IMPROVED STREAMING FAIR SUBMODULAR MAXIMIZATION UNDER A KNAPSACK CONSTRAINT: APPLICATION TO BUDGETED FAIR INFLUENCE MAXIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Given a ground set N partitioned into groups $G_1, \ldots, G_h$, we seek a subset $S \subseteq N$ maximizing a monotone submodular function $f(S)$, subject to a knapsack constraint $\sum_{e \in S} w_e \leq B$ and fairness constraints $\ell_i \leq |S \cap G_i| \leq u_i$ for every group i, with particular focus on the streaming setting. This problem arises in many applications, such as budgeted fair influence maximization in social networks. We propose an operation-based reduction method which yields a $0.387-O(\varepsilon)$ approximation, improving the previous best guarantee 0.241 in Zhou et al. (2026). Moreover, we give the first constant-factor streaming approximations that satisfy both the knapsack constraint and all fairness constraints exactly. The previous best streaming result (Cui et al., 2024) achieves an approximation ratio of $1/12 - O(\varepsilon)$ while relaxing each fairness lower bound to $\lfloor \ell_i/2 \rfloor$. Our two-pass algorithm improves the ratio to $1/(7+\varepsilon)$ using $O(K)$ memory, where $K = \sum_i u_i$. We further propose a new technique called density-truncated local search, which achieves a $1/(5+\varepsilon)$ approximation using $O(1/\varepsilon)$ passes and $\widetilde{O}_\varepsilon(K)$ memory. Experiments on budgeted fair influence maximization over real-world social networks show that our algorithms maintain exact feasibility in all runs while achieving competitive influence spread. Additional evaluations show that our algorithms retain over 96% of the influence spread that can be achieved by a greedy method without considering fairness.

1 INTRODUCTION

We study monotone submodular maximization under a knapsack constraint and fairness constraints (FSMK), with particular focus on its streaming variant (S-FSMK). One important application is *budgeted fair influence maximization* (BFIM) in social networks. Classical influence maximization selects seed nodes to maximize expected influence spread (Kempe et al., 2003). Under the independent cascade and linear threshold models, the expected spread is monotone and submodular (Kempe et al., 2003). In practical influence campaigns, candidate seeds may have heterogeneous recruitment costs, leading to a knapsack constraint (Nguyen & Zheng, 2013; Bian et al., 2020), while fairness requirements are modeled by lower and upper group quotas on the number of selected seeds. These fairness constraints prevent the selected seeds from being overly concentrated in large or highly connected groups. Thus, BFIM is an instance of FSMK. In S-FSMK, candidate elements arrive sequentially in an arbitrary order, and the algorithm may make only a small number of passes over the stream while using limited working memory. Figure 1 illustrates this setting for BFIM: candidate seeds arrive one by one with their group labels and costs, and the algorithm selects a seed set under the budget and fairness constraints.

Existing results reveal a gap between streaming efficiency and exact feasibility. The closest streaming result, due to Cui et al. (2024), gives a two-pass $1/12 - O(\varepsilon)$ approximation using $O(K)$ memory, where $K = \sum_i u_i$, but relaxes each fairness lower bound from ℓ_i to $\lfloor \ell_i/2 \rfloor$. The main difficulty is that fairness lower bounds destroy downward closedness, a structural property used by many greedy and streaming algorithms for constrained submodular maximization. Under a downward-closed constraint, every subset of a feasible solution remains feasible, allowing solutions to be built

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

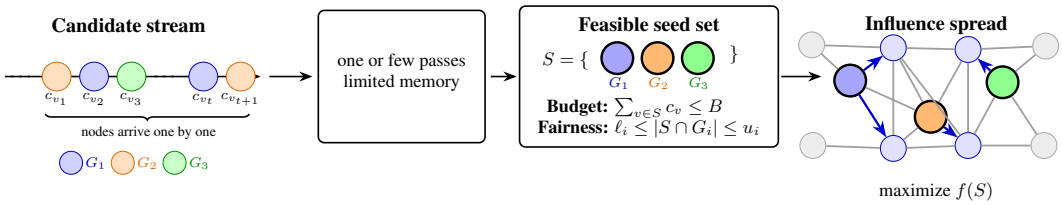


Figure 1: Streaming BFIM: candidate seeds arrive with group labels and costs; the selected seeds satisfy both the budget (knapsack) and group quotas (fairness). Diffusion takes place on a fixed network accessed through a value oracle.

incrementally. Fairness lower bounds break this property: early selections may leave insufficient budget to satisfy some groups, while fixing a fair base in advance may exclude better elements arriving later.

To address this gap, we develop an operation-based reduction that preserves exact feasibility and applies to an arbitrary number of groups. The reduction provides a common framework for both offline and streaming algorithms, leading to improved offline guarantees and the first constant-factor streaming guarantees that satisfy the knapsack constraint and all fairness constraints exactly. More broadly, the framework may provide a useful tool for other combinatorial optimization problems with fairness constraints.

Contributions. Our algorithmic contributions are threefold. First, we introduce a general operation-based framework that preserves exact feasibility. It yields a randomized $0.387 - O(\varepsilon)$ offline approximation, improving the previous best 0.241 guarantee (Zhou et al., 2026). Second, we obtain a two-pass $1/(7+\varepsilon)$ streaming approximation using $O(K)$ memory. This is the first constant-factor streaming guarantee that satisfies both the knapsack constraint and all fairness constraints exactly; the previous $1/12 - O(\varepsilon)$ streaming guarantee (Cui et al., 2024) relaxes each lower bound to $\lfloor \ell_i/2 \rfloor$. Third, we develop a new density-truncated local search method that further improves the approximation ratio to $1/(5+\varepsilon)$ using $O(1/\varepsilon)$ passes and $\widetilde{O}_\varepsilon(K)$ memory. The analysis uses a two-case argument: knapsack rejections serve as value certificates, whereas in their absence a blockwise charging argument bounds rejected optimal operations and establishes a multi-pass progress recurrence. Table 1 summarizes the theoretical guarantees. Experiments on real-world social networks show that our algorithms maintain exact feasibility in all runs while achieving competitive influence spread. Additional evaluations show that they retain over 96% of the influence spread achieved by a greedy method without fairness constraints.

Table 1: Guarantees for monotone fair submodular maximization under a knapsack. All rows satisfy the budget exactly. Here $\beta = 1 - 1/e - \varepsilon$; $\widetilde{O}_\varepsilon$ suppresses polylogarithmic and ε-dependent factors. Memory bounds treat ε as fixed unless shown explicitly.

Setting / work	Approximation	Groups	Passes / memory	Fairness
Offline: Li et al. (2025)	$\beta/2 \approx 0.316 - O(\varepsilon)$	Constant	–	Exact
Offline: Zhou et al. (2026)	$\frac{1-e^{-3}}{4-e^{-3}} \approx 0.241$	Any	–	Exact
Offline: this paper	$\frac{\beta}{1+\beta} \approx 0.387 - O(\varepsilon)$	Any	–	Exact
Streaming: Cui et al. (2024)	$1/12 - O(\varepsilon)$	Any	2 / $O(K)$	Relaxed[†]
Two-pass: this paper	$1/(7+\varepsilon)$	Any	2 / $O(K)$	Exact
Multi-pass: this paper	$1/(5+\varepsilon)$	Any	$O(1/\varepsilon)$ / $\widetilde{O}_\varepsilon(K)$	Exact

[†]The lower bounds are relaxed to $\lfloor \ell_i/2 \rfloor$; the upper bounds remain u_i.

Related work. Classical cardinality constrained submodular maximization admits the greedy $(1-1/e)$ approximation (Nemhauser et al., 1978); knapsack extensions account for heterogeneous costs (Sviridenko, 2004; Kulik et al., 2013). Streaming cardinality algorithms achieve $1/2 - \varepsilon$ with memory linear in the cardinality bound up to accuracy factors (Badanidiyuru et al., 2014; Kazemi

et al., 2019). For matroids, one-pass and multi-pass methods offer progressively stronger guarantees (Chakrabarti & Kale, 2015; Huang et al., 2020; Feldman et al., 2026). Fair streaming optimization has been studied under cardinality and matroid constraints (El Halabi et al., 2020; 2023), with related offline matroid results in El Halabi et al. (2024).

Most closely related to our setting, Cui et al. (2024) give a two-pass $1/12 - O(\varepsilon)$ approximation with relaxed lower bounds. Offline, Li et al. (2025) obtain $\beta/2$ with exact feasibility for a constant number of groups, where $\beta = 1 - 1/e - \varepsilon$. Zhou et al. (2026) obtain $(1-e^{-3})/(4-e^{-3})$ with exact feasibility for arbitrary groups. Our reduction improves the latter offline guarantee while preserving exact feasibility and gives stronger streaming guarantees without relaxing the lower bounds. Fair influence maximization has also been studied with fairness defined over diffusion outcomes (Tsang et al., 2019; Fish et al., 2019; Stoica et al., 2020; Ali et al., 2021; Becker et al., 2022); in contrast, our fairness constraints govern the composition of the selected seed set. Appendix D provides further context and additional application illustrations.

2 Problem and Preliminaries

Let N be partitioned into groups $G_1, \ldots, G_h$. Each $e \in N$ has a nonnegative weight $w(e)$, and $w(S) = \sum_{e \in S} w(e)$. A nonnegative function $f : 2^N \to \mathbb{R}_{\geq 0}$ is monotone if $f(A) \leq f(B)$ whenever $A \subseteq B$, and submodular if $f(e \mid A) \geq f(e \mid B)$ for $A \subseteq B$ and $e \notin B$. We write $f(T \mid S) = f(S \cup T) - f(S)$, with $f(e \mid S)$ for a singleton, and assume value-oracle access to f. Normalization means $f(\varnothing) = 0$.

Definition 2.1 (FSMK and its streaming version)**.** Given a budget $B \geq 0$ and integer bounds $0 \leq \ell_i \leq u_i \leq |G_i|$, $\forall i \in [h]$, fair submodular maximization with a knapsack (FSMK) asks for

$$\max_{S \subseteq N} f(S) \quad \text{subject to} \quad w(S) \leq B, \qquad \ell_i \leq |S \cap G_i| \leq u_i \quad (i \in [h]). \tag{2.1}$$

In the streaming version (S-FSMK), the elements of N arrive sequentially in an arbitrary order. The algorithm accesses the elements only in stream order, using limited memory, and may make one or more passes over the stream. After the final pass, it must output a set S satisfying the knapsack and fairness constraints in (2.1).

Let O be an optimal feasible set of definition 2.1, $\mathrm{opt} = f(O)$, $K = \sum_i u_i$, and $L = \max_i \ell_i$. A partition matroid divides its ground set into disjoint blocks and limits the number selected from each block. Its rank is at most the sum of these capacities. Groups with $u_i = 0$ can be discarded; if $K = 0$, the empty set is the only feasible solution. We assume $K \geq 1$ below.

Influence maximization specialization. For BFIM, take $N = V$, $w(v) = c_v$, and $f(S)$ equal to expected influence spread on a fixed social network. The stream contains candidate seeds, not graph edges. Under exact value-oracle access to the expected influence spread, our approximation guarantees apply directly to the independent cascade and linear threshold models. In our experiments, we estimate influence using reverse-reachable sets (Borgs et al. (2014); Tang et al. (2014; 2015)).

We use the following two existing subroutines, with fixed accuracy parameters.

Lemma 2.1 (Offline subroutine (Chekuri et al., 2010))**.** *For $0 < \varepsilon < 1 - 1/e$, $\mathsf{MatKnap}(\varepsilon)$ is a randomized polynomial-time β-approximation for monotone submodular maximization under one partition matroid and one knapsack, where $\beta = 1-1/e-\varepsilon$. The objective guarantee is in expectation and the returned set is feasible.*

Lemma 2.2 (Streaming subroutine (Cui et al., 2024))**.** *For fixed $\eta > 0$, $\mathsf{MatKnapStream}(\eta)$ gives a one-pass $1/[6(1+\eta)]$ approximation under one partition matroid and one knapsack, using $O(K)$ memory and $O(K)$ value-oracle queries per stream element, where K bounds the sum of the block capacities.*

3 An Approximation Framework for FSMK

3.1 Encoding feasible operations

For each group G_i, let $C_i \subseteq G_i$ be a minimum-weight subset of size ℓ_i, and let $C = \bigcup_i C_i$. Equivalently, C_i consists of the ℓ_i lightest elements of G_i. Since every feasible solution must select

at least ℓ_i elements from G_i, its weight within G_i is at least $w(C_i)$. Hence, C is the minimum-weight set satisfying all fairness lower bounds. Moreover, $|C_i| = \ell_i \le u_i$ for every i, so C also satisfies the fairness upper bounds. Therefore, the instance is feasible if and only if $w(C) \le B$. Assuming feasibility, we define the residual budget $R = B - w(C) \ge 0$.

