Fairness k-submodular maximization subject to matroid constraint

Abstract

Fairness k-submodular maximization has attracted increasing interest due to its broad relevance in artificial intelligence and machine learning. However, most existing work is limited to monotone objectives or simple size constraints, while the non-monotone setting with richer constraints remains largely unexplored. In this paper, we first introduce a constant-ratio approximation algorithm for the problem under a general non-monotone objective function and a matroid constraint. Our approach is built upon a two-stage algorithmic framework. Specifically, we first develop an algorithm that guarantees feasibility with respect to upper fairness bounds only. We then show how this algorithm can be systematically extended to simultaneously enforce fairness bounds, while preserving provable approximation guarantees. Comprehensive experiments on standard benchmark datasets demonstrate that our algorithm achieves high-quality objective values while maintaining a favorable balance between fairness guarantees and query efficiency consistently outperforming existing state-of-the-art methods.

1 Introduction

Given a finite ground set V of size n, an integer k, and a non-negative (possibly non-monotone) k-submodular function $f : (k+1)^V \to \mathbb{R}_+$, where $(k+1)^V = \{(X_1, \ldots, X_k) \mid X_i \subseteq V \ \forall i \in [k], X_i \cap X_j = \emptyset \ \forall i \neq j\}$. A k-set $\mathbf{x} = (X_1, \ldots, X_k) \in (k+1)^V$ is said to be *fair* if each component X_i satisfies the fairness bounds $l_i \leq |X_i| \leq u_i \quad \forall i \in [k]$, for given lower and upper fairness bounds l_i and u_i such that $l_i \leq u_i$. In this work, we study the problem of **fairness k-submodular maximization under a matroid constraint**, (denoted as FkSM): find a fair k-set $\mathbf{x}$ such that $\bigcup_i X_i \in \mathcal{M}$ that maximizes $f(\mathbf{x})$.

This problem forms a fundamental optimization framework with broad applicability in artificial intelligence and machine learning, including revenue optimization and influence maximization [Ha *et al.*, 2024; Ohsaka and Yoshida, 2015; Qian *et al.*, 2017; Spaeh *et al.*, 2025], sensor placement [Ha *et al.*, 2024; Nguyen and Thai, 2020], recommendation systems and image and document summarization [Spaeh *et al.*, 2025; Feldman *et al.*, 2018].

In the above applications, the fairness factor across components is essential for obtaining balanced and practically meaningful solutions. In recommendation systems, fairness constraints prevent the over-selection of items from a small number of genres, sellers, or product categories, thereby promoting diversity and improving user experience [Spaeh *et al.*, 2025; Mirzasoleiman *et al.*, 2016]. Similarly, in revenue and influence maximization, fairness ensures equitable representation across topics, which may correspond to demographic attributes such as gender, ethnicity, or region, and helps mitigate biased influence propagation [Ha *et al.*, 2024; Ohsaka and Yoshida, 2015; Nguyen and Thai, 2020]. In sensor placement, fairness constraints balance the deployment of different sensor types, preventing excessive allocation to a single type and improving coverage quality, cost efficiency, and robustness of the sensing system [Nguyen and Thai, 2020; Ha *et al.*, 2024].

Beyond fairness, the objective function in many real-world applications is inherently non-monotone. Such non-monotonicity naturally arises in recommendation maximization, revenue maximization, and image summarization, where adding an element may reduce the overall utility due to redundancy or competition [Spaeh *et al.*, 2025; Mirzasoleiman *et al.*, 2016; Kuhnle, 2019; Pham *et al.*, 2023]. While fairness k-submodular maximization has been extensively studied under simple cardinality constraints with monotone objectives [Zhu *et al.*, 2024], the non-monotone setting remains largely unexplored.

Moreover, many practical constraints cannot be captured by cardinality limits and require more general feasibility models such as matroid constraints. Despite their broad applicability, no constant-ratio approximation algorithm is currently known for FkSM with general non-monotone objectives; consequently, the existence of provably guaranteed approximation algorithms for this problem remains a fundamental open question.

1.1 Our Contributions and Techniques

In this work, we address the above open question by developing constant-factor approximation algorithms for the FkSM problem. Our main contributions are summarized as follows.

- We first study a special case of FkSM in which the lower

Table 1: Our algorithms for the FkSM problem. In the **Fairness Approximation** column, the entry θ indicates that the algorithm's output $\mathbf{x} = (X_1, \ldots, X_k)$ satisfies $|X_i| \geq \lfloor \theta\, l_i \rfloor$.

PROBLEM	FUNCTION	ALGORITHM	APPROXIMATION RATIO	FAIRNESS APPROXIMATION
UFkSM	MONOTONE	UFair$k$Sub	$\frac{1}{4.582} - \epsilon$	–
	NON-MONOTONE	UFair$k$Sub	$\frac{1}{6.494} - \epsilon$	–
FkSM	MONOTONE	Fair$k$Sub	$\frac{1}{9.164} - \frac{\epsilon}{2}$	$\frac{1}{2}$
	NON-MONOTONE	Fair$k$Sub	$\frac{1}{25.976} - \frac{\epsilon}{4}$	$\frac{1}{2}$

fairness bounds satisfy $l_i = 0$ for all $i \in [k]$, referred to as the *Upper* FkSM (UFkSM) problem. For this setting, we propose the UFair$k$Sub algorithm, which achieves an approximation ratio of $\frac{1}{4.582} - \epsilon$ for monotone objective functions and $\frac{1}{6.494} - \epsilon$ for non-monotone objectives.

- For the general FkSM problem with full fairness bounds, we design the Fair$k$Sub algorithm that attains an approximation ratio of $\frac{1}{9.164} - \frac{\epsilon}{2}$ for monotone objectives and $\frac{1}{25.976} - \frac{\epsilon}{4}$ for non-monotone objectives, while allowing a factor-$\frac{1}{2}$ violation of the lower fairness bounds. An overview of our results is provided in Table 1.

Our techniques. It is worth noting that a recent approach to the fairness k-submodular maximization problem with monotone objectives relies on a greedy strategy over the space of *extendable* k-sets under simple size constraints [Zhu *et al.*, 2024]. However, this approach critically depends on monotonicity and simple size constraints. To overcome this limitation, we propose a two-phase algorithmic framework together with a new theoretical analysis. Our framework, embodied in the Fair$k$Sub algorithm, is purely combinatorial and is designed to efficiently handle both monotone and non-monotone objective functions under a matroid constraint.

In the **first phase** of our algorithm, we develop the UFair$k$Sub algorithm as a core subroutine for generating candidate solutions to FkSM that satisfy only the upper fairness constraints, i.e., $l_i = 0$ for all $i \in [k]$. The algorithm employs a greedy threshold strategy to iteratively select elements and construct a solution $\mathbf{s} = (S_1, S_2, \ldots, S_k)$, proceeding until exactly one component remains underfilled, namely when $|S_i| < u_i$ for some $i \in [k]$. We further show that this construction can be combined with classical techniques for submodular maximization under matroid constraints to establish provable approximation guarantees. A key ingredient of this phase is a *transformation* that projects the optimal solution onto the algorithm's current solution at each iteration, allowing us to bound the loss relative to the optimum while preserving matroid feasibility. In the **second phase**, we refine the solution obtained from the first phase to achieve feasibility with respect to both upper and lower fairness bounds, as well as the underlying constraints. Specifically, we show how to combine the UFkSM to procedure systematically produce a final solution with provable guarantees. Our analysis exploits a structural decomposition of the optimal solution and key properties of matroids, together with a refinement mechanism that improves solution quality while controlling violations of the lower fairness bounds.

2 Related Work

This section discusses related work on the FkSM problem.

k-submodular maximization. The notion of k-submodularity was originally proposed by [Singh *et al.*, 2012] and subsequently developed in a series of works that primarily concentrated on unconstrained k-submodular maximization [Ward and Zivnỳ, 2014; Ward and Zivnỳ, 2016; Iwata *et al.*, 2016; Oshima, 2017; Soma, 2019]. [Ohsaka and Yoshida, 2015] first considered k-submodular maximization under a total size constraint B (denoted as kSMTS), requiring $|\cup_{i=1}^{k} S_i| \leq B$, as well as under component size constraints $B_1, B_2, \ldots, B_k$ (denoted as kSMIS), where $|S_i| \leq B_i$ for all $i \in [k]$. The authors demonstrated that greedy-based algorithms achieve approximation ratios of $\frac{1}{2}$ for the total size constraint and $\frac{1}{3}$ for the component size constraints, respectively.

The kSMIS problem has further been studied in the non-monotone setting. Specifically, [Spaeh *et al.*, 2025] derived an approximation ratio of $\frac{0.3178}{2}(1 - \frac{1}{\min_i B_i}) \geq \frac{1}{8}$, which asymptotically improves to 0.1589 as $\min_i B_i \to \infty$. Beyond cardinality constraints, k-submodular maximization has been explored under general constraints. Under matroid constraints, [Sakaue, 2017] showed that a greedy algorithm attains a $1/2$-ratio. The knapsack constraint was initiated by [Tang *et al.*, 2022], who proposed a $\frac{e-1}{2e}$-approximation algorithm for monotone objectives. Subsequent work improved this guarantee to $\frac{1}{2}$ for monotone objectives and $\frac{1}{3}$ for non-monotone objectives [Ha *et al.*, 2024; Zhou *et al.*, 2025]. Among the most relevant works, [Zhu *et al.*, 2024] studied fairness k-submodular maximization under total size constraint with monotone objective functions and obtained a $\frac{1}{3}$-approximation guarantee. Nevertheless, their method critically depends on monotonicity and cannot be extended to non-monotone objectives.

Submodular maxization under fairness constraint (SMF). This problem is the closest variant to FkSM, defined as follows: Given a submodular function $f : 2^V \to \mathbb{R}_+$ and a ground set V partitioned into k disjoint subsets $V_1, V_2, \ldots, V_k$, the objective is to find a set $S \subseteq V$ that maximizes $f(S)$ subject to the fairness constraints $l_i \leq |S \cap V_i| \leq u_i$, where l_i and u_i denote the lower and upper fairness bounds of V_i. The study of the SMF was initiated in [Celis *et al.*, 2018], which focused on monotone objectives and established a tight $1 - 1/e$ approximation guarantee via the continuous greedy paradigm. Later, [Halabi *et al.*, 2020] examined a streaming formulation and derived approximation algorithms for both monotone and non-monotone objectives, achieving a $\frac{1}{2}$ ratio in the monotone case and a $\frac{1}{5.82}(1 - \max_i \frac{l_i}{|V_i|})$ ratio when the objective is

non-monotone. These guarantees were improved to $\gamma/3$, with γ denoting the best-known approximation ratio for submodular maximization under matroid constraints [Yuan and Tang, 2023]. Subsequent work has extended SMF to richer constraints, including matroid constraints [El Halabi *et al.*, 2023; Halabi *et al.*, 2024] and knapsack constraints [Cui *et al.*, 2024]. A common characteristic of these approaches, however, is that they relax the lower-bound fairness requirements, typically guaranteeing only $|S \cap V_i| \geq \lfloor l_i/2 \rfloor$ for each group $i \in [k]$. Existing algorithmic techniques for SMF do not readily extend to kSMF for two fundamental reasons. First, k-submodularity exhibits structural properties that differ substantially from classical submodularity. Second, fairness constraints in kSMF are intrinsically more involved, as each element must be assigned to exactly one of the k components, rather than being selected from pre-defined partitions of the ground set. As a result, the existence of constant-factor approximation algorithms for the FkSM problem remains an open question.

3 Preliminaries

k-submodular notations. Given a finite set V of size n, and an integer number k, let $[k] = \{1, 2, \ldots, k\}$ and $(k+1)^V = \{(X_1, X_2, ...X_k) \mid X_i \subseteq V \; \forall i \in [k], X_i \cap X_j = \emptyset \; \forall i \neq j\}$ be a family of k disjoint subsets of V. For simplicity, we call a tuple of k disjoint subsets as a k-**set**. For a k-set $\mathbf{x} = (X_1, X_2, \ldots, X_k)$, we call X_i the i-th *component* of $\mathbf{x}$. For each $e \in V$, let $\mathbf{x}(e) = i$ if $e \in X_i$ for some $i \in [k]$, and 0 otherwise. We define $supp(\mathbf{x}) = \bigcup_{i \in [k]} X_i$, denote $(\mathbf{x})_i = X_i$ for each $i \in [k]$, and let $|\mathbf{x}| = \sum_{i=1}^{k} |X_i|$. Furthermore, if $X_i = \{e\}$ and $X_j = \emptyset$ for all $j \neq i$, we denote the corresponding k-set by $\langle e, i \rangle$. For $e \notin supp(\mathbf{x})$, adding e to X_i is denoted by $\mathbf{x} + \langle e, i \rangle$. Similarly, if $e \in X_i$, then $\mathbf{x} - \langle e, i \rangle$ denotes removing e from X_i in the k-set $\mathbf{x}$.

