming**: a one-pass heuristic algorithm based on the ideas of our two-pass algorithm (see Appendix A.3 for details).
- **Matroid-Intersection**: streaming algorithm for submodular maximization over two matroid constraints (Chakrabarti & Kale, 2015) with \( I \) and \( I^C \).
- **Random**: randomly selects a base set in \( I \); no fairness constraints.

We describe below the setup of our experiments. We select fairness bounds \( \ell_c, u_c \) which yield instances with feasible solutions, and enforce either that each color group \( V_c \) comprises a similar portion of the solution set \( S \) (in exemplar-based clustering) or that they have a similar representation in \( S \) as in the entire dataset (in maximum coverage and movie recommendation). We report the results in Figure 1, and discuss them in Section 5.4. Varying the specific values of the bounds yields qualitatively very similar results. The code is available at https://github.com/google-research/google-research/tree/master/fair_submodular_matroid.

#### 5.1. Maximum coverage

Given a graph \( G = (V, E) \), we aim to select a subset of nodes \( S \subseteq V \) that maximize the coverage of \( S \) in the graph, given by the monotone submodular function \( f(S) = \bigcup_{v \in S} N(v) \), where \( N(v) = \{ u : (v, u) \in E \} \) denote the set of neighbors of \( v \). We use the Pokec social network (Leskovec & Krevl, 2014), which consists of 1,632,803 nodes, representing users, and 30,622,564 edges, representing friendships. Each user profile contains attributes such as age, gender, height and weight, which can have “null” value. We impose a partition matroid constraint with respect to body mass index (BMI), which is calculated as the squared ratio between weight (in kg) and height (in m). We discard all profiles with no set height or weight (around 60%), as well as profiles with clearly fake data (fewer than 2%). The resulting graph has 582,289 nodes and 5,834,695 edges. We partition users into four standard BMI categories (underweight, normal weight, overweight and obese). We set the upper bound for each BMI group \( V_c \) to \( k_i = \frac{|V_c|}{V} k \).

The resulting rank of the matroid is then roughly \( k \). We also impose a fairness constraint with respect to age, with 7 groups: \([1, 10], [11, 17], [18, 25], [26, 35], [36, 45], [46+]\), and the last group comprised of records with “null” age (around 30%). We set \( \ell_c = 0.9 \frac{|V_c|}{V} k \) and \( u_c = 1.5 \frac{|V_c|}{V} k \). We vary \( k \) between 10 and 200. The results are shown in Figure 1a and 1d.
5.2. Exemplar-based clustering

We consider a dataset containing 4,521 records of phone calls in a marketing campaign ran by a Portuguese banking institution (Moro et al., 2014). We aim to find a representative subset of calls \( S \subseteq V \) in order to assess the quality of service. We use 7 features with numeric values (age, account balance, last contact day, duration, number of contacts during the campaign, number of days that passed by after the client was last contacted from a previous campaign, number of contacts performed before this campaign) to represent each record \( e \in V \) in the Euclidean space as \( x_e \in \mathbb{R}^7 \). We impose a partition matroid constraint with respect to account balance, with 5 groups: \((-\infty, 0), [0, 2000), [2000, 4000), [4000, 6000), [6000, \infty)\). We choose equal upper bounds \( k_i = k/5 \) for each age group \( V_i \). The resulting rank of the matroid is then at most \( k \). We also impose a fairness constraint with respect to age, with 6 groups: \([0, 29], [30, 39], [40, 49], [50, 59], [60, 69], [70+]\). We set our fairness bounds as \( \ell_c = 0.1k + 2 \) and \( u_c = 0.4k \) for each color \( 1 \leq c \leq 6 \). Then we maximize the following monotone submodular function (Gomes & Krause, 2010):

\[
f(S) = \sum_{e' \in V} \left( d(e', 0) - \min_{e' \in S \cup \{0\}} d(e', e) \right)
\]

where \( d(e', e) = \|x_{e'} - x_0\|_2^2 \) and \( x_0 \) is a phantom exemplar, which we choose to be the origin. We vary \( k \) between 25 and 60. The results are shown in Figure 1b and 1e.

5.3. Movie recommendation

We emulate a movie recommendation system using the Movielens 1M dataset (Harper & Konstan, 2016), which includes approximately one million ratings for 3,900 movies by 6,040 users. We implement the experimental design of previous research (Mitrović et al., 2017; Norouzi-Fard et al., 2018; El Halabi et al., 2020) by computing a low-rank completion of the user-movie rating matrix (Troyanskaya et al., 2001), resulting in feature vectors \( w_u, v_m \in \mathbb{R}^{20} \) for each user \( u \) and \( v_m \in \mathbb{R}^{20} \) for each movie \( m \). The product \( v_m^T w_u \) approximates the rating of movie \( m \) by user \( u \). The monotone submodular utility function \( f_u(S) \) tailored to user \( u \) for a set \( S \) of movies is defined as:

\[
\alpha \cdot \sum_{m' \in M} \max \left( \max_{m \in S} (v_m^T v_{m'}), 0 \right) + (1 - \alpha) \cdot \sum_{m \in S} w_u^T v_m.
\]

The parameter \( \alpha = 0.85 \) interpolates between optimizing the coverage of the entire movie collection and selecting movies that maximize the user’s score. We enforce a proportional representation in terms of movie release dates using a laminar matroid with 9 groups for each decade \( d \) between 1911 and 2000, and three groups for each 30-year period \( t \): 1911–1940, 1941–1970, 1971–2000. We set an upper bound of \( \left\lceil 1.2 \frac{|V_d|}{|V|} k \right\rceil \) for each decade group \( V_d \), and an upper bound of roughly \( \frac{|V_t|}{|V|} k \) for each 30-year period group \( V_t \). The resulting rank of the matroid is then roughly \( k \). We also partition the movies into 18 genres \( c \), which we model using colors. As fairness constraint, we set \( \ell_c = \left\lceil 0.8 \frac{|V_d|}{|V|} k \right\rceil \) and \( u_c = \left\lceil 1.4 \frac{|V_t|}{|V|} k \right\rceil \). We vary \( k \) between 10 and 200. The results are shown in Figure 1c and 1f.
5.4. Results

We compare the results of our proposed algorithms, **TWOPASS-FAIR-STREAMING** and **GREEDY-FAIR-STREAMING**, with the baselines, **MATROID-INTERSECTION** and **RANDOM**—see Figure 1. We observe that the value of the submodular function for **TWOPASS-FAIR-STREAMING** and **GREEDY-FAIR-STREAMING** is lower than **MATROID-INTERSECTION** by at most 15% and 26%, respectively, while the violation in the fairness constraint is significantly higher for **MATROID-INTERSECTION**. Indeed, **GREEDY-FAIR-STREAMING** does not violate the fairness constraint in any of the experiments, as guaranteed theoretically (see Appendix A.3). And the violation of **TWOPASS-FAIR-STREAMING** is often 2 − 3 times lower than **MATROID-INTERSECTION**. The objective value of **RANDOM** is significantly lower than the other three algorithms on maximum coverage and movie recommendation. Surprisingly, on exemplar-based clustering, **RANDOM** obtains objective values better than **GREEDY-FAIR-STREAMING** and **MATROID-INTERSECTION**, and similar to **TWOPASS-FAIR-STREAMING**, for several $k$ values. This comes however at the cost of significant fairness violations.