Starting from the base set C, we represent all remaining choices by two types of operations. For each $e \in G_i \setminus C_i$, an *addition* a_e inserts e into the current solution, while a *replacement* $r_{c,e}$, for $c \in C_i$, replaces the base element c by e. Their incremental weights are

$$\widetilde{w}(a_e) = w(e), \qquad \widetilde{w}(r_{c,e}) = w(e) - w(c) \ge 0. \tag{3.1}$$

The replacement weight is nonnegative because every element of C_i is no heavier than any element of $G_i \setminus C_i$. Let Ω denote the set of all addition and replacement operations.

For an operation $o \in \Omega$, let $e(o)$ denote the element introduced by o. For $M \subseteq \Omega$, define $A(M) = \{e(o) : o \in M\}$ as the set of introduced elements, and $D(M) = \{c : r_{c,e} \in M \text{ for some } e\}$ as the set of base elements removed by replacements. For an operation set in which each original element is introduced at most once, the corresponding solution is

$$S(M) = (C \setminus D(M)) \cup A(M). \tag{3.2}$$

We later show that duplicate operations introducing the same element can be removed without loss. A concrete example illustrating the operations and the decoding rule is given in Appendix A.1.

It remains to encode the fairness upper bounds. For each $c \in C_i$, all replacements of c form a block $\Omega_c = \{r_{c,e} : e \in G_i \setminus C_i\}$, from which at most one operation may be selected. This ensures that each base element is replaced at most once. In addition, all additions in group G_i form a block $\Omega_i^{\mathrm{add}} = \{a_e : e \in G_i \setminus C_i\}$ with capacity $u_i - \ell_i$, since the base C_i already contains ℓ_i elements and at most $u_i - \ell_i$ further elements may be added. These disjoint blocks define the partition matroid

$$\mathcal{I}_{\mathrm{par}} = \left\{ M \subseteq \Omega : |M \cap \Omega_c| \le 1 \ (c \in C), \quad |M \cap \Omega_i^{\mathrm{add}}| \le u_i - \ell_i \ (i \in [h]) \right\}. \tag{3.3}$$

The replacement blocks ensure that each base element is removed at most once, while the addition blocks enforce the fairness upper bounds. The sum of the block capacities is $|C| + \sum_i (u_i - \ell_i) = K$. Details are provided in Appendix A.2.

We now define the auxiliary optimization problem over the operation set Ω. For $M \subseteq \Omega$, define

$$h(M) = f(C \cup A(M)) - f(C). \tag{3.4}$$

Thus, $h(M)$ measures the gain of the introduced elements relative to the base set C. We use $C \cup A(M)$ instead of the decoded set $S(M)$ because replacements may remove elements from C; this definition keeps h monotone and submodular.

The auxiliary problem is

$$\max_{M \subseteq \Omega} \ h(M) \qquad \text{s.t.} \qquad M \in \mathcal{I}_{\mathrm{par}}, \quad \widetilde{w}(M) \le R. \tag{3.5}$$

We denote its optimal value by $\mathrm{OPT}_{\mathrm{aux}}$.

We state the two structural properties needed below and defer their proofs to Appendix A.

Lemma 3.1. *The auxiliary objective h is normalized, monotone, and submodular.*

Lemma 3.2 (Duplicate pruning)**.** *Every feasible auxiliary set M has a subset $\overline{M}$ that introduces each original element at most once, with $h(\overline{M}) = h(M)$ and $\widetilde{w}(\overline{M}) \le \widetilde{w}(M)$.*

Therefore, the auxiliary problem is monotone submodular maximization subject to one partition matroid and one knapsack constraint, so we can apply $\mathsf{MatKnap}(\varepsilon)$. The complete offline procedure is summarized in Algorithm 1.

3.2 Analysis of the Reduction

The following three properties connect the auxiliary problem to FSMK. We include their key arguments so the approximation transfer can be read without the appendix; full proofs are in Appendix A.

Lemma 3.3 (Exact feasibility)**.** *For every pruned feasible auxiliary set $\overline{M}$, the set $S(\overline{M})$ satisfies all fairness bounds and the knapsack constraint.*

Algorithm 1 Operation reduction with an auxiliary solver

1: For each group G_i, compute a minimum-weight set C_i of size ℓ_i, and set $C \leftarrow \bigcup_i C_i$.
2: **if** $w(C) > B$ **then**
3: **return** infeasible.
4: **end if**
5: Set $R \leftarrow B - w(C)$ and construct the auxiliary problem (3.5).
6: Apply $\mathsf{MatKnap}(\varepsilon)$ to obtain M, and prune duplicates to obtain $\overline{M}$.
7: Decode $S(\overline{M}) \leftarrow (C \setminus D(\overline{M})) \cup A(\overline{M})$.
8: **return** $\widehat{S} \in \arg\max_{T \in \{C, S(\overline{M})\}} f(T)$.

Indeed, distinct additions and distinct replaced base elements give

$$|S(\overline{M}) \cap G_i| = \ell_i + |\overline{M} \cap \Omega_i^{\mathrm{add}}|, \qquad w(S(\overline{M})) = w(C) + \widetilde{w}(\overline{M}) \le B. \tag{3.6}$$

Lemma 3.4 (Optimal benchmark)**.** *For an optimal feasible solution O to FSMK, there exists a feasible auxiliary set M_O such that $S(M_O) = O$ and $h(M_O) \ge f(O) - f(C)$.*

Lemma 3.5 (Value transfer)**.** *For any feasible auxiliary set M, let $\overline{M}$ be the pruned set given by Lemma 3.2. Then $f(S(\overline{M})) \ge h(M)$.*

Since $S(\overline{M}) \cup C = C \cup A(\overline{M})$, submodularity and nonnegativity imply $f(S(\overline{M})) + f(C) \ge f(C \cup A(\overline{M}))$. Subtracting $f(C)$ and applying duplicate pruning proves the lemma.

Approximation transfer. If an auxiliary routine returns $h(M) \ge \alpha \mathrm{OPT}_{\mathrm{aux}}$, decode it and return the better of C and $S(\overline{M})$. The preceding lemmas give

$$f(\widehat{S}) \ge \max\{f(C), \alpha(f(O) - f(C))\} \ge \frac{\alpha}{1+\alpha} f(O). \tag{3.7}$$

For a randomized routine with an expected guarantee, the same conclusion holds in expectation: $\mathbb{E}[f(\widehat{S})] \ge \max\{f(C), \mathbb{E}[h(M)]\}$. Feasibility holds for every returned set, independently of this expectation.

Theorem 3.1. *For fixed $0 < \varepsilon < 1 - 1/e$, Algorithm 1 is a randomized polynomial time algorithm that satisfies the knapsack constraint and all fairness constraints exactly, and achieves $\mathbb{E}[f(\widehat{S})] \ge \frac{\beta}{1+\beta} f(O)$, where $\beta = 1 - 1/e - \varepsilon$.*

The theorem follows from Lemma 2.1 and (3.7). There are at most $|N|(L+1)$ operations, so the explicit auxiliary instance has polynomial size. The first-pass construction of C and the decoding step are shared by both streaming algorithms below.

4 Two-Pass Streaming Algorithm

The reduction need not materialize Ω. In the first pass, we maintain the ℓ_i lightest elements of each group to construct C. In the second pass, each $e \in G_i \setminus C_i$ generates one addition and ℓ_i replacement operations, which are processed immediately by $\mathsf{MatKnapStream}(\varepsilon/6)$. Queries to h are evaluated via (3.5), so the operation stream is never stored explicitly.

Theorem 4.1. *For fixed $\varepsilon > 0$, Algorithm 2 uses two passes, satisfies all constraints exactly, and returns $f(\widehat{S}) \ge f(O)/(7+\varepsilon)$.*

Proof. Lemma 2.2 gives $h(M) \ge \mathrm{OPT}_{\mathrm{aux}}/(6+\varepsilon)$. Applying (3.7) with $\alpha = 1/(6+\varepsilon)$ gives the ratio; Lemma 3.3 gives exact feasibility. □

Lemma 4.1. *For fixed $\varepsilon > 0$, the algorithm uses $O(K)$ memory and $O(K(L+1))$ value-oracle queries per original stream element.*

The first pass stores at most $|C| \le K$ elements. In the second pass, $\mathsf{MatKnapStream}$ uses $O(K)$ memory because the auxiliary partition matroid has total capacity K. Each $e \in G_i \setminus C_i$ generates one addition and ℓ_i replacement operations, and hence at most $L+1$ operations. Since each operation requires $O(K)$ value-oracle queries, the total is $O(K(L+1))$ queries per original stream element.

Algorithm 2 Two-pass streaming algorithm for S-FSMK

1: **Pass 1:** construct the base set C as in Section 3.
2: **if** the lower bounds cannot be satisfied or $w(C) > B$ **then**
3: **return** infeasible.
4: **end if**
5: Set $R \leftarrow B - w(C)$ and $\eta \leftarrow \varepsilon/6$.
6: **Pass 2:** generate the addition and replacement operations associated with each arriving element and process them using $\mathsf{MatKnapStream}(\eta)$ with budget R.
7: Let M be the operation set returned by $\mathsf{MatKnapStream}(\eta)$.
8: Obtain $\overline{M}$ from M using Lemma 3.2.
9: Set $S_{\mathrm{str}} \leftarrow (C \setminus D(\overline{M})) \cup A(\overline{M})$.
10: **return** $\widehat{S} \in \arg\max_{T \in \{C, S_{\mathrm{str}}\}} f(T)$.

5 A Density-Truncated Local Search Algorithm

The two-pass algorithm relies on an existing auxiliary solver. We now exploit the partition structure directly and develop a density-truncated local search. When a block is full, a new operation may replace the currently selected operation with the smallest incremental value, provided that it gives sufficient improvement and value per unit weight. Repeating this procedure over several passes leads to a stronger approximation guarantee.

5.1 Update rule and threshold selection

Let $\mathcal{P}$ denote the blocks of the partition matroid, and let $P(o)$ and $b(o)$ denote the block containing operation o and its capacity, respectively. For a fixed threshold τ, pass t starts from M_{t-1}. Define an order $\prec_t$ by placing the operations in M_{t-1} first, preserving their previous relative order, followed by all other operations in their generation order. For $o \in Q$, define

$$Q_t^{\prec o} = \{x \in Q : x \prec_t o\}, \qquad \nu_t(o, Q) = h(o \mid Q_t^{\prec o}). \tag{5.1}$$

These incremental values satisfy $\sum_{o \in Q} \nu_t(o, Q) = h(Q)$. The order $\prec_t$ is used only to define these values and does not require an additional scan. A simple example is given in Appendix A.11.

During pass t, operations in the initial set M_{t-1} are skipped when they reappear in the stream, even if they have been removed earlier in the pass. Operations with $b(o) = 0$ or $\widetilde{w}(o) > R$ are also ignored. Hence, every newly processed operation appears after all operations currently maintained in M under $\prec_t$.

For an incoming operation o, let $\Delta_o = h(o \mid M)$. If $P(o)$ is not full, set $X_o = \varnothing$. Otherwise, remove the operation with the smallest incremental value in the same block:

$$x_o \in \arg\min_{x \in M \cap P(o)} \nu_t(x, M), \qquad X_o = \{x_o\}.$$

Given a density threshold τ and an improvement parameter $\beta_t > 0$, we decide whether to insert the incoming operation o into the current set M. Recall that X_o is the operation removed from the same block when the block is full, and $X_o = \varnothing$ otherwise. The parameter β_t controls the required improvement over the removed operation, while τ sets a minimum marginal value per unit weight. We accept the update $M \leftarrow (M \setminus X_o) \cup \{o\}$ if

$$\Delta_o \geq (1 + \beta_t) \sum_{x \in X_o} \nu_t(x, M), \tag{5.2}$$

$$\Delta_o \geq \tau \widetilde{w}(o), \tag{5.3}$$

$$\widetilde{w}((M \setminus X_o) \cup \{o\}) \leq R. \tag{5.4}$$

The first condition requires sufficient gain relative to the operation being removed, the second enforces the density threshold, and the third preserves the residual knapsack constraint. The complete $\mathsf{DensityLocalSearchPass}$ procedure is given in Appendix B.1.