Given two k-sets $\mathbf{x} = (X_1, \ldots, X_k)$ and $\mathbf{y} = (Y_1, \ldots, Y_k)$, we write $\mathbf{x} \sqsubseteq \mathbf{y}$ if $X_i \subseteq Y_i$ for all $i \in [k]$. The difference between $\mathbf{x}$ and $\mathbf{y}$ is defined as $\mathbf{x} - \mathbf{y} = (X_1 \setminus Y_1, \ldots, X_k \setminus Y_k)$. A function $f : (k+1)^V \to \mathbb{R}$ is k-**submodular** iff

$$f(\mathbf{x}) + f(\mathbf{y}) \geq f(\mathbf{x} \sqcap \mathbf{y}) + f(\mathbf{x} \sqcup \mathbf{y})$$

where $\mathbf{x} \sqcap \mathbf{y} = (X_1 \cap Y_1, ..., X_k \cap Y_k)$, and $\mathbf{x} \sqcup \mathbf{y} = (Z_1, ...Z_k)$ where $Z_i = X_i \cup Y_i \setminus (\bigcup_{j \neq i} X_j \cup Y_j)$.

The function f is **monotone** if $f(\mathbf{x}) \geq f(\mathbf{y})$, for all $\mathbf{x} \sqsubseteq \mathbf{y}$. The marginal gain of adding e to the i-th set of $\mathbf{x}$ denote as $\Delta_{e,i} f(\mathbf{x}) = f(\mathbf{x} + \langle e, i \rangle) - f(\mathbf{x})$. We assume that there exists an **oracle query** that returns $f(\mathbf{x})$ when queried for any set k-set $\mathbf{x}$.

Given lower and upper fairness bounds l_i and u_i (with $l_i \leq u_i$ for all $i \in [k]$). A k-set $\mathbf{x} = (X_1, \ldots, X_k) \in (k+1)^V$ is said to be **fair** if each component X_i satisfies $l_i \leq |X_i| \leq u_i \quad \forall i \in [k]$. Without loss of generality, we assume that $u_i \geq 1$ for all $i \in [k]$.

Matroids. Let V be a finite ground set. A collection of subsets $\mathcal{M} \subseteq 2^V$ is said to define a *matroid* if it is non-empty and satisfies the following two axioms:

- *Hereditary property:* for any sets A and B with $A \subseteq B$, if $B \in \mathcal{I}$, then $A \in \mathcal{M}$;
- *Exchange property:* for any $A, B \in \mathcal{M}$ such that $|A| < |B|$, there exists an element $e \in B \setminus A$ for which $A \cup \{e\} \in \mathcal{M}$.

For convenience, we use the notation $A + e$ to denote $A \cup \{e\}$ and $A - e$ to denote $A \setminus \{e\}$. A subset $A \subseteq V$ is called *independent* if $A \in \mathcal{M}$. Throughout this work, we assume that the matroid is accessed via an independence oracle. An independent set that cannot be further extended by inclusion is referred to as a *basis*. A fundamental property of matroids is that all bases have identical cardinality, denoted by $\mathfrak{r}$, which is called the *rank* of the matroid.

Problem definition. In this work, we study *fairness k-submodular maximization under a matroid constraint* (FkSM), which aims to find a fair k-set $\mathbf{x}$ satisfying $\bigcup_{i \in [k]} X_i \in \mathcal{M}$ that maximizes $f(\mathbf{x})$.

The following Lemmas are useful for our theoretical analysis.

Lemma 1. *[Ward and Zivnỳ, 2014] $f : (k+1)^V \to \mathbb{R}_+$ is k-submodular if and only if f satisfies the two following properties: (1) **orthant submodularity**, i.e. $\Delta_{e,i} f(\mathbf{x}) \geq \Delta_{e,i} f(\mathbf{y})$ for any $\mathbf{x}, \mathbf{y} \in (k+1)^V$ with $\mathbf{x} \sqsubseteq \mathbf{y}$ and $e \notin supp(\mathbf{y})$; (2) **pairwise monotonicity**, i.e. $\Delta_{e,i} f(\mathbf{x}) + \Delta_{e,j} f(\mathbf{x}) \geq 0$ for any $\mathbf{x} \in (k+1)^V$, $e \notin supp(\mathbf{x})$ and $i, j \in [k]$ with $i \neq j$.*

Lemma 2 ([Buchbinder *et al.*, 2014]). *Let $f : 2^V \mapsto \mathbb{R}^+$ be submodular. Denote by $A(p)$ a random subset of A where each element appears with probability at most p (not necessary independently). Then $\mathbb{E}[f(A(p))] \geq (1-p) f(\emptyset)$.*

Lemma 3 (Exchange property of matroid bases [Schrijver, 2003]). *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 exists 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.*

We recall the submodular maximization under a matroid constraint (SMM), whose solution serves as a building block in our algorithmic design. Let $f : 2^V \to \mathbb{R}_+$ be a (possibly non-monotone) submodular function (corresponding to the case $k = 1$) and a matroid $\mathcal{M}$. The SMM problem is to find $\arg\max_{S \in \mathcal{M}} f(S)$.

4 Algorithm for UFkSM problem

In this section, we first introduce the Upper Fair Approximation (UFair$k$Sub) algorithm for the UFkSM problem, which serves as a key building block for computing solutions to the general FkSM problem. Given a ground set V, a k-submodular function f, upper bounds u_i, a matroid $\mathcal{M}$, and a parameter $\epsilon > 0$, the UFair$k$Sub algorithm operates in a two-stage framework. In the **first stage** (Lines 1-10), it initializes the solution $\mathbf{s} \leftarrow \mathbf{0}$, sets the set of unfilled components to $I = [k]$, and computes the maximum singleton value M (Line 1).

The algorithm subsequently adopts a threshold-based greedy strategy to incorporate elements with large marginal contributions into $\mathbf{s}$. Concretely, during each iteration of the **while** loop, every candidate element e is examined. If inserting e into the current solution preserves feasibility with respect to the matroid $\mathcal{M}$, the algorithm determines the index $i_e \in I$ that maximizes the marginal gain of e. When this marginal gain is no smaller than the current threshold

θ, the element e is assigned to component i_e and included in the solution (Line 6). Once a component i_e reaches its capacity, namely $|(\mathbf{s})i_e| = u_{i_e}$, it is removed from the set of available components I (Line 8). The threshold θ is initialized to M and is progressively reduced by a multiplicative factor of $(1-\epsilon)$ in successive iterations, until it drops below $2\epsilon M/(3\mathfrak{r})$. The first stage concludes either after all iterations are exhausted or when exactly one component remains unfilled, i.e., $|I| = 1$. Notably, this stage always selects pairs $\langle e, i_e\rangle$ with non-negative marginal gain in to $\mathbf{s}$ due to the *pairwise monotonicity* of f. In particular, for any element e, there exist two distinct labels $i, j \in [k]$ such that $\Delta_{e,i} f(\mathbf{s}) + \Delta_{e,j} f(\mathbf{s}) \geq 0$.

If $|I| = 1$, say $I = \{i\}$, the algorithm moves into the **Second stage** (Lines 11-17): We first define a new matroid $\mathcal{M}' = \{S \subseteq V : S \in \mathcal{M}, |S| \leq u_i\}$ and adapts an $1/\alpha$-approximation algorithm for SMM problem for a submodular function $g(S) \leftarrow f\left(\sum_{e\in S}\langle e, i\rangle\right)$ over ground set V and a matroid constraint $\mathcal{M}'$. Let S be the solution returned by the corresponding approximation algorithm, and define $\mathbf{s}' := \{\langle e, i\rangle \mid e \in S\}$. Finally, the algorithm outputs the better solution between $\mathbf{s}$ and $\mathbf{s}'$.

Algorithm 1: UFair$k$Sub algorithm for UFkSM

Input: $V, k, f, \mathcal{M}, u_1, \ldots, u_k, \epsilon$

1: $\mathbf{s} \leftarrow \mathbf{0}, I \leftarrow [k], \mathbf{s}' \leftarrow \mathbf{0}$, $\theta \leftarrow M \leftarrow \max_{e\in V,\, i\in[k]} f(\langle e, i\rangle)$, $\mathfrak{r} = rank(\mathcal{M})$
2: **while** $\theta \geq \frac{2\epsilon M}{3\mathfrak{r}}$ **do**
3: **for** $e \in V \setminus supp(\mathbf{s})$ **do**
4: **if** $supp(\mathbf{s}) \cup \{e\} \in \mathcal{M}$ **then**
5: $i_e \leftarrow \arg\max_{i\in I} \Delta_{e,i} f(\mathbf{s})$
6: **if** $\Delta_{e,i_e} f(\mathbf{s}) \geq \theta$ **then**
7: $\mathbf{s} \leftarrow \mathbf{s} + \langle e, i_e\rangle$
8: **If** $|(\mathbf{s})_{i_e}| = u_{i_e}$ **then** $I \leftarrow I \setminus \{i_e\}$
9: **If** $|I| = 1$ **then break**
10: $\theta \leftarrow (1-\epsilon)\theta$
11: **if** $|I| = 1$ **then**
12: $i \leftarrow$ only number in I
13: Define a matroid $\mathcal{M}' = \{S \subseteq V : S \in \mathcal{M}, |S| \leq u_i\}$
14: Define $g : 2^V \to \mathbb{R}_+$ as $g(S) \leftarrow f\left(\sum_{e\in S}\langle e, i\rangle\right)$
15: $S \leftarrow \mathtt{Alg}_{\mathtt{SMM}}(g, V, \mathcal{M}')$
16: $\mathbf{s}' \leftarrow \sum_{e\in S}\langle e, i\rangle$
17: **break**
18: $\mathbf{x} \leftarrow \arg\max_{\mathbf{x}'\in\{\mathbf{s},\mathbf{s}'\}} f(\mathbf{x})$
19: **return** $\boldsymbol{x}$

Theoretical Analysis. Denote $\langle e_j, i_j\rangle$ as the j-th pair added to $\mathbf{s}$ and $t = |supp(\mathbf{s})|$ after the while loop of the Algorithm 1. Let $\mathbf{s}_j = (\langle e_1, i_1\rangle, \langle e_2, i_2\rangle, \ldots, \langle e_j, i_j\rangle)$ represent the solution $\mathbf{s}$ after adding $j \leq t$ pairs.

Let $\mathbf{o}$ denote an optimal solution to the UFkSM problem. For each iteration j of the **while** loop in Algorithm 1, we define a **transformation** that produces a k-set $\mathbf{o}_j$ satisfying the following properties: (1) $|(\mathbf{o}_j)_i| \leq u_i$ for all i, and (2) $\mathbf{s}_j \sqsubseteq \mathbf{o}_j$.

The transformation is defined as follows. For $j = 0$, set $\mathbf{o}_0 = \mathbf{o}$. For $j \geq 1$, we consider the following cases:

- If $\mathbf{o}_{j-1}(e_j) = i_j$, then set $\mathbf{o}_j = \mathbf{o}_{j-1}$.
- If $\mathbf{o}_{j-1}(e_j) = 0$, set $\mathbf{o}'_j = \mathbf{o}_{j-1} + \langle e_j, i_j\rangle$. If $|(\mathbf{o}'_j)_{i_j}| \leq u_{i_j}$, then set $\mathbf{o}_j = \mathbf{o}'_j$. Otherwise, select an arbitrary element $u \in (\mathbf{o}'_j)_{i_j} \setminus (\mathbf{s}_j)_{i_j}$ and set $\mathbf{o}_j := \mathbf{o}'_j - \langle u, i_j\rangle$.
- If $i_j \neq \mathbf{o}_{j-1}(e_j) \neq 0$, let $p := \mathbf{o}_{j-1}(e_j)$ and set $\mathbf{o}'_j := \mathbf{o}_{j-1} - \langle e_j, p\rangle + \langle e_j, i_j\rangle$. If $|(\mathbf{o}'_j)_{i_j}| \leq u_{i_j}$, then set $\mathbf{o}_j := \mathbf{o}'_j$. Otherwise, select an arbitrary element $u \in (\mathbf{o}'_j)_{i_j} \setminus (\mathbf{s}_j)_{i_j}$ and set $\mathbf{o}_j := \mathbf{o}'_j - \langle u, i_j\rangle$.

The transformation projects the optimal solution $\mathbf{o}$ onto the evolving solution $\mathbf{s}_j$. At each iteration, it retains $\mathbf{s}_j$ as a fixed backbone while allowing $\mathbf{o}_j$ to include additional elements within the given bounds. This alignment enables controlled synchronization between $\mathbf{o}$ and $\mathbf{s}_j$, which is key to establishing both approximation and fairness guarantees.

Lemma 4 ensures that the residual set $E_j = supp(\mathbf{o}_j) \setminus supp(\mathbf{s}_j)$ is always a subset of the matroid-feasible set $supp(\mathbf{o})$. As a result, E_j remains matroid-independent at every iteration, which is crucial for proving that solutions is can be extended without violating matroid constraint.

Lemma 4. *For each iteration j, define $E_j = supp(\mathbf{o}_j) \setminus supp(\mathbf{s}_j)$. Then we have $E_j \subseteq supp(\mathbf{o})$.*

Lemma 5 establishes a per-iteration bound between the decrease in the value of the transformed solution $\mathbf{o}_j$ and the marginal gain achieved by the constructed solution $\mathbf{s}_j$. This inequality enables a charging argument, where the loss of $\mathbf{o}_j$ is paid for by the progress of $\mathbf{s}_j$, with constants depending on whether f is monotone or non-monotone. By summing over all iterations, the lemma directly links local improvements to a global approximation guarantee, thereby playing a central role in the overall analysis.