6. Broader Impact

Recent studies have shown that automated data-driven methods can have unintended biases and discriminatory effects. Our proposed algorithms aim to prevent these issues in applications which can be modeled as a submodular maximization over a matroid constraint problem. Such applications arise in a variety of contexts, including selection of political representatives, committees, candidates for outreach programs, and content selection for search engines and news feeds. As in prior work, we observe that enforcing fairness may come at a slight cost in utility value (see Section 5). However, this trade-off between fairness and utility should not be viewed as a less desirable outcome, but rather as a balance between the two metrics. Our algorithms provide solutions that achieve a good such balance. Lastly, while the fairness notion we consider is broad, it does not encompass all fairness metrics in the literature. As we noted earlier, defining the right notion of algorithmic fairness is an active line of research. There is no universal definition of fairness, and the choice of which metric to use will depend on the application.

Acknowledgements

We thank Michael Kapralov for helpful discussions. The work of Federico Fusco is partially supported by ERC Advanced Grant 788893 AMDROMA “Algorithmic and Mechanism Design Research in Online Markets”, PNRR MUR project PE0000013-FAIR”, and PNRR MUR project IR0000013-SoBigData.it.

References


A. Heuristics

In this section, we propose two heuristics which can improve the performance of our two-pass algorithm from Section 4, and a one-pass heuristic algorithm for FMMSM.

A.1. Alternative algorithm for finding a feasible set

We propose an alternative algorithm, Greedy-Fair-Reservoir, to Fair-Reservoir (Algorithm 1) for finding a feasible solution \( S \in \mathcal{F} \) in a single pass. Greedy-Fair-Reservoir is similar to Fair-Reservoir, but instead of collecting a maximal independent set \( I_c \) of arbitrary elements for each color \( c \), it picks elements greedily.

Algorithm 3 Greedy-Fair-Reservoir

1: \( I_c \leftarrow \emptyset \) for all \( c = 1, ..., C \)
2: for the next element \( e \) on the stream do
3: \( \quad \text{Let} \ c \ \text{be the color of} \ e \)
4: \( \quad \text{if} \ I_c + e \in \mathcal{I} \ \text{then} \)
5: \( \quad \ I_c \leftarrow I_c + e \)
6: \( \quad \text{else} \)
7: \( \quad \text{for} \ e' \in I_c \ \text{in order of increasing} \ f(e') \ \text{do} \)
8: \( \quad \quad \text{if} \ I_c + e - e' \in \mathcal{I} \ \text{and} \ f(I_c + e - e') \geq f(I_c) \ \text{then} \)
9: \( \quad \quad I_c \leftarrow I_c + e - e' \)
10: \( \quad \ \text{break} \)
11: \( \) Consider the partition matroid \( \mathcal{I}_C \) on \( V \) defined in (1)
12: \( \text{Let} \ S \subseteq \cup_{c=1}^{C} I_c \ \text{be any max-cardinality set in} \ \mathcal{I} \cap \mathcal{I}_C \) (Lemma 2.2)
13: \( \text{Return set} \ S \)

Note that Theorem 4.2 still holds; Greedy-Fair-Reservoir is guaranteed to return a feasible solution in polynomial time and using \( O(k \cdot C) \) memory. Using Greedy-Fair-Reservoir instead of Fair-Reservoir in Fair-Streaming yielded better performance in our empirical evaluation (Section 5). Furthermore, in Appendix B we show that Greedy-Fair-Reservoir can be adapted to return an optimal solution to the problem of fair modular maximization over a matroid constraint.

A.2. Alternative filling-up procedure

Algorithm 4 Fair-Streaming+

1: \( \text{Input:} \) Set \( \mathcal{S} \) from Fair-Reservoir and routine \( \mathcal{A} \)
2: \( \mathcal{S}_1 \leftarrow \emptyset, \mathcal{S}_2 \leftarrow \emptyset \)
3: for element \( e \) in \( \mathcal{S} \) do
4: \( \quad \text{Let} \ c \ \text{be the color of} \ e \)
5: \( \quad \text{if} \ |S_1 \cap V_c| < |S_2 \cap V_c| \ \text{then} \)
6: \( \quad S_1 \leftarrow S_1 + e \)
7: \( \quad \text{else} \)
8: \( \quad S_2 \leftarrow S_2 + e \)
9: Define matroids \( \mathcal{I}_C, \mathcal{I}_1, \mathcal{I}_2 \) as in Equations (2) and (3)
10: Run two copies of \( \mathcal{A} \), one for matroids \( \mathcal{I}_C, \mathcal{I}_1 \) and one for matroids \( \mathcal{I}_C, \mathcal{I}_2 \), and let \( \mathcal{S}_1' \) and \( \mathcal{S}_2' \) be the two outputs
11: Run two copies of \( \mathcal{A} \), one for matroids \( \mathcal{I}_1, \mathcal{I}_0 \) and objective \( f_1 \), and one for matroids \( \mathcal{I}_2, \mathcal{I}_0 \) and objective \( f_2 \), and let \( \mathcal{S}_1'' \) and \( \mathcal{S}_2'' \) be the two outputs
12: Return \( \mathcal{S}' = \arg \max (f(S'_1 \cup S''_1), f(S'_2 \cup S''_2)) \)

We propose an alternative way to augment sets \( S'_1, S'_2 \) at the end of Fair-Streaming (Algorithm 2, lines 11-14). Instead of adding arbitrary elements from \( S_1, S_2 \), we again run \( \mathcal{A} \) with objective \( f_i(S) = f(S \cup S'_i) \) (which is still monotone submodular), matroid \( \mathcal{I}_C = \{X \subseteq S_i \mid |X \cup S'_i| \in \mathcal{I}_C\} \), and a second dummy matroid \( \mathcal{I}_0 = \{X \subseteq S_i \mid |X| \leq |S_i|\} \), for \( i = 1, 2 \). If \( \mathcal{A} \) outputs a base \( S''_i \in \mathcal{I}_C \) (which can be easily enforced), then the output set \( S' = \arg \max (f(S'_1 \cup S''_1), f(S'_2 \cup S''_2)) \)
would still satisfy Lemma 4.3 and 4.4. We refer to this algorithm as FAIR-STREAMING+ and it is presented in Algorithm 4.

A.3. One-pass heuristic algorithm

When $f$ is not modular, we can employ a greedy heuristic to augment the set returned by GREEDY-FAIR-RESERVOIR to obtain a simple one-pass heuristic, GREEDY-FAIR-STREAMING, for FMMSM.

**Algorithm 5 GREEDY-FAIR-STREAMING**

1: Run GREEDY-FAIR-RESERVOIR, let $S$ be its output and $I_c$ the set collected for each color $c = 1, \ldots, C$
2: for $e \in \bigcup_{c \in \mathbb{C}} I_c$ in order of decreasing $f(e \mid S)$ do
3:   if $S + e \in \mathcal{I}$ and $S + e \in \mathcal{I}^C$ then
4:      $S \leftarrow S + e$
5:  Return $S$

Since GREEDY-FAIR-STREAMING only adds elements to the feasible set $S$ as long as they do not violate the matroid and fairness upper bounds, Theorem 4.2 still holds; GREEDY-FAIR-STREAMING is guaranteed to output a feasible solution in one pass, using $O(k \cdot C)$ memory. It also performs quite well in terms of objective values in our empirical evaluation (Section 5), though in general it does not provide any worst-case guarantee on the objective value.