The choice of τ is tied to the scale of the auxiliary optimum. Our analysis requires a threshold close to $\mathrm{OPT}_{\mathrm{aux}}/(2R)$, which is unknown. We estimate this scale using the best feasible singleton

Algorithm 3 MultiPassSFMK(ε)

Pass 1: construct C and R; return infeasible if $w(C) > B$.
Pass 2: find the best feasible singleton $o^\star$ and set $m \leftarrow h(\{o^\star\})$; return C if $m = 0$.
Construct the candidate thresholds $\{\tau_j : j \in \mathcal{J}\}$ as described in Appendix B.2.
Set $d \leftarrow \lceil 4/\varepsilon \rceil$ and $M_0^{(j)} \leftarrow \varnothing$ for all $j \in \mathcal{J}$.
for $t = 1, \ldots, d$ **do**
    In one stream pass, run DensityLocalSearchPass$(\tau_j, 1/t, M_{t-1}^{(j)})$ for all $j \in \mathcal{J}$.
end for
Let M_{out} maximize h over $\{M_d^{(j)} : j \in \mathcal{J}\} \cup \{\{o^\star\}\}$, and prune it to $\overline{M}_{\text{out}}$.
Decode $S_{\text{out}} \leftarrow (C \setminus D(\overline{M}_{\text{out}})) \cup A(\overline{M}_{\text{out}})$.
return $\widehat{S} \in \arg\max_{T \in \{C, S_{\text{out}}\}} f(T)$.

and consider a geometric set of candidate thresholds. All candidates are processed in parallel over the same stream scans. The construction of the thresholds and the boundary cases are given in Appendix B.2. The complete procedure is summarized in Algorithm 3. It first constructs the base set and estimates the scale of the auxiliary optimum, then runs the DensityLocalSearchPass for all candidate thresholds in parallel, and finally returns the best decoded solution.

5.2 Approximation analysis

Fix a threshold τ and suppress its index. The proof has two cases, depending on whether a candidate operation passes the local-improvement and density tests but fails the budget test. We call such an operation *budget rejected*. Individually infeasible operations discarded before the tests are not budget rejected. Complete proofs of the results in this section are given in Appendix A.

Lemma 5.1 (Incremental values)**.** *For $Q \subseteq W$, (1) $\sum_{o \in Q} \nu_t(o, Q) = h(Q)$, (2) $\nu_t(o, W) \leq \nu_t(o, Q)$ for $o \in Q$, and (3) $h(Q \mid W \setminus Q) \leq \sum_{o \in Q} \nu_t(o, W)$.*

Lemma 5.2 (Maintained invariants)**.** *Every maintained M is feasible, $h(M)$ never decreases, and $\nu_t(x, M) \geq \tau \widetilde{w}(x)$ for each $x \in M$. Consequently, $h(M) \geq \tau \widetilde{w}(M)$.*

An accepted exchange loses at most the incremental value of its removed operation and gains at least $(1 + \beta_t)$ times that value. The inserted operation comes last in the current order and satisfies the density test. Removing an earlier operation can only increase the incremental values of survivors. Preserving their relative order between passes preserves the invariant. Appendix A.13 supplies the complete induction.

Lemma 5.3 (Budget rejection)**.** *If any operation is budget rejected, then* $\max\{h(M_d), h(\{o^\star\})\} \geq \tau R/2$.

At rejection, nonnegative costs imply $\widetilde{w}(M) + \widetilde{w}(o) > R$. The invariant and density test give $h(M) \geq \tau \widetilde{w}(M)$ and $h(\{o\}) \geq \tau \widetilde{w}(o)$. Their maximum is at least $\tau R/2$; subsequent improvements and the best singleton preserve this bound.

Lemma 5.4 (One-pass progress)**.** *If pass t contains no budget rejection, then*

$$\mathrm{OPT}_{\text{aux}} - \tau R \leq \frac{h(M_t) - h(M_{t-1})}{\beta_t} + (2 + \beta_t) h(M_t). \tag{5.5}$$

The proof separates optimal operations according to how they are handled during the pass. Density failures contribute at most τR, rejected operations are charged blockwise to the final solution, and removed operations are charged to the objective increase during the pass. Combining these bounds gives (5.5). The complete proof is given in Appendix A.15.

Lemma 5.5 (Multi-pass recurrence)**.** *If no budget rejection occurs in the first t passes with $\beta_t = 1/t$, then* $\mathrm{OPT}_{\text{aux}} - \tau R \leq 2(1 + 1/t) h(M_t)$.

For $t = 1$, (5.5) gives a factor of four. For the induction step, combine the previous bound with (5.5) and write $\theta = h(M_{t-1})/h(M_t)$. The resulting coefficient is at most $\min\{2t\theta/(t-1),\ t(1-\theta) +$

$2+1/t\} \le 2(1+1/t)$. The complete proof handles zero values separately in Appendix A.16. Thus, after $d = \lceil 4/\varepsilon \rceil$ passes without a budget rejection,

$$h(M_d) \ge \frac{\mathrm{OPT}_{\mathrm{aux}} - \tau R}{2 + \varepsilon/2}. \tag{5.6}$$

Theorem 5.1 (Auxiliary approximation). *For $0 < \varepsilon \le 1$, if $m = 0$, then $\mathrm{OPT}_{\mathrm{aux}} = 0$; otherwise Algorithm 3 selects a feasible M_{out} with $h(M_{\mathrm{out}}) \ge \mathrm{OPT}_{\mathrm{aux}}/(4+\varepsilon)$.*

Proof. For $R > 0$, choose $j^\star$ from (B.2). With no budget rejection, (5.6) gives $h(M_d^{(j^\star)}) \ge (\mathrm{OPT}_{\mathrm{aux}}/2)/(2+\varepsilon/2)$. With a budget rejection, Lemma 5.3 gives a candidate of value at least $\tau_{j^\star} R/2 \ge \mathrm{OPT}_{\mathrm{aux}}/(4+\varepsilon)$. When $R = 0$, all processed operations have zero weight, so no budget rejection occurs and (5.6) with $\tau = 0$ gives the stronger factor $1/(2+\varepsilon/2)$. The case $m = 0$ follows from submodularity, as above. □

Theorem 5.2. *For $0 < \varepsilon \le 1$, Algorithm 3 returns an exactly feasible set with $f(\widehat{S}) \ge f(O)/(5+\varepsilon)$.*

Apply (3.7) with $\alpha = 1/(4+\varepsilon)$. If $m = 0$, the returned base is optimal. Thus the approximation transfer is identical to the two-pass case; the stronger guarantee comes entirely from the improved auxiliary solver.

Theorem 5.3. *The algorithm uses at most $2+\lceil 4/\varepsilon \rceil$ passes, $O(K\log(K+1)/\varepsilon) = \widetilde{O}_\varepsilon(K)$ memory, and $O(K(L+1)\log(K+1)/\varepsilon^2)$ value-oracle queries per original stream element over all passes.*

6 Experiments

We evaluate our two streaming algorithms on four public social networks and additional coverage benchmarks. We report normalized influence $\widehat{f}(S)/|V|$, estimated using reverse-reachable (RR) sets under the independent-cascade model (Borgs et al., 2014; Tang et al., 2014), with separate RR samples for selection and evaluation. Detailed experimental settings, dataset sources, and additional results on fairness violations, runtime and oracle calls, are provided in Appendix C.

Exact feasibility. A set is exactly feasible if $w(S) \le B$ and $\ell_i \le |S \cap G_i| \le u_i$ for every group. On Epinions (real data) and Pokec (real data), we use five budgets and five stream orders shared by all methods, giving 50 runs per method. Figure 2 shows that TwoPass and MultiPass are exactly feasible in all 50 runs. The released FairKnapStream method of Cui et al. (2024) is exactly feasible in 6 runs. SmkStream (Han et al., 2021) and RandomStream, which enforce only the budget, are exactly feasible in none of the runs. All 250 outputs satisfy the budget; every infeasible output therefore violates the fairness constraints.

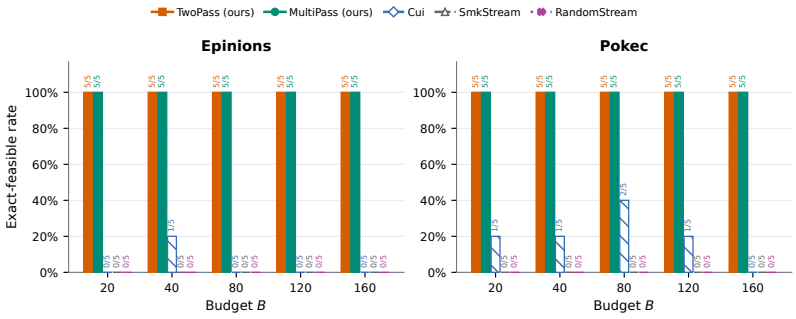


Figure 2: Exact-feasibility rate on Epinions and Pokec over five shared stream orders per budget.

Utility. We compare TwoPass and MultiPass with Cui+ExactRepair and two exactly feasible heuristics, FairBase+Density and FairBase+1Swap. Both methods achieve positive paired coverage gains against all three comparators, with all six 95% confidence intervals lying strictly above zero. The gains are particularly pronounced over FairBase+Density, while consistent improvements are also observed over Cui+ExactRepair and FairBase+1Swap. Complete results including the memory comparison are reported in Appendix C.

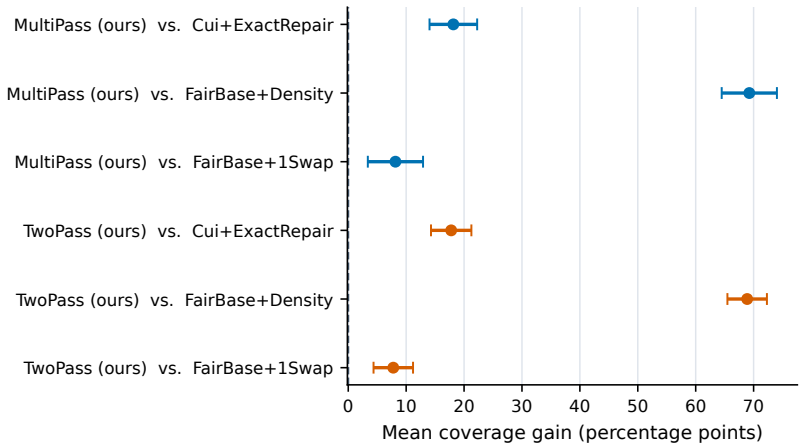


Figure 3: Paired coverage gains over three baselines. Bars show 95% confidence intervals over five shared stream orders.

Influence retained under group quotas. We compare our methods with a deterministic budget-only heuristic that repeatedly selects the node with the largest marginal influence gain per unit cost. On LastFM Asia and Twitch EN, seed budgets are 1–3% of the network size and every edge has IC probability 0.1. TwoPass retains 96.85–99.30% of the heuristic's total reach, while MultiPass retains 98.92–100% (Figure 4).

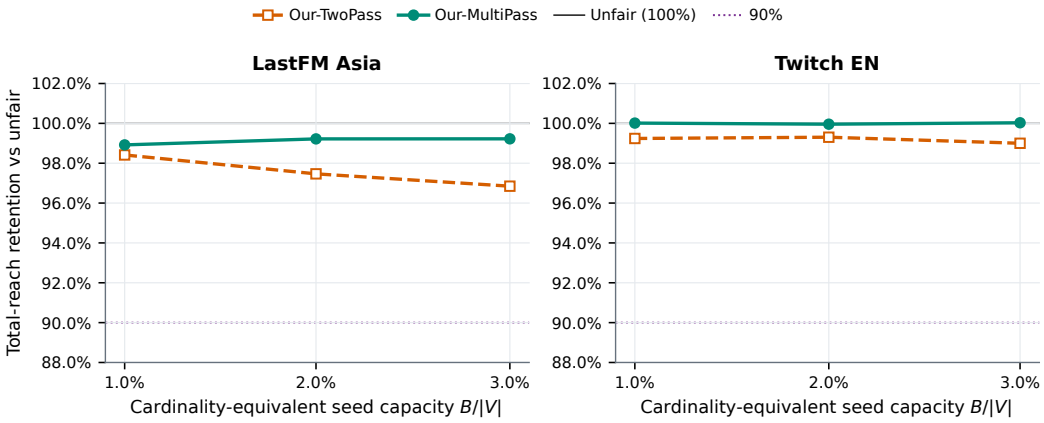


Figure 4: Reach relative to the budget-only density-greedy heuristic on LastFM Asia and Twitch EN at 1–3% seed budgets; lines mark 90% and 100% retention.

7 Conclusion

We studied monotone submodular maximization under a knapsack constraint and fairness constraints for an arbitrary number of groups, with particular focus on streaming. Our operation-based framework preserves exact feasibility, improves the best known offline approximation from 0.241 to $0.387 - O(\varepsilon)$, and yields the first constant-factor streaming guarantees satisfying all constraints exactly: a two-pass $1/(7+\varepsilon)$ approximation and a $1/(5+\varepsilon)$ approximation using $O(1/\varepsilon)$ passes. Experiments on budgeted fair influence maximization show that our algorithms are exactly feasible in all runs and retain over 96% of the influence spread achieved by a greedy method without fairness constraints. A promising direction for future work is to achieve stronger approximation guarantees while preserving exact feasibility in the streaming setting.

AI USE STATEMENT

We used ChatGPT (OpenAI) to assist with wording, literature searches, and code checking.

REPRODUCIBILITY STATEMENT

To facilitate reproducibility, we provide an anonymized implementation of the proposed algorithms and the experimental code at https://anonymous.4open.science/r/iclr-code-E1C4/.

