FULLY DYNAMIC REGULARIZED SUBMODULAR MAXIMIZATION WITH POLYLOGARITHMIC UPDATE

Anonymous authors
Paper under double-blind review

ABSTRACT

Regularized submodular maximization provides a principled framework for cost-aware subset selection in machine learning and has attracted substantial research attention. Meanwhile, fully dynamic submodular maximization has received increasing attention for settings in which elements may be inserted or deleted over time. However, existing fully dynamic algorithms are designed for nonnegative submodular objectives and therefore do not apply to potentially negative regularized objectives, while existing algorithms for regularized submodular maximization do not support arbitrary deletions. To fill this gap, we study regularized submodular maximization under a cardinality constraint in the fully dynamic setting. We propose a novel randomized algorithm, BATCHREG, that achieves an approximation ratio of $\frac{3-\sqrt{5}}{2} - \epsilon$ under the commonly adopted regularized approximation notion, while using $\mathrm{poly}(\log n, \log k, 1/\epsilon)$ expected amortized oracle queries per update, where n is the maximum number of active elements and k is the cardinality constraint. To the best of our knowledge, this is the first algorithm to achieve a nontrivial approximation guarantee for fully dynamic regularized submodular maximization with polylogarithmic update complexity. We further conduct extensive experiments on the maximum coverage problem using real-world datasets, and the results demonstrate that our algorithm maintains high-quality solutions while supporting efficient dynamic updates.

1 INTRODUCTION

Submodular maximization provides a fundamental framework for subset selection in machine learning, as its diminishing-returns property naturally captures coverage, representativeness, and diversity. It has been widely applied to recommendation (Wang et al., 2021; El Halabi et al., 2020; Nikolakaki et al., 2021), influence maximization (Kempe et al., 2003; Stoica et al., 2020), data summarization (Fan et al., 2025; Nguyen et al., 2025; Renduchintala et al., 2024; Bhatt et al., 2024), and many other machine learning tasks.

In many applications, however, selecting an element not only provides utility but also incurs a cost, such as acquisition, storage, computation, or service cost. This motivates *regularized submodular maximization*, which seeks to solve $\max_{S \subseteq V,\, |S| \le k} f(S) - \ell(S)$, where f is a monotone submodular utility function and ℓ is a nonnegative modular cost. Regularized submodular maximization provides a natural way to balance utility and cost in applications such as profit maximization, cost-aware recommendation, and data summarization, and thus has received considerable attention (Sviridenko et al., 2017; Harshaw et al., 2019; Feldman, 2021; Nikolakaki et al., 2021; Kazemi et al., 2021; Cui et al., 2023; Zhu et al., 2024; Guo et al., 2026). A fundamental difficulty is that the regularized objective $f - \ell$ may take negative values even when f is nonnegative and monotone. Existing hardness results imply that no constant-factor multiplicative approximation guarantee is possible in polynomial time for maximizing a potentially negative submodular function, with or without constraints (Nikolakaki et al., 2021). Consequently, the regularized submodular maximization literature typically adopts a weaker approximation notion and seeks a solution S satisfying $f(S) - \ell(S) \ge \rho f(O) - \ell(O)$, where O is an optimal solution and ρ is the approximation ratio (Sviridenko et al., 2017; Harshaw et al., 2019; Feldman, 2021; Nikolakaki et al., 2021).

Along a separate line of research, there has been growing interest in *fully dynamic submodular maximization*, motivated by applications in which the underlying dataset evolves over time. In the standard fully dynamic model, elements may be inserted into or deleted from the current ground set, and the algorithm is required to efficiently maintain a high-quality solution after every update. Monemizadeh (2020) initiated the study of this model and proposed an algorithm for monotone sub-

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.

modular maximization under a cardinality constraint with polynomial amortized query complexity $\mathrm{poly}(k, \log n, 1/\epsilon)$, where n is the maximum number of active elements, k is the cardinality constraint, and ϵ is the error parameter. Subsequent work reduced the amortized query complexity to efficient polylogarithmic complexity $\mathrm{poly}(\log k, \log n, 1/\epsilon)$ and extended fully dynamic algorithms to more general constraints (Dütting et al., 2023; Banihashem et al., 2023a;b; 2025; 2026; Dütting et al., 2026). Nevertheless, these algorithms optimize nonnegative submodular objectives and therefore do not directly apply to the regularized objective $f - \ell$. Conversely, existing algorithms for regularized submodular maximization do not support the fully dynamic setting, where elements may be arbitrarily deleted over time. A straightforward workaround is to rerun a regularized submodular maximization algorithm on the current active elements after relevant deletions, but doing so requires rescanning all active elements and thus defeats the purpose of efficient dynamic maintenance. Recently, Qiang & Liu (2026) considered dynamic submodular maximization under a model different from the standard setting above, where the objective is a DR-submodular function defined over the integer lattice. However, their algorithm has limited practical applicability, since their result relies on the impractical assumption that the optimal value is known in advance and has undesirable polynomial update complexity. Therefore, an efficient polylogarithmic-update algorithm for fully dynamic regularized submodular maximization is still missing.

Our Contributions. To fill this gap, we study **f**ully **d**ynamic **r**egularized **s**ubmodular maximization under a **c**ardinality constraint (abbreviated as the FDRSC problem) and propose a novel randomized algorithm, BATCHREG. Under the commonly adopted approximation notion defined above, our algorithm achieves an approximation ratio of $(3 - \sqrt{5})/2 - \epsilon$ using $\mathrm{poly}(\log n, \log k, 1/\epsilon)$ expected amortized oracle queries per update. To the best of our knowledge, this is the first algorithm that achieves a nontrivial approximation guarantee for fully dynamic regularized submodular maximization with polylogarithmic update complexity.

We also conduct extensive experiments on the maximum coverage problem using real-world datasets. The results demonstrate that our algorithm maintains high-quality solutions while supporting efficient dynamic updates.

Technical Overview. A natural starting point is to bring the distorted objective $D = f - a\ell$, as commonly used in prior work on regularized submodular maximization, to the fully dynamic setting. However, the distorted objective used for selection does not directly characterize an element's contribution under the original regularized objective. In particular, since $F(e \mid G) = D(e \mid G) + (a-1)\ell_e$, elements with similar distorted marginal gains may still incur substantially different losses under F when their costs differ. Consequently, marginal-gain bucketing alone is insufficient in our setting. Our first ingredient is therefore a regularization-aware source construction that jointly groups candidate elements by their distorted marginal gains and costs. This allows us to control the deletion loss under F, while selecting a largest joint bucket also ensures sufficient progress on the candidate set.

A second challenge is to achieve sufficient progress at each level without violating the marginal-gain threshold. Selecting one element at a time may create too many levels, whereas adding an entire random batch directly is unsafe because earlier selected elements may decrease the marginal gains of later ones. We therefore construct each level from a filtered ordered batch, accepting an element only if its distorted marginal gain remains above the threshold. We further use shared random samples to estimate the batch size efficiently, allowing multiple candidate batch sizes to be evaluated simultaneously. This construction provides the progress needed to bound the expected number of levels while maintaining the value certificate required by the approximation analysis.