On one hand, this is due to the fact that the output $S$ of GREEDY-FAIR-RESERVOIR is an arbitrary maximum-cardinality set in $\mathcal{I} \cap \mathcal{I}^C$ (line 12 in Algorithm 3); thus it may pick only zero-value elements from $\bigcup_c I_c$. On the other hand, any algorithm that collects high-value independent sets $I_c$ in each color without considering the objective-value interactions between these sets, and then returns some subset of $\bigcup_c I_c$, is doomed to obtain $O(1/C)$-approximation. To see this, consider an instance where each color has two elements: $V_c = \{e_c, e'_c\}$, with matching lower and upper bounds $\ell_c = u_c = 1$, a matroid $\mathcal{I}$ that encodes the same constraints as the color upper bounds (i.e., $\mathcal{I} = \mathcal{I}^C$), and a monotone submodular function $f$ that assigns a value 1 to every element $e_c$ and a value 1.01 to every element $e'_c$, but in such a way that the 1.01 value is shared between all elements $e'_c$; more formally, $f(S) = |S \cap \{e_c : c = 1, \ldots, C\}| + 1.01 \cdot \min(1, |S \cap \{e'_c : c = 1, \ldots, C\}|)$. The optimal solution, of value $C$, is to pick all elements $e_c$, but GREEDY-FAIR-STREAMING will pick $I_c = \{e'_c\}$. Then there is no subset of $\bigcup_c I_c$ with value higher than 1.01, which gives a multiplicative gap of $O(1/C)$.

B. Streaming Modular Case

In this section, we present a one-pass algorithm, GREEDY-FAIR-STREAMING-M, for the fair matroid modular maximization (F3M) problem in the streaming setting. In what follows, we assume that $f$ is modular, but not necessarily monotone. GREEDY-FAIR-STREAMING-M collects maximal independent sets $I_c$ for each color $c$ in the same way as in GREEDY-FAIR-RESERVOIR, but then it returns an optimal feasible solution in $\bigcup_c I_c$.

**Algorithm 6 GREEDY-FAIR-STREAMING-M**

1: $I_c \leftarrow \emptyset$ for all $c = 1, \ldots, C$
2: for the next element $e$ on the stream do
3:   Let $c$ be the color of $e$
4:   if $I_c + e \in \mathcal{I}$ then
5:       $I_c \leftarrow I_c + e$
6:   else
7:      for $e' \in I_c$ in order of increasing $f(e')$ do
8:          if $I_c + e - e' \in \mathcal{I}$ and $f(I_c + e - e') \geq f(I_c)$ then
9:              $I_c \leftarrow I_c + e - e'$
10:             break
11:    Let $S \subseteq \bigcup_{c \in \mathbb{C}} I_c$ be any set in $\mathcal{F}$ which maximizes $f(S)$.
12: Return set $S$

We start by recalling the following notions for matroids: A circuit of a matroid $\mathcal{I}$ is a dependent set that is minimal with respect to inclusion. We say that an element $u \in V$ is spanned by a set $S \in \mathcal{I}$ if the maximum size independent subsets of $S$ and $S + u$ are of the same size. It follows from these definitions that every element $u$ of a circuit $S$ is spanned by $S - u$.  

Fairness in Streaming Submodular Maximization over a Matroid Constraint

13
Before proving the result, we need the following lemma which relates independence in matroids with reachability on graphs. Given a directed graph $G = (V, E)$, we denote with $\delta^+(u)$ the out-neighborhood of a vertex $u \in V$, i.e. $\delta^+(u) = \{v \in V \mid (u, v) \in E\}$.

**Lemma B.1** (Lemma 13 of (Feldman et al., 2018)). Consider an arbitrary directed acyclic graph $G = (V, E)$ whose vertices are elements of some matroid $\mathcal{I}$. If every non-sink vertex $u$ of $G$ is spanned by $\delta^+(u)$ in $\mathcal{I}$, then for every set $S$ of vertices of $G$ which is independent in $\mathcal{I}$ there must exist an injective function $\psi$ such that, for every vertex $u \in S$, $\psi(u)$ is a sink of $G$ which is reachable from $u$.

Using Lemma B.1, we can make the following key observation about GREEDY-FAIR-STREAMING-M.

**Lemma B.2.** For each color $c$, the independent set $I_c$ output by GREEDY-FAIR-STREAMING-M maximizes $f$ in $V_c$ over the matroid constraint $\mathcal{I}$. Moreover, if any element $x \in V_c$ is not in $I_c$, then there must exist a subset $I'_c$ of $I_c$ such that

- $I'_c + x \not\in \mathcal{I}$,
- for all $y \in I'_c$, $f(y) \geq f(x)$.

**Proof.** For each color $c$, let $\text{OPT}_c$ be any independent set in $V_c$ which maximizes $f$ in $V_c$, we show that $f(I_c) \geq f(\text{OPT}_c)$, thus proving the optimality of $I_c$. Consider the following directed graph $G_c = (V_c, E_c)$, whose nodes are the elements in $V_c$. For $t = 1, \ldots, |V_c|$, let $x^t$ be the $t$-th element of color $c$ to arrive on the stream. $I^t_c$ the set of elements in $Y^t$ which can be swapped with $x^t$, i.e., $Y^t := \{y \in I^t_c \mid I^t_c + x^t - y \not\in \mathcal{I}\}$. If $x^{t'}$ was swapped with $y^{t'} \in Y^{t'}$, then we add directed edges from $y^{t'}$ to all the elements in $Y^{t'} - y^{t'} + x^{t'}$. If $x^{t'}$ was not added, then we add directed edges from $x^{t'}$ to each element in $Y^{t'}$. If $x^{t'}$ was added without any swap, then its out-neighborhood is empty. Note that every edge in the graph $G_c$ points from a vertex dropped or swapped out at some time to a vertex that is either never deleted or removed at a later time. This time component makes the underlying graph a DAG. Note also that because of the design of the algorithm (lines 7-8), the value of $f$ is non-increasing along every edge $(u, v) \in E_c$, i.e., $f(u) \leq f(v)$.

We want to apply Lemma B.1 on the graph $G_c$. To that end, we argue that for any $t$ where $x^t$ is not a sink, and thus $I^t_c + x^t \not\in \mathcal{I}$, the set $Y^t + x^t$ is a circuit in $I^t_c + x^t$. First, we show that any proper subset of $Y^t + x^t$ is independent. For any $y \in I^t_c + x^t$, we have that $Y^t + x^t - y \subseteq I^t_c + x^t - y \in \mathcal{I}$ by the definition of $Y^t$ and since $I^t_c \subseteq \mathcal{I}$. Hence, $Y^t + x^t - y \not\in \mathcal{I}$. Next, we show that $Y^t + x^t$ is a dependent set. To see this, assume towards contradiction that this is not the case, i.e. that $Y^t + x^t \subseteq \mathcal{I}$; then we could repeatedly apply the augmentation property and add to $Y^t$ some elements $\{y_1, y_2, \ldots, y_j\} \subseteq I_c \setminus Y^t$ until $|Y^t + x^t + y_1 + \cdots + y_j| = |I_c|$, while maintaining independence. We get a contradiction: the remaining element $z \in I_c \setminus (Y^t + x^t + y_1 + \cdots + y_j)$ satisfies $Y^t + x^t + y_1 + \cdots + y_j = I_c + x - z \not\in \mathcal{I}$, while on the other hand $z \not\in Y^t$, which implies $I_c + x - z \not\in \mathcal{I}$ by definition of $Y^t$. It follows then that for ever non-sink vertex $u$ of $G_c$, its out-neighborhood $\delta^+(u) = Y^t + x^t$ is a circuit, hence $u$ is spanned by $\delta^+(u)$ in $\mathcal{I}$.

By Lemma B.1, there exists an injective function $\psi$ which associates each element $u$ in $\text{OPT}_c$ to an element $\psi(u)$ in $I_c$, which is reachable from $u$. As discussed earlier, the value of $f$ is non-increasing along every edge in the graph, and in particular along each $u$-$\psi(u)$ path. Hence $f(u) \leq f(\psi(u))$ for all $u \in \text{OPT}_c$, and $f(I_c) \geq f(\text{OPT}_c)$.