REFERENCES

Junaid Ali, Mahmoudreza Babaei, Abhijnan Chakraborty, Baharan Mirzasoleiman, Krishna P Gummadi, and Adish Singla. On the fairness of time-critical influence maximization in social networks. *IEEE Transactions on Knowledge and Data Engineering*, 35(3):2875–2886, 2021.

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

Ruben Becker, Gianlorenzo D'angelo, Sajjad Ghobadi, and Hugo Gilbert. Fairness in influence maximization through randomization. *Journal of Artificial Intelligence Research*, 73:1251–1283, 2022.

Song Bian, Qintian Guo, Sibo Wang, and Jeffrey Xu Yu. Efficient algorithms for budgeted influence maximization on massive social networks. *Proceedings of the VLDB Endowment*, 13(9):1498–1510, 2020.

Christian Borgs, Michael Brautbar, Jennifer Chayes, and Brendan Lucier. Maximizing social influence in nearly optimal time. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*, pp. 946–957. SIAM, 2014.

Damon Centola. The spread of behavior in an online social network experiment. *science*, 329(5996): 1194–1197, 2010.

Amit Chakrabarti and Sagar Kale. Submodular maximization meets streaming: matchings, matroids, and more. *Mathematical Programming*, 154(1):225–247, 2015.

Chandra Chekuri, Jan Vondrák, and Rico Zenklusen. Dependent randomized rounding via exchange properties of combinatorial structures. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, pp. 575–584. IEEE, 2010.

Wei Chen, Yajun Wang, and Siyu Yang. Efficient influence maximization in social networks. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 199–208, 2009.

Wei Chen, Chi Wang, and Yajun Wang. Scalable influence maximization for prevalent viral marketing in large-scale social networks. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1029–1038, 2010.

Shubham Chowdhary, Giulia De Pasquale, Nicolas Lanzetti, Ana-Andreea Stoica, and Florian Dörfler. Fairness in social influence maximization via optimal transport. *Advances in Neural Information Processing Systems*, 37:10380–10413, 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*, pp. 514–525, 2024.

Pedro Domingos and Matt Richardson. Mining the network value of customers. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 57–66, 2001.

Marwa El Halabi, Slobodan Mitrović, Ashkan Norouzi-Fard, Jakab Tardos, and Jakub M Tarnawski. Fairness in streaming submodular maximization: Algorithms and hardness. *Advances in Neural Information Processing Systems*, 33:13609–13622, 2020.

Marwa El Halabi, Federico Fusco, Ashkan Norouzi-Fard, Jakab Tardos, and Jakub Tarnawski. Fairness in streaming submodular maximization over a matroid constraint. In *International Conference on Machine Learning*, pp. 9150–9171. PMLR, 2023.

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*, pp. 1027–1035. PMLR, 2024.

Moran Feldman, Paul Liu, Ashkan Norouzi-Fard, Ola Svensson, and Rico Zenklusen. Streaming submodular maximization under matroid constraints. *Mathematics of Operations Research*, 51 (1):299–332, 2026.

Benjamin Fish, Ashkan Bashardoust, Danah Boyd, Sorelle Friedler, Carlos Scheidegger, and Suresh Venkatasubramanian. Gaps in information access in social networks? In *The world wide web conference*, pp. 480–490, 2019.

Jacob Goldenberg, Barak Libai, and Eitan Muller. Talk of the network: A complex systems look at the underlying process of word-of-mouth. *Marketing letters*, 12(3):211–223, 2001.

Mark Granovetter. Threshold models of collective behavior. *American journal of sociology*, 83(6): 1420–1443, 1978.

Kai Han, Shuang Cui, Tianshuai Zhu, Enpei Zhang, Benwei Wu, Zhizhuo Yin, Tong Xu, Shaojie Tang, and He Huang. Approximation algorithms for submodular data summarization with a knapsack constraint. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 5 (1):1–31, 2021.

Chien-Chung Huang, Théophile Thiery, and Justin Ward. Improved multi-pass streaming algorithms for submodular maximization with matroid constraints. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, volume 176 of *LIPIcs*, pp. 62:1–62:19, 2020. doi: 10.4230/LIPIcs.APPROX/RANDOM.2020.62.

Ehsan Kazemi, Marko Mitrovic, Morteza Zadimoghaddam, Silvio Lattanzi, and Amin Karbasi. Submodular streaming in all its glory: Tight approximation, minimum memory and low adaptive complexity. In *International Conference on Machine Learning*, pp. 3311–3320. PMLR, 2019.

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

Ariel Kulik, Hadas Shachnai, and Tami Tamir. Approximations for monotone and nonmonotone submodular maximization with knapsack constraints. *Mathematics of Operations Research*, 38 (4):729–739, 2013.

Lijun Li, Chenyang Xu, Liuyi Yang, and Ruilong Zhang. Fair submodular maximization over a knapsack constraint. In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence*, pp. 3934–3942. International Joint Conferences on Artificial Intelligence Organization, 2025. doi: 10.24963/ijcai.2025/438.

Wenyue Ma, Maximilian K Egger, Andreas Pavlogiannis, Yuchen Li, and Panagiotis Karras. Reachability-aware fair influence maximization. In *Asia-Pacific Web (APWeb) and Web-Age Information Management (WAIM) Joint International Conference on Web and Big Data*, pp. 342–359. Springer, 2024.

George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. An analysis of approximations for maximizing submodular set functions—i. *Mathematical programming*, 14(1):265–294, 1978.

Huy Nguyen and Rong Zheng. On budgeted influence maximization in social networks. *IEEE Journal on Selected Areas in Communications*, 31(6):1084–1094, 2013.

Matthew Richardson and Pedro Domingos. Mining knowledge-sharing sites for viral marketing. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 61–70, 2002.

Xiaobin Rui, Zhixiao Wang, Hao Peng, Wei Chen, and Philip S Yu. A scalable algorithm for fair influence maximization with unbiased estimator. *IEEE transactions on knowledge and data engineering*, 37(7):3881–3895, 2025.

Don W Stacks and Michael B Salwen. *An integrated approach to communication theory and research*. Routledge, 2014.

Ana-Andreea Stoica, Jessy Xinyi Han, and Augustin Chaintreau. Seeding network influence in biased networks and the benefits of diversity. In *Proceedings of The Web Conference 2020*, pp. 2089–2098, 2020.

Maxim Sviridenko. A note on maximizing a submodular set function subject to a knapsack constraint. *Operations Research Letters*, 32(1):41–43, 2004.

Youze Tang, Xiaokui Xiao, and Yanchen Shi. Influence maximization: Near-optimal time complexity meets practical efficiency. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pp. 75–86, 2014.

Youze Tang, Yanchen Shi, and Xiaokui Xiao. Influence maximization in near-linear time: A martingale approach. In *Proceedings of the 2015 ACM SIGMOD international conference on management of data*, pp. 1539–1554, 2015.

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

Thomas W Valente. Social network thresholds in the diffusion of innovations. *Social networks*, 18 (1):69–89, 1996.

Xiangqi Wang, Shaokun Zhang, Jose Efraim Aguilar Escamilla, Qingyun Wu, Xiangliang Zhang, Jian Kang, and Huazheng Wang. Fair online influence maximization. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856.

Zhixiao Wang, Jiayu Zhao, Chengcheng Sun, Xiaobin Rui, and Philip S Yu. A general concave fairness framework for influence maximization based on poverty reward. *ACM Transactions on Knowledge Discovery from Data*, 19(1):1–23, 2024.

Mingchao Zhou, Chensheng Ma, Hao Xiao, and Zhao Zhang. A constant approximation algorithm for fair submodular maximization under a knapsack constraint. *Operations Research Letters*, pp. 107507, 2026.

A Complete Proofs

A.1 Example of the operation encoding

Consider one group with bounds $\ell = 1$ and $u = 2$, containing elements c, e_1, e_2 of weights $1, 3, 4$, respectively, and let $B = 5$. The base is $C = \{c\}$ and the residual budget is $R = 4$. The replacement block has capacity one and contains r_{c,e_1} and r_{c,e_2} with operation weights 2 and 3, while the addition block has capacity one and contains a_{e_1} and a_{e_2} with operation weights 3 and 4.

For example, let

$$M = \{r_{c,e_1}, a_{e_2}\}.$$

Then

$$A(M) = \{e_1, e_2\}, \qquad D(M) = \{c\},$$

and hence

$$S(M) = (C \setminus D(M)) \cup A(M) = \{e_1, e_2\}.$$

Although M satisfies both block capacities, its total operation weight is

$$\widetilde{w}(M) = (3 - 1) + 4 = 6 > R.$$

Thus, M is infeasible for the auxiliary problem. Correspondingly, the decoded set has total weight

$$w(S(M)) = w(C) + \widetilde{w}(M) = 1 + 6 = 7 > B.$$

In contrast, $\{r_{c,e_1}\}$ has operation weight 2 and decodes to $\{e_1\}$ with total weight 3, while $\{a_{e_2}\}$ has operation weight 4 and decodes to $\{c, e_2\}$ with total weight 5. Both operation sets are feasible for the auxiliary problem.

Finally, the feasible operation set $\{r_{c,e_2}\}$ decodes to $\{e_2\}$, whose total weight is 4. Its auxiliary value, however, is $f(C \cup \{e_2\}) - f(C)$, evaluated using the set $C \cup \{e_2\} = \{c, e_2\}$ of total weight 5. This illustrates that the evaluation set $C \cup A(M)$ and the decoded solution $S(M)$ need not coincide.

A.2 Partition matroid structure

Every addition operation a_e, where $e \in G_i \setminus C_i$, belongs to exactly one addition block Ω_i^{add}. Every replacement operation $r_{c,e}$ belongs to exactly one replacement block Ω_c, determined by the base element c being replaced. Since addition and replacement operations are distinct, these blocks are pairwise disjoint and together cover Ω.

The family $\mathcal{I}_{\mathrm{par}}$ imposes an independent cardinality bound on each block: capacity one for every replacement block Ω_c and capacity $u_i - \ell_i$ for every addition block Ω_i^{add}. Hence, by definition, $\mathcal{I}_{\mathrm{par}}$ is a partition matroid. The sum of its block capacities is

$$|C| + \sum_i (u_i - \ell_i) = \sum_i \ell_i + \sum_i (u_i - \ell_i) = \sum_i u_i = K,$$

so its rank is at most K.

A.3 Proof of Lemma 3.1

Proof. Normalization follows from $h(\varnothing) = f(C) - f(C) = 0$. Monotonicity follows from $A(M) \subseteq A(M')$ for $M \subseteq M'$ and the monotonicity of f. For $M \subseteq M'$ and an operation o, if $e(o) \notin A(M')$, diminishing returns follows from $C \cup A(M) \subseteq C \cup A(M')$. Otherwise its marginal on M' is zero, while its marginal on M is nonnegative. □

A.4 Proof of Lemma 3.2

Proof. Keep one operation per element of $A(M)$. Deleting the others preserves matroid independence, cannot increase the nonnegative total weight, leaves $A(M)$ unchanged. □

A.5 Proof of Lemma 3.3

Proof. By the definition of the operations, each replacement in G_i removes one element of C_i and adds one element of $G_i \setminus C_i$, while each addition adds one element without removing any element. Since $|C_i| = \ell_i$,

$$|S(\overline{M}) \cap G_i| = \ell_i + |\overline{M} \cap \Omega_i^{\mathrm{add}}|.$$

Since $\overline{M} \subseteq M$ and $\mathcal{I}_{\mathrm{par}}$ is downward closed, $\overline{M} \in \mathcal{I}_{\mathrm{par}}$. Therefore,

$$\ell_i \le |S(\overline{M}) \cap G_i| \le u_i$$

for every $i \in [h]$.

Each added element appears in exactly one operation of $\overline{M}$, and each element of C is removed at most once. Hence,

$$w(S(\overline{M})) = w(C) + \sum_{o \in \overline{M}} \widetilde{w}(o).$$

By Lemma 3.2,

$$\sum_{o \in \overline{M}} \widetilde{w}(o) \le \sum_{o \in M} \widetilde{w}(o) \le R.$$

Thus,

$$w(S(\overline{M})) \le w(C) + R = B.$$

Therefore, $S(\overline{M})$ is feasible for FSMK. □

A.6 Proof of Lemma 3.4

Proof. For each $i \in [h]$, let $O_i = O \cap G_i, \quad X_i = C_i \setminus O_i, \quad Y_i = O_i \setminus C_i$. Since $|C_i| = \ell_i \le |O_i|$, we have $|X_i| \le |Y_i|$. Fix an injection $\pi_i : X_i \to Y_i$ and define

$$M_{O,i} = \{r_{c,\pi_i(c)} : c \in X_i\} \cup \{a_e : e \in Y_i \setminus \pi_i(X_i)\}.$$

Let $M_O = \bigcup_{i=1}^{h} M_{O,i}$. Each element of C is replaced at most once. Moreover,

$$\begin{aligned} |M_O \cap \Omega_i^{\mathrm{add}}| &= |Y_i| - |X_i| \\ &= |O_i| - |C_i| \\ &\le u_i - \ell_i. \end{aligned}$$

Thus, $M_O \in \mathcal{I}_{\mathrm{par}}$.