Lemma 5. *For each $j \in [t]$, the following holds:*

- *If f is non-monotone, then*

$$f(\mathbf{o}_{j-1}) - f(\mathbf{o}_j) \;\leq\; 3\frac{f(\mathbf{s}_j) - f(\mathbf{s}_{j-1})}{1-\epsilon}.$$

- *If f is monotone, then*

$$f(\mathbf{o}_{j-1}) - f(\mathbf{o}_j) \;\leq\; 2\frac{f(\mathbf{s}_j) - f(\mathbf{s}_{j-1})}{1-\epsilon}.$$

Theorem 1. *For any $\epsilon > 0$, Algorithm 1 satisfies the following properties.*

- *For non-monotone objective functions f, the algorithm achieves an approximation ratio of $\frac{1}{6.494} - \epsilon$.*
- *For monotone objective functions f, the algorithm achieves an approximation ratio of $\frac{1}{4.582} - \epsilon$.*
- *Algorithm 1 runs in $O\left(\frac{nk}{\epsilon}\log r\right) + Q(\mathtt{Alg}(\mathtt{SMM}))$, where $Q(\mathtt{Alg}(\mathtt{SMM}))$ denotes the query complexity of the underlying SMM algorithm used as a subroutine.*

Proof. We first prove the approximation ratio for the case non-monotone functions f. Since $\mathbf{s}_t \sqsubseteq \mathbf{o}_t$, one can write $\mathbf{o}_t = \mathbf{s}_t \sqcup \mathbf{z}_r$, where residual k-set $\mathbf{z}_r = \{\langle z_1, i_{z_1}\rangle, \langle z_2, i_{z_2}\rangle, \ldots, \langle z_r, i_{z_r}\rangle\}, z_i \in supp(\mathbf{o}_t) \setminus supp(\mathbf{s}_t), i_{z_i} = \mathbf{o}_t(z_i), i \in [r]$. Consider the following cases: **Case 1.** If $|\mathbf{s}_t| = \mathfrak{r}$, then $r \leq \mathfrak{r}$. Since $supp(\mathbf{s}_t) \in \mathcal{M}$, it follows from the down-closed property that $supp(\mathbf{s}_r) \in \mathcal{M}$. By the augmentation property, there exists an ordering $supp(\mathbf{z}) = \{z_1, z_2, \ldots, z_r\}$ such that $supp(\mathbf{s}_{i-1}) + z_i = \{e_1, e_2, \ldots, e_{i-1}, z_i\} \in \mathcal{M}, \forall i \leq r$. The algorithm awlays add pairs with non-negative marignal gain so $f(\mathbf{s}_t) \geq f(\mathbf{s}_r)$, therefore

$$
\begin{aligned}
&f(\mathbf{o}_t) - f(\mathbf{s}_t) \leq f(\mathbf{o}_t) - f(\mathbf{s}_r) = f(\mathbf{s}_t \sqcup \mathbf{z}_r) - f(\mathbf{s}_r) \\
&= \sum_{i=1}^{r} \Delta_{z_i, i_{z_i}} f(\mathbf{s}_t \sqcup \mathbf{z}_{i-1}) \overset{(a)}{\leq} \sum_{i=1}^{r} \Delta_{z_i, i_{z_i}} f(\mathbf{s}_{i-1}) \\
&\overset{(b)}{\leq} \sum_{i=1}^{r} \frac{\theta_{(i)}}{1-\epsilon} \overset{(c)}{\leq} \sum_{i=1}^{r} \frac{\Delta_{e,i} f(\mathbf{s}_{i-1})}{1-\epsilon} = \frac{f(\mathbf{s}_r)}{1-\epsilon} \qquad (1)
\end{aligned}
$$

where (a) follows from the orthant submodularity of f, (b) holds since the pair $\langle z_i, i_{z_i}\rangle$ was not selected into $\mathbf{s}_t$ in earlier iterations, and (c) follows from the selection rule of the pair $\langle e, i\rangle$ into $\mathbf{s}_{i-1}$. On other hand, by Lemma 5, we obtain

$$
\begin{aligned}
f(\mathbf{o}) - f(\mathbf{o}_t) &= \sum_{j=1}^{t} (f(\mathbf{o}_{j-1}) - f(\mathbf{o}_j)) \\
&\leq \sum_{i=1}^{t} 3\frac{f(\mathbf{s}_j) - f(\mathbf{s}_{j-1})}{1-\epsilon} = \frac{3f(\mathbf{s}_t)}{1-\epsilon}. \qquad (2)
\end{aligned}
$$

Therefore $f(\mathbf{o}) - f(\mathbf{s}_t) = f(\mathbf{o}) - f(\mathbf{o}_t) + f(\mathbf{o}_t) - f(\mathbf{s}_t) \leq \frac{3f(\mathbf{s}_t)}{1-\epsilon} + \frac{f(\mathbf{s}_r)}{1-\epsilon} \leq \frac{4f(\mathbf{s}_t)}{1-\epsilon}$ which implies $f(\mathbf{s}_t) \geq \frac{1-\epsilon}{5} f(\mathbf{o})$.

Case 2: If $|\mathbf{s}_t| < \mathfrak{r}$, we consider the following cases: If $|I| = 1$ and i is only number in I. From (2) we have:

$$
f(\mathbf{o}) = f(\mathbf{o}_t) + f(\mathbf{o}) - f(\mathbf{o}_t) \leq f(\mathbf{o}_t) + 3\frac{f(\mathbf{s}_t)}{1-\epsilon}. \qquad (3)
$$

By the properties of $\mathbf{o}_t$, we have $(\mathbf{s}_t)_l = (\mathbf{o}_v)_l, \forall l \neq i$ and $(\mathbf{s}_t)_i \sqsubseteq (\mathbf{o}_t)_i$. By Lemma 3 $supp(\mathbf{z}_r) = E_t \in \mathcal{M}$ and $|\mathbf{z}_r| \leq u_i$. Thus $\mathbf{z}_r$ is a feasible solution of UFkSM problem. Since g is submodular, assuming $1/\alpha$ be the approximation ratio of $\mathcal{A}$ for SMM, we have $f(\mathbf{z}_r) \leq \alpha g(S) = \alpha f(\mathbf{s}')$. Therefore $f(\mathbf{o}_t) = f(\mathbf{s}_t \sqcup \mathbf{z}_r) \leq f(\mathbf{s}_t) + f(\mathbf{z}_r) \leq f(\mathbf{s}_t) + \alpha f(\mathbf{s}')$. Put it into (3) we have $f(\mathbf{o}) \leq (1 + \frac{3}{1-\epsilon}) f(\mathbf{s}_t) + \alpha f(\mathbf{s}') \leq (\alpha + \frac{4-\epsilon}{1-\epsilon}) f(\mathbf{x})$. Therefore $f(\mathbf{x}) \geq \frac{1-\epsilon}{\alpha(1-\epsilon)+4-\epsilon} f(\mathbf{o}) \geq \frac{1-\epsilon}{4+\alpha} f(\mathbf{o})$. If $|I| > 1$, we assume that the algorithm continues the main loop until $|I| = 1$ **or** $|supp(\mathbf{s})| = \mathfrak{r}$. This is possible due to the *pairwise monotonicity* of the function f. Specifically, for each element e and the current solution $\mathbf{x}$, there exist two distinct labels $i, j \in [k]$ such that $\Delta_{e,i} f(\mathbf{x}) + \Delta_{e,j} f(\mathbf{x}) \geq 0$. Therefore, at least one label $i \in \{i, j\}$ satisfies $\Delta_{e,i} f(\mathbf{x}) > 0$, which guarantees that the algorithm can make progress in each iteration and thus continue the main loop until $|\mathbf{s}| = \mathfrak{r}$ or $|I| = 1$. Let $\mathbf{s}_T = (\langle e_1, i_1\rangle, \langle e_2, i_2\rangle, \ldots, \langle e_T, i_T\rangle), T > t$ as $\mathbf{s}$ in this case. By the similarity argument of **Case 1** and **Case 2** with $|I| = 1$, we have $f(\mathbf{s}_T) \geq \frac{1-\epsilon}{4+\alpha} f(\mathbf{o})$. We have

$$
\begin{aligned}
f(\mathbf{s}_T) - f(\mathbf{s}_t) &\leq \sum_{e \in supp(\mathbf{s}_T) \setminus supp(\mathbf{s}_t)} \Delta_{e, \mathbf{s}_T(e)} f(\mathbf{s}_t) \\
&\overset{(d)}{\leq} \frac{2\epsilon M}{3\mathfrak{r}} \leq \frac{2\epsilon}{3} f(\mathbf{o})
\end{aligned}
$$

where (d) holds since the marginal gain of each additional pair is smaller than the final threshold $2\epsilon M/(3\mathfrak{r})$. This implies that $f(\mathbf{s}_t) \geq f(\mathbf{s}_T) - \frac{2\epsilon}{3} f(\mathbf{o}) \geq (\frac{1}{4+\alpha} - \epsilon) f(\mathbf{o})$. Combining all cases, we obtain an approximation ratio of $\frac{1}{4+\alpha} - \epsilon$. By instantiating the best-known approximation algorithm for SMM with $\alpha = \frac{1}{0.401}$ [Buchbinder and Feldman, 2024], the approximation ratio becomes $\frac{1}{6.494} - \epsilon$. The proofs of approximation ratio for monotone functions and query complexity are presented in the Appendix. □

5 Algorithm for general FkSM problem

We now present a novel algorithmic framework, denoted by Fair$k$Sub (Algorithm 2), for the FkSM problem. The framework illustrates how the previously proposed UFair$k$Sub algorithm can be systematically adapted as a key subroutine to achieve the desired theoretical guarantees. Consequently, the resulting algorithm has a more intricate structure and adopts a two-phase strategy to construct solutions while rigorously enforcing the required bound guarantees.

Given an instance of FkSM with parameter ϵ, the algorithm proceeds in two stages. In **Phase 1**, the algorithm first computes a maximum-cardinality independent set $L \in \mathcal{M}$, which can be obtained in polynomial time and satisfies $|L| = \mathfrak{r} \geq \sum_{i \in [k]} l_i$. It then adapts UFair$k$Sub to compute a candidate solution $\mathbf{u}$ for UFkSM on the restricted ground set L, subject to upper fairness bounds l_i and a matroid constraint that is reduced to a cardinality constraint of size $\sum_{i \in [k]} l_i$. For each component i of $\mathbf{u}$, the set $(\mathbf{u})_i$ is filled with l_i elements (Line 5). The algorithm then randomly splits $\mathbf{u}$ into two solutions, $\mathbf{u}^{(1)}$ and $\mathbf{u}^{(2)}$, such that each component $(\mathbf{u}^{(j)})_i$ has size $\lfloor l_i/2 \rfloor$ (Line 6). The algorithm continues adapting UFair$k$Sub twice to find candidate solutions $\mathbf{x}^{(j)}, j \in \{1, 2\}$ with setting constraints: upper fairness bounds u_i, an induced matroid of $\mathcal{M}$ defined by $supp(\mathbf{u}^{(j)})$ as $\mathcal{M}_j = \{T \subseteq V : T \cup supp(\mathbf{u}^{(j)}) \in \mathcal{M}\}$. This step yields a potential solutions $\mathbf{x}^{(j)}$ that satisfies all upper fairness constraints while still leaving room for improvement (Line 8).

In **Phase 2**, the algorithm initializes $\mathbf{y}^{(j)} \leftarrow \mathbf{x}^{(j)}$ and then refines the solution by selectively adding elements to any component i from $(\mathbf{u}^{(j)})_i$ satisfying $|(\mathbf{y}^{(j)})_i| < u_i$, with the dual objectives of *(1) enhancing satisfaction of the lower fairness bounds* and *(2) ensuring a provable bounded guarantee on the quality* of the resulting solutions $\mathbf{y}^{(j)}$. By construction, this process always preserves matroid feasibility, due to the definition of the induced matroids $\mathcal{M}_j$ and the construction of $\mathbf{u}^{(j)}$. Finally, the algorithm outputs the best solution among all candidates $\{\mathbf{y}^{(1)}, \mathbf{y}^{(2)}, \mathbf{u}\}$, thereby balancing upper and lower fairness requirements while guaranteeing bounded approximation performance.

We present the theoretical guarantees stated in Theorem 2.