These components are incorporated into a layered dynamic structure with local suffix reconstruction, so that rebuilding is triggered only after sufficiently many updates have accumulated. Relative to existing layered dynamic frameworks, the main new ingredients of BATCHREG are the joint distorted-gain/cost source construction, the filtered batch mechanism, and the shared-sample batch-size estimator, together with the corresponding regularized deletion analysis. Thanks to these techniques, we obtain a $((3 - \sqrt{5})/2 - \epsilon)$-approximation using $\mathrm{poly}(\log n, \log k, 1/\epsilon)$ expected amortized oracle queries per update.

2 RELATED WORK

Regularized Submodular Maximization. Following the seminal work of Sviridenko et al. (2017), regularized submodular maximization has received considerable attention (e.g., Sviridenko et al.

(2017); Harshaw et al. (2019); Feldman (2021); Zhu et al. (2024); Guo et al. (2026)). Among them, (Nikolakaki et al., 2021; Kazemi et al., 2021; Cui et al., 2023) studied the insertion-only streaming models, which can be viewed as special cases of the fully dynamic model considered in this paper: elements arrive sequentially, but arbitrary deletions are not supported. More specifically, both (Nikolakaki et al., 2021) and (Kazemi et al., 2021) studied the problem under a cardinality constraint and developed algorithms achieving a $((3-\sqrt{5})/2-\epsilon)$-approximation, with $O(\epsilon^{-1}\log k)$ oracle queries per inserted element. (Cui et al., 2023) further considered the more general κ-system constraint, and proposed a streaming algorithm achieving a $1/O(\kappa \log k)$-approximation with $O(\log^2 k)$ oracle queries per inserted element.

Fully Dynamic Submodular Maximization. Monemizadeh (2020) and Lattanzi et al. (2020) independently initiated the study of fully dynamic submodular maximization, both obtaining a $(1/2-\epsilon)$-approximation for monotone submodular maximization under a cardinality constraint. The former has polynomial update complexity of $\mathrm{poly}(k, \log n, 1/\epsilon)$, whereas the latter claimed a more efficient polylogarithmic update complexity of $\mathrm{poly}(\log k, \log n, 1/\epsilon)$. However, Banihashem et al. (2023a) subsequently identified an error in the theoretical analysis of Lattanzi et al. (2020) and proposed a new algorithm that restored polylogarithmic expected amortized query complexity. Later work further extended fully dynamic submodular maximization to more general objective functions and constraints (Banihashem et al., 2023b; 2025; 2026). Nevertheless, as discussed above, these works assume that the objective function is nonnegative and therefore do not provide non-trivial approximation guarantees for regularized submodular objectives that may take negative values. Therefore, their algorithms and analyses cannot be directly applied to the FDRSC problem studied in this paper.

3 Preliminaries

Let $\mathcal{N}$ be a finite ground set. For a set function $h: 2^{\mathcal{N}} \to \mathbb{R}$, an element $e \notin S$, and sets $A, S \subseteq \mathcal{N}$, we define the marginal gains as $h(e \mid S) := h(S \cup \{e\}) - h(S)$.

Definition 3.1 (Submodular function). *A set function $f: 2^{\mathcal{N}} \to \mathbb{R}_{\geq 0}$ is* submodular *if $f(e \mid X) \geq f(e \mid Y)$ for all $X \subseteq Y \subseteq \mathcal{N}$ and $e \in \mathcal{N} \setminus Y$. It is* monotone *if $f(X) \leq f(Y)$ whenever $X \subseteq Y$, and* normalized *if $f(\varnothing) = 0$.*

Definition 3.2 (Modular function). *A set function $\ell: 2^{\mathcal{N}} \to \mathbb{R}_{\geq 0}$ is* modular *if $\ell(e \mid X) = \ell(e \mid Y)$ for all $X \subseteq Y \subseteq \mathcal{N}$ and $e \in \mathcal{N} \setminus Y$. When $\ell(\varnothing) = 0$, we have $\ell(S) = \sum_{e \in S} \ell_e$ where $\ell_e := \ell(\{e\}) \geq 0$.*

In this paper, we consider a nonnegative, normalized, and monotone submodular function f, together with a nonnegative normalized modular cost function ℓ. Following the standard fully dynamic model used in prior work on submodular maximization (Monemizadeh, 2020; Lattanzi et al., 2020; Banihashem et al., 2023a;b), we consider an update sequence $\mathcal{I} = (\sigma_1, \sigma_2, \ldots)$ consisting of element insertions and deletions, and the update sequence is *oblivious*, i.e., $\mathcal{I}$ is fixed independently of the internal randomness of the algorithm. Let $V_t \subseteq \mathcal{N}$ denote the set of active elements after the first t updates, and let $n := \max_t |V_t|$ denote the maximum number of active elements over the update sequence. The fully dynamic regularized submodular maximization problem under a cardinality constraint is defined as $\max_{S \subseteq V_t,\ |S| \leq k} F(S) = \max_{S \subseteq V_t,\ |S| \leq k} \{f(S) - \ell(S)\}$ at each time t. We use O_t to denote an optimal solution, i.e., $O_t \in \arg\max\{F(S) : S \subseteq V_t,\ |S| \leq k\}$ at time t. Consistent with prior work on dynamic submodular maximization (Monemizadeh, 2020; Lattanzi et al., 2020; Banihashem et al., 2023a), we assume a value oracle query to the submodular function f and measure computational efficiency by the number of oracle queries. An algorithm has *expected amortized query complexity* q if, for every prefix of L updates, the expected total number of oracle queries is at most Lq.

4 Algorithms for the FDRSC Problem

In this section, we present BatchReg, our fully dynamic algorithm for the FDRSC problem. We first describe the procedure for a fixed value of H. It maintains a layered structure constructed through source bucketing and randomized batch selection, and supports dynamic updates via suffix reconstruction. We then remove the requirement of knowing an appropriate value of H in advance by maintaining structures for a geometric sequence of estimates in Section 4.2. The complete pseudocode of BatchReg is given in Algorithm 4 in Appendix A; we defer this outer maintenance procedure because it mainly coordinates the geometric estimates and the corresponding outputs.

For clarity and notational convenience, we define the distorted objective $D(S) = f(S) - a\ell(S)$, where $a = (3+\sqrt{5})/2$. Equivalently, for each element e, we define its scaled cost as $c_e = (a-1)\ell_e$, such that $D(S) = F(S) - c(S)$, where $c(S) = \sum_{e \in S} c_e$. We use D to guide element selection, while the original objective $F(S) = f(S) - \ell(S)$ is used to evaluate the maintained solutions. At time t, let $\Delta_t = \max\{D(S) : S \subseteq V_t,\ |S| \le k\}$. For any set function h used below, write $h(A \mid G) = h(A \cup G) - h(G)$. The parameters $\epsilon_o, \epsilon_b, \epsilon_s, \epsilon_d$ control estimation, bucketing, sampling, and deletion, respectively; δ_c is the batch-size failure parameter. We use the ranges in Appendix A.1 and the final choices in equation 2. Each insertion uses a fresh record; reinserting an underlying item does not reactivate its old record. Appendix C.2 explains this convention and its oracle implementation.