The operations in $M_{O,i}$ remove the elements in $C_i \setminus O_i$ and add the elements in $O_i \setminus C_i$. Therefore,

$$S(M_O) = O.$$

By the definition of the operation weights,

$$\begin{aligned} \sum_{o \in M_O} \widetilde{w}(o) &= \sum_{i=1}^{h} \left(\sum_{e \in Y_i} w(e) - \sum_{c \in X_i} w(c) \right) \\ &= w(O) - w(C) \\ &\le B - w(C) \\ &= R. \end{aligned}$$

Hence, M_O is feasible for the auxiliary problem.

Finally,

$$A(M_O) = O \setminus C.$$

By monotonicity of f,

$$\begin{aligned} h(M_O) &= f(C \cup A(M_O)) - f(C) \\ &= f(C \cup O) - f(C) \\ &\ge f(O) - f(C). \end{aligned}$$

This completes the proof. □

A.7 Proof of Lemma 3.5

Proof. By Lemma 3.2,

$$h(\overline{M}) = h(M).$$

Moreover,

$$S(\overline{M}) \cup C = C \cup A(\overline{M}).$$

By submodularity and the nonnegativity of f,

$$\begin{aligned} f(S(\overline{M})) + f(C) &\geq f(S(\overline{M}) \cup C) + f(S(\overline{M}) \cap C) \\ &\geq f(C \cup A(\overline{M})). \end{aligned}$$

Therefore,

$$\begin{aligned} f(S(\overline{M})) &\geq f(C \cup A(\overline{M})) - f(C) \\ &= h(\overline{M}) \\ &= h(M). \end{aligned}$$

This completes the proof. □

A.8 Proof of Theorem 3.1

Proof. Every realization of the auxiliary solver is feasible, so Lemmas 3.2 and 3.3 give an exactly feasible decoded solution. The randomized subroutine guarantees

$$\mathbb{E}[h(M)] \geq \beta \mathrm{OPT}_{\mathrm{aux}} \geq \beta(f(O) - f(C)).$$

For every realization, $f(\widehat{S}) \geq f(C)$ and $f(\widehat{S}) \geq f(S(\overline{M})) \geq h(M)$. Therefore

$$\mathbb{E}[f(\widehat{S})] \geq \max\{f(C), \mathbb{E}[h(M)]\} \geq \max\{f(C), \beta(f(O) - f(C))\} \geq \frac{\beta}{1+\beta} f(O).$$

The last inequality follows by taking a convex combination with weights $\beta/(1+\beta)$ and $1/(1+\beta)$. For fixed ε, the auxiliary routine is polynomial-time, and $|\Omega| \leq |N|(L+1)$. This establishes both the expected approximation and the running time. □

A.9 Proof of Theorem 4.1

Proof. Set $\rho = \frac{1}{6(1+\eta)} = \frac{1}{6+\varepsilon}$. The guarantee of MatKnapStream gives

$$h(M) \geq \rho \, \mathrm{OPT}_{\mathrm{aux}}.$$

By Lemma 3.4,

$$\mathrm{OPT}_{\mathrm{aux}} \geq f(O) - f(C).$$

After duplicate pruning, Lemma 3.5 gives

$$f(S_{\mathrm{str}}) \geq h(M) \geq \rho\big(f(O) - f(C)\big).$$

The set C is feasible, and Lemma 3.3 shows that S_{str} is also feasible. Since the algorithm returns the better of these two sets,

$$\begin{aligned} f(\widehat{S}) &\geq \max\big\{f(C), \rho\big(f(O) - f(C)\big)\big\} \\ &\geq \frac{\rho}{1+\rho} f(O) \\ &= \frac{1}{7+\varepsilon} f(O). \end{aligned}$$

This completes the proof. □

A.10 Proof of Lemma 4.1

Proof. For each $c \in C$, the auxiliary partition matroid contains a replacement block Ω_c of capacity 1, and for each group G_i, it contains an addition block Ω_i^{add} of capacity $u_i - \ell_i$. Hence, the sum of all block capacities is

$$|C| + \sum_{i=1}^{h} (u_i - \ell_i) = \sum_{i=1}^{h} u_i = K,$$

where $|C| = \sum_{i=1}^{h} \ell_i$. Since $\eta = \varepsilon/6$, Lemma 2.2 implies that $\mathsf{MatKnapStream}(\eta)$ uses $O(K)$ memory. The first pass stores at most $|C| \le K$ elements, so the total memory is $O(K)$.

For each $e \in G_i \setminus C_i$, the algorithm generates one addition operation and ℓ_i replacement operations, and hence at most $L + 1$ operations. By Lemma 2.2, $\mathsf{MatKnapStream}(\eta)$ uses $O(K)$ value-oracle queries per operation. Therefore, the number of value-oracle queries per original stream element is

$$O\big(K(L+1)\big).$$

This completes the proof. □

A.11 Example of the order $\prec_t$

Suppose pass t starts with

$$M_{t-1} = \{o_2, o_5\},$$

where $o_2 \prec_t o_5$ is their relative order inherited from the previous pass. Suppose the remaining operations are generated in the order o_1, o_3, o_4, o_6. Then

$$o_2 \prec_t o_5 \prec_t o_1 \prec_t o_3 \prec_t o_4 \prec_t o_6.$$

Thus, the operations already in M_{t-1} are placed first, while all other operations follow their generation order.

For example, if

$$Q = \{o_2, o_4, o_5, o_6\},$$

then

$$Q_t^{\prec o_4} = \{o_2, o_5\}, \qquad \nu_t(o_4, Q) = h(o_4 \mid \{o_2, o_5\}),$$

and

$$Q_t^{\prec o_6} = \{o_2, o_5, o_4\}, \qquad \nu_t(o_6, Q) = h(o_6 \mid \{o_2, o_5, o_4\}).$$

The order $\prec_t$ is only used to define these incremental values and does not require an additional scan of the stream.

A.12 Proof of Lemma 5.1

Proof. List the operations of Q as $o_1, \dots, o_r$ according to $\prec_t$. By the definition of ν_t,

$$\sum_{j=1}^{r} \nu_t(o_j, Q) = \sum_{j=1}^{r} h\big(o_j \mid \{o_1, \dots, o_{j-1}\}\big) = h(Q),$$

where the last equality follows from $h(\varnothing) = 0$.

For every $o \in Q$, we have $Q_t^{\prec o} \subseteq W_t^{\prec o}$. Hence, by submodularity,

$$\nu_t(o, W) = h(o \mid W_t^{\prec o}) \le h(o \mid Q_t^{\prec o}) = \nu_t(o, Q).$$

For (3), add $o_1, \dots, o_r$ to $W \setminus Q$ in this order. Immediately before o_j is added, the current set contains $W_t^{\prec o_j}$. Therefore, by submodularity, the marginal value of o_j at that time is at most $\nu_t(o_j, W)$. Summing over $j = 1, \dots, r$ gives

$$h(Q \mid W \setminus Q) \le \sum_{o \in Q} \nu_t(o, W).$$

This completes the proof. □

A.13 Proof of Lemma 5.2

Proof. The first two properties follow directly from the update rule. If the block $P(o)$ of an accepted operation o is full, the algorithm removes one operation from the same block before inserting o; otherwise, no operation is removed. Hence the partition matroid constraint is preserved. The knapsack-feasibility condition ensures that $\widetilde{w}(M) \le R$ after every accepted update.

Let M^- be the maintained set immediately before an accepted update, and let

$$M^+ = (M^- \setminus X_o) \cup \{o\}.$$

By submodularity and Lemma 5.1,

$$\begin{aligned} h(M^+) - h(M^-) &= h(o \mid M^- \setminus X_o) - h(X_o \mid M^- \setminus X_o) \\ &\ge h(o \mid M^-) - \sum_{x \in X_o} \nu_t(x, M^-) \\ &\ge \beta_t \sum_{x \in X_o} \nu_t(x, M^-) \ge 0. \end{aligned}$$

The second inequality follows from the local-improvement condition, and the last inequality follows from the monotonicity of h. Thus, the objective value does not decrease.

We next prove the density property. Since o is generated after all operations currently in M^- under $\prec_t$,

$$\nu_t(o, M^+) = h(o \mid M^- \setminus X_o) \ge h(o \mid M^-) \ge \tau \widetilde{w}(o),$$

where the last inequality is the density condition.

Now consider any $x \in M^- \setminus X_o$. The newly inserted operation o appears after x under $\prec_t$, and removing X_o can only shrink the set of operations preceding x. Hence, by submodularity,

$$\nu_t(x, M^+) \ge \nu_t(x, M^-).$$

Thus, every accepted update preserves the density property for all operations in the maintained set.

It remains to verify that this property is preserved between consecutive passes. At the beginning of pass $t+1$, the operations in M_t appear first under $\prec_{t+1}$ and retain their relative order from pass t. Therefore, for every $x \in M_t$,

$$\nu_{t+1}(x, M_t) = \nu_t(x, M_t).$$

Since $M_0 = \varnothing$, induction over the updates and the passes shows that every maintained operation $x \in M$ satisfies

$$\nu_t(x, M) \ge \tau \widetilde{w}(x).$$

Finally, Lemma 5.1 gives

$$h(M) = \sum_{x \in M} \nu_t(x, M) \ge \tau \sum_{x \in M} \widetilde{w}(x) = \tau \widetilde{w}(M).$$

This completes the proof. □

A.14 Proof of Lemma 5.3

Proof. Suppose that an operation o is budget rejected when the maintained set is M. Then

$$\widetilde{w}\big((M \setminus X_o) \cup \{o\}\big) > R.$$

Since all operation weights are nonnegative,

$$\widetilde{w}(M) + \widetilde{w}(o) > R.$$

By Lemma 5.2,

$$h(M) \ge \tau \widetilde{w}(M).$$

Moreover, by submodularity and the density condition,

$$h(\{o\}) = h(o \mid \varnothing) \ge h(o \mid M) \ge \tau \widetilde{w}(o).$$

Hence,

$$h(M) + h(\{o\}) \geq \tau R,$$

and therefore

$$\max\{h(M), h(\{o\})\} \geq \frac{\tau R}{2}.$$

By Lemma 5.2, the maintained objective value does not decrease in later updates or passes, so $h(M_d) \geq h(M)$. Moreover, since o is budget rejected, it satisfies $b(o) > 0$ and $\widetilde{w}(o) \leq R$, and hence $\{o\}$ is a feasible singleton. By the choice of $o^\star$,

$$h(\{o^\star\}) \geq h(\{o\}).$$

The result follows. □

A.15 Proof of Lemma 5.4

Proof. Let A_t contain every operation held at any time during pass t, including M_{t-1}. For $x \in A_t \setminus M_t$, put $\chi_t(x) = \nu_t(x, M_x^-)$, where M_x^- is the maintained set immediately before removal. Initial operations are not processed again, and every other operation is generated once, so each operation is removed at most once in this pass.

Let $M^\star$ be an optimal solution to the auxiliary problem. We establish four claims.

Claim 1.

$$\sum_{x \in A_t \setminus M_t} \chi_t(x) \leq \frac{h(M_t) - h(M_{t-1})}{\beta_t}.$$

Indeed, by the proof of Lemma 5.2, every accepted update satisfies

$$h(M^+) - h(M^-) \geq \beta_t \sum_{x \in X_o} \nu_t(x, M^-).$$

Whenever $x \in X_o$, we have $\nu_t(x, M^-) = \chi_t(x)$. Since each operation is removed at most once during the pass, summing over all accepted updates proves the claim.

Claim 2. There exists a set $L_t \subseteq M^\star \setminus A_t$ such that

$$h(L_t \mid A_t) \leq \tau R.$$

For each $o \in M^\star \setminus A_t$, let M_o^- denote the maintained set when o is processed, and define

$$L_t = \left\{ o \in M^\star \setminus A_t : h(o \mid M_o^-) < \tau \widetilde{w}(o) \right\}.$$

Since $M_o^- \subseteq A_t$, submodularity gives

$$\begin{aligned} h(L_t \mid A_t) &\leq \sum_{o \in L_t} h(o \mid A_t) \\ &\leq \sum_{o \in L_t} h(o \mid M_o^-) \\ &\leq \tau \widetilde{w}(L_t) \leq \tau R, \end{aligned}$$

where the last inequality follows from $L_t \subseteq M^\star$ and the feasibility of $M^\star$.