Algorithm 2: Fair$k$Sub algorithm for FkSM

Input: $V, k, f, l_i, u_i, \forall i \in [k], \mathcal{M}, \epsilon$

// Phase 1: Initializing solution under upper fairness bounds

$L \leftarrow \arg\max_{S \subseteq V, S \in \mathcal{M}} |S|$, $\mathbf{u}^{(1)} \leftarrow \mathbf{0}$, $\mathbf{u}^{(2)} \leftarrow \mathbf{0}$

$\mathbf{u} \leftarrow$ Adapt UFair$k$Sub with upper bounds l_i and $\mathcal{M}$ is size constraint $\sum_{i \in [k]} l_i$

for $i \in [k]$ **do**

  if $|(\mathbf{u})_i| < l_i$ **then**

    Randomly select $l_i - |(\mathbf{u})_i|$ elements from $L \setminus supp(\mathbf{u})$, assign them to $(\mathbf{u})_i$, and remove them from L

  Uniformly at random select $2\lfloor \frac{l_i}{2} \rfloor$ elements from $(\mathbf{u})_i$ and divide them evenly into two subsets $(\mathbf{u}^{(1)})_i$ and $(\mathbf{u}^{(2)})_i$

Define matroids $\mathcal{M}_j = \{T \subseteq V : T \cup supp(\mathbf{u}^{(j)}) \in \mathcal{M}\}, j \in \{1, 2\}$

$\mathbf{x}^{(j)} \leftarrow$ UFair$k$Sub$(V, f, k, f, u_1, \ldots, u_k, \mathcal{M}_j, \epsilon)$

// Phase 2: Solution refinement under fairness bounds

for $j \in \{1, 2\}$ **do**

  $\mathbf{y}^{(j)} \leftarrow \mathbf{x}^{(j)}$

  for $i \in [k]$ **do**

    foreach $e \in (\mathbf{u}^{(j)})_i$ **do**

      if $|(\mathbf{y}^{(j)})_i| < u_i$ **then**

        $\mathbf{y}^{(j)} \leftarrow \mathbf{y}^{(j)} + \langle e, i \rangle$

$\mathbf{s} \leftarrow \arg\max_{\mathbf{x} \in \{\mathbf{y}^{(1)}, \mathbf{y}^{(2)}, \mathbf{u}\}} f(\mathbf{x})$

return $\mathbf{s}$

Theorem 2. *For any $\epsilon > 0$, the following statements about Algorithm 2 hold.*

- *It returns a solution $\mathbf{s}$ that satisfies $supp(\mathbf{s}) \in \mathcal{M}$ and $\lfloor \frac{l_i}{2} \rfloor \leq |(\mathbf{s})_i| \leq u_i$ for the FkSM problem.*
- *For non-monotone functions f, it attains an approximation ratio of $\frac{1}{25.976} - \frac{\epsilon}{4}$.*
- *For monotone functions f, the algorithm achieves an approximation ratio of $\frac{1}{9.164} - \frac{\epsilon}{2}$.*
- *Algorithm 1 runs in $O\left(\frac{nk}{\epsilon} \log \mathfrak{r}\right) + Q(\text{Algorithm for SMM})$, where $Q(\text{Algorithm for SMM})$ denotes the query complexity of the underlying SMM algorithm used as a subroutine.*

Proof. We present the approximation ratio analysis below. The proofs of the fairness bounds and the query complexity can be found in the appendix. **Idea of the proof.** The approximation ratio is obtained by splitting the optimal solution $\mathbf{o}$ into two k-sets and using the matroid base exchange property to transform them into feasible subproblem solutions. Applying UFair$k$Sub to these subproblems and aggregating the results yields the claimed guarantee.

By Lemma 3, the optimal solution $\mathbf{o}$ can be decomposed as $\mathbf{o} = \mathbf{o}_1 \sqcup \mathbf{o}_2$, where $\mathbf{o}_1 = (O_1^1, \ldots, O_k^1)$ and $\mathbf{o}_2 = (O_1^2, \ldots, O_k^2)$, with $supp(\mathbf{o}_1) \cap supp(\mathbf{o}_2) = \emptyset$, such that $L_1 \cup supp(\mathbf{o}_1) \in \mathcal{M}$ and $L_2 \cup supp(\mathbf{o}_2) \in \mathcal{M}$, where $L_1 = supp(\mathbf{u}^{(1)})$ and $L_2 = supp(\mathbf{u}^{(2)})$. This implies that each $\mathbf{o}_j$ is a feasible solution to the UFkSM problem with instance $\mathcal{I}_j = (f, V, u_1, \ldots, u_k, \mathcal{M}_j)$ for $j \in 1, 2$. Let ρ denote the approximation ratio of UFair$k$Sub. After Line 6 of Algorithm 2, we have

$$f(\mathbf{x}^{(j)}) \geq \rho f(\mathbf{o}_j), \forall j \in \{1, 2\}. \tag{4}$$

For the case where f is non-monotone. For any k-set $\mathbf{v} \in (k+1)^V$ that $supp(\mathbf{v}) = \{v_1, v_2, \ldots, v_m\}$, we define a finite ground set $T(\mathbf{v}) = \{\langle v_1, \mathbf{v}(v_1)\rangle, \ldots, \langle v_m, \mathbf{v}(v_m)\rangle\}$, where each pair $\langle v_i, \mathbf{v}(v_i)\rangle$ is treated as an individual element of $T(\mathbf{v})$. We then define a function $h(\cdot) : 2^{T(\mathbf{v})} \mapsto \mathbb{R}_+$ by $h(T(\mathbf{v}')) = f(\mathbf{v}')$ for any $\mathbf{v}' \sqsubseteq \mathbf{v}$. By the orthant submodularity of f, it follows that h is a non-negative and submodular function. For $j \in \{1, 2\}$, consider the function $h'(\cdot) = h(T(\mathbf{x}^{(j)}) \sqcup \cdot)$, h' is also non-negative and submodular function. Denote $\mathbf{u}'^{(j)} \sqsubseteq \mathbf{u}^{(j)}$ is a k-set that contains all pairs added into $\mathbf{y}^{(j)}$ in **for** loop (Lines 9-14). By the construction of $\mathbf{u}^{(j)}$ in Alg. 2, for any pair $\langle e, i\rangle \sqsubseteq \mathbf{u}'^{(j)}$:

$$\Pr[\langle e, i\rangle \text{ was selected into } \mathbf{y}^{(j)}] \leq \Pr[\langle e, i\rangle \text{ was selected into } \mathbf{u}^{(j)}]$$
$$= \frac{1}{2} \frac{2\lfloor \frac{l_i}{2} \rfloor}{l_i} \leq \frac{1}{2}.$$

Applying Lemma 2 gives

$$\mathbb{E}[f(\mathbf{y}^{(j)})] = \mathbb{E}[f(\mathbf{x}^{(j)} \sqcup \mathbf{u}'^{(j)})] = \mathbb{E}[h'(T(\mathbf{u}'^{(j)}))]$$
$$\geq (1 - \frac{1}{2}) h'(\emptyset) = \frac{1}{2} h(T(\mathbf{x}^{(j)})) = \frac{1}{2} f(\mathbf{x}^{(j)}).$$

By the selection of final solution, we obtain

$$\mathbb{E}[f(\mathbf{s})] \geq \max\{\mathbb{E}[f(\mathbf{y}^{(1)})], \mathbb{E}[f(\mathbf{y}^{(2)})]\}$$
$$\geq \frac{1}{2} \max\{f(\mathbf{x}^{(1)}), f(\mathbf{x}^{(2)})\} \geq \frac{1}{4}(f(\mathbf{x}^{(1)}) + f(\mathbf{x}^{(2)}))$$
$$\geq \frac{\rho}{4}(f(\mathbf{o}_1) + f(\mathbf{o}_2)) \text{ (Due to (4))}$$
$$\geq \frac{\rho}{4}(f(\mathbf{o}_1 \sqcup \mathbf{o}_2)) \geq \frac{\rho}{4} f(\mathbf{o}) \text{ (Due to the } k\text{-submodularity)} .$$

This implies the approximation ratio $\rho/4 = 1/25.976 - \epsilon/4$. For the case where f is **monotone**, since $\mathbf{x}^{(j)} \sqsubseteq \mathbf{y}^{(j)}$, we have $f(\mathbf{y}^{(j)}) \geq f(\mathbf{x}^{(j)})$, and hence the approximation ratio is $\rho/2 = 1/9.164 - \epsilon/2$. □

6 Experimental Evaluation

In this section, we conduct comprehensive experiments to demonstrate the performance of our Fair$k$Sub algorithm on three representative applications of the general FkSM problem, namely Max-k-Cut (MkC), k-topic Influence Maximization (kTIM), and k-measurement Sensor Placement (kMSP). We set up a partition matroid (a special case of a matroid) in our experiments, following prior study [El Halabi *et al.*, 2023]. We parameterize the fairness constraint by a budget ratio $B \in (0, 1]$. Specifically, we set the cardinality budget to $b = \lfloor B|V| \rfloor$, and define the fairness bounds for each label $i \in [k]$ by $l_i = \left\lfloor 0.8 \frac{b}{k} \right\rfloor, u_i = \left\lfloor 1.4 \frac{b}{k} \right\rfloor, \forall i \in [k]$. In all figures, B is reported on the horizontal axis; accordingly,

B serves as the control parameter of the fairness constraint. Since the FkSM problem has not been explicitly studied in prior work, we compare our approach with the most closely related state-of-the-art algorithms:

- **FairGreedy**: a traditional greedy algorithm that iteratively selects the pair with the highest marginal gain while satisfying both the matroid and fairness constraints. This approach is commonly used for general k-submodular maximization and has been adopted in recent works [Ohsaka and Yoshida, 2015; Zhu *et al.*, 2024].
- **kGreedyIS**: the algorithm for k-submodular maximization under individual size constraints [Ohsaka and Yoshida, 2015].
- **kGreedyTS**: the algorithm for k-submodular maximization under a total size constraint [Ohsaka and Yoshida, 2015].

To adapt **kGreedyIS** and **kGreedyTS** to our problem setting, we proceed as follows: *(i)* we first construct a maximum-size feasible solution satisfying the matroid constraint; *(ii)* we adapt **kGreedyIS** by treating the upper fairness bounds as individual size constraints; and *(iii)* we adapt **kGreedyTS** by setting the total size constraint to B. These adapted algorithms serve as heuristic baselines and do not provide any theoretical approximation guarantees. We evaluate all algorithms using four criteria: (i) objective value, (ii) number of oracle queries, (iii) running time, and (iv) fairness error $\texttt{FairErrors}(\mathbf{x}) = \sum_{i\in[k]} \max\{|(\mathbf{x})_i| - u_i, ; l_i - |(\mathbf{x})_i|; 0\}$, as adopted in [Zhu *et al.*, 2024]. Due to page limitations, detailed experimental settings and matroid specifications are deferred to the Appendix.

Experimental results. We report the main results (Fig. 1) for two representative scenarios: (i) MkC (ii) the kTIM on Amazon and Facebook datasets. More detailed results are included in the Appendix.

Objective value. Fig. 1(a,d) shows that Fair$k$Sub provides stable solution quality as B increases. On Amazon (MkC), Fair$k$Sub achieves objective values close to FairGreedy (a gap of about 5%), while it clearly outperforms kGreedyIS and kGreedyTS (about 36.9%–85.0% higher than kGreedyIS, and 37.0%–69.0% higher than kGreedyTS). On Facebook, Fair$k$Sub remains competitive with FairGreedy, while consistently outperforming kGreedyIS by about 0.7%–7.6% and substantially surpassing kGreedyTS by approximately 12.2%–31.8%.

Number of queries. Fig. 1(b,e) highlights the query efficiency of Fair$k$Sub, especially for MkC. On Amazon, Fair$k$Sub reduces the number of queries by about 79–278 times compared to FairGreedy, and by about 35–93 times compared to kGreedyIS/kGreedyTS. On Facebook, the reduction is more moderate: compared to FairGreedy, Fair$k$Sub uses about 1.3–3.7 times fewer queries; compared to kGreedyIS, Fair$k$Sub uses fewer queries when $B \geq 0.04$ (about 1.3–1.7 times fewer), while at $B = 0.02$ it may require more queries.

Fairness Errors. Fig. 1(c,f) indicates that Fair$k$Sub generally controls the fairness constraints well. On Amazon, Fair$k$Sub attains $\texttt{FairErrors} = 0$ for most cases B and only shows a very small violation at $B = 0.5$ (fair $\texttt{FairErrors} = 17$), whereas kGreedyTS has substantial violations that grow with B (e.g., 350 $\rightarrow$ 1366), and kGreedyIS/FairGreedy may also violate fairness at larger B. On Facebook, Fair$k$Sub keeps fair error equal to 0 for all tested B, matching FairGreedy/kGreedyIS, and improving over kGreedyTS (whose violations increase from 72 to 534 as B grows). Overall, the results suggest that Fair$k$Sub strikes a good balance between solution quality and computational cost: **while remaining competitive in objective value, Fair$k$Sub substantially reduces the number of queries, and maintains very small fairness violations.**

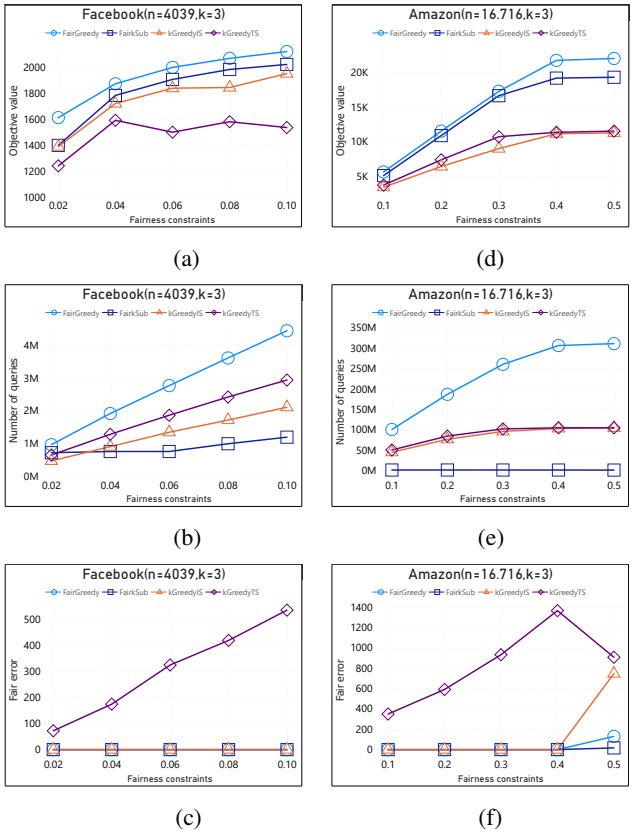


Figure 1: Performance of the algorithms in terms of objective value, query complexity, and fairness error.