4.1 Algorithm for a Fixed Value of H

For a fixed positive estimate H of Δ_t, define $\tau = \frac{H}{ak}$ and $b(H) = \frac{a-1}{a}(\frac{H}{1+\epsilon_o} - \frac{H}{a})$. The procedure is defined for every $H > 0$; its approximation analysis uses $\Delta_t \le H \le (1+\epsilon_o)\Delta_t$. The threshold τ determines the minimum distorted marginal gain required for an element to enter this instance, while $b(H)$ denotes the F-value threshold corresponding to the approximation guarantee of the H-instance. By submodularity, if $D(\{e\}) < \tau$, then for any subsequent prefix G not containing e, we have $D(e \mid G) < \tau$. Therefore, element e can be ignored by this instance. Moreover, if there exists an active element e satisfying $F(\{e\}) \ge b(H)$, then this singleton solution already achieves the required target quality. Hence, the algorithm maintains the best singleton separately, without including such elements in the H-instance. Therefore, only elements satisfying both $\tau \le D(\{e\}) \le H$ and $F(\{e\}) < b(H)$ are assigned to the corresponding H-instance. For a suitable estimate, $D(\{e\}) > H$ is impossible for an active element, since a singleton is feasible and $\Delta_t \le H$. Furthermore, for any element e assigned to this instance, since $D(\{e\}) = F(\{e\}) - c_e$, we have $0 \le c_e < F(\{e\}) < b(H) \le k\tau$. This property ensures that every element admitted to the H-instance has a bounded regularized gain and cost scale, which facilitates the subsequent layered structure and approximation analysis.

Layered Structure. The elements assigned to the H-instance are organized into T nonempty levels. The root set P_0 contains all active elements currently assigned to the instance. Each level i consists of a construction-time candidate set P_i, which remains fixed throughout the current version of the level, a source bucket $C_i \subseteq P_i$, an ordered sample U_i drawn from C_i, and an accepted set $A_i \subseteq U_i$. The levels are constructed recursively through the historical prefix. Let $G_0 = \varnothing$ and define $G_i = G_{i-1} \cup A_i$, so G_{i-1} contains all elements accepted before level i is constructed. During construction or reconstruction, we use the following filter to construct the next candidate set:

$$\textsc{Eligible}(P, G) = \begin{cases} \{e \in (P \cap V_t) \setminus G : D(e \mid G) \ge \tau\}, & |G| < k, \\ \varnothing, & |G| = k. \end{cases} \tag{1}$$

After constructing level i, we set $P_{i+1} = \bar{P}_{i+1} = \textsc{Eligible}(P_i, G_i)$. Thus, when $|G_i| < k$, only elements that remain active, have not yet been accepted, and have distorted marginal gain at least τ are retained for the next level. The process terminates when the candidate set becomes empty; T denotes the last nonempty level, while $T+1$ is kept as an empty terminal slot for future insertions.

Since P_{i+1} depends on the historical prefix G_i, a change at an earlier level may affect all subsequent levels. To handle insertions without immediately reconstructing the entire structure, we additionally maintain an expanded candidate set $\bar{P}_i \supseteq P_i$ at each level and propagate a new element into it only while $|G_{i-1}| < k$ and $D(e \mid G_{i-1}) \ge \tau$. These insertions leave P_i unchanged. For deletions, inactive elements are not immediately removed from G_i, since doing so would change the marginal gains on which subsequent levels are constructed. Therefore, G_i may temporarily contain inactive elements, and the instance returns the better of $\varnothing$ and $G_T \cap V_t$ under F at time t.

Source Bucket Construction. To determine the source bucket C_i at level i, we partition the candidate set P_i into joint buckets. Since $F(e \mid G_{i-1}) = D(e \mid G_{i-1}) + (a-1)\ell_e$, elements with similar distorted marginal gains may still have substantially different marginal contributions under F when their costs differ. Thus, bucketing only by distorted marginal gain is insufficient for controlling the F-value loss caused by deletions. We therefore partition P_i according to $D(e \mid G_{i-1})$ and $(a-1)\ell_e$, and select C_i from these joint buckets.

Specifically, the distorted marginal gains are partitioned into geometric intervals $D(e \mid G_{i-1}) \in [\eta_p, (1+\epsilon_b)\eta_p)$, where $\eta_p = \tau(1+\epsilon_b)^p$. The costs are partitioned similarly, except that we introduce

Algorithm 1 Batch selection for a source C and a selected prefix G

function SCANBATCH($(u_1, \dots, u_m), G, \eta$)
  $A \leftarrow \varnothing$
  for $j = 1, \dots, m$ **do**
    if $D(u_j \mid G \cup A) \geq \eta$ **then** append u_j to A
  return A
function BATCHSIZE(C, G, η, r)
  $M \leftarrow \min\{|C|, \lceil r/2 \rceil\}$
  if $M = 1$ **then return** 1
  $q \leftarrow \lceil 16\epsilon_s^{-1} \ln(2M/\delta_c) \rceil$; $Z_1, \dots, Z_M \leftarrow 0$
  for $h = 1, \dots, q$ **do**
    draw an independent uniform ordered M-sample $(w_1, \dots, w_M)$ without replacement from C
    $A \leftarrow \varnothing$
    for $j = 1, \dots, M$ **do**
      if $D(w_j \mid G \cup A) \geq \eta$ **then**
        append w_j to A
      else
        $Z_j \leftarrow Z_j + 1$
  $h^\star \leftarrow \min\{j : Z_j/q > 3\epsilon_s/2\}$, or $M + 1$ if no such index exists
  return $\min\{M, h^\star - 1\}$

a separate low-cost interval $[0, \epsilon_b\tau)$ and use $c_e \in [\zeta_q, (1+\epsilon_b)\zeta_q)$, with $\zeta_q = \epsilon_b\tau(1+\epsilon_b)^q$, for elements with $c_e \geq \epsilon_b\tau$. This avoids making the number of buckets depend on the smallest positive cost. With nonnegative integer indices, the two partitions contain at most $L_D = 1 + \lceil \log_{1+\epsilon_b}(ak) \rceil$ and $L_c = 1 + \lceil \log_{1+\epsilon_b}(k/\epsilon_b) \rceil$ groups, respectively. Combining them yields at most $J = L_D L_c$ joint buckets. At each level i, we select a largest nonempty joint bucket as the source bucket C_i, and denote the lower endpoint of its distorted-marginal-gain interval by η_i. Since there are at most J joint buckets, the largest source bucket satisfies $|C_i| \geq |P_i|/J$. This lower bound connects progress within C_i to progress on the entire candidate set P_i, and is later used to bound both the number of levels and the reconstruction cost. The pair (C_i, η_i) determines the batch-selection procedure.

Batch Selection. After determining the source bucket C_i and its threshold η_i, we select a batch of elements from C_i to construct level i. Since D is submodular, accepting earlier elements may decrease the distorted marginal gains of later ones. We therefore use SCANBATCH to ensure that each element satisfies the threshold condition when it is accepted, and BATCHSIZE to choose a batch size that balances threshold feasibility and per-level progress. Algorithm 1 gives the SCANBATCH and BATCHSIZE procedures used for batch selection.