Claim 3. Let

$$T_t = M^\star \setminus (A_t \cup L_t).$$

Then

$$h(T_t \mid A_t) \leq (1 + \beta_t) h(M_t).$$

Fix $o \in T_t$. Since $o \in M^\star$, we have $b(o) > 0$ and $\widetilde{w}(o) \leq R$. Moreover, $o \notin A_t$ implies $o \notin M_{t-1}$, so o is processed by the algorithm. Since $o \notin L_t$, it satisfies the density condition. As

o is not accepted and no operation is budget rejected in this pass, o must fail the local-improvement condition. Hence $P(o)$ is full when o is processed and

$$h(o \mid M_o^-) < (1+\beta_t)\nu_t(x_o, M_o^-).$$

For each block P, introduce its abstract slots when it is first full during the pass, including at the beginning of the pass. Whenever an accepted operation replaces another operation, the new operation inherits the slot of the removed operation. These slots are only used for the analysis and are independent of the order $\prec_t$.

The incremental value associated with each slot never decreases. For an operation that remains in its slot, this follows from Lemma 5.2. If an accepted operation y replaces x_y, then

$$\begin{aligned}\nu_t(y, M^+) &= h(y \mid M^- \setminus \{x_y\}) \\ &\geq h(y \mid M^-) \\ &\geq (1+\beta_t)\nu_t(x_y, M^-) \geq \nu_t(x_y, M^-).\end{aligned}$$

Moreover, once a block becomes full, it remains full for the rest of the pass.

When $o \in T_t$ is rejected, x_o has minimum incremental value in $M_o^- \cap P(o)$. Hence every slot of $P(o)$ has value at least $\nu_t(x_o, M_o^-)$ at that time, and by the slot monotonicity just established the same is true at the end of the pass.

Since $T_t \subseteq M^\star$ and $M^\star$ satisfies the partition matroid constraint,

$$|T_t \cap P(o)| \leq b(o).$$

We can therefore assign the operations of T_t in each block to distinct final slots of that block. Let $\phi(o) \in M_t$ denote the operation occupying the slot assigned to o. Then ϕ is injective and

$$\nu_t(x_o, M_o^-) \leq \nu_t(\phi(o), M_t).$$

Using submodularity and the preceding rejection and assignment inequalities, we obtain

$$\begin{aligned}h(T_t \mid A_t) &\leq \sum_{o \in T_t} h(o \mid M_o^-) \\ &\leq (1+\beta_t) \sum_{o \in T_t} \nu_t(x_o, M_o^-) \\ &\leq (1+\beta_t) \sum_{o \in T_t} \nu_t(\phi(o), M_t) \\ &\leq (1+\beta_t) h(M_t),\end{aligned}$$

where the last inequality follows from the injectivity of ϕ and Lemma 5.1(1).

Claim 4.

$$h(A_t) \leq h(M_t) + \sum_{x \in A_t \setminus M_t} \chi_t(x).$$

By Lemma 5.1(1),

$$h(A_t) = \sum_{x \in A_t} \nu_t(x, A_t).$$

For $x \in M_t$, since $M_t \subseteq A_t$, Lemma 5.1(2) gives

$$\nu_t(x, A_t) \leq \nu_t(x, M_t).$$

For $x \in A_t \setminus M_t$, since $M_x^- \subseteq A_t$,

$$\nu_t(x, A_t) \leq \nu_t(x, M_x^-) = \chi_t(x).$$

Summing these inequalities proves the claim.

Finally, since

$$M^\star \cup A_t = A_t \cup T_t \cup L_t,$$

monotonicity and submodularity imply

$$\begin{aligned}\mathrm{OPT}_{\mathrm{aux}} = h(M^\star) &\le h(A_t) + h(T_t \mid A_t) + h(L_t \mid A_t) \\ &\le \sum_{x \in A_t \setminus M_t} \chi_t(x) + (2+\beta_t) h(M_t) + \tau R.\end{aligned}$$

Applying Claim 1 yields

$$\mathrm{OPT}_{\mathrm{aux}} - \tau R \le \frac{h(M_t) - h(M_{t-1})}{\beta_t} + (2+\beta_t) h(M_t),$$

as desired. □

A.16 Proof of Lemma 5.5

Proof. Let $\Lambda = \mathrm{OPT}_{\mathrm{aux}} - \tau R$. If $\Lambda \le 0$, the result follows immediately since $h(M_t) \ge 0$. Assume therefore that $\Lambda > 0$.

For $t = 1$, Lemma 5.4, $M_0 = \varnothing$, and $\beta_1 = 1$ give

$$\Lambda \le 4 h(M_1),$$

which is the desired bound.

Now let $t \ge 2$ and suppose that the result holds for $t-1$. Since $h(M_t) \ge h(M_{t-1})$ by Lemma 5.2, and $\Lambda > 0$ implies $h(M_t) > 0$, define

$$\theta_t = \frac{h(M_{t-1})}{h(M_t)} \in [0, 1].$$

By the induction hypothesis,

$$\Lambda \le 2\left(1 + \frac{1}{t-1}\right) h(M_{t-1}) = \frac{2t}{t-1} \theta_t h(M_t).$$

On the other hand, Lemma 5.4 with $\beta_t = 1/t$ gives

$$\begin{aligned}\Lambda &\le t\big(h(M_t) - h(M_{t-1})\big) + \left(2 + \frac{1}{t}\right) h(M_t) \\ &= \left(t(1-\theta_t) + 2 + \frac{1}{t}\right) h(M_t).\end{aligned}$$

Therefore,

$$\Lambda \le \min\left\{ \frac{2t}{t-1}\theta_t,\ t(1-\theta_t) + 2 + \frac{1}{t} \right\} h(M_t).$$

The first expression is increasing in θ_t, whereas the second is decreasing. They are equal when

$$\theta_t = \frac{t^2 - 1}{t^2},$$

at which both are equal to

$$\frac{2(t+1)}{t}.$$

Hence,

$$\Lambda \le 2\left(1 + \frac{1}{t}\right) h(M_t).$$

This completes the proof. □

A.17 Proof of Theorem 5.1

Proof. If $m = 0$, every feasible singleton has value zero. For any feasible auxiliary solution M, submodularity gives

$$h(M) \le \sum_{o \in M} h(\{o\}) = 0.$$

Hence $\mathrm{OPT}_{\mathrm{aux}} = 0$.

Assume now that $m > 0$. First suppose that $R > 0$. By the choice of the geometric thresholds, there exists an index $j^\star \in \mathcal{J}$ such that

$$\frac{\mathrm{OPT}_{\mathrm{aux}}}{2(1+\varepsilon/4)R} \le \tau_{j^\star} \le \frac{\mathrm{OPT}_{\mathrm{aux}}}{2R}.$$

Suppose first that no operation is budget rejected during the d passes for threshold $\tau_{j^\star}$. By the no-rejection bound and the upper bound on $\tau_{j^\star}$,

$$\begin{aligned} h(M_d^{(j^\star)}) &\ge \frac{\mathrm{OPT}_{\mathrm{aux}} - \tau_{j^\star} R}{2+\varepsilon/2} \\ &\ge \frac{\mathrm{OPT}_{\mathrm{aux}}/2}{2+\varepsilon/2} = \frac{1}{4+\varepsilon}\mathrm{OPT}_{\mathrm{aux}}. \end{aligned}$$

Suppose instead that some operation is budget rejected during these d passes. By Lemma 5.3 and the lower bound on $\tau_{j^\star}$,

$$\begin{aligned} \max\{h(M_d^{(j^\star)}), h(\{o^\star\})\} &\ge \frac{\tau_{j^\star} R}{2} \\ &\ge \frac{\mathrm{OPT}_{\mathrm{aux}}}{4(1+\varepsilon/4)} = \frac{1}{4+\varepsilon}\mathrm{OPT}_{\mathrm{aux}}. \end{aligned}$$

Since M_{out} has maximum h-value among $\{M_d^{(j)} : j \in \mathcal{J}\} \cup \{\{o^\star\}\}$, the desired bound follows in both cases.

Finally, suppose that $R = 0$. Then every processed operation has zero weight, so no budget rejection can occur. Since $\tau_0 = 0$, the no-rejection bound gives

$$h(M_d^{(0)}) \ge \frac{1}{2+\varepsilon/2}\mathrm{OPT}_{\mathrm{aux}}.$$

Therefore,

$$h(M_{\mathrm{out}}) \ge h(M_d^{(0)}) \ge \frac{1}{2+\varepsilon/2}\mathrm{OPT}_{\mathrm{aux}} \ge \frac{1}{4+\varepsilon}\mathrm{OPT}_{\mathrm{aux}}.$$

This completes the proof. □

A.18 Proof of Theorem 5.2

Proof. If $m = 0$, then $\mathrm{OPT}_{\mathrm{aux}} = 0$ by Theorem 5.1. Lemma 3.4 then gives

$$f(O) \le f(C).$$

Since the algorithm returns C and C is feasible, the result follows.

Assume that $m > 0$. By Lemma 5.2, every $M_d^{(j)}$ is feasible for the auxiliary problem, and $\{o^\star\}$ is feasible by its definition. Hence, M_{out} is feasible for the auxiliary problem.

By Lemma 3.2,

$$h(\overline{M}_{\mathrm{out}}) = h(M_{\mathrm{out}}) \qquad \text{and} \qquad \widetilde{w}(\overline{M}_{\mathrm{out}}) \le \widetilde{w}(M_{\mathrm{out}}).$$

Therefore, Lemma 3.3 shows that S_{out} satisfies the knapsack constraint and all group bounds.

By Lemma 3.5, Theorem 5.1, and Lemma 3.4,

$$\begin{aligned} f(S_{\text{out}}) &\geq h(\overline{M}_{\text{out}}) \\ &= h(M_{\text{out}}) \\ &\geq \frac{1}{4+\varepsilon}\text{OPT}_{\text{aux}} \\ &\geq \frac{1}{4+\varepsilon}\big(f(O) - f(C)\big). \end{aligned}$$

Let $\alpha = 1/(4+\varepsilon)$. Since the algorithm returns the better of C and S_{out},

$$\begin{aligned} f(\widehat{S}) &\geq \max\big\{f(C), \alpha\big(f(O) - f(C)\big)\big\} \\ &\geq \frac{\alpha}{1+\alpha} f(O) = \frac{1}{5+\varepsilon} f(O). \end{aligned}$$

Since both C and S_{out} are feasible, $\widehat{S}$ is feasible. This completes the proof. □

A.19 Proof of Theorem 5.3

Proof. The first pass constructs C, and the second pass finds $o^\star$. The algorithm then performs $d = \lceil 4/\varepsilon \rceil$ local search passes. Hence, the total number of passes is at most $2 + \lceil 4/\varepsilon \rceil$.

If $R > 0$, the number of thresholds is $|\mathcal{J}| = O(\log(K+1)/\varepsilon)$. For each $j \in \mathcal{J}$, the maintained set $M^{(j)}$ contains at most K operations, since the rank of the auxiliary partition matroid is at most K. Moreover, the base set C contains $\sum_{i=1}^{h} \ell_i \leq K$ elements. Therefore, the total memory is $O(K \log(K+1)/\varepsilon) = \widetilde{O}_\varepsilon(K)$. If $R = 0$, only one threshold is maintained, so the same upper bound continues to hold.

Each original element $e \in G_i \setminus C_i$ generates one addition operation and ℓ_i replacement operations, and hence at most $L+1$ operations. For a fixed threshold, a direct implementation may examine at most K maintained operations to find an operation of minimum incremental value in $P(o)$. Thus, during one local search pass, the number of value-oracle queries per original stream element is $O(K(L+1)\log(K+1)/\varepsilon)$. Since there are $d = O(1/\varepsilon)$ local search passes, the total number of value-oracle queries per original stream element over all passes is $O(K(L+1)\log(K+1)/\varepsilon^2)$. This completes the proof. □

B Implementation Details

B.1 The local search Pass

For completeness, we give the full implementation of one density-truncated local search pass.

B.2 Threshold selection and boundary cases

The second pass finds a feasible singleton $o^\star$ of maximum value and sets

$$m = h(\{o^\star\}).$$

If no feasible singleton exists, we set $m = 0$.

Suppose first that $m > 0$. Since every feasible auxiliary solution contains at most K operations and h is normalized, monotone, and submodular,

$$m \leq \text{OPT}_{\text{aux}} \leq Km.$$

The lower bound follows because $\{o^\star\}$ is feasible. For the upper bound, any feasible M satisfies

$$h(M) \leq \sum_{o \in M} h(\{o\}) \leq |M| m \leq Km.$$

Algorithm 4 DensityLocalSearchPass(τ, β_t, M_{t-1})

Save the ordered initial set M_{t-1} and set $M \leftarrow M_{t-1}$.
Define the order $\prec_t$ by placing the operations in M_{t-1} first, followed by all other operations in the order in which they are generated during the current pass.
for each arriving element e **do**
  for each addition or replacement operation o generated by e **do**
    if $b(o) = 0$, $\widetilde{w}(o) > R$, or $o \in M_{t-1}$ **then**
      Continue.
    end if
    if $|M \cap P(o)| < b(o)$ **then**
      Set $X_o \leftarrow \varnothing$.
    else
      Choose

$$x_o \in \operatorname*{arg\,min}_{x \in M \cap P(o)} \nu_t(x, M)$$

      and set $X_o \leftarrow \{x_o\}$.
    end if
    Set $\Delta_o \leftarrow h(o \mid M)$.
    if $\Delta_o \geq (1+\beta_t) \sum_{x \in X_o} \nu_t(x, M)$, $\Delta_o \geq \tau \widetilde{w}(o)$, and $\widetilde{w}((M \setminus X_o) \cup \{o\}) \leq R$ **then**
      Set $M \leftarrow (M \setminus X_o) \cup \{o\}$.
    end if
  end for
end for
return M.