Next, we prove the remaining statement of the lemma. For any $x \in V_c \setminus I_c$, we define $I'_c := \{y \in I_c \mid I_c + x - y \not\in \mathcal{I}\}$. We show that $I'_c$ satisfies the two required properties. First, it is clear that for all $y \in I'_c$ it holds that $f(y) \geq f(x)$, because otherwise the independent set $I_c + x - y$ would have value strictly larger than $I_c$, violating the optimality of $I_c$. Second, using a similar argument as above, we can show that $I'_c + x$ is a circuit, and hence $I'_c + x \not\in \mathcal{I}$.

We are now ready to prove the optimality of GREEDY-FAIR-STREAMING-M.

**Theorem 4.5.** There exists a one-pass streaming algorithm for F3M, which finds an optimal solution, uses $O(k \cdot C)$ memory, and runs in polynomial time.

**Proof.** For every color $c$, the set $I_c$ collected by GREEDY-FAIR-STREAMING-M is a maximal independent set in $V_c$; therefore, by Lemma 4.1 there always exists a feasible set in $\bigcup_c I_c$. We need to prove that when $f$ is modular, $\bigcup_c I_c$ also contains an optimal feasible set.

The proof proceeds similarly to that of Lemma 4.1. Let $R \in \mathcal{F}$ be the optimal feasible set such that $|R \setminus \bigcup_c I_c|$ is minimal. We will prove that $|R \setminus \bigcup_c I_c|$ is actually 0. Assume towards contradiction $|R \setminus \bigcup_c I_c| > 0$. We will show how to exchange
an element \( x \in R \setminus \cup I_e \) for an element \( y \in \bigcup I_c \setminus R \). Without loss of generality, assume that \( (R \cap V_1) \setminus I_1 \neq \emptyset \), and let \( x \) be any of its elements. It is enough to show that there exists an element \( y \in I_1 \setminus R \) such that \( R - x + y \in \mathcal{I} \) and 
\[
f(R - x + y) \geq f(R).
\]

Let \( I_1 \) be the set guaranteed by Lemma B.2. Further let \( I'_1 \) be a maximal set with \( I_1 \subseteq I'_1 \subseteq I_1 \cup R \) that is independent. By maximality of \( I'_1 \), and since \( R, I'_1 \in \mathcal{I} \) we have \( |R| \leq |I'_1| \) and \( |R - x| < |I'_1| \). By the matroid augmentation property, there is \( y \in I'_1 \setminus (R - x) \) such that \( R - x + y \in \mathcal{I} \). Because
\[
I'_1 \setminus (R - x) \subseteq (I_1 \cup R) \setminus (R - x) \subseteq I'_1 + x,
\]
we must have \( y \in I'_1 \setminus R \) or \( y = x \). The latter is impossible, since this would imply that \( x \in I'_1 \); however, this is impossible because \( I'_1 + x \) is not independent by Lemma B.2. So we have found an element \( y \in I'_1 \setminus R \) such that \( R - x + y \in \mathcal{I} \) and 
\[
f(R - x + y) \geq f(R)
\]
(by Lemma B.2). This contradicts the original assumption, and concludes the proof that the output of \textsc{Greedy-Fair-Streaming-M} is optimal.

Finally, in the following section, we show that \( \textsc{F3M} \) can be solved in polynomial time in the offline setting. Hence, line 11 in \textsc{Greedy-Fair-Streaming-M} can be done in polynomial time when \( f \) is modular, and hence \textsc{Greedy-Fair-Streaming-M} runs in polynomial time in this case.

\section{C. Centralized Modular Case}

In this section, we present two polynomial-time algorithms for \( \textsc{F3M} \), in the \textit{centralized} setting. One is based on linear programming and the other reduces the problem to modular maximization over two matroid constraints. We again do not assume monotonicity here.

\subsection{C.1. Linear programming algorithm}

Given a modular function \( f \), we show that the \( \textsc{F3M} \) problem
\[
\max_{S \subseteq V} \left\{ f(S) = \sum_{e \in S} f(e) : S \in \mathcal{F} \right\}
\]
(6)

admits an exact Linear Programming relaxation which can be solved in polynomial time.

Let \( I_S \in \mathbb{R}^n \) denote the vector whose \( i \)-th entry is 1 if \( i \in S \) and 0 otherwise. We show in particular that the linear program relaxation of (7) given by
\[
\max_{x \in [0,1]^n} \left\{ \sum_{e \in V} x_e f(e) : x \in \text{conv}(\{ I_S : S \in \mathcal{F} \}) \right\}
\]
(7)
is integral, i.e., it has at least one integral optimal solution \( x^* \in \{0,1\}^n \). Hence, the relaxation is exact. For that it is enough to show that the polytope \( \text{conv}(\{ I_S : S \in \mathcal{F} \}) \) is integral, i.e. all of its extreme points are integral (Nemhauser & Wolsey, 1999, Proposition 1.3 in Part III.1, Section 1). This result generalizes the one given in Edmonds (1970, Theorems 35 and 45) for the intersection of two matroids. Our proof follow a similar structure to the proof given in Schrijver (2003, Section 41.4) of this result.

Let \( \mathcal{I}_F \) denote the family of fair sets, i.e.,
\[
\mathcal{I}_F = \{ S \subseteq V : \ell_c \leq |S \cap V_c| \leq u_c \forall c = 1, \ldots, C \}.
\]

Recall that \( \mathcal{F} = \mathcal{I} \cap \mathcal{I}_F \). Let \( P \) be the matroid polytope of \( \mathcal{I} \) defined as \( P_M = \{ x \in \mathbb{R}_+^n : x(A) \leq r(A), \forall A \subseteq V \} \), where \( x(A) = \sum_{e \in A} x_e \), and \( r \) is the rank function of \( \mathcal{I} \). The matroid polytope \( P \) corresponds to the convex-hull of indicator vectors of independent sets, i.e., \( P = \text{conv}(\{ I_S : S \in \mathcal{I} \}) \).

The following lemma provides the convex-hull of indicator vectors of fair sets.

\textbf{Lemma C.1.} Let \( P_F = \{ x \in [0,1]^n : x(V_c) \in [\ell_c, u_c], \forall c = 1, \ldots, C \} \), then \( P_F = \text{conv}(\{ I_S : S \in \mathcal{I}_F \}) \).

15
Proof. Since \( \{1_S : S \in \mathcal{I}_F \} \subseteq P_F \), then \( \text{conv}(\{1_S : S \in \mathcal{I}_F \}) \subseteq P_F \). To prove the other direction, we show that for any \( \theta \in \mathbb{R}^n \), the linear program \( \max_{x \in P_F} \theta^\top x \) is integral, hence \( P_F \) is integral (Nemhauser & Wolsey, 1999, Proposition 1.3 in Part III.1, Section 1).

Let \( V_+ \) be the set of indices \( i \in V \) where \( \theta_i > 0 \). For each color \( c \), let \( J_c \) be the set of indices corresponding to the largest \( \ell_c \) coefficients \( \theta_i \) for \( i \in V_c \), and \( J^+_c \) be the set of indices corresponding to the largest \( \min\{u_c - \ell_c, |V_c \setminus J_c| \cap V_+| \} \) coefficients \( \theta_i \) for \( i \in V_c \setminus J_c \cap V_+ \). Then it is easy to see that the integral vector \( x^* = \bigcup_c \mathbb{1}_{J_c} \cup \mathbb{1}_{J^+_c} \) is an optimal solution of \( \max_{x \in P_F} \theta^\top x \). Hence, \( P_F \subseteq \text{conv}(\{1_S : S \in \mathcal{I}_F \}) \), which concludes the proof. \( \square \)

Next we show that the linear system corresponding to \( P_F \cap P \) is totally dual integral (TDI), and hence \( P_F \cap P \) is integral. We recall first the definitions of TDI and box-TDI.