Every call has $C \neq \varnothing$, $C \cap G = \varnothing$, $r = k - |G| \geq 1$, and $D(e \mid G) \geq \eta > 0$ for $e \in C$. All ordered samples are without replacement. Accepted sets retain the order induced by the sample. For an ordered batch $U = (u_1, \dots, u_m)$, SCANBATCH examines the elements sequentially. Let A denote the set of elements accepted so far. An element u_j is accepted only if $D(u_j \mid G_{i-1} \cup A) \geq \eta_i$. Rechecking the threshold after each acceptance is necessary because previously accepted elements may decrease the distorted marginal gains of the remaining elements. Consequently, every element in A_i satisfies the required threshold at the moment it is accepted, even though the accepted subset itself need not be uniformly distributed. To determine the batch size, let J_m denote the probability that the last element of a uniform ordered m-sample from C_i passes threshold η_i after SCANBATCH processes the first $m-1$ elements from G_{i-1}. By submodularity, J_m is nonincreasing in m. We restrict the batch size to $M_i = \min\{|C_i|, \lceil \frac{k-|G_{i-1}|}{2} \rceil\}$, which accounts for both the size of the source bucket and the remaining cardinality capacity. BATCHSIZE uses shared random samples to estimate the rejection probabilities at different positions simultaneously and truncates the batch before the estimated rejection probability becomes too large, yielding a batch size m_i. Sharing the same samples across positions avoids estimating each candidate batch size separately. Finally, after determining m_i, we draw a fresh independent ordered batch U_i of size m_i uniformly from C_i, and apply SCANBATCH to obtain A_i. The actual batch is sampled independently of the trials used to determine m_i; this separation is important because the deletion analysis conditions on the batch-size decision while requiring U_i to retain its uniform sampling law.

Algorithm 2 Initialization and reconstruction of one instance

procedure StartInstance(e, H)
  set $\tau = \frac{H}{ak}, b(H) = \frac{a-1}{a}\left(\frac{H}{1+\epsilon_o} - \frac{H}{a}\right)$; $G_0 \leftarrow \varnothing$; $P_0 \leftarrow \{e\}$
  $P_1 \leftarrow \bar{P}_1 \leftarrow$ Eligible(P_0, G_0); $T \leftarrow 0$; Rebuild(1)
procedure Rebuild(i)
  $Q \leftarrow \bar{P}_i \cap V_t$ ▷ save the input before discarding its records
  discard levels $i, \ldots, T+1$ and their references
  $P_i \leftarrow \bar{P}_i \leftarrow Q$; $j \leftarrow i$
  while $P_j \neq \varnothing$ **do**
    group P_j by $D(e \mid G_{j-1})$ and c_e
    choose a largest bucket C_j; let η_j be its D-interval lower endpoint
    $z_j \leftarrow 0$; $r_j \leftarrow k - |G_{j-1}|$; $M_j \leftarrow \min\{|C_j|, \lceil r_j/2 \rceil\}$
    $m_j \leftarrow$ BatchSize($C_j, G_{j-1}, \eta_j, r_j$)
    draw U_j, a fresh uniform ordered m_j-sample without replacement from C_j
    $A_j \leftarrow$ ScanBatch(U_j, G_{j-1}, η_j); $G_j \leftarrow G_{j-1} \cup A_j$
    $P_{j+1} \leftarrow \bar{P}_{j+1} \leftarrow$ Eligible(P_j, G_j); $j \leftarrow j+1$
  $T \leftarrow j-1$

Initialization and Reconstruction. Building on the above single-level construction, we now present the initialization and reconstruction procedures for an H-instance, as shown in Algorithm 2. When reconstruction is triggered at level i, Rebuild(i) preserves the first $i-1$ levels and reconstructs the subsequent levels starting from level i. Specifically, Rebuild(i) takes the currently active elements in the expanded candidate set at level i as the new candidate set, discards the existing level i and all subsequent levels, and repeatedly applies the single-level construction procedure until the candidate set becomes empty. Initialization is a special case of this procedure: after initializing the root active candidate set and the first candidate set, invoking Rebuild(1) constructs the layered structure.

Insertions and Deletions. Algorithm 3 gives the update procedures. An inserted element assigned to the current H-instance is first added to P_0 and then propagated through the levels for which it remains eligible, where it is added to the corresponding expanded candidate sets. To avoid reconstruction after every insertion, these elements are accumulated until $|\bar{P}_i| \geq \frac{3}{2}|P_i|$; reconstruction is also triggered when an inserted element reaches the terminal slot $T+1$. For an existing level, this growth condition ensures that sufficiently many insertions have accumulated before reconstruction, allowing the reconstruction cost to be amortized over these updates.

For deletions, the global procedure first marks e inactive, and Delete(e) removes it from P_0, while the existing layered structure is retained until reconstruction is triggered. For each level i, z_i counts the elements deleted from its source bucket C_i since the level was constructed. Once $z_i \geq \epsilon_d |C_i|$, Rebuild($i$) reconstructs level i and all subsequent levels. This condition is used to bound the deletion-induced loss in the subsequent analysis. The decisions that discard level i depend on candidate growth and source deletions at levels at most i, not on U_i or A_i. Decisions at later levels may depend on A_i, but do not discard level i. This distinction preserves the sampling law under the specific histories defined in Appendix B.2; Lemma B.6 establishes the conditional uniformity.

4.2 Estimating the Value of H

The previous subsection assumes that a suitable estimate H of Δ_t is known. To remove this assumption, we consider geometric estimates $H_p = (1+\epsilon_o)^p$, where $p \in \mathbb{Z}$. For any time t with $\Delta_t > 0$, there exists an estimate H_p satisfying $\Delta_t \leq H_p \leq (1+\epsilon_o)\Delta_t$. We later show that maintaining the corresponding instance together with the best active singleton is sufficient to achieve the desired approximation guarantee. Algorithm 4 in Appendix A gives the overall multi-instance maintenance procedure. If $\Delta_t = 0$, then $a^{-1}f(O_t) - \ell(O_t) \leq 0$, so the empty-set candidate already meets the target guarantee. Although the estimate sequence is defined globally, each element is relevant to only a small number of estimates. For an inserted element e, let $x_e = D(\{e\})$. By the admission conditions in Section 4.1, only instances satisfying $x_e > 0$, $x_e \leq H_p \leq akx_e$, and $F(\{e\}) < b(H_p)$ need to process e. Since the estimates grow geometrically, each element is associated with at most $G_H = 1 + \lceil \log_{1+\epsilon_o}(ak) \rceil$ instances. Thus, the algorithm maintains instances on demand: an instance is created when it first receives an element and removed when its root active candidate set P_0 becomes empty. The element-to-instance associations are recorded so that subsequent updates are

Algorithm 3 Insertion and deletion for one estimate H

procedure INSERT(e)
  if e does not satisfy $\tau \le D(\{e\}) \le H$ and $F(\{e\}) < b(H)$ **then return**
  $P_0 \leftarrow P_0 \cup \{e\}$
  for $i = 1, \dots, T+1$ **do**
    if $|G_{i-1}| = k$ or $D(e \mid G_{i-1}) < \tau$ **then break**
    $\bar{P}_i \leftarrow \bar{P}_i \cup \{e\}$
    if $i = T+1$ or $|\bar{P}_i| \ge 3|P_i|/2$ **then**
      REBUILD(i); **break**

procedure DELETE(e) ▷ e has already been marked inactive
  $P_0 \leftarrow P_0 \setminus \{e\}$
  if $P_0 = \varnothing$ **then return** ▷ the global algorithm removes the instance
  for $i = 1, \dots, T$ **do**
    if $e \in C_i$ **then** $z_i \leftarrow z_i + 1$
    if $z_i \ge \epsilon_d |C_i|$ **then**
      REBUILD(i); **break**

function INSTANCEOUTPUT
  return the better of $\varnothing$ and $G_T \cap V_t$ under F

forwarded only to the relevant instances. After each update, the algorithm returns the solution with the largest F-value among all active instance outputs, the best active singleton, and the empty set.