7 Conclusions

We studied the FkSM problem with possibly non-monotone objectives, which has broad significance and impact in artificial intelligence and machine learning. Our combinatorial two-phase framework provides the first constant-factor approximation guarantees while ensuring matroid feasibility, full upper-bound satisfaction, and a factor-$1/2$ guarantee on the lower bounds. Experiments on standard benchmark datasets further demonstrate the effectiveness of our approach in terms of solution quality, query efficiency, and running time. An important direction for future work is to extend this framework to richer and more general constraint settings.

References

[Buchbinder and Feldman, 2024] Niv Buchbinder and Moran Feldman. Constrained submodular maximization via new bounds for dr-submodular functions. 2024.

[Buchbinder *et al.*, 2014] Niv Buchbinder, Moran Feldman, Joseph Naor, and Roy Schwartz. Submodular maximization with cardinality constraints. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*, pages 1433–1452. SIAM, 2014.

[Celis *et al.*, 2018] L. Elisa Celis, Lingxiao Huang, and Nisheeth K. Vishnoi. Multiwinner voting with fairness constraints. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 144–151. International Joint Conferences on Artificial Intelligence Organization, 7 2018.

[Chen and Kuhnle, 2024] Yixin Chen and Alan Kuhnle. Practical and parallelizable algorithms for non-monotone submodular maximization with size constraint. *Journal of Artificial Intelligence Research*, 79:599–637, 2024.

[Cualinescu *et al.*, 2011] Gruia Cualinescu, Chandra Chekuri, Martin Pál, and Jan Vondrák. Maximizing a monotone submodular function subject to a matroid constraint. *SIAM J. Comput.*, 40(6):1740–1766, 2011.

[Cui *et al.*, 2024] Shuang Cui, Kai Han, Shaojie Tang, Feng Li, and Jun Luo. Fairness in streaming submodular maximization subject to a knapsack constraint. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, page 514–525, New York, NY, USA, 2024. Association for Computing Machinery.

[El Halabi *et al.*, 2023] Marwa El Halabi, Federico Fusco, Ashkan Norouzi-Fard, Jakab Tardos, and Jakub Tarnawski. Fairness in streaming submodular maximization over a matroid constraint. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 9150–9171. PMLR, 23–29 Jul 2023.

[Ene and Nguyen, 2022] Alina Ene and Huy Nguyen. Streaming algorithm for monotone k-submodular maximization with cardinality constraints. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 5944–5967. PMLR, 17–23 Jul 2022.

[Feldman *et al.*, 2018] Moran Feldman, Amin Karbasi, and Ehsan Kazemi. Do less, get more: Streaming submodular maximization with subsampling. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018*, pages 730–740, 2018.

[Frieze and Jerrum, 1997] Alan Frieze and Mark Jerrum. Improved approximation algorithms for max k-cut and max bisection. *Algorithmica*, 18(1):67–81, 1997.

[Ha *et al.*, 2024] Dung T.K. Ha, Canh V. Pham, and Tan D. Tran. Improved approximation algorithms for k-submodular maximization under a knapsack constraint. *Computers & Operations Research*, 161:106452, 2024.

[Halabi *et al.*, 2020] Marwa El Halabi, Slobodan Mitrovic, Ashkan Norouzi-Fard, Jakab Tardos, and Jakub Tarnawski. Fairness in streaming submodular maximization: Algorithms and hardness. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[Halabi *et al.*, 2024] Marwa El Halabi, Jakub Tarnawski, Ashkan Norouzi-Fard, and Thuy-Duong Vuong. Fairness in submodular maximization over a matroid constraint. In *International Conference on Artificial Intelligence and Statistics, 2-4 May 2024, Palau de Congressos, Valencia, Spain*, volume 238 of *Proceedings of Machine Learning Research*, pages 1027–1035. PMLR, 2024.

[Han *et al.*, 2020] Kai Han, Zongmai Cao, Shuang Cui, and Benwei Wu. Deterministic approximation for submodular maximization over a matroid in nearly linear time. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[Iwata *et al.*, 2016] Satoru Iwata, Shin-ichi Tanigawa, and Yuichi Yoshida. Improved approximation algorithms for k-submodular function maximization. In *Proceedings of the twenty-seventh annual ACM-SIAM symposium on Discrete algorithms*, pages 404–413. Society for Industrial and Applied Mathematics, 2016.

[Kuhnle, 2019] Alan Kuhnle. Interlaced greedy algorithm for maximization of submodular functions in nearly linear time. *arXiv preprint arXiv:1902.06179*, 2019.

[Mirzasoleiman *et al.*, 2016] Baharan Mirzasoleiman, Ashwinkumar Badanidiyuru, and Amin Karbasi. Fast constrained submodular maximization: Personalized data summarization. In Maria-Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 1358–1367. JMLR.org, 2016.

[Nguyen and Thai, 2020] Lan Nguyen and My T Thai. Streaming k-submodular maximization under noise subject to size constraint. In *International Conference on Machine Learning*, pages 7338–7347. PMLR, 2020.

[Ohsaka and Yoshida, 2015] Naoto Ohsaka and Yuichi Yoshida. Monotone k-submodular function maximization with size constraints. In *Advances in Neural Information Processing Systems*, pages 694–702, 2015.

[Oshima, 2017] Hiroki Oshima. Derandomization for k-submodular maximization. In *International Workshop on Combinatorial Algorithms*, pages 88–99. Springer, 2017.

[Pham *et al.*, 2023] Canh V. Pham, Tan D. Tran, Dung T. K. Ha, and My T. Thai. Linear query approximation algorithms for non-monotone submodular maximization under knapsack constraint. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence,*

IJCAI 2023, 19th-25th August 2023, Macao, SAR, China, pages 4127–4135. ijcai.org, 2023.

[Qi, 2023] Benjamin Qi. On maximizing sums of non-monotone submodular and linear functions. *Algorithmica*, 86(4):1080–1134, November 2023.

[Qian *et al.*, 2017] Chao Qian, Jing-Cheng Shi, Ke Tang, and Zhi-Hua Zhou. Constrained monotone k-submodular function maximization using multiobjective evolutionary algorithms with theoretical guarantee. *IEEE Transactions on Evolutionary Computation*, 22(4):595–608, 2017.

[Sakaue, 2017] Shinsaku Sakaue. On maximizing a monotone k-submodular function subject to a matroid constraint. *Discrete Optimization*, 23:105–113, 2017.

[Schrijver, 2003] Alexander Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency*. Springer Science and Business Media, 2003.

[Singh *et al.*, 2012] Ajit Singh, Andrew Guillory, and Jeff Bilmes. On bisubmodular maximization. In *Artificial Intelligence and Statistics*, pages 1055–1063, 2012.

[Soma, 2019] Tasuku Soma. No-regret algorithms for online k-submodular maximization. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1205–1214, 2019.

[Spaeh *et al.*, 2025] Fabian Christian Spaeh, Alina Ene, and Huy Nguyen. Online and streaming algorithms for constrained k-submodular maximization. In *AAAI-25, Sponsored by the Association for the Advancement of Artificial Intelligence, February 25 - March 4, 2025, Philadelphia, PA, USA*, pages 20567–20574. AAAI Press, 2025.

[Tang *et al.*, 2022] Zhongzheng Tang, Chenhao Wang, and Hau Chan. On maximizing a monotone k-submodular function under a knapsack constraint. *Operations Research Letters*, 50(1):28–31, 2022.

[Ward and Zivnỳ, 2014] Justin Ward and Stanislav Zivnỳ. Maximizing bisubmodular and k-submodular functions. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*, pages 1468–1481. Society for Industrial and Applied Mathematics, 2014.

[Ward and Zivnỳ, 2016] Justin Ward and Stanislav Zivnỳ. Maximizing k-submodular functions and beyond. *ACM Transactions on Algorithms (TALG)*, 12(4):47, 2016.

[Yuan and Tang, 2023] Jing Yuan and Shaojie Tang. Group fairness in non-monotone submodular maximization. *J. Comb. Optim.*, 45(3):88, 2023.

[Zhou *et al.*, 2025] Huanjian Zhou, Lingxiao Huang, and Baoxiang Wang. Improved approximation algorithms for k-submodular maximization via multilinear extension. In *The Thirteenth International Conference on Learning Representations*, 2025.

[Zhu *et al.*, 2024] Yanhui Zhu, Samik Basu, and Aduri Pavan. Fairness in monotone k-submodular maximization: Algorithms and applications. In *IEEE International Conference on Big Data, BigData 2024, Washington, DC, USA, December 15-18, 2024*, pages 173–178. IEEE, 2024.

Appendix

A Missed Proofs of Lemmas and Theorems

Lemma 4. For each iteration j, define $E_j = supp(\mathbf{o}_j) \setminus supp(\mathbf{s}_j)$. Then we have $E_j \subseteq supp(\mathbf{o})$.

Proof. We proof this Lemma by the induction
If $j = 0$, $\mathbf{o}_j = \mathbf{o}$, the Lemma holds. Asumme that The Lemma holds with $j - 1 \geq 1$, we consider following cases:

- If $\mathbf{o}_{j-1}(e_j) = i_j$, $\mathbf{o}_j = \mathbf{o}_{j-1}$. Thus $E_j = E_{j-1} \in supp(\mathbf{o})$.
- If $\mathbf{o}_{j-1}(e_j) = 0$, $\mathbf{o}'_j = \mathbf{o}_{j-1} + \langle e_j, i_j \rangle$. We consider the following cases:
  - If $|(\mathbf{o}'_j)_{i_j}| \leq u_{i_j}$, then set $\mathbf{o}_j = \mathbf{o}'_j = \mathbf{o}_{j-1} + \langle e_j, i_j \rangle$. Thus $E_j = supp(\mathbf{o}_j) \setminus supp(\mathbf{s}_j) = supp(\mathbf{o}_{j-1}) \setminus supp(\mathbf{s}_{j-1}) = E_{j-1}$.
  - If $|(\mathbf{o}'_j)_{i_j}| > u_{i_j}$, $\mathbf{o}_j := \mathbf{o}'_j - \langle u, i_j \rangle$. Thus $E_j = supp(\mathbf{o}_j) \setminus supp(\mathbf{s}_j) = ((supp(\mathbf{o}_{j-1}) \cup \{e_j\} \setminus \{u\})) \setminus supp(\mathbf{s}_j)) = (supp(\mathbf{o}_{j-1}) \setminus \{u\}) \setminus supp(\mathbf{s}_{j-1})) = E_{j-1} \setminus \{u\} \in supp(\mathbf{o})$.
- If $i_j \neq \mathbf{o}_{j-1}(e_j) \neq 0$, $p := \mathbf{o}_{j-1}(e_j)$ $\mathbf{o}'_j := \mathbf{x}_{j-1} - \langle e_j, p \rangle + \langle e_j, i_j \rangle$, we always have $supp(\mathbf{o}_j) \subseteq supp(\mathbf{o}_{j-1})$ and thus $E_j = supp(\mathbf{o}_j) \setminus supp(\mathbf{s}_j) = supp(\mathbf{o}_{j-1}) \setminus (supp(\mathbf{s}_{j-1}) \cup \{e\}) = E_{j-1} \setminus \{e\} \in supp(\mathbf{o})$.