**Definition C.2** (Sections 5.17 and 5.20 in (Schrijver, 2003)). A system \( Mx \leq b \) is called totally dual integral (TDI) if \( M \) and \( b \) are rational, and the dual of \( \max\{c^\top x : Mx \leq b\} \) has an integer optimal solution (if finite), for each \( c \in \mathbb{Z}^n_+ \). Furthermore, a system \( Mx \leq b \) is called box-totally dual integral (box-TDI) if the system \( Mx \leq b, d_1 \leq x \leq d_2 \) is TDI for each \( d_1, d_2 \in \mathbb{Z}^n_+ \).

**Theorem C.3.** The linear system \( \{0 \leq x \leq 1, x(A) \leq r(A), \forall A \subseteq V, -x(V_c) \leq -\ell_c, x(V_c) \leq u_c, \forall c = 1, ..., C \} \) is TDI. Hence, \( P_F \cap P \) is integral.

**Proof.** We first show that the linear system \( \{x(A) \leq r(A), \forall A \subseteq V, -x(V_c) \leq -\ell_c, x(V_c) \leq u_c, \forall c = 1, ..., C \} \) is box-TDI. We can write the linear system as \( Mx \leq b \). Given any \( \theta \in \mathbb{R}^n \), the dual of \( \max_x \{\theta^\top x : Mx \leq b\} \), is given by:

\[
\min_{\lambda \geq 0, \alpha \geq 0, \beta \geq 0} \left\{ \sum_{A \subseteq V} \lambda_A r(A) + \sum_{c=1}^C (\alpha_c u_c - \beta_c \ell_c) : \sum_{A \subseteq V} \lambda_A \mathbb{1}_A + \sum_{c=1}^C (\alpha_c - \beta_c) \mathbb{1}_{V_c} = \theta \right\}
\]

We argue that the dual has an optimal solution \( \lambda^*, \alpha^*, \beta^* \) for which the collection of sets \( C = \{A \subseteq V : \lambda_A > 0\} \) form a chain, i.e., if \( A, B \in C \) then \( A \subseteq B \) or \( A \supseteq B \). Given any optimal dual solution, let \( \delta = \min\{\lambda_A^*, \lambda_B^*\} \), then decrease \( \lambda_A^*, \lambda_B^* \) by \( \delta \), and increase \( \lambda_{A \cup B}^*, \lambda_{A \cap B}^* \) by \( \delta \). The modified solution is still feasible since \( \mathbb{1}_A + \mathbb{1}_B = \mathbb{1}_{A \cup B} + \mathbb{1}_{A \cap B} \), and it has an equal or lower cost since \( r(A) + r(B) \geq r(A \cup B) + r(A \cap B) \). Applying this uncrossing operation for all pairs of sets in \( C \), results in a chain. The submatrix of \( M \) with rows corresponding to the constraints \( x(A) \leq r(A), \forall A \subseteq C \), and \( x(V_c) \leq u_c, \forall c = 1, ..., C \) is the incidence matrix of the union of two laminar families, hence it is totally unimodular (TU) (Schrijver, 2003, Theorem 41.11). Adding the rows corresponding to the constraints \( -x(V_c) \leq -\ell_c, \forall c = 1, ..., C \) preserves the TU property (Nemhauser & Wolsey, 1999, Proposition 2.1 in Part III.1, Section 2). It follows then by Schrijver (2003, Theorem 5.35) that the linear system \( Mx \leq b \) is box-TDI.

By definition of box-TDI, we then have that the linear system corresponding to \( P_F \cap P \) is TDI, which implies that \( P_F \cap P \) is integral by (Schrijver, 2003, Theorem 5.22).

**Corollary C.4.** We have \( \text{conv}(\{1_S : S \in F\}) = P_F \cap P \) and hence it is integral.

**Proof.** We note that any integral vector in \( P_F \cap P \) must also belong to \( \{1_S : S \in F\} \). Since \( P_F \cap P \) is integral (Theorem C.3), all its vertices are integral. Hence \( P_F \cap P \subseteq \text{conv}(\{1_S : S \in F\}) \), and since \( P_F = \text{conv}(\{1_S : S \in I_F\}) \) and \( P = \text{conv}(\{1_S : S \in I\}) \), we also have \( \text{conv}(\{1_S : S \in F\}) \subseteq P_F \cap P \). \( \square \)

**Theorem C.5.** There is an exact polynomial-time algorithm for F3M.

**Proof.** Since \( \text{conv}(\{1_S : S \in F\}) \) is integral, We can solve problem (6) by solving its exact LP relaxation (7). The latter can be solved in polynomial time using the ellipsoid method, since \( \text{conv}(I_M \cap I_F) \) admits a polynomial time separation oracle (which simply queries the separation oracles of \( P_F \) and \( P \)). \( \square \)

### C.2. Reduction to submodular (modular) maximization over matroid intersection bases

In this section we show that fair matroid submodular maximization (FMSM) reduces to a version of submodular maximization over an intersection of two matroids (with an extra “full-rank” constraint). This will imply another polynomial-time exact algorithm for F3M.

Let us define the **submodular maximization over matroid intersection bases (SMOMIB)** problem as follows:
We will define two matroids $I_1$ and $I_2$ on the same ground set $V$, with equal ranks $k_1 = k_2$, and a submodular objective function $f : 2^V \rightarrow \mathbb{R}_+$.

- output: a set $S \subseteq V$ that is independent and full-rank in both matroids: $S \in I_1 \cap I_2$, $|S| = k_1 = k_2$.

- objective: maximize $f(S)$.

**Proposition C.6.** Let $A$ be an $\alpha$-approximation to SMOMIB. Then there is an $\alpha$-approximation to FMSM.

**Proof.** Let $V = \bigcup I_c, (\ell_c, u_c) \in \mathcal{C}$, $f$ be an instance of FMSM. For every guess $x \in [\sum_c \ell_c, \sum_c u_c]$ we will try to find a good solution of size exactly $x$ using $A$.

Let us first sketch the idea: we clone every element $v \in V$ into two elements $v_{\ell}$ and $v_u$ that are copies (in the sense of the matroids and the function), so that only one of the two can be in a solution, the intuition being that $v_{\ell}$ is used to satisfy the lower bound on $v$’s color and $v_u$ is used to take elements beyond the lower bound. We can enforce the necessary constraints using a second (laminar) matroid, which will be defined so that a solution of size $x$ must have all its bounds satisfied with equality. We also truncate the first matroid to cardinality $x$, so that the ranks are equal.

Now we formalize the above. Fix $x$, and let $V' = \{v_{\ell}, v_u : v \in V\}$ be the new universe. For a set $S' \subseteq V'$ denote its projection $\pi(S')$ to $V$ as

$$\pi(S') = \{v \in V : v_{\ell} \in S' \lor v_u \in S'\}.$$  

We will define two matroids $I_1$ and $I_2$ on $V'$. Let

$$I_1 = \{S' \subseteq V' : \pi(S') \in I \land (\forall v \in V) \{v_{\ell}, v_u\} \not\subseteq S' \land |S'| \leq x\}.$$  

It is easy to see that $I_1$ is a matroid. Next, we define $I_2$ to be the following laminar matroid:

- for each color $c$, set $\{v_{\ell} : v \in V_c\}$ with bound $\ell_c$,
- for each color $c$, set $\{v_u : v \in V_c\}$ with bound $u_c - \ell_c$,
- the union of the latter sets, that is $\{v_u : v \in V\}$, with bound $x - \sum_c \ell_c$.