5 THEORETICAL ANALYSIS

In this section, we establish the performance guarantee of our algorithm. We first analyze the approximation performance for a fixed value of H, and then extend the result to the full algorithm via geometric estimates. To keep the presentation concise and focused, we present only the main proof ideas here and defer the complete proofs to Appendix B.

Throughout this section, for an error parameter $\epsilon \in (0, 1/10]$, set

$$\alpha = \tfrac{2\epsilon}{5}, \qquad \epsilon_o = \epsilon_b = \epsilon_s = \alpha, \qquad \epsilon_d = \tfrac{\alpha}{4}, \qquad \delta_c = \tfrac{\alpha}{100k}. \tag{2}$$

The subscripts o, b, s, d correspond to estimation, bucketing, sampling, and deletion, respectively, while δ_c denotes the failure-probability parameter used in the batch-size estimation.

We first focus on a fixed value of H satisfying $\Delta_t \le H \le (1+\epsilon_o)\Delta_t$. Our goal is to show that the solution associated with this value has expected F-value at least $\kappa b(H)$ for a suitable factor κ. The argument has three main steps. First, we lower-bound the current active value by the value certified when the levels were constructed, minus the loss caused by subsequent deletions. Second, we show that each level retains a constant fraction of this construction-time lower bound in expectation. Finally, if the accumulated construction value is insufficient, termination of the layered process provides a complementary lower bound in the underfull case. These ingredients are then combined into a single bound in terms of the fixed target $b(H)$.

Fix an update time with $\Delta_t > 0$ and a suitable estimate H satisfying $\Delta_t \le H \le (1+\epsilon_o)\Delta_t$. If any active singleton has F-value at least $b(H)$, then this singleton already provides the desired value. Hence, in the remainder of the analysis, we assume $F(\{e\}) < b(H)$ for every active e. Let T be the number of nonempty levels maintained for the fixed value H. Since the accepted sets $A_1, \dots, A_T$ are disjoint and every nonempty level accepts at least one element, $T \le |G_T| \le k$.

Before analyzing the maintained value, we record the two properties that allow the construction-time argument to remain valid after dynamic updates. For any stored set X, we write $X^\circ := X \cap V_t$ for its active part at the current time t. The dynamic updates preserve the structural properties needed in the analysis. In particular, the active expanded candidate sets satisfy $\bar{P}^\circ_{i+1} = \text{ELIGIBLE}(\bar{P}_i, G_i)$, the terminal active expanded candidate set satisfies $\bar{P}^\circ_{T+1} = \varnothing$, and every current level satisfies $z_i < \epsilon_d |C_i|$. Moreover, conditional on the history $\mathcal{H}_i(t)$ defined in Appendix B.2, U_i remains a uniform ordered sample without replacement from C_i. The accuracy probability is at least $1 - \delta_c$ when conditioned on the smaller history $\mathcal{H}_i^{-m}(t)$, which also omits the count-estimation outcome. These distinct conditioning statements follow from Lemma B.6.

We first derive a pointwise lower bound on the value that survives from the constructed levels. For level i, define $v_i = |A_i|\eta_i + c(A_i)$ and $d_i = \sum_{e \in A_i \setminus V_t} \big((1+\epsilon_b)\eta_i + c_e\big)$. The quantity v_i is a lower bound on the contribution certified when level i is constructed: since every accepted element has distorted marginal gain at least η_i at the time of acceptance, $F(A_i \mid G_{i-1}) \geq v_i$. The quantity d_i, in turn, upper-bounds the contribution that may be lost when accepted elements subsequently become inactive. By submodularity, removing deleted predecessors can only increase the marginal gain of every surviving accepted element. Thus only deleted accepted elements need to be charged, and Lemma B.7 yields $F(G_T^\circ) \geq \sum_{i=1}^{T}(v_i - d_i)$. This bound is effective when the accumulated construction value is already sufficiently large, but it may be weak when only a small number of elements have been accepted. In the latter case, termination itself provides useful complementary information. Let O_D be an optimal solution for D under the cardinality constraint. If $|G_T| < k$, the admission and eligibility invariants together with submodularity imply that every $e \in O_D \setminus G_T$ has $D(e \mid G_T) < \tau$. Otherwise, such an element would remain eligible and be propagated through the layered structure to the empty terminal candidate set. Hence, by the monotonicity and submodularity of f, $f(G_T) \geq \Delta_t - k\tau$, which yields the following lemma.

Lemma 5.1. *If $|G_T| < k$, then $F(G_T) \geq b(H) + \frac{1}{a}D(G_T) \geq b(H)$.*

We next show that subsequent deletions cannot erase too much of the construction-time value in expectation. The conditional uniformity of the sampled order allows the deletion fraction of the source bucket to control the expected number of deleted sampled elements, while the joint distorted marginal gain/cost grouping bounds the loss contributed by each such element. On the other hand, when the batch-size estimate is accurate, the batch-selection rule guarantees that a sufficiently large fraction of the sampled elements is accepted in expectation. Thus, both the accepted value and the deletion loss can be compared against the same source-level quantity.

Let $\gamma = (1-2\epsilon_s)/(1+\epsilon_b)$, $\lambda_d = 1 + 2\epsilon_b$, and $\kappa = \gamma - \lambda_d\epsilon_d - \gamma k\delta_c$. Here γ captures the retained fraction on an accurate batch-size call after accounting for the bucketing slack, $\lambda_d\epsilon_d$ accounts for deletion loss, and $\gamma k\delta_c$ accounts for inaccurate batch-size estimates. Consequently, κ is the per-level retention factor. Equation 2 ensures $0 < \kappa \leq 1$. Lemmas B.8 and B.9 give the following lemma.