In all cases, $E_j \in supp(\mathbf{o})$. This completes the induction and the proof. □

Lemma 5. For each $j \in [t]$, the following holds:

- If f is non-monotone, then

$$f(\mathbf{x}_{j-1}) - f(\mathbf{x}_j) \;\leq\; \frac{3\big(f(\mathbf{s}_j) - f(\mathbf{s}_{j-1})\big)}{1-\epsilon}.$$

- If f is monotone, then

$$f(\mathbf{x}_{j-1}) - f(\mathbf{x}_j) \;\leq\; \frac{2\big(f(\mathbf{s}_j) - f(\mathbf{s}_{j-1})\big)}{1-\epsilon}.$$

Proof. **Prove for case f is non-monotone.**
We now prove the Lemma by the definition of the **transformation** for $\mathbf{o}_j$ and exploring the pairwise monotonicity property of the k-submodular function f. We prove the lemma by consider each cases of the transformation:

- **Case 1.** If $\mathbf{o}_{j-1}(e_j) = i_j$, $\mathbf{o}_j = \mathbf{o}_{j-1}$. Thus $f(\mathbf{o}_{j-1}) - f(\mathbf{o}_j) = 0 \leq f(\mathbf{s}_j) - f(\mathbf{s}_{j-1})$.
- **Case 2.** If $\mathbf{o}_{j-1}(e_j) = 0$. Recap that $\mathbf{o}'_j = \mathbf{o}_{j-1} + \langle e_j, i_j \rangle$. Let q be a value in I that $q \neq i_j$ in this case, we have $|(\mathbf{o}_{j-1})_q| < u_q$, we consider following cases:
  - If $|(\mathbf{o}'_j)_{i_j}| \leq u_{i_j}$, with a notice that $\mathbf{s}_{j-1} \sqsubseteq \mathbf{o}_{j-1}$, we have

$$\begin{aligned} f(\mathbf{o}_{j-1}) - f(\mathbf{o}_j) &= \Delta_{e_j,q} f(\mathbf{o}_{j-1}) - (\Delta_{e_j,q} f(\mathbf{o}_{j-1}) + \Delta_{e_j,i_j} f(\mathbf{o}_{j-1})) && (5)\\ &\leq \Delta_{e_j,q} f(\mathbf{o}_{j-1}) \leq \Delta_{e_j,q} f(\mathbf{s}_{j-1}) && (6)\\ &\leq \Delta_{e_j,i_j} f(\mathbf{s}_{j-1}) = f(\mathbf{s}_j) - f(\mathbf{s}_{j-1}). && (7) \end{aligned}$$

    The sequence of inequalities in (6) follows from pairwise monotonicity and orthant submodularity, while the sequence of inequalities in (7) is a consequence of the selection rule employed by Algorithm 1.
  - If $|(\mathbf{o}'_j)_{i_j}| > u_{i_j}$, by the construction of $\mathbf{o}_j$ and selection rule of u we have $\mathbf{o}_{j-1} - \langle u, i_j \rangle = \mathbf{o}_j - \langle e_j, i_j \rangle$. Therefore,

$$\begin{aligned} f(\mathbf{o}_{j-1}) - f(\mathbf{o}_j) &= \Delta_{u,i_j} f(\mathbf{o}_{j-1} - \langle u, i_j \rangle) + \Delta_{e_j,q} f(\mathbf{o}_j - \langle e_j, i_j \rangle) && (8)\\ &\quad - (\Delta_{e_j,i_j} f(\mathbf{o}_j - \langle e_j, i_j \rangle) + \Delta_{e_j,q} f(\mathbf{o}_j - \langle e_j, i_j \rangle)) && (9)\\ &\leq \Delta_{u,i_j} f(\mathbf{o}_{j-1} - \langle u, i_j \rangle) + \Delta_{e_j,q} f(\mathbf{o}_j - \langle e_j, i_j \rangle) && (10)\\ &\leq \Delta_{u,i_j} f(\mathbf{s}_{j-1}) + \Delta_{e_j,q} f(\mathbf{s}_{j-1}) && (11)\\ &\leq 2\frac{\theta_j}{1-\epsilon} \leq 2\frac{f(\mathbf{s}_j) - f(\mathbf{s}_{j-1})}{1-\epsilon} && (12) \end{aligned}$$

    where inequality (10) is due to pairwise monotonicity. The inequality (11) are due to orthant submodularity, the fact that $\mathbf{s}_{j-1} \sqsubseteq \mathbf{o}_{j-1} - \langle u, i_j \rangle$ and the inequality (12) is due to the selection rule of Algorithm 1: For any pair $\langle u, l \rangle$ that is not added to $\mathbf{s}$ at iteration j, its marginal gain is smaller than the threshold θ in all previous iterations, thus $\Delta_{u,l} f(\mathbf{s}_{j-1}) \leq \frac{\theta_j}{1-\epsilon}$ for any $u \notin supp(\mathbf{s}_{j-1}), l \in [k]$.
- **Case 3.** If $i_j \neq \mathbf{o}_{j-1}(e_j) \neq 0$, then $\mathbf{o}'_j := \mathbf{o}_{j-1} - \langle e_j, p \rangle + \langle e_j, i_j \rangle$ and $p := \mathbf{o}_{j-1}(e_j)$.

– If $|(\mathbf{o}'_j)_{i_j}| \leq u_{i_j}$, $\mathbf{o}_j = \mathbf{o}'_j$ so we have:

$$
\begin{aligned}
f(\mathbf{o}_{j-1}) - f(\mathbf{o}_j) &= 2\Delta_{e_j,p} f(\mathbf{o}_{j-1} - \langle e_j, p\rangle) - \big(\Delta_{e_j,p} f(\mathbf{o}_{j-1} - \langle e_j, p\rangle) + \Delta_{e_j,i_j} f(\mathbf{o}_{j-1} - \langle e_j, p\rangle)\big) && (13)\\
&\leq 2\Delta_{e_j,p} f(\mathbf{o}_{j-1} - \langle e_j, p\rangle) && (14)\\
&\leq 2\ \Delta_{e_j,p} f(\mathbf{s}_{j-1}) && (15)\\
&\leq 2\frac{\theta_j}{1-\epsilon} \leq 2\frac{f(\mathbf{s}_j) - f(\mathbf{s}_{j-1})}{1-\epsilon} && (16)
\end{aligned}
$$

where inequality (14) is due to pairwise monotonicity. The inequality (15) is due to orthant submodularity the fact that $\mathbf{s}_{j-1} \sqsubset \mathbf{o}_{j-1} - \langle e_j, p\rangle$.

– If $|(\mathbf{o}'_j)_{i_j}| > u_{i_j}$, recall that $\mathbf{o}_j := \mathbf{o}'_j - \langle u, i_j\rangle + \langle u, p\rangle$. With a notice that $\mathbf{o}_{j-1} - \langle e_j, p\rangle - \langle u, i_j\rangle = \mathbf{o}_j - \langle e_j, i_j\rangle$ and by the pairwise monotonicity, the selection rule of the algorithm we have

$$
\begin{aligned}
f(\mathbf{o}_{j-1}) - f(\mathbf{o}_j) &= (f(\mathbf{o}_{j-1}) - f(\mathbf{o}_{j-1} - \langle u, i_j\rangle)) + (f(\mathbf{o}_{j-1} - \langle u, i_j\rangle) - f(\mathbf{o}_j) && (17)\\
&= \Delta_{u,i_j} f(\mathbf{o}_{j-1} - \langle u, i_j\rangle) + 2\Delta_{e_j,p} f(\mathbf{o}_{j-1} - \langle u, i_j\rangle - \langle e_j, p\rangle) && (18)\\
&\quad - \big(\Delta_{e_j,p} f(\mathbf{o}_{j-1} - \langle u, i_j\rangle - \langle e_j, p\rangle) + \Delta_{e_j,i_j} f(\mathbf{o}_{j-1} - \langle u, i_j\rangle - \langle e_j, p\rangle)\big) && (19)\\
&\leq \Delta_{u,i_j} f(\mathbf{s}_{j-1}) + 2\Delta_{e_j,p} f(\mathbf{o}_{j-1} - \langle e_j, p\rangle) && (20)\\
&\leq \frac{f(\mathbf{s}_j) - f(\mathbf{s}_{j-1})}{1-\epsilon} + 2\Delta_{e_j,p} f(\mathbf{s}_{j-1}) && (21)\\
&\leq \frac{f(\mathbf{s}_j) - f(\mathbf{s}_{j-1})}{1-\epsilon} + 2\frac{f(\mathbf{s}_j) - f(\mathbf{s}_{j-1})}{1-\epsilon} && (22)\\
&\leq 3\frac{f(\mathbf{s}_j) - f(\mathbf{s}_{j-1})}{1-\epsilon}. && (23)
\end{aligned}
$$

Therefore, by combining all cases, we obtain $f(\mathbf{o}_{j-1}) - f(\mathbf{o}_j) \leq 3\frac{f(\mathbf{s}_j) - f(\mathbf{s}_{j-1})}{1-\epsilon}$.

Prove for case f is monotone. We first reuse the arguments from the previous case, which remain valid when f is monotone. If any of **Case 1**, **Case 2**, or **Case 3** with $|(\mathbf{o}'_j)_{i_j}| \leq u_{i_j}$ occurs, then the lemma holds for monotone f, since

$$
f(\mathbf{o}_{j-1}) - f(\mathbf{o}_j) \ \leq\ \frac{2\big(f(\mathbf{s}_j) - f(\mathbf{s}_{j-1})\big)}{1-\epsilon}.
$$

We consider following remain case **Case 3** with $|(\mathbf{o}'_j)_{i_j}| > u_{i_j}$

If $|(\mathbf{o}'_j)_{i_j}| > u_{i_j}$, recall that $\mathbf{o}_j := \mathbf{o}'_j - \langle u, i_j\rangle + \langle u, p\rangle$. With a notice that $\mathbf{o}_{j-1} - \langle e_j, p\rangle - \langle u, i_j\rangle = \mathbf{o}_j - \langle e_j, i_j\rangle$, therefore

$$
\begin{aligned}
f(\mathbf{o}_{j-1}) - f(\mathbf{o}_j) &\leq f(\mathbf{o}_{j-1}) - f(\mathbf{o}_j - \langle e_j, i_j\rangle) && (24)\\
&= f(\mathbf{o}_{j-1}) - f(\mathbf{o}_j - \langle e_j, i_j\rangle + \langle u, i_j\rangle) + f(\mathbf{o}_j - \langle e_j, i_j\rangle + \langle u, i_j\rangle) - f(\mathbf{o}_j - \langle e_j, i_j\rangle) && (25)\\
&= \Delta_{e,p} f(\mathbf{o}_j - \langle e_j, i_j\rangle + \langle u, i_j\rangle) + \Delta_{u,i_j} f(\mathbf{o}_j - \langle e_j, i_j\rangle) && (26)\\
&\leq \Delta_{e,p} f(\mathbf{s}_{j-1}) + \Delta_{u,i_j} f(\mathbf{s}_{j-1}) && (27)\\
&\leq 2\frac{\theta_j}{1-\epsilon} && (28)\\
&\leq 2\frac{f(\mathbf{s}_j) - f(\mathbf{s}_{j-1})}{1-\epsilon} && (29)
\end{aligned}
$$

where inequality (24) follows from the monotonicity of f, inequality (27) follows from the orthant submodularity of f, and inequality (28) follows from the selection rule of the algorithm. □

Theorem 2. For any $\epsilon > 0$, Algorithm 1 satisfies the following properties.

- For non-monotone objective functions f, the algorithm achieves an approximation ratio of $\frac{1}{6.494} - \epsilon$.
- For monotone objective functions f, the algorithm achieves an approximation ratio of $\frac{1}{4.582} - \epsilon$.
- Algorithm 1 runs in $O\big(\frac{nk}{\epsilon}\log r\big) + Q(\texttt{Alg}(\texttt{SMM}))$, where $Q(\texttt{Alg}(\texttt{SMM}))$ denotes the query complexity of the underlying SMM algorithm used as a subroutine.

Proof. Proof the approximation ratio for monotone case in Theorem 1. We also consider two cases

Case 1. If $|\mathbf{s}_t| = \mathfrak{r}$, apply Lemma 5 for monotone case, we have

$$f(\mathbf{o}) - f(\mathbf{o}_t) = \sum_{j=1}^{t}(f(\mathbf{o}_{j-1}) - f(\mathbf{o}_j)) \leq \sum_{i=1}^{t} 2\frac{f(\mathbf{s}_j) - f(\mathbf{s}_{j-1})}{1-\epsilon} = \frac{2f(\mathbf{s}_t)}{1-\epsilon}. \tag{30}$$

Combine this with (1) we obtain

$$f(\mathbf{o}) = (f(\mathbf{o}) - f(\mathbf{o}_t)) + (f(\mathbf{o}_t) - f(\mathbf{s}_t)) + f(\mathbf{s}_t) \tag{31}$$

$$= \frac{2f(\mathbf{s}_t)}{1-\epsilon} + \frac{f(\mathbf{s}_r)}{1-\epsilon} + f(\mathbf{s}_t) \tag{32}$$

$$\leq \frac{4-\epsilon}{1-\epsilon} f(\mathbf{s}_t) \tag{33}$$

which implies that $f(\mathbf{s}_t) \geq \frac{1-\epsilon}{4-\epsilon} f(\mathbf{o})$.