Having those two matroids, we verify if each has rank $x$; if not, we skip this guess of $x$. Finally, we define $f' : 2^{V'} \rightarrow \mathbb{R}_+$ in the natural way: $f'(S') := f(\pi(S'))$. One can check that $f'$ is submodular if $f$ was. Now call $A$ on instance $V', I_1, I_2, f'$.

To verify that the reduction works, we need to check:

- If $S$ is a feasible solution to FMSM, then for guess $x = |S|$, the following “lift” $S'$ of $S$ is feasible for SMOMIB: from each color $c$, pick some $\ell_c$ elements $v \in V_c \cap S$ and take $v_{\ell}$ into $S'$, while taking $v_u$ for the other $|V_c \cap S| - \ell_c$ elements. Then $\pi(S') = S$ so $S' \in I_1$, we also have $S' \in I_2$ by construction, and $|S'| = |S| = x = k_1 = k_2$. Also $f(S) = f'(S')$.

- Conversely, for any $x$ and any $S' \in I_1 \cap I_2$ with $|S'| = x$, we have that $S := \pi(S')$ has $|S| = |S'|$ and one can check that $S$ is feasible for the fair problem (in particular, we must then have $|S' \cap \{v_{\ell} : v \in V_c\}| = \ell_c$ for all $c$). Also $f(S) = f'(S')$.

Now we are ready to give another proof of Theorem C.5, which we restate for convenience.

**Theorem C.5.** There is an exact polynomial-time algorithm for F3M.

**Proof.** Note that if $f$ is a modular function, then we can instead define $f'$ in the proof of Proposition C.6 as $f'(S') = \sum_{v \in v_{\ell} \in S'} f(v) + \sum_{v \in V_c \cap S'} f(v)$, which is also modular (and equal to $f(\pi(S'))$ for sets $S' \in I_1$). Thus it is enough to give a polynomial-time algorithm for SMOMIB in the special case of modular objective, which is easy: set an objective function $f''(S') = \lambda |S'| + f'(S')$, where $\lambda$ is very large. This is still a modular function. Now we run any exact weighted matroid intersection algorithm (see Lemma 2.2): $f''$ will enforce that the optimal set has the maximum cardinality $|S'| = x$ and, subject to that, maximum $f'$-value.
D. Proof of Theorem 3.2

In this section, we prove Theorem 3.2 which is based on a reduction to the hardness result of Kapralov (2021, Theorem 1).

**Theorem 3.2.** There is no one-pass semi-streaming algorithm that, given as input the edges of a perfectly-matchable bipartite graph \( G = (P \cup Q, E) \), with probability at least 2/3 finds a matching of size at least \( \frac{2}{3} |P| \).

**Proof.** The main result (Theorem 1) of Kapralov (2021) states that no single-pass semi-streaming algorithm can find a \((1/(1+\ln 2)+\eta)\)-approximate maximum matching in a bipartite graph, for any absolute constant \( \eta > 0 \), with probability at least 1/2. This differs from the statement of the theorem in two ways: i) Theorem 3.2 requires the existence of a perfect matching in the input graph which it not the case in Kapralov (2021, Theorem 1); ii) the approximation factors are different.

The lower bound of Kapralov (2021, Theorem 1) is achieved using a hard input distribution on graphs which contain a bipartite graph \( G \). Hence, running a single-pass semi-streaming algorithm on \( G \) with probability at least \( 999/1000 \) almost surely produces a matching of size \( \Theta(|P|/L) \) from this matching, and still retain a better-than-3/4-approximate \( P \)-extreme matching in \( G \) with probability at least \( 999/1000 \). We can use the hard instance distribution of Kapralov (2021). We simply augment \( G \) to solve the hard instance distribution of Kapralov (2021). We simply augment \( G \) with a small number of additional vertices and edges:

1. \( \hat{n}/100 \) new vertices added to \( P \), called \( P^+ \);
2. \( |P| + |P^+| - |Q| \) new vertices added to \( Q \), called \( Q^+ \);
3. a complete bipartite graph between \( P \) and \( Q^+ \);
4. a complete bipartite graph between \( P^+ \) and \( Q \);
5. a complete bipartite graph between \( P^+ \) and \( Q^+ \).

We call the added edges
\[
E^+ = (P^+ \times Q) \cup (P \times Q^+) \cup (P^+ \times Q^+) .
\]
We call the augmented graph
\[
\hat{G}^+ = (P \cup P^+, Q \cup Q^+, E \cup E^+) .
\]

We show that \( \hat{G}^+ \) is guaranteed to have a perfect matching with probability at least 0.999. Let \( M_{OPT} \) be the maximum matching in \( G \). \( P_0 \) and \( Q_0 \) the corresponding unmatched vertices of \( P \) and \( Q \), respectively. Note that \( |P_0| = |P| - |M_{OPT}| \) and \( |Q_0| = |Q| - |M_{OPT}| \). We can augment \( M_{OPT} \) with edges connecting vertices of \( P_0 \) to \( Q^+ \), \( Q_0 \) to \( P^+ \), and all the remaining unmatched vertices in \( P^+ \) to the ones in \( Q^+ \). To do so we need \( |Q^+| = |P^+| - |Q_0| + |P_0| = |P^+| + |P| - |Q| \), which is satisfied. We also need \( |P^+| \geq |Q_0| = |Q| - |M_{OPT}| \), which holds with probability at least 0.999, since \( |M_{OPT}| \geq 0.999 \cdot \hat{n}/2 \) with probability at least 0.999.

Hence, running \( A \) on \( \hat{G}^+ \) is guaranteed to find a matching of size at least \( 2/3 \cdot |P \cup P^+| \) with probability at least \( 2/3 \cdot 0.999 > 1/2 \). We can simply discard edges of \( E^+ \) from this matching, and still retain a better-than-0.6-approximate

---

1Assuming any constant success probability here is equivalent, as such an algorithms can always be ran in parallel multiple times, with independent sources of randomness, to boost its success probability.
matching in $\hat{G}$, leading to a contradiction. To see this note that the number of edges in $E^+$ in the matching returned by $A$ is at most \( \max\{|P^+|, |Q^+|\} \). So the size of the matching after removing these edges is at least \( 2/3 \cdot |P \cup P^+| - |P^+| - (|P| - |Q|) \) which is larger than \( 0.6 \cdot |P| \).

\[ \square \]

E. Exponential-Memory Algorithm

In this section we present an algorithm for achieving a nearly \( 1/2 \)-approximate solution for FMMSM in the streaming setting, albeit with exponential memory in \( k \) and \( C \). Our algorithm and proof closely follow the result of (Huang et al., 2022).

**Theorem 1.1.** For any constant \( \eta \in (0, 1/2) \), there exists a one-pass streaming \( (1/2 - \eta) \)-approximation algorithm for FMMSM that uses \( 2^{O(k^2+k \log C)} \cdot \log \Delta \) memory, where \( \Delta = \frac{\max_{e \in V} f(e)}{\min_{e \in V : f(e) > 0}} \).

As is standard technique with exponential-memory streaming algorithms, we will first consider our algorithm to have access to hidden information about some optimal solution. We will then replace decisions based on hidden information with random guessing, and show that our algorithm succeeds with positive probability while consuming a bounded amount of randomness. Finally, we run our algorithm in parallel using all possible sequences of random bits, and conclude that at least one instance of the algorithm succeeds.