Lemma 5.2. *For every current level i, with $\mathcal{H}_i^{-m}(t)$ defined precisely in Appendix B.2, $\mathbb{E}[v_i - d_i \mid \mathcal{H}_i^{-m}(t)] \geq \kappa\mathbb{E}[v_i \mid \mathcal{H}_i^{-m}(t)]$.*

Summing Lemma 5.2 controls the retained value in terms of $\sum_i v_i$, but this quantity is random and need not always reach the fixed target $b(H)$. We therefore combine the two regimes above through the auxiliary quantity $\Phi = \sum_i (v_i - d_i) + \kappa\left(b(H) - \sum_i v_i\right)$. If the accumulated construction value already satisfies $\sum_i v_i \geq b(H)$, the correction term is nonpositive, and the deletion bound directly gives $F(G_T^\circ) \geq \sum_i (v_i - d_i) \geq \Phi$. Otherwise, the shortfall in construction value forces the historical prefix to be underfull: $|G_T|\tau \leq \sum_i v_i < b(H) \leq k\tau$, so $|G_T| < k$. Lemma 5.1 and the deletion-loss bound then give $F(G_T^\circ) \geq b(H) - \sum_i d_i \geq \Phi$; the complete argument is given in Lemma B.10. Thus the same pointwise lower bound holds in both regimes: $F(G_T^\circ) \geq \Phi$. Taking expectations and applying Lemma 5.2 yields $\mathbb{E}[\Phi] \geq \kappa b(H)$.

It remains to translate this fixed-H bound, which is expressed through the distorted optimum Δ_t, back to the original regularized benchmark involving O_t. Define $\beta = \kappa\frac{a-1}{a}\left(\frac{1}{1+\epsilon_o} - \frac{1}{a}\right) \leq \frac{1}{a}$. Since $H \geq \Delta_t$, we have $\kappa b(H) \geq \beta\Delta_t$. Moreover, $\Delta_t \geq f(O_t) - a\ell(O_t)$ and $a\beta \leq 1$, yielding the following theorem.

Theorem 5.3. *Let $\Delta_t > 0$ and let $H = (1+\epsilon_o)^p$, $p \in \mathbb{Z}$, satisfy $\Delta_t \leq H \leq (1+\epsilon_o)\Delta_t$. Let Q be the best under F among $\varnothing$, the best active singleton, and the maintained output associated with H, if it exists. Then $\mathbb{E}[F(Q)] \geq \kappa b(H) \geq \beta\Delta_t$. Consequently, $\mathbb{E}[F(Q)] \geq \beta f(O_t) - \ell(O_t)$.*

Theorem 5.3 establishes the approximation guarantee for any suitable value of H. The geometric-estimate argument in Appendix B guarantees that such a value is represented unless the best active singleton already meets the target. The appendix further shows that the number of maintained levels and the cost of suffix reconstructions can be bounded and amortized over the updates that trigger them. Combining these results with the parameter choices in equation 2 yields our main theorem.

Theorem 5.4. *For the FDRSC problem,* BATCHREG *maintains, after every update time t, a feasible set $S_t \subseteq V_t$ with $|S_t| \leq k$ such that $\mathbb{E}[F(S_t)] \geq \big((3-\sqrt{5})/2 - \epsilon\big) f(O_t) - \ell(O_t)$. Moreover, its expected amortized oracle query complexity is $\mathrm{poly}(\log n, \log k, 1/\epsilon)$ per update.*

6 EXPERIMENTS

In this section, we evaluate BATCHREG on a maximum coverage problem over a real-world network. To empirically examine the two aspects captured by our theoretical guarantees, namely solution quality and update efficiency, we measure the average objective value over the update sequence and the total number of oracle queries, respectively. The source code is available at `https://anonymous.4open.science/r/BatchReg-426C`.

Baselines. We compare against two state-of-the-art streaming algorithms for regularized submodular maximization under a cardinality constraint, namely STREAMING-CSG Nikolakaki et al. (2021) and THRESHOLD-STREAMING (Kazemi et al., 2021). Both algorithms are designed for the insertion-only streaming model and do not support arbitrary deletions. Thus, following (Banihashem et al., 2023b; Lattanzi et al., 2020), we adapt them to the fully dynamic setting while preserving their original approximation guarantees. Specifically, insertions follow the original streaming procedures, whereas each deletion triggers a rerun on all currently active elements in their original arrival order.

Application and Implementation. Following prior work on regularized submodular maximization (Harshaw et al., 2019; Kazemi et al., 2021), we consider the classic maximum coverage problem. Given a graph $G = (\mathcal{N}, E)$, let $N(u)$ denote the neighbor set of node u. For any $S \subseteq \mathcal{N}$, define $g(S) = \left|\bigcup_{u \in S} N(u)\right|$ as its coverage function. Following (Nikolakaki et al., 2021; Cui et al., 2023), the regularized objective function is defined as $F(S) = g(S) - \ell(S) = \left|\bigcup_{u \in S} N(u)\right| - \sum_{u \in S} \ell_u$, subject to $|S| \leq k$. Each node $u \in \mathcal{N}$ is assigned a nonnegative cost ℓ_u, sampled uniformly from $[50, 500]$. We use the Slashdot dataset from SNAP (Leskovec et al., 2009), which contains 82,168 nodes and 948,464 edges. To simulate a fully dynamic environment, we follow (Lattanzi et al., 2020; Banihashem et al., 2023b) and adopt a bounded-active-set update scheme. We first randomly permute all nodes and insert them sequentially. Once the active set size reaches $W = 3000$, each subsequent insertion is followed by the random deletion of one node from the current active set. After all nodes have been inserted, we continue deleting nodes until the active set becomes empty. We set ϵ to the default value 0.1 for all algorithms. All experiments were conducted on a Windows workstation with an Intel Core Ultra 9 285K @ 3.70 GHz CPU.

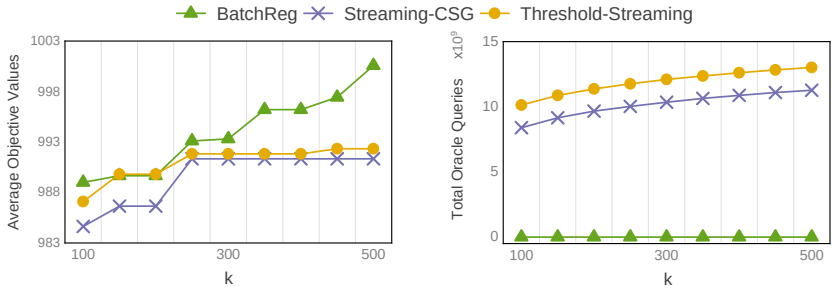


Figure 1: Maximum Coverage on Social Networks.

Experimental Results. Figure 1 presents the experimental results under different cardinality constraints k. As shown in the figure, BATCHREG achieves comparable or better objective values than STREAMING-CSG and THRESHOLD-STREAMING, indicating that it maintains high-quality solutions despite fully dynamic updates. More importantly, BATCHREG requires only about 1/14459 and 1/16922 as many oracle queries as the naive fully dynamic adaptations of STREAMING-CSG and THRESHOLD-STREAMING, respectively. These results demonstrate that BATCHREG substantially improves update efficiency without sacrificing solution quality.

7 CONCLUSION

We present BATCHREG for fully dynamic regularized submodular maximization with a provable approximation guarantee and polylogarithmic expected amortized query complexity. Experiments demonstrate competitive solution quality and efficient updates. An interesting direction for future work is to extend the framework to non-monotone regularized submodular objectives and more general feasibility constraints.

AI USE STATEMENT

Generative AI tools were used to polish the writing of the manuscript and to provide limited assistance with software implementation. All AI-assisted text and code were reviewed, edited, and verified by the authors.