Case 2: If $|\mathbf{s}_t| < \mathfrak{r}$, we consider the following cases: If $|I| = 1$ and i is only number in I. From (30) we have:

$$f(\mathbf{o}) = f(\mathbf{o}_t) + f(\mathbf{o}) - f(\mathbf{o}_t) \leq f(\mathbf{o}_t) + \frac{2f(\mathbf{s}_t)}{1-\epsilon}. \tag{34}$$

By the properties of $\mathbf{o}_t$, we have $(\mathbf{s}_t)_l = (\mathbf{o}_v)_l, \forall l \neq i$ and $(\mathbf{s}_t)_i \sqsubseteq (\mathbf{o}_t)_i$. By Lemma 3 $supp(\mathbf{z}_r) = E_t \in \mathcal{M}$ and $|\mathbf{z}_r| \leq u_i$. Thus $\mathbf{z}_r$ is a feasible solution of UFkSM problem. Since g is submodular, by applying the best approximation algorithm $\mathcal{A}$ for SMM for monotone submodular objective function $1/\alpha = 1 - 1/e$ [Cualinescu *et al.*, 2011], we have $f(\mathbf{z}_r) \leq \alpha g(S) = \alpha f(\mathbf{s}')$, where $1/\alpha = 1 - 1/e$ [Qi, 2023]. Therefore

$$f(\mathbf{o}_t) = f(\mathbf{s}_t \sqcup \mathbf{z}_r) \leq f(\mathbf{s}_t) + f(\mathbf{z}_r) \leq f(\mathbf{s}_t) + \alpha f(\mathbf{s}').$$

Put it into (34) we have

$$f(\mathbf{o}) \leq (1 + \frac{2}{1-\epsilon}) f(\mathbf{s}_t) + \alpha f(\mathbf{s}') \leq (\alpha + \frac{3-\epsilon}{1-\epsilon}) f(\mathbf{x}).$$

Therefore

$$f(\mathbf{x}) \geq \frac{1-\epsilon}{\alpha(1-\epsilon) + 3 - \epsilon} f(\mathbf{o}) \geq \frac{1-\epsilon}{3+\alpha} f(\mathbf{o}).$$

If $|I| > 1$, we assume that the algorithm continues the main loop until $|I| = 1$ **or** $|supp(\mathbf{s})| = \mathfrak{r}$. This is possible due to the *pairwise monotonicity* of the function f. Specifically, for each element e and the current solution $\mathbf{s}$, there exist two distinct labels $i, j \in [k]$ such that $\Delta_{e,i} f(\mathbf{s}) + \Delta_{e,j} f(\mathbf{s}) \geq 0$. Therefore, at least one label $i \in \{i, j\}$ satisfies $\Delta_{e,i} f(\mathbf{x}) > 0$, which guarantees that the algorithm can make progress in each iteration and thus continue the main loop until $|\mathbf{s}| = \mathfrak{r}$ or $|I| = 1$. Let $\mathbf{s}_T = (\langle e_1, i_1\rangle, \langle e_2, i_2\rangle, \ldots, \langle e_T, i_T\rangle), T > t$ as $\mathbf{s}$ in this case. By the similarity argument of **Case 1** and **Case 2** with $|I| = 1$, we have $f(\mathbf{s}_T) \geq \frac{1-\epsilon}{3+\alpha} f(\mathbf{o})$. We have

$$f(\mathbf{s}_T) - f(\mathbf{s}_t) \leq \sum_{e \in supp(\mathbf{s}_T) \setminus supp(\mathbf{s}_t)} \Delta_{e,\mathbf{s}_T(e)} f(\mathbf{s}_t) \overset{(d)}{\leq} \frac{2\epsilon M}{3\mathfrak{r}} \leq \frac{2\epsilon}{3} f(\mathbf{o})$$

where (d) holds since the marginal gain of each additional pair is smaller than the final threshold $2\epsilon M/(3\mathfrak{r})$. This implies that

$$f(\mathbf{s}_t) \geq f(\mathbf{s}_T) - \frac{2\epsilon}{3} f(\mathbf{o}) \geq (\frac{1}{4+\alpha} - \epsilon) f(\mathbf{o}).$$

Combining all cases, we obtain an approximation ratio of

$$\frac{1-\epsilon}{3+\alpha} - \epsilon \geq \frac{e-1}{4e-3} - \epsilon \approx \frac{1}{4.582} - \epsilon.$$

Prove for query complexity. The algorithm first finds the maximal singleton M within $O(n)$ queries. It then conducts $\log_{1-\epsilon}(\frac{2\epsilon}{3\mathfrak{r}})$ iterations of the while loop (Lines 2-10) in which each iteration takes $O(nk)$ queries. After the while loop, if $|I| = 1$, the algorithm calls algorithm for SMM requiring $O(\texttt{Alg}(\texttt{SMM}))$. The total queries required is at most

$$O(n) + O(nk \log_{1-\epsilon}(\frac{2\epsilon}{3\mathfrak{r}})) + O(\texttt{Alg}(\texttt{SMM})) \tag{35}$$

$$= O(n + \frac{nk}{\epsilon} \log(\mathfrak{r})) + O(\texttt{Alg}(\texttt{SMM})) \tag{36}$$

The algorithm first finds the maximal singleton M using $O(n)$ value-oracle queries. It then executes the while loop (Lines 2–10) for $\log_{1-\epsilon}\left(\frac{2\epsilon}{3\mathfrak{r}}\right)$ iterations, where each iteration requires $O(nk)$ queries. After the while loop, if $|I| = 1$, the algorithm invokes the subroutine for SMM, which requires $O(\texttt{Alg}(\texttt{SMM}))$ queries. Consequently, the total number of oracle queries is bounded by

$$O(n) + O\left(nk \log_{(1-\epsilon)}\left(\frac{2\epsilon}{3\mathfrak{r}}\right)\right) + O(\texttt{Alg}(\texttt{SMM})) = O\left(n + \frac{nk}{\epsilon}\log(\mathfrak{r})\right) + O(\texttt{Alg}(\texttt{SMM})). \tag{37}$$

The proof is completed. □

Theorem 2. For any $\epsilon > 0$, the following statements about Algorithm 2 hold.

- It returns a solution $\mathbf{s}$ $\mathbf{s}$ that satisfies $supp(\mathbf{s}) \in \mathcal{M}$ and $\lfloor \frac{l_i}{2} \rfloor \leq |(\mathbf{s})_i| \leq u_i$ for the FkSM problem.
- For non-monotone functions f, it attains an approximation ratio of $\frac{1}{21.976} - \frac{\epsilon}{4}$.
- For monotone functions f, the algorithm achieves an approximation ratio of $\frac{1}{9.164} - \frac{\epsilon}{2}$.
- Algorithm 1 runs in $O\left(\frac{nk}{\epsilon}\log \mathfrak{r}\right) + Q(\text{Algorithm for SMM})$, where $Q(\text{Algorithm for SMM})$ denotes the query complexity of the underlying SMM algorithm used as a subroutine.

Proof. **Proof of matroid feasibility.** In **Phase 2**, the algorithm initializes $\mathbf{y}^{(j)} \leftarrow \mathbf{x}^{(j)}$ and then refines the solution by selectively adding elements from $(\mathbf{u}^{(j)})_i$ to any component i satisfying $|(\mathbf{y}^{(j)})_i| < u_i$. This operation always preserves matroid feasibility. Indeed, since $\mathbf{x}^{(j)}$ is feasible in the induced matroid $\mathcal{M}_j$ and every added element belongs to $\mathbf{u}^{(j)}$, whose support satisfies $supp(\mathbf{u}^{(j)}) \cup supp(\mathbf{x}^{(j)}) \in \mathcal{M}$, it follows that the support of $\mathbf{y}^{(j)}$ remains independent in $\mathcal{M}$.

Prove fairness bounds. By construction, both $\mathbf{x}^{(j)}$ and $\mathbf{y}^{(j)}$ always satisfy the upper fairness constraints. Since $|L| \geq \sum_i l_i$ and $\mathbf{u}^{(j)}$ is constructed from $\mathbf{u}$ in Lines 3–4, we have $|(\mathbf{u}^{(j)})_i| \geq \lfloor l_i/2 \rfloor$ for all i. Moreover, as pairs from $\mathbf{u}^{(j)}$ are added to $\mathbf{y}^{(j)}$, it follows that $|(\mathbf{y}^{(j)})_i| \geq |(\mathbf{u}^{(j)})_i| \geq \lfloor l_i/2 \rfloor$.

Prove the query complexity. Algorithm 2 first computes the set L without making any value-oracle query to f, and then invokes Fair$k$Sub once in Line 2 and twice in Line 8, incurring a query complexity of $O!\left(n + \frac{nk}{\epsilon}\log(\mathfrak{r})\right) + O(\texttt{Alg}(\texttt{SMM}))$. It then refines the solution in Lines 9–12 using at most $2\sum_{i\in[k]} u_i$ additional queries. Therefore, the total number of queries required is at most

$$O\left(n + \frac{nk}{\epsilon}\log(\mathfrak{r})\right) + O(\texttt{Alg}(\texttt{SMM})) + 2\sum_{i\in[k]} u_i = O\left(n + \frac{nk}{\epsilon}\log(\mathfrak{r})\right) + O(\texttt{Alg}(\texttt{SMM}))$$

which completes the proof.

□

B Additional Experimental Details

B.1 Applications and Datasets

We begin by describing representative applications of the FkSM problem.

- **Max-k-Cut** (MkC): MkC [Frieze and Jerrum, 1997] is defined as follows: Given a graph $G = (V, E)$ and an integer $k \geq 2$, the goal is to partition the vertex set V into k disjoint subsets $V_1, \ldots, V_k$ such that the total weight of edges crossing between different partitions is maximized. Formally, the objective is to maximize $f(\mathbf{s}) = \sum_{(u,v)\in E,\, u\in V_i,\, v\in V_j,\, i\neq j} w_{uv}$, where w_{uv} denotes the weight of edge (u, v). This weighted formulation generalizes the cut-size (unweighted) Max-k-Cut: setting $w_{uv} \equiv 1$ for all $(u, v) \in E$ recovers the objective that counts the number of edges crossing different parts. The associated objective function $f(\cdot)$ used in our setting corresponds to a non-monotone k-submodular function [Spaeh *et al.*, 2025]. In our experiments, we assign a random edge weight $w_{uv} \sim \text{Unif}(0, 1)$ independently to each edge $(u, v) \in E$ for all MkC datasets. Weights are generated once per dataset and fixed across all algorithms. In this application, we use the Email, Amazon, and Twitch datasets.
- **k-topics Influence Maximization** (kTIM). We consider the k-topic independent cascade model on a directed graph $G = (V, E)$, as introduced by [Ohsaka and Yoshida, 2015]. For each topic $i \in [k]$ and edge $(u, v) \in E$, an activation probability $p^i_{u,v} \in [0, 1]$ is given. A seed assignment is $s \in (k + 1)^V$, where $s(v) = i$ means that v is chosen as a seed for topic i and $s(v) = 0$ otherwise; let $supp_i(s) = \{v \in V : s(v) = i\}$. For each topic i, the diffusion starts from $supp_i(s)$ and follows the standard independent cascade rule with probabilities $\{p^i_{u,v}\}$, independently across topics. Let $A_i(supp_i(s))$ be the (random) set of vertices activated by topic i. We define the multi-topic influence spread by

$$\sigma(\mathbf{s}) = \mathbb{E}\left[\left|\bigcup_{i\in[k]} A_i(supp_i(s))\right|\right].$$

It is known that σ is monotone k-submodular [Ohsaka and Yoshida, 2015]. Our objective is to find $s \in (k+1)^V$ maximizing $\sigma(s)$ subject to the matroid and fairness constraints defined below. We used Facebook dataset in this application.

- **k-measurements Sensor Placement** (kMSP). We consider the sensor placement problem studied in [Ohsaka and Yoshida, 2015]. There are k sensor types and a ground set V of candidate locations. A feasible deployment is represented by $(S_1, \ldots, S_k)$ with $S_i \subseteq V$, where at most one sensor can be installed at each location (i.e., the sets $S_1, \ldots, S_k$ are pairwise disjoint). When a type-i sensor is installed at location $e \in V$, it generates a random measurement $X_e^{(i)}$, and we define $\Omega = \{ X_e^{(i)} : e \in V,\ i \in [k] \}$. For any $A \subseteq \Omega$, the entropy is

$$H(A) = -\sum_{\mathbf{x} \in \mathrm{dom}(A)} \Pr[A = \mathbf{x}] \log \Pr[A = \mathbf{x}].$$

  Our objective is to maximize the entropy of the collected measurements,

$$f(S_1, \ldots, S_k) = H\Big(\{ X_e^{(i)} : i \in [k],\ e \in S_i \} \Big),$$

  which is monotone k-submodular [Ohsaka and Yoshida, 2015]. In our experiments, we use the Intel Lab dataset and follow the same evaluation protocol as [Ene and Nguyen, 2022].

Matroid setting. Across all datasets and applications, we impose a partition matroid constraint together with per-label fairness bounds. We parameterize the fairness constraint by a budget ratio $B \in (0, 1]$. Specifically, we set the cardinality budget to $b = \lfloor B|V| \rfloor$, and define the fairness bounds for each label $i \in [k]$ as

$$l_i = \left\lfloor 0.8\, \frac{b}{k} \right\rfloor, \qquad u_i = \left\lfloor 1.4\, \frac{b}{k} \right\rfloor, \qquad \forall i \in [k].$$

In all figures, B is reported on the horizontal axis; accordingly, B (through the induced bounds l_i and u_i) serves as the control parameter of the fairness constraint. Details of the datasets used in our experiments are summarized in Table 2. The column "#Matroid parts" in Table 2 denotes the number of parts p in the partition matroid. For Email, we use the part labels provided by the dataset. For the other datasets, we randomly assign vertices to p parts as evenly as possible (with part sizes differing by at most one vertex). We generate the partition once and keep it fixed across runs. The original Amazon graph contains $262{,}111$ vertices; following the common practice of extracting a representative subgraph, in our experiments we restrict to the top-5000 communities, resulting in a reduced instance with $16,716$ vertices.

Table 2: The datasets used in experiments

Database	**#Nodes**	**#Edges**	**#Matroid parts**	**Applications**
Email	1,005	25,571	42	MkC
Amazon	16,716	48,739	10	MkC
Twitch	168,114	6,797,557	10	MkC
Facebook	4,039	88,234	10	kTIM
Intel Lab sensors	55	-	3	kMSP

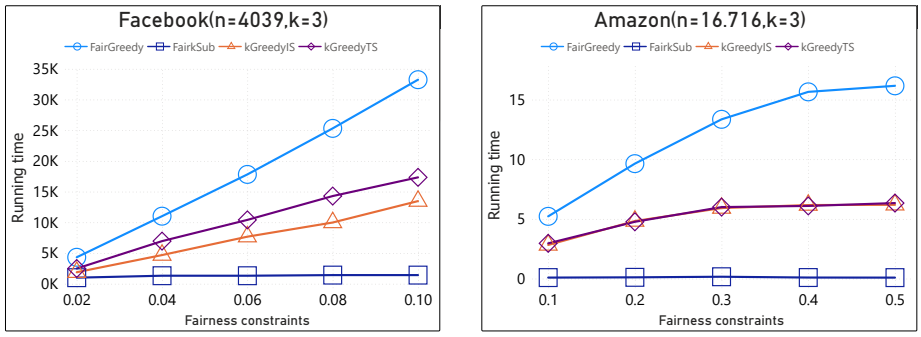
Other experimental settings. All methods were implemented in **C++17** and compiled with `g++` using `-O2`; for the kTIM objective, we enabled OpenMP (`-fopenmp`) to parallelize Monte Carlo simulations. All experiments were executed on our HPC cluster using the `small` partition with a single compute node, allocating **32 CPU cores** and **128 GB** memory. We report wall-clock running time measured by `std::chrono`, and we used the same objective oracle implementation and data structures across algorithms to ensure a fair runtime comparison. For all implemented algorithms, we set the parameter ϵ to 0.1 and $k = 3$.