Let \( \text{OPT} \) be a canonical optimal feasible solution, which appears in the stream in the order \( o_1, o_2, \ldots, o_\ell \). We will first present an algorithm that assumes approximate knowledge of \( f(\text{OPT}) \); specifically we assume that our algorithm receives as input some \( \gamma \) where \( f(\text{OPT}) \in [(1 - \eta) \cdot \gamma, \gamma] \) is guaranteed. In the, we will show how to get rid of this assumption at a small cost to memory complexity in terms of the so-called aspect ratio, \( \Delta \).

Initially our algorithm will also rely on the following pieces of hidden information:

1. The cardinality \( \ell \) of \( \text{OPT} \); we call this the **cardinality oracle**.

2. The color of any opt element, \( c(o_i) \); we call this the **color oracle**.

3. The \( f \)-value of any opt element, conditioned on a set \( S \) (that we fix later on), \( f(o_i|S) \); we call this the **function oracle**. Here we need only that the oracle returns the value up to an additive error of \( f(\text{OPT}) \cdot \eta / \ell \); this will be crucial in bounding the amount of randomness guessing needed to replace the oracle.

4. The independence in \( \mathcal{I} \) of some set which may contain opt elements, as well as elements form the algorithm’s memory, \( S \cup \{o_{i_1}, \ldots, o_{i_m}\} \in \mathcal{I} \); we call this the **matroid oracle**.

With this in mind, the algorithm is presented in Algorithm 7.

Note the invariant that

\[ \forall i : \{s_1, \ldots, s_i, o_{i+1}, \ldots, o_\ell\} \in \mathcal{I} \quad (8) \]

is guaranteed by the matroid oracle call on Line 15.

**Lemma E.1.** In Algorithm 7, the **if** clause on Line 13 is satisfied (and thus the algorithm does not proceed to Line 15) only if \( \{s_1, \ldots, s_{i-1}, e, o_{i+1}, \ldots, o_\ell\} \notin \mathcal{I} \).

**Proof.** Consider the first element \( e \) for which the Lemma’s statement is violated. Recall that at this point \( S = \{s_1, \ldots, s_{i-1}\} \) and let \( O = \{o_{i+1}, \ldots, o_\ell\} \) for simplicity of notation. Suppose for contradiction that \( S \cup T + e \notin \mathcal{I} \) but \( S \cup O + e \in \mathcal{I} \). Notice also that for all elements \( t \in T \) it must be the case that \( S \cup O + t \notin \mathcal{I} \), otherwise \( t \) never would have been added to \( T \); this is because, by our assumption, the Lemma statement was true for all previous elements.

If \( |T| > |O| \) we immediately get a contradiction: Both \( S \cup T \) and \( S \cup O \) are independent, and \( |S \cup T| > |S \cup O| \) so my the augmentation property of matroids there exists a set \( S \cup O + t \in \mathcal{I} \) for \( t \in T \).

If, on the other hand, \( |T| \leq |O| \), \( |S \cup T| < |S \cup O + e| \) (also independent), so by the augmentation property of matroids, there exists a set \( S \cup T \cup O' \in \mathcal{I} \) where \( O' \subseteq O + e \). However, \( e \) cannot be in \( O' \), since \( S \cup T + e \notin \mathcal{I} \) by assumption, so
Algorithm 7 Exponential algorithm

1: **Input:** Cardinality, color, function, and matroid oracles, and $\gamma$.
2: $\ell \leftarrow |\text{OPT}|$ \hspace{1em} // Query the cardinality oracle.
3: $S \leftarrow \emptyset$
4: for $i \leftarrow 1 \ldots \ell$ do
5: \hspace{1em} $c \leftarrow$ color of $o_i$ \hspace{1em} // Query color oracle.
6: \hspace{1em} Set $\theta \in \mathbb{Z}$ such that $f(o_i|S) \in [\theta \eta \gamma/\ell, (\theta + 1) \eta \gamma/\ell)$ \hspace{1em} // Query function oracle.
7: $T \leftarrow \emptyset$
8: for $e$ element in the stream do
9: \hspace{1em} if $e$ is not color $c$ then
10: \hspace{2em} continue
11: \hspace{1em} if $f(e|S) \notin [\theta \eta \gamma/\ell, (\theta + 1) \eta \gamma/\ell)$ then
12: \hspace{2em} continue
13: \hspace{1em} if $S \cup T + e \notin \mathcal{I}$ then
14: \hspace{2em} continue
15: \hspace{1em} if $S \cup \{o_{i+1}, \ldots, o_{\ell}\} + e \notin \mathcal{I}$ then \hspace{1em} // Query matroid oracle.
16: \hspace{1em} $T \leftarrow T + e$
17: \hspace{1em} else
18: \hspace{2em} $s_i \leftarrow e$
19: \hspace{1em} $S \leftarrow S + s_i$
20: Return $S$

$O' \subseteq O$. Furthermore, $|S \cup T + e|$ and $|S \cup T \cup O'| = |S \cup O + e| > |S \cup O|$. Therefore, by applying the augmentation property again, we can get an independent set of the form $S \cup O + t$ where $t \in T$; this is a contradiction.

Lemma E.2. Algorithm 7 will always find an appropriate element $s_i$, and break out of the loop on Line 4.

Proof. We can prove the following stronger claim through induction over $i$: The algorithm will break out of the loop on Line 4 no later than $o_i$’s arrival in the stream.

For any $i$ (both base case and inductive step), we know that $o_i$ is still in the stream when the loop at Line 4 begins. Then the inductive statement follows simply due to the fact that $o_i$ itself will pass all the filters on Lines 9, 11, 13 (due to Lemma E.1), and 15: It is the right color, the right size, and $e = o_i$ satisfies the condition $\{s_1, \ldots, s_{i-1}, e, o_{i+1}, \ldots, o_{\ell}\}$.

From this it follows that Algorithm 7 will indeed always output a feasible solution of $\ell$ elements, due to Equation (8) when $i = \ell$.

We now turn to showing a lower bound on the quality in terms of $f(\text{OPT})$ of the solution output by Algorithm 7.

Lemma E.3. The solution output by Algorithm 7 has value at least $(1/2 - \eta) \cdot f(\text{OPT})$.

Proof. We prove the following statement by induction over $i$:

$$2 \cdot f(\{s_1, \ldots, s_i\}) + f(\{o_{i+1}, \ldots, o_{\ell}\}|\{s_1, \ldots, s_i\}) + \frac{i \eta \gamma}{\ell} \geq f(\text{OPT}). \hspace{2em} (9)$$

The base case of $i = 0$ holds trivially, and by substituting in $i = \ell$, we get the statement of the Lemma.

For simplicity denote $S = \{s_1, \ldots, s_{i-1}\}$ and $O = \{o_{i+1}, \ldots, o_{\ell}\}$. To prove the inductive step, it suffices to show that the
change in the left hand size of Equation (9) is positive when moving from \(i - 1\) to \(i\):

\[
\text{LHS}_i - \text{LHS}_{i-1} = 2 \cdot f(S + s_i) - 2f(S) + f(O|S + s_i) - f(O|s_i) + \frac{\eta \gamma}{\ell} \\
\geq 2 \cdot f(s_i|S) - f(o_i|S) + \frac{\eta \gamma}{\ell} \\
\geq f(s_i|S) + \frac{\eta}{\ell} - f(o_i|S) \\
\geq 0,
\]

since we know that \(f(o_i|S)\) and \(f(e|S)\) are both in \([\theta \eta \gamma / \ell, (\theta + 1) \eta \gamma / \ell]\) from Lines 6 and 11.