REPRODUCIBILITY STATEMENT

Complete proofs of all theoretical results are provided in the appendix. A detailed description of the experimental setup, dataset, baseline methods, evaluation metrics, and parameter settings is provided in Section 6. The source code necessary to reproduce the experimental results is available in the anonymous repository linked in the paper.

REFERENCES

Kiarash Banihashem, Leyla Biabani, Samira Goudarzi, MohammadTaghi Hajiaghayi, Peyman Jabbarzade, and Morteza Monemizadeh. Dynamic constrained submodular optimization with polylogarithmic update time. In *International Conference on Machine Learning (ICML)*, volume 202, pp. 1660–1691, 2023a.

Kiarash Banihashem, Leyla Biabani, Samira Goudarzi, MohammadTaghi Hajiaghayi, Peyman Jabbarzade, and Morteza Monemizadeh. Dynamic non-monotone submodular maximization. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, 2023b.

Kiarash Banihashem, Samira Goudarzi, MohammadTaghi Hajiaghayi, Peyman Jabbarzade, and Morteza Monemizadeh. Non-monotone submodular optimization: p-matchoid constraints and fully dynamic setting. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 38, 2025.

Kiarash Banihashem, Samira Goudarzi, MohammadTaghi Hajiaghayi, Peyman Jabbarzade, and Morteza Monemizadeh. Improved dynamic algorithm for non-monotone submodular maximization under cardinality constraint. In *International Conference on Machine Learning (ICML)*, volume 306, 2026.

Gantavya Bhatt, Yifang Chen, Arnav Das, Jifan Zhang, Sang Truong, Stephen Mussmann, Yinglun Zhu, Jeff Bilmes, Simon Du, Kevin Jamieson, et al. An experimental design framework for label-efficient supervised finetuning of large language models. In *Findings of the Association for Computational Linguistics (Findings of ACL)*, pp. 6549–6560, 2024.

Xi Chen and Binghui Peng. On the complexity of dynamic submodular maximization. In *ACM Symposium on Theory of Computing (STOC)*, pp. 1685–1698, 2022.

Shuang Cui, Kai Han, Jing Tang, and He Huang. Constrained subset selection from data streams for profit maximization. In *International World Wide Web Conference (WWW)*, pp. 1822–1831, 2023.

Paul Dütting, Federico Fusco, Silvio Lattanzi, Ashkan Norouzi-Fard, and Morteza Zadimoghaddam. Fully dynamic submodular maximization over matroids. In *International Conference on Machine Learning (ICML)*, pp. 8821–8835, 2023.

Paul Dütting, Federico Fusco, Silvio Lattanzi, Ashkan Norouzi-Fard, Ola Svensson, and Morteza Zadimoghaddam. A general framework for dynamic consistent submodular maximization. In *International Conference on Machine Learning (ICML)*, 2026.

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

Ziqing Fan, Siyuan Du, Shengchao Hu, Pingjie Wang, Li Shen, Ya Zhang, Dacheng Tao, and Yanfeng Wang. Combatting dimensional collapse in llm pre-training data via submodular file selection. In *International Conference on Learning Representations (ICLR)*, 2025.

Moran Feldman. Guess free maximization of submodular and linear sums. *Algorithmica*, pp. 853–878, 2021.

Qintian Guo, Chen Feng, Jieming Shi, Jing Tang, Xiaofang Zhou, and Sibo Wang. Efficient algorithms for budgeted profit maximization with theoretical guarantees. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 2026.

Chris Harshaw, Moran Feldman, Justin Ward, and Amin Karbasi. Submodular maximization beyond non-negativity: Guarantees, fast algorithms, and applications. In *International Conference on Machine Learning (ICML)*, pp. 2634–2643. PMLR, 2019.

Ehsan Kazemi, Shervin Minaee, Moran Feldman, and Amin Karbasi. Regularized submodular maximization at scale. In *International Conference on Machine Learning (ICML)*, pp. 5356–5366, 2021.

David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pp. 137–146, 2003.

Silvio Lattanzi, Slobodan Mitrović, Ashkan Norouzi-Fard, Jakub M. Tarnawski, and Morteza Zadimoghaddam. Fully dynamic algorithm for constrained submodular optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, 2020.

Jure Leskovec, Kevin J. Lang, Anirban Dasgupta, and Michael W. Mahoney. Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters. *Internet Mathematics*, 6(1):29–123, 2009.

Morteza Monemizadeh. Dynamic submodular maximization. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, 2020.

Dang Nguyen, Wenhan Yang, Rathul Anand, Yu Yang, and Baharan Mirzasoleiman. Mini-batch coresets for memory-efficient language model training on data mixtures. In *International Conference on Learning Representations (ICLR)*, 2025.

Sofia Maria Nikolakaki, Alina Ene, and Evimaria Terzi. An efficient framework for balancing submodularity and cost. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pp. 1256–1266, 2021.

Yuanyuan Qiang and Bin Liu. Dynamic algorithms for maximizing a DR-submodular function subtracted by a linear function over the integer lattice. *Theoretical Computer Science*, 1072: 115877, 2026.

HSVNS Kowndinya Renduchintala, Sumit Bhatia, and Ganesh Ramakrishnan. Smart: Submodular data mixture strategy for instruction tuning. In *Findings of the Association for Computational Linguistics (Findings of ACL)*, pp. 12916–12934, 2024.

Ana-Andreea Stoica, Jessy Xinyi Han, and Augustin Chaintreau. Seeding network influence in biased networks and the benefits of diversity. In *International World Wide Web Conference (WWW)*, pp. 2089–2098, 2020.

Maxim Sviridenko, Jan Vondrák, and Justin Ward. Optimal approximation for submodular and supermodular optimization with bounded curvature. *Mathematics of Operations Research*, 42(4): 1197–1218, 2017.

Yanhao Wang, Francesco Fabbri, and Michael Mathioudakis. Fair and representative subset selection from data streams. In *International World Wide Web Conference (WWW)*, pp. 1340–1350, 2021.

Yanhui Zhu, Samik Basu, and A Pavan. Regularized unconstrained weakly submodular maximization. In *ACM International Conference on Information and Knowledge Management (CIKM)*, pp. 3537–3548, 2024.

APPENDIX

A ADDITIONAL ALGORITHM DETAILS

The main text has presented the main procedures for batch selection, suffix reconstruction, and dynamic updates. This appendix supplements the parameter ranges used by the algorithm, bounds on the number of element groups, and the maintenance of instances corresponding to geometric estimates. All marginal tests are performed with respect to the distorted objective $D = f - a\ell$, while different candidate solutions are compared according to the original objective $F = f - \ell$.