For $R > 0$, let

$$\mathcal{J} = \{0, \dots, \lceil \log_{1+\varepsilon/4} K \rceil\}, \qquad \tau_j = \frac{m(1+\varepsilon/4)^j}{2R}. \tag{B.1}$$

Since $\mathrm{OPT}_{\mathrm{aux}}/m \in [1, K]$, there exists $j^\star \in \mathcal{J}$ such that

$$\frac{\mathrm{OPT}_{\mathrm{aux}}}{2(1+\varepsilon/4)R} \leq \tau_{j^\star} \leq \frac{\mathrm{OPT}_{\mathrm{aux}}}{2R}. \tag{B.2}$$

Thus the grid contains a threshold within a factor $1 + \varepsilon/4$ of the desired value $\mathrm{OPT}_{\mathrm{aux}}/(2R)$.

If $m = 0$, every feasible singleton has value zero. By submodularity, every feasible auxiliary set M satisfies

$$h(M) \leq \sum_{o \in M} h(\{o\}) = 0,$$

and hence $\mathrm{OPT}_{\mathrm{aux}} = 0$. Lemma 3.4 then implies $f(O) \leq f(C)$, so the base set C is optimal.

Finally, if $R = 0$, only zero-weight operations can belong to a feasible auxiliary solution. In this case we use the single threshold $\tau_0 = 0$ and process only zero-weight operations.

B.3 Zero residual budget and degenerate instances

If $K = 0$, every upper bound is zero, and the algorithm returns $\varnothing$. Otherwise construct C before filtering operations. If no eligible auxiliary operation exists, $\mathrm{OPT}_{\mathrm{aux}} = 0$ and Lemma 3.4 makes C optimal. When $R = 0$, discard all positive-weight operations. On the remaining ground set, every operation has weight zero. A routine requiring a strictly positive budget can therefore be called with an artificial budget of one: every subset still has total operation weight zero, so the feasible family and optimum are unchanged. Decode using the original residual budget $R = 0$. The multi-pass algorithm instead uses its explicit zero-budget branch with $\tau = 0$.

For $R > 0$, a routine implementing density tests through division must treat zero-weight operations by the equivalent multiplication-form inequality, or as infinite density when the marginal is positive. Zero weight is not evidence of zero value. The memory bounds exclude the input group-bound table and oracle internals; at most K groups have positive upper bounds.

C Experimental Details and Additional Results

Datasets and group labels. Table 2 lists the network sizes, group labels, and data sources. Epinions and Pokec are directed networks, while LastFM Asia and Twitch EN are undirected.

Table 2: Networks and group labels used in the experiments.

Dataset	$\lvert V\rvert$	$\lvert E\rvert$	Groups	Group label	Source
Epinions	131,828	841,372	5	Released group IDs	SNAP, code
Pokec	1,632,803	30,622,564	3	Gender: 0, 1, or unknown	SNAP
LastFM Asia	7,624	27,806	18	Country/location	SNAP
Twitch EN	7,126	35,324	2	Explicit-language label (`mature`)	SNAP

For Epinions, we use the coverage instance released with the FairKnapStream code. It includes five group IDs, but their meanings are not specified. Its edge list does not include the signs of the original network. For Pokec, we treat missing gender values as a separate group. LastFM Asia groups are based on country/location. Twitch EN is the English-language Twitch network; its binary `mature` label describes explicit-language content, not a demographic attribute.

Influence model and RR sampling. An RR set consists of the vertices that can reach a sampled target in a random live-edge graph. The fraction of RR sets intersecting S estimates $f(S)/|V|$. For Epinions and Pokec, we use the weighted-cascade IC model with $p(u,v) = 1/d^-(v)$. We generate 20,000 RR sets for selection and an independent collection of 200,000 RR sets for evaluation. The selection and evaluation seeds are 7301 and 19777 for the feasibility experiment, and 26091501 and 26091502 for the comparison of exactly feasible solutions. Within each experiment, all methods share the same selection and evaluation RR sets for each dataset.

For LastFM Asia and Twitch EN, we set the IC probability to 0.1 on every edge. We use 10,000 RR sets for selection and an independent collection of 20,000 RR sets per group for evaluation. All returned solutions are evaluated using the same RR sets. Overall reach is the average activation fraction across groups, weighted by group size. Worst-group reach is the smallest activation fraction among the groups.

Costs and group quotas. For Epinions and Pokec, let $r(v)$ be the rank of v in a fixed random permutation (seed 9127). We set

$$\widetilde{w}(v) = 1 + \sqrt{d^+(v)} + \frac{0.1r(v)}{|V|+1}, \qquad w(v) = \frac{\widetilde{w}(v)}{|V|^{-1}\sum_{x\in V}\widetilde{w}(x)}.$$

Thus the mean node cost is one. The final term is a deterministic perturbation that makes equal degree-based costs distinct. Every method receives the same cost vector. For a group G_i, we set

$$\ell_i = \min\left\{|G_i|, \left\lceil 0.8B\frac{|G_i|}{|V|}\right\rceil\right\}, \qquad u_i = \max\left\{\ell_i, \min\left\{|G_i|, \left\lceil 1.2B\frac{|G_i|}{|V|}\right\rceil\right\}\right\}.$$

We use $\varepsilon = 0.1$ in these two experiments.

Baselines. Cui denotes the FairKnapStream implementation released by Cui et al. (2024), which we use without changing its selection rule. SmkStream is the two-pass knapsack streaming baseline from the same release. RandomStream is a one-pass baseline that accepts each arriving element with probability $1/2$ if adding it satisfies the budget. Neither SmkStream nor RandomStream enforces group quotas.

Cui+ExactRepair applies our repair procedure to the output S_{Cui} of FairKnapStream. It starts from S_{Cui} if that set is exactly feasible. Otherwise, it starts from the minimum-cost fair base C and makes one improvement sweep over $C \cup S_{\mathrm{Cui}}$, preserving feasibility. In both cases, it then makes up to five additional sweeps over the full stream, using the same order as the other methods. It accepts only additions or one-for-one swaps that strictly increase the selection objective and preserve the budget and all group quotas.

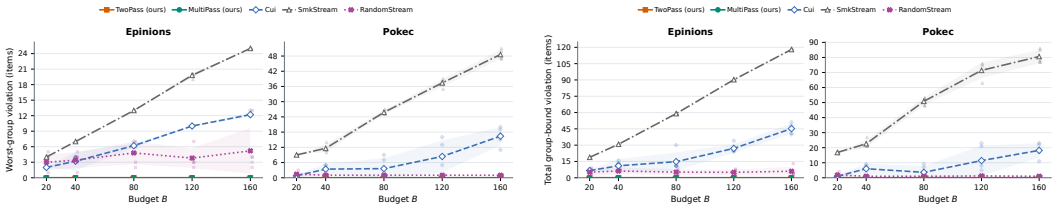
FairBase+Density starts from C and repeatedly adds the node with the largest positive marginal gain per unit cost among those whose addition preserves feasibility. FairBase+1Swap starts from the resulting set and further improves it through additions and one-for-one swaps with strictly positive gain, while preserving the budget and all group quotas.

For the downstream experiment, we use a deterministic budget-only density-greedy heuristic as the reference. It repeatedly adds the node with the largest positive marginal gain per unit cost among those whose addition satisfies the budget.

Exact-feasibility results. The budgets are $20, 40, 80, 120, 160$. For each dataset–budget pair, we use stream-order seeds $\{11, 29, 47, 83, 101\}$. For each group, define

$$v_i = \max\{\ell_i - |S \cap G_i|, 0\} + \max\{|S \cap G_i| - u_i, 0\}.$$

Exact feasibility requires $w(S) \leq B$ and $v_i = 0$ for every group.



(a) Worst single-group violation.

(b) Total violation over groups.

Figure 5: Group-quota violations measured in numbers of seeds. TwoPass and MultiPass have zero violation in every run; lower is better. Lines show means over five stream orders, faint points show individual orders, and shading shows the mean plus or minus one sample standard deviation, truncated at zero.

Across the 50 runs per method, the mean worst-group violations are 0 for both of our methods, 6.62 for Cui, 20.12 for SmkStream, and 2.56 for RandomStream. The corresponding mean total violations are 0, 14.48, 55.90, and 3.36. Figure 5 reports these magnitudes by dataset and budget.

Coverage under exact feasibility. We compare TwoPass and MultiPass with Cui+ExactRepair, FairBase+Density, and FairBase+1Swap with $n \in \{2016, 2520, 3024\}$ nodes.

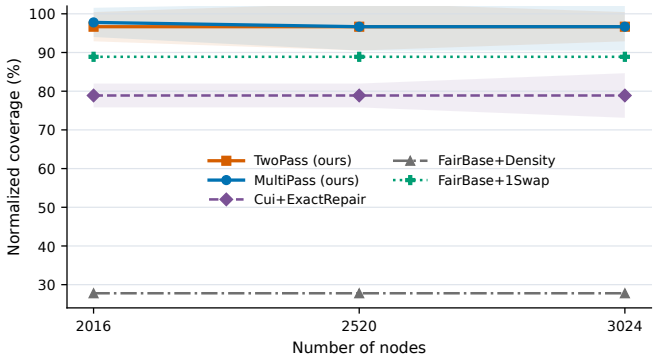


Figure 6: Normalized coverage.

Each instance has two groups, budget $B = 8$, and group bounds $\ell_i = 4$ and $u_i = 5$. We use five shared stream orders at each size. All 75 returned solutions satisfy the budget and group bounds. Figure 6 reports normalized coverage, defined as the fraction of nodes covered by the selected set. TwoPass achieves mean coverage of 96.67% at all three sizes, while MultiPass achieves 96.67–97.78%. The corresponding means are 78.89% for Cui+ExactRepair, 88.89% for FairBase+1Swap, and 27.78% for FairBase+Density. Thus, both streaming methods exceed the strongest baseline in this comparison, FairBase+1Swap, by at least 7.78 percentage points at every tested size.

Stored items. We record the largest number of elements retained simultaneously by each implementation on the complete five-budget Epinions grid. For each budget we first average over the five stream orders when applicable, and then average over budgets. MultiPass stores 8,018 elements on average and TwoPass stores 12,461, compared with 40,091 for FairBase+Density and 40,075 for FairBase+1Swap. This counter measures retained elements, not bytes used internally by the value oracle.

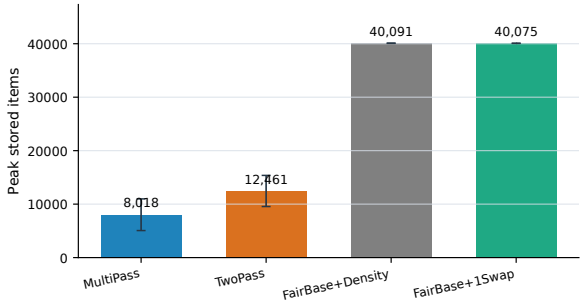


Figure 7: Peak stored elements on Epinions, averaged over five budgets.

Runtime and oracle calls. We compare TwoPass, MultiPass, and Cui+ExactRepair on Epinions and Pokec for $B \in \{20, 40, 80, 120, 160\}$, using five shared stream orders for each dataset–budget pair. All 150 returned solutions are exactly feasible. As shown in Figure 8, TwoPass uses a similar number of logical value-oracle calls to Cui+ExactRepair; on Pokec, the ratio ranges from 0.89 to 2.21. MultiPass uses substantially more oracle calls because it maintains several threshold instances over multiple passes. In wall-clock time, MultiPass is faster than Cui+ExactRepair at the two smallest budgets on both networks, but becomes slower as the budget grows. TwoPass is the slowest of the three implementations, especially at larger budgets.