For the subroutine solving submodular maximization under a matroid constraint (`SMM`) in `UFair`k`Sub` and `Fair`k`Sub`, the best-known approximation algorithms in [Cualinescu *et al.*, 2011; Buchbinder and Feldman, 2024] rely on the multilinear extension to estimate the objective function. However, their approximation guarantees are largely of theoretical interest and do not translate well into practical performance, as these algorithms incur a prohibitively large polynomial query complexity Poly(n) [Chen and Kuhnle, 2024]. Therefore, we employ a practical algorithm proposed in [Han *et al.*, 2020], which achieves a $1/4$-approximation ratio for non-monotone submodular functions and a $1/2$-approximation ratio for monotone submodular functions.

B.2 Experimental Results

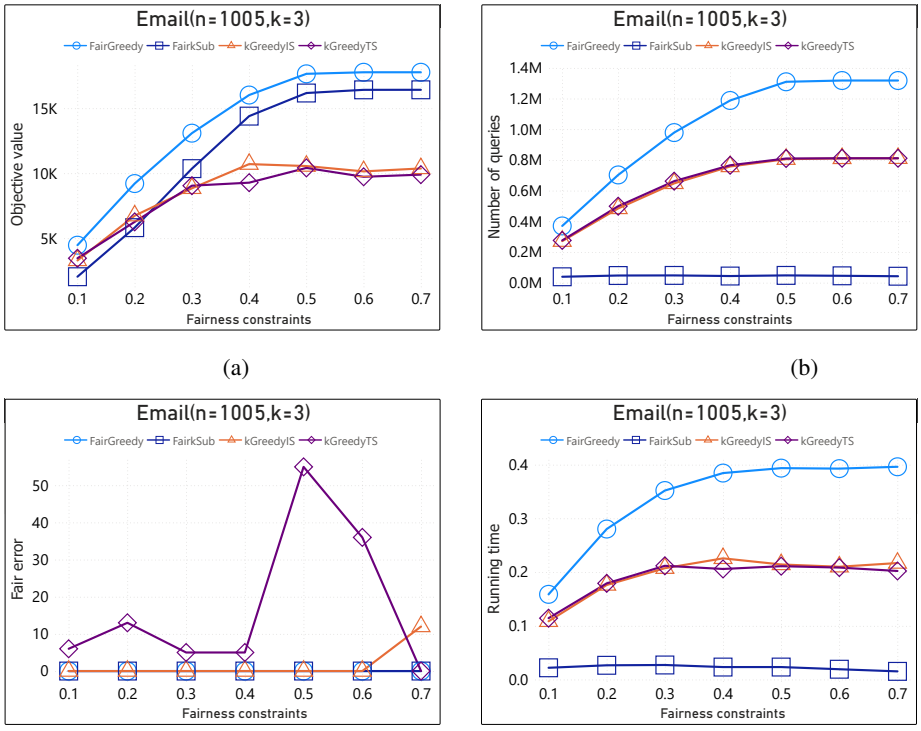
This subsection reports additional experimental results for the three applications, following the four evaluation criteria: objective value, number of queries, fairness error, and running time. Unless otherwise stated, the trends in running time are consistent with those of number of queries.

For the kTIM experiment on the Facebook dataset, the main paper (Fig. 1) already reports objective value, number of queries, and fairness error; here we only provide the additional running-time results (Fig. 2(a)). Overall, Fair$k$Sub is consistently faster than the baselines across all tested budgets (e.g., 996 seconds at $B = 0.02$ and 1415 seconds at $B = 0.10$), while FairGreedy, kGreedyIS, and kGreedyTS require substantially more time (e.g., $4317/1881/2487$ seconds at $B = 0.02$ and $33238/13469/17335$ seconds at $B = 0.10$, respectively). Similarly, for Amazon dataset, the main paper (Fig. 1(d,e,f)) already reports objective value, number of queries, and fairness error; Fig. 2(b) shows that Fair$k$Sub also enjoys a clear running-time advantage, completing in about 0.086–0.160 seconds over $B \in [0.1, 0.5]$, compared to 5.22–16.16 seconds for FairGreedy and about 2.82–6.33 seconds for kGreedyIS/kGreedyTS.



(a) (b)

Figure 2: Additional running-time results corresponding to Fig. 1: Facebook (kTIM, a) and Amazon (MkC, b).



(a) (b)

(c) (d)

Figure 3: Performance of the algorithms for the MkC application on the Email dataset.

MkC on Email. Fig. 3 summarizes the results on Email. In terms of objective value, FairGreedy achieves the best values across budgets, while Fair$k$Sub becomes increasingly competitive as B grows: the relative gap to FairGreedy decreases from

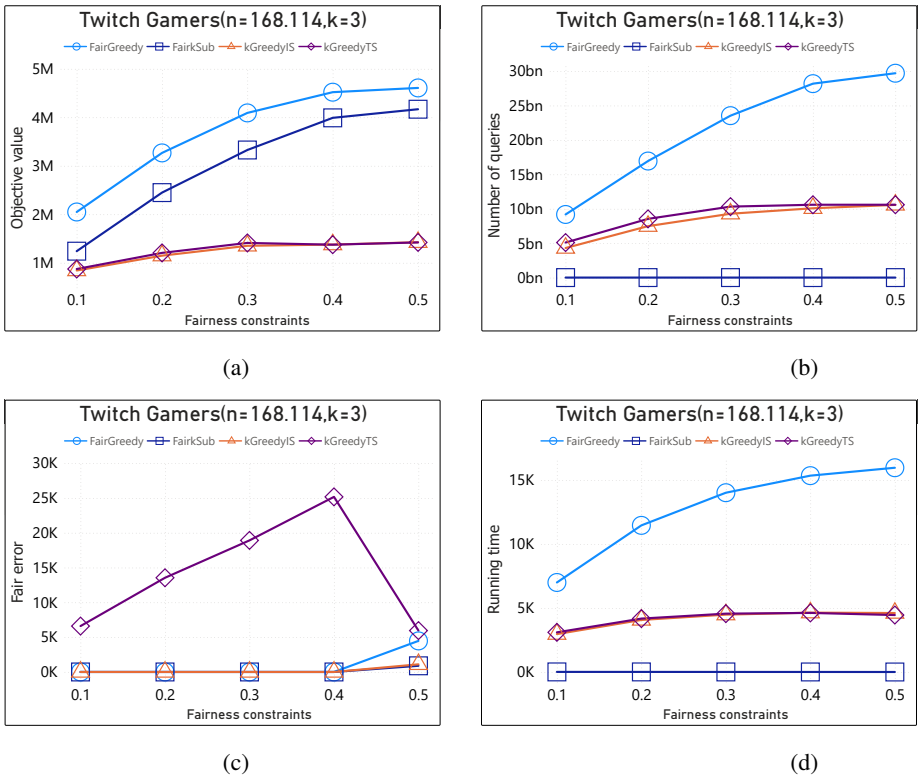


Figure 4: Performance of the algorithms for the MkC application on the Twitch dataset.

about 54% at $B = 0.1$ to around 7.6% at $B = 0.7$. Moreover, for moderate-to-large budgets ($B \geq 0.3$), Fair$k$Sub consistently outperforms kGreedyIS and kGreedyTS in objective value.

Regarding number of queries, Fair$k$Sub is significantly more query-efficient. Across all budgets, Fair$k$Sub uses about 3.9×10^4–4.8×10^4 queries, while FairGreedy requires about 3.7×10^5–1.3×10^6 queries. This corresponds to roughly 8–27 times fewer queries for Fair$k$Sub (and similarly large improvements over kGreedyIS/kGreedyTS). This is reflected in running time: Fair$k$Sub runs in about 0.015–0.027 seconds, compared with roughly 0.16–0.40 seconds for FairGreedy.

In terms of fairness, Fair$k$Sub attains zero violation for all tested budgets. In contrast, kGreedyTS exhibits noticeable violations at several budgets (e.g., up to 55 at $B = 0.5$), and kGreedyIS may also violate fairness at larger budgets (e.g., at $B = 0.7$).

MkC on Twitch. Fig. 4 reports results on the largest MkC instance (Twitch), where computational differences are most visible. For objective value, FairGreedy attains the highest values, while Fair$k$Sub remains competitive and clearly dominates kGreedyIS/kGreedyTS. For example, over $B \in [0.1, 0.5]$, Fair$k$Sub achieves about 1.24M–4.17M, which is typically much higher than kGreedyIS/kGreedyTS, while staying within a moderate gap to FairGreedy.

For the number of queries, Fair$k$Sub uses only about 1.56×10^7–1.71×10^7 queries, whereas the baselines require billions of queries. For example, FairGreedy needs about 9.2×10^9–3.0×10^{10} queries, meaning that Fair$k$Sub uses roughly 540–$1{,}900$ times fewer queries across all budgets (with similarly large gaps compared to the other baselines). Consequently, Fair$k$Sub achieves dramatic running-time improvements: it finishes in about 20–27 seconds, compared with thousands of seconds for the baselines (e.g., $\sim$ 7008–15966 seconds for FairGreedy), i.e., speedups on the order of 10^2–10^3.

Regarding fairness, Fair$k$Sub attains zero violation for $B \leq 0.4$ and shows a small violation at $B = 0.5$ (fair error $= 887$), which remains smaller than the violations of FairGreedy and kGreedyTS at the same budget.

kMSP on Intel Lab sensors. Fig. 5 presents results for sensor placement. In terms of objective value, Fair$k$Sub is consistently close to FairGreedy, with a small relative gap of roughly 0.6%–3.2% depending on B. kGreedyTS sometimes attains slightly higher objective values, but this comes with substantial fairness violations (e.g., fair error 6–15 across budgets), whereas Fair$k$Sub maintains zero violation for $B \leq 0.4$ and only a minimal violation at $B = 0.5$ (total error $= 1$). Regarding number of queries and running time, all methods are extremely fast on this small instance. Fair$k$Sub uses a moderate number of queries (about 1200–2235), which can be higher than kGreedyIS/kGreedyTS, but the corresponding running times remain negligible in absolute terms (on the order of 10^{-5} seconds). Overall, on kMSP, Fair$k$Sub provides a favorable trade-off between objective quality and fairness feasibility, while keeping computation lightweight.

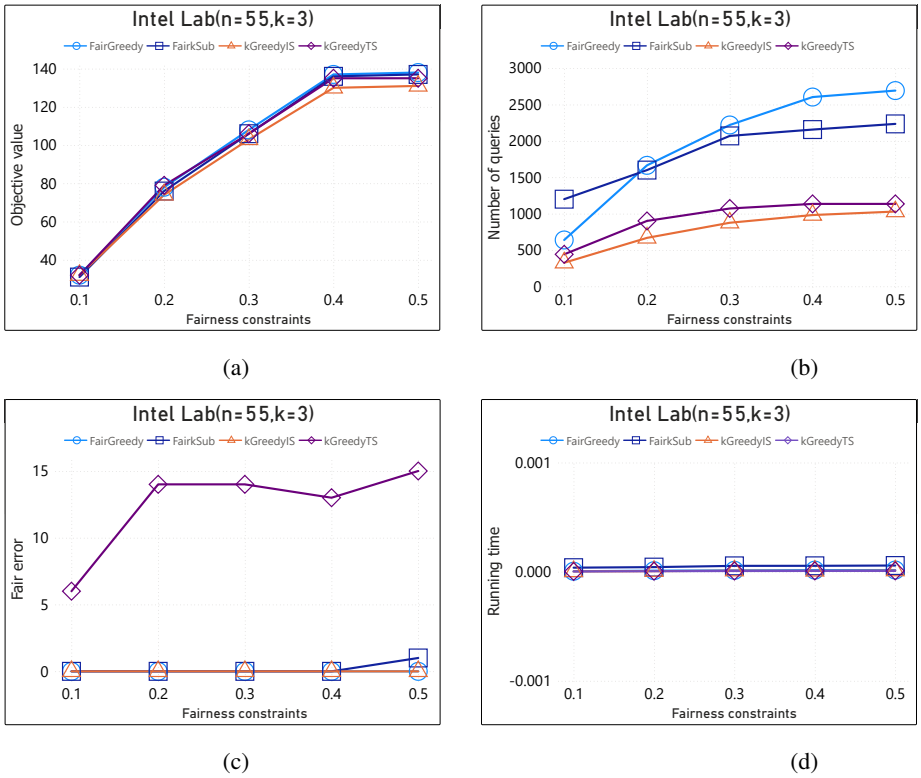


Figure 5: Performance of the algorithms for the kMSP application on the IntelLab dataset.