A.1 PARAMETERS AND ELEMENT GROUPING

Throughout the algorithm, we use four accuracy parameters $\epsilon_o, \epsilon_b, \epsilon_s, \epsilon_d$ and a failure probability δ_c for batch-size estimation, and assume that

$$0 < \epsilon_o, \epsilon_b, \epsilon_s \le \frac{1}{20}, \quad 0 < \epsilon_d \le \frac{\epsilon_s}{4}, \quad 0 < \delta_c \le \min\left\{\frac{1}{10}, \frac{\epsilon_s}{100k}\right\}. \tag{3}$$

Their final values are specified in equation 2 of the main text. For any fixed value H, the distorted marginal gains of candidate elements lie between τ and $H = ak\tau$. Since we geometrically group distorted marginal gains by a factor of $1 + \epsilon_b$, the number of marginal-gain groups is at most

$$L_D = 1 + \left\lceil \log_{1+\epsilon_b}(ak) \right\rceil. \tag{4}$$

On the other hand, the scaled costs of candidate elements satisfy $0 \le c_e < k\tau$. We place elements with $c_e < \epsilon_b\tau$ into a separate low-cost group, and geometrically partition the remaining costs using the same factor $1 + \epsilon_b$. Hence, the number of cost groups is at most

$$L_c = 1 + \left\lceil \log_{1+\epsilon_b} \frac{k}{\epsilon_b} \right\rceil. \tag{5}$$

Combining the distorted-marginal-gain groups and the cost groups, each candidate set is partitioned into at most $J = L_D L_c$ joint buckets. This bound will be used later in the analysis of the largest source bucket and the number of levels.

A.2 MAINTAINING MULTIPLE ESTIMATES

The complete dynamic algorithm is given in Algorithm 4. Initially $V_0 = \varnothing$, and the instance dictionary and best-singleton dictionary are empty. The parameters are fixed before any updates. A run is keyed by its current INSTANCEOUTPUT value; if the active set is empty, OUTPUT returns $\varnothing$. All ties use fixed deterministic rules. Whether an element should be assigned to a geometric-estimate instance depends only on its singleton value and cost. Therefore, when an element is inserted, all instances to which it should be registered can be determined and recorded immediately. If a relevant instance does not yet exist, the algorithm creates it when it is first needed; otherwise, the element is processed by the insertion procedure described in the main text. When an element is deleted, the algorithm only needs to access the instances recorded at the time of its insertion and execute the corresponding deletion procedure for each of them. If an instance no longer contains any active root candidates after the deletion, the instance and its maintained state can be discarded; otherwise, only the record of its current output needs to be updated. In addition, the algorithm separately maintains the best active singleton and the F-values of the current outputs of all active instances. Hence, after each update, it can return the candidate with the largest F-value among the empty set, the best active singleton, and the current outputs of all maintained instances.

Algorithm 4 BATCHREG: the complete dynamic algorithm

procedure INSERTALL(e)
  evaluate $f(\{e\})$; $x_e \leftarrow f(\{e\}) - a\ell_e$; mark e active
  insert e into the best-singleton dictionary with key $F(\{e\})$
  for H_p such that $x_e > 0$, $x_e \le H_p \le akx_e$, and $F(\{e\}) < b(H_p)$ **do**
    if run p is absent **then**
      STARTINSTANCE(e, H_p)
    else
      INSERT(e) in run p
    record the registration and refresh run p's heap key

procedure DELETEALL(e)
  mark e inactive and delete it from the best-singleton dictionary
  for runs in which e was registered **do**
    call DELETE(e) in that run
    if the run's root is empty **then**
      destroy the run and all of its live state
    else
      refresh its heap key

function OUTPUT
  return the best under F among $\varnothing$, the best active singleton, and all run outputs

B OMITTED PROOFS

We give the full proofs in the order used by the main analysis. We begin with the sampling procedure, then justify the dynamic state and conditional distributions. We next derive the approximation bound, followed by the bounds on levels and reconstruction. Finally, we combine these results for the algorithm with all threshold estimates.

B.1 PROPERTIES OF BATCH SELECTION

We first verify the effect of checking every sampled element against the accepted prefix. Summing these checked marginals gives a deterministic lower bound on the gain of the accepted set. Submodularity also ensures that a rejected element remains below the threshold after later acceptances.

Lemma B.1. *If $A = \text{SCANBATCH}(U, G, \eta)$, then*

$$D(A \mid G) \ge |A|\eta \ge 0. \tag{6}$$

Moreover, every tentative element rejected during the scan has scaled marginal below η with respect to the final accepted prefix.

Proof. Expand $D(A \mid G)$ in acceptance order. Every accepted marginal is at least η by the acceptance test. If an element is rejected, the final accepted prefix contains the accepted prefix present at the rejection time, so submodularity can only decrease its marginal. □

We next show that extending the sampled prefix cannot increase the acceptance probability of its final test element. The comparison uses one fixed test element and one random order of the others, so both experiments have the same accepted initial prefix.

Lemma B.2. *The sequence $J_1, J_2, \ldots, J_{|C|}$ is nonincreasing. If every element of C satisfies $D(e \mid G) \ge \eta$, then $J_1 = 1$.*

Proof. Fix the uniformly chosen test element s and expose one random order of $C \setminus \{s\}$. Couple the experiments for m and $m+1$ by using the first $m-1$ and first m elements of this order. After the common prefix, the latter accepted set contains the former accepted set; processing one additional tentative element cannot remove an accepted element. By submodularity, the final marginal of s

in the longer experiment is no larger. Averaging the pointwise indicator inequality over s and the random order proves monotonicity. The last statement is the definition with an empty predecessor sequence. □

For a fixed source C, prefix G, and threshold η, let J_j be the last-element acceptance probability defined in Section 4.1. Write $p_j = 1 - J_j$ and $\widehat{p}_j = Z_j/q$. We call a count estimate accurate if, for all $j \le M$,

$$\widehat{p}_j \le \tfrac{3}{2}\epsilon_s \Rightarrow p_j < 2\epsilon_s, \qquad \widehat{p}_j > \tfrac{3}{2}\epsilon_s \Rightarrow p_j > \epsilon_s. \tag{7}$$

A call with $M = 1$ is accurate by convention.

The estimator only distinguishes two ranges of small rejection probabilities. For each fixed position, independent trials give a binomial count; a union bound then makes the classification valid simultaneously at all positions.

Lemma B.3. *Assume $\epsilon_s \le 1/8$. One call to* BATCHSIZE *is accurate with probability at least $1 - \delta_c$. The returned sample size satisfies $1 \le m \le M$. On the accuracy event,*

$$m = M \implies J_m \ge 1 - 2\epsilon_s, \tag{8}$$

$$m < M \implies J_m \ge 1 - 2\epsilon_s \quad \textit{and} \quad J_{m+1} \le 1 - \epsilon_s. \tag{9}$$

Proof. In one ordered M-sample, the rejection indicator at position j is Bernoulli with mean p_j: its first $j-1$ elements and its test element have exactly the experiment defining J_j. The q sampled orders are independent, so $Z_j \sim \mathrm{Bin}(q, p_j)$ for every fixed j. Indicators at different positions of the same order may be dependent; no independence across positions is used.

If $p_j \ge 2\epsilon_s$, the multiplicative lower-tail Chernoff bound, whose exponent is smallest at $p_j = 2\epsilon_s$, gives

$$\mathbb{P}\left[\widehat{p}_j \le \frac{3}{2}\epsilon_s\right] \le \exp(-q\epsilon_s/16).$$

If $p_j \le \epsilon_