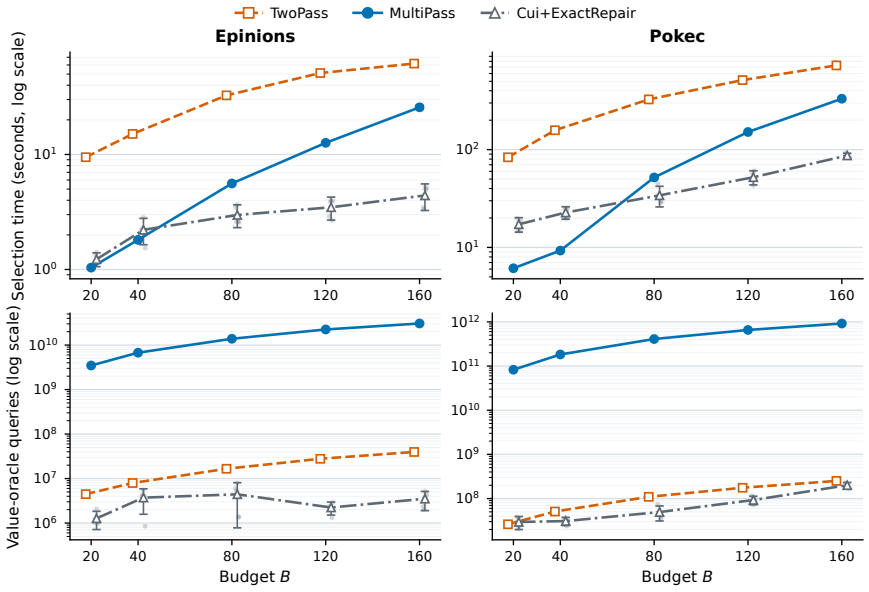


Figure 8: Selection time and logical value-oracle calls on Epinions and Pokec. Points show means over five shared stream orders; error bars are two-sided 95% Student-t confidence intervals. Both vertical axes use logarithmic scales.

D INFLUENCE-MAXIMIZATION CONTEXT AND ILLUSTRATIONS

The following background and examples distinguish influence spread, group representation among seeds, and the candidate-stream model. The diagrams illustrate the problem setting rather than empirical results. Information diffusion has been studied from several perspectives: in sociology through innovation diffusion and threshold models (Stacks & Salwen, 2014; Granovetter, 1978; Valente, 1996; Centola, 2010), in marketing through word of mouth and viral marketing models (Goldenberg et al., 2001; Domingos & Richardson, 2001; Richardson & Domingos, 2002), and in network science through models of diffusion over graphs and seed selection (Kempe et al., 2003).

Once diffusion is modeled on a social graph, the central problem is to find a small set of seed nodes that maximizes the spread of influence. This is the influence maximization problem: given a graph $G = (V, E)$ and a seed budget k, choose a seed set $S \subseteq V$ with $|S| \leq k$ to maximize the expected influence spread $f(S)$, namely the expected number of nodes activated at the end of the propagation process under a given diffusion model (Kempe et al., 2003; Chen et al., 2009). As illustrated in Figure 9, orange nodes form the seed set S, blue nodes are activated nodes, and arrows indicate influence propagation along social ties (Kempe et al., 2003).

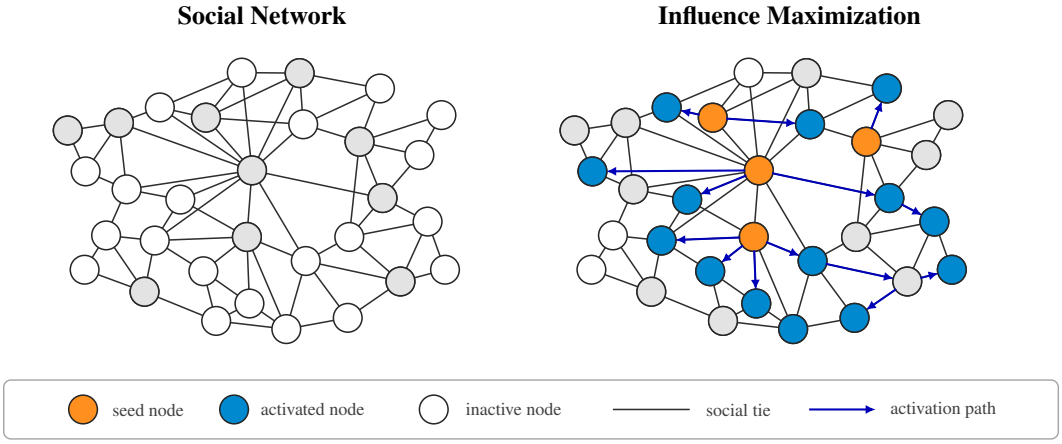


Figure 9: Influence maximization

For the independent cascade model and the linear threshold model, Kempe et al. (2003) showed that the expected influence spread function f is monotone and submodular. Monotonicity means that adding more seeds cannot decrease the expected spread, while submodularity captures diminishing returns: adding a node to a smaller seed set can bring more additional spread than adding the same node to a larger seed set. Thus, influence maximization is a canonical instance of monotone submodular maximization. By the result of Nemhauser et al. (1978), the monotonicity and submodularity of f imply that the greedy algorithm achieves a $(1 - 1/e)$ approximation when at most k seeds can be selected.

The classical influence maximization model uses the cardinality constraint $|S| \leq k$, which limits only the number of selected seed nodes (Kempe et al., 2003). This model is natural when all seeds have the same cost, but it is less suitable for budgeted campaigns, where different users may have different incentive or recruitment costs (Nguyen & Zheng, 2013; Bian et al., 2020). Budgeted influence maximization captures this setting by assigning a cost c_v to each node v and requiring the seed set to satisfy $\sum_{v \in S} c_v \leq B$. In submodular optimization, this gives a monotone submodular maximization problem under a knapsack constraint (Sviridenko, 2004; Kulik et al., 2013). However, the knapsack constraint alone does not control how the selected seeds are distributed across different groups or communities. When the objective is only to maximize the total influence spread, the selected seeds may concentrate in a large or well-connected community, leaving other communities with little representation. Fairness has therefore received increasing attention in influence maximization (Tsang et al., 2019; Fish et al., 2019; Stoica et al., 2020; Ali et al., 2021; Becker et al., 2022).

In this paper, we consider fairness directly at the seed-selection stage. The nodes are partitioned into groups $G_1, \ldots, G_h$, and the selected seed set is required to satisfy $\ell_i \leq |S \cap G_i| \leq u_i$ for every group G_i. Together with the knapsack constraint $\sum_{v \in S} c_v \leq B$, the goal is to maximize the expected influence spread while ensuring prescribed representation of every group among the selected seeds. We refer to this problem as *budgeted fair influence maximization* (BFIM), where fairness is imposed on the composition of the selected seed set.

Figure 10 illustrates the role of the fairness constraints. Without fairness constraints, the selected seeds may all come from the majority community. With group lower bounds, the seed set is required to include nodes from the minority communities as well.

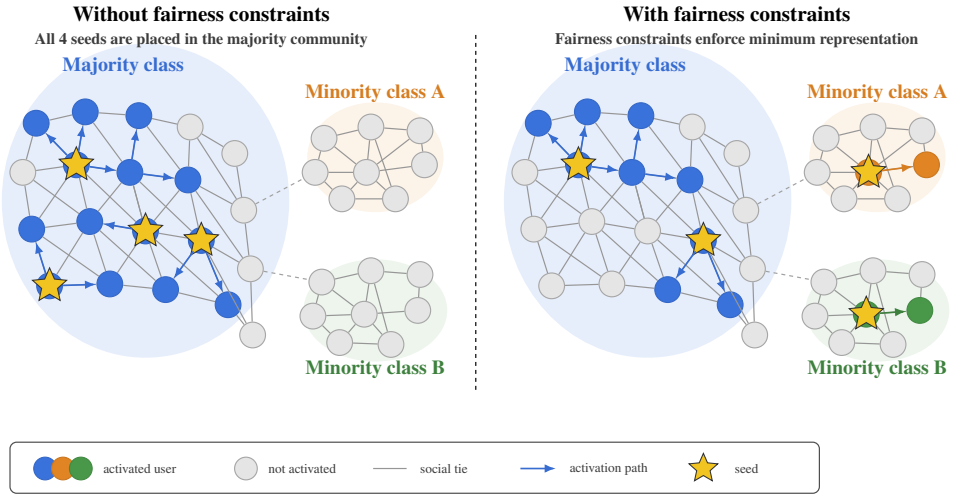


Figure 10: Fairness constraints enforce minimum representation.

The formulation above assumes that all candidate seed nodes can be stored and processed together. For large-scale instances, however, storing the entire candidate set may be impractical. Motivated by streaming submodular maximization (Badanidiyuru et al., 2014; Kazemi et al., 2019), we consider the streaming version of BFIM. The social network and diffusion model are fixed, while candidate seed nodes are processed sequentially using limited memory and only one or a few passes over the input. The algorithm must identify nodes with large marginal influence while accounting for heterogeneous seed costs and satisfying all fairness constraints.

Since the expected influence spread is monotone and submodular (Kempe et al., 2003), streaming BFIM is a special case of a more general problem: streaming monotone submodular maximization subject to one knapsack constraint and fairness constraints. This general problem is formalized in Definition 2.1.

Streaming BFIM is obtained directly from Definition 2.1 by taking $N = V$, setting $w(v) = c_v$ for every candidate node $v \in V$, and letting $f(S)$ be the expected influence spread generated by the seed set S. Therefore, under value-oracle access to the expected influence spread, every approximation, feasibility, and memory guarantee established for S-FSMK immediately yields the corresponding guarantee for streaming BFIM under the independent cascade and linear threshold models.

Figure 1 illustrates this specialization. Candidate seed nodes arrive sequentially, and each node has a cost and belongs to one group. The streaming algorithm processes these candidates using limited memory and returns a seed set satisfying both the knapsack constraint and the group representation constraints, while maximizing the expected influence spread.

D.1 Extended related work

Kempe et al. (2003) gave the seminal formulation and approximation framework for influence maximization. Subsequent work developed efficient greedy and heuristic methods (Chen et al., 2009; 2010) and scalable sampling-based methods (Borgs et al., 2014; Tang et al., 2014; 2015). Fairness has also received increasing attention in influence maximization (Chowdhary et al., 2024; Ma et al.,

2024; Wang et al., 2024; Rui et al., 2025; Wang et al., 2025). Much of this literature defines fairness through the diffusion outcome or its distribution, for example by balancing outreach across groups, improving the reachability of poorly reached individuals, or giving greater weight to groups with low utility. These approaches therefore evaluate fairness through the influence received by individuals or groups. Our fairness requirement, in contrast, is imposed at the seed-selection stage.

Beyond the influence-maximization application, our work is closely related to streaming submodular maximization and fair submodular maximization. For streaming monotone submodular maximization under a cardinality constraint k, Badanidiyuru et al. (2014) obtained a one-pass $(1/2-\varepsilon)$ approximation using $O(k\log k/\varepsilon)$ memory. Kazemi et al. (2019) subsequently reduced the memory requirement to $O(k/\varepsilon)$ while retaining the same one-pass $(1/2-\varepsilon)$ guarantee. Beyond cardinality constraints, the framework of Chakrabarti & Kale (2015) yields a one-pass algorithm for monotone submodular maximization under a matroid constraint with approximation guarantee $1/4$ and $O(r(\log|E|)^{O(1)})$ memory, where r denotes the matroid rank. Huang et al. (2020) subsequently improved the approximation guarantee to $1/(2+\varepsilon)$ using $O(1/\varepsilon)$ passes and storing $O(r)$ elements. More recently, Feldman et al. (2026) obtained a one-pass 0.3178-approximation and a multi-pass $(1-1/e-\varepsilon)$ approximation using $\widetilde{O}(r)$ memory. Fair submodular maximization in the streaming setting was studied by El Halabi et al. (2020) under a cardinality constraint. For monotone functions, they obtained a $1/2$-approximation using $k^{O(k)}$ memory and a $1/4$-approximation using $O(k)$ memory. El Halabi et al. (2023) subsequently extended the setting to a matroid constraint, establishing trade-offs between solution quality, fairness, and memory. The corresponding offline matroid setting was studied by El Halabi et al. (2024).

The works most closely related to ours study fair submodular maximization under a knapsack constraint. Cui et al. (2024) gave a two-pass streaming algorithm with approximation ratio $1/12-O(\varepsilon)$ for monotone submodular functions using $O(K)$ memory, where $K=\sum_{i=1}^{h}u_i$ and u_i is the upper fairness bound for group G_i. Their algorithm satisfies the knapsack constraint exactly, but relaxes the fairness lower bounds to $\lfloor \ell_i/2\rfloor \le |S\cap G_i|\le u_i$. For the offline problem, Li et al. (2025) obtained a $1/2(1-1/e-\varepsilon)$-approximation, satisfying both the knapsack and fairness constraints when the number of groups is constant. For an arbitrary number of groups, they obtained a $(1-1/e-\varepsilon)$ approximation when one of the two constraint families is required only in expectation. Zhou et al. (2026) subsequently considered an arbitrary number of groups while satisfying both the knapsack and fairness constraints exactly, and obtained an approximation ratio of $\frac{1-e^{-3}}{4-e^{-3}}\approx 0.241$.