Finally, taking Equation (9) with \(i = \ell\) gives us

\[
2 \cdot f(\{s_1, \ldots, s_\ell\}) + \eta \gamma \geq f(\text{OPT}),
\]

and therefore

\[
f(\{s_1, \ldots, s_\ell\}) \geq f(\text{OPT}) \cdot (1/2 - \eta \cdot (1 + \eta)/2) \geq f(\text{OPT}) \cdot (1/2 - \eta).
\]

This concludes the proof of correctness of Algorithm 7 in the presence of the four oracles (cardinality oracle, color oracle, function oracle, and matroid oracle). We are now ready to prove Theorem 1.1.

**Lemma E.4.** There exists a randomized single-pass streaming algorithm using \(O(k)\) memory, and outputting a \(1/2 - \eta\)-approximately optimal feasible solution with positive probability, while consuming \(O(k^2 + k \log C)\) bits of randomness.

**Proof.** Algorithm 7 is such an algorithm when replacing the three oracles with uniformly random choices. Indeed, it produces the correct output with positive probability (when all random choices happen to be correct).

The cardinality oracle is called only once and chooses between \(k\) options, so a random implementation consumes \(\log k\) random bits. The color oracle is called at most \(k\) times and chooses between \(C\) options, so a random implementation consumes \(O(k \log C)\) random bits. The function oracle is called at most \(k\) times and chooses between \(O(k / \eta)\) options. This is because \(\theta\) is at least \(0\) and at most \(\ell / \eta \leq k / \eta\), since \(\gamma \geq f(\text{OPT}) \geq f(o_i) \geq f(o_i|S)\). Therefore, a random implementation consumes \(O(k \log(k / \eta))\) random bits. The matroid oracle is called at most \(k^2 + k\) times; at most \(k + 1\) times every iteration of the for loop in Line 4. This is because every time it is called and returns true (that is \(\{s_1, \ldots, s_{i-1}, e, o_{i+1}, \ldots, o_{\ell}\} \in F\)), the current iteration of the loop is terminated; every time it is called and returns false, \(T\) is incremented, and since \(T \in \mathcal{I}\), it can have size at most \(k\). Therefore, a random implementation of this consumes \(k^2 + k\) random bits.

In total this is \(O(\log k + k^2 + k \log(k / \eta) + k \log C) = O(k^2 + k \log C)\) random bits.

**Proof of Theorem 1.1.** We simply run \(2^{O(k^2 + k \log C)}\) parallel copies of the algorithm guaranteed by Lemma E.4, each with a different stream of bits as randomness. At least one is guaranteed to succeed. We can then find and return the highest valued feasible set output by the algorithms.

However, all versions of Algorithm 7 assume access to \(\gamma\) with the guarantee that \(f(\text{OPT}) \in [(1 - \eta) \cdot \gamma, \gamma]\). For the purposes of this proof, we call the above algorithm the \(\gamma\)-dependent algorithm; it satisfies the requirements postulated by Theorem 1.1, but only under the condition that \(\gamma\) is set correctly. We will now show how to drop this requirement while losing a \(O(\log \Delta)\) factor in the memory complexity. (What follows is standard technique often used in the literature in the context of streaming submodular maximization.) We again run multiple copies of the \(\gamma\)-dependent algorithm, with different guesses of \(\gamma\). In fact, we run a copy for \(\gamma = (1 - \eta)^t\) for every \(t \in \mathbb{Z}\), thus guaranteeing that in at least one of the cases \(f(\text{OPT}) \in [(1 - \eta) \cdot \gamma, \gamma]\) is satisfied.

Although this is potentially an infinite number of parallel copies, we show that all but \(O(\log(k \Delta))\) of them fall into one of two classes, such that copies within the same class look identical to each other; thus we require only \(2^{O(k^2)} \cdot \log \Delta\) memory in the end. Recall that \(\Delta\) is the ratio between the value of the larges and smallest (non-zero) elements on the
stream, that is $\Delta = \max f(e)/\min_{f(e)\neq 0} f(e)$. Let the largest and smallest elements be $e_{\text{max}}$ and $e_{\text{min}}$ respectively, such that $\Delta = f(e_{\text{max}})/f(e_{\text{min}})$.

For values of $\gamma$ that are less than $f(e_{\text{min}})/2$, an element $e$ can only pass the filter at Line 11 if $f(e) = 0$. This can be proven by induction. As long as $S$ contains only elements with $f$-value 0, $f(S) = 0$, and $\forall e : f(e|S) = f(e)$. For any $e$ such that $f(e) > 0$ it follows that $f(e|S) \geq f(e_{\text{min}}) > (\theta + 1)\eta\gamma/\ell$ so $e$ does not pass the filter at Line 11. As a result all copies of the $\gamma$-dependent algorithm with $\gamma \leq f(e_{\text{min}})/2$ look identical to each other and can be stored as one.

For values of $\gamma$ that are greater than $f(e_{\text{max}}) \cdot k/\eta$ it is also true that all copies of the $\gamma$-dependent algorithm look identical. In this case, if $\theta$ is anything other than 0 on Line 6, all elements will be filtered out on Line 11, since $\forall e : f(e) < \gamma\eta/\ell$. On the other hand, if $\theta$ is 0 on Line 6, all elements $e$ pass the filter on Line 11 for the same reason. Therefore, when $\gamma > f(e_{\text{max}}) \cdot k/\eta$, the exact value of $\gamma$ is irrelevant to the execution of the $\gamma$-dependent algorithm, and all such copies can be stored as one.

In summary, copies of the $\gamma$-dependent algorithm for $\gamma$’s over $f(e_{\text{max}}) \cdot k/\eta$ as well as $\gamma$’s under $f(e_{\text{min}})/2$ are stored as though they constituted only two total copies of the $\gamma$-dependent algorithm. (This can be done without foreknowledge of $f(e_{\text{max}})$, $f(e_{\text{min}})$ or even $\Delta$.) All other copies of the algorithm are stored explicitly — a total of $O(\log(k\Delta))$ copies. Once again, the correct solution among all possibilities can be selected by simply picking the largest $f$-valued feasible set. The total memory complexity is $2O(k^2) \cdot \log \Delta$.

\[ \square \]

**F. Proof of Theorem 1.2**

Towards a contradiction assume that $A$ is an algorithm as in the statement of Theorem 1.2. We then describe how to construct an algorithm $B$ to find a perfect matching. Consider any instance of the perfect bipartite matching streaming problem, and let $\langle (l_1, r_1), (l_2, r_2), \ldots, (l_{|E|}, r_{|E|}) \rangle$ denote the stream of edges of a bipartite graph $G(L \cup R, E)$. Define the following matroid constraint on $E$: a subset of edges is independent if it has at most one edge incident to any left vertex $l \in L$. Note that this is a partition matroid, and its rank $k = n$, as we can assume that each left vertex has at least one edge (otherwise there is no perfect matching). Moreover, we use the fairness constraint to ensure that exactly one edge incident to each vertex $r \in R$ is selected; we have $C = n$. With these two constraints on the set of edges, we have that any solution $S \subseteq E$ is feasible if and only if $S$ is a perfect matching. The submodular function $f$ does not play a role in this reduction and can be defined arbitrarily. $B$ can simulate the behavior of algorithm $A$ on the edge set with the constraints defined above and returns that there exists a perfect matching if and only $A$ return that there exists a feasible solution. This contradicts Theorem 3.1 and concludes the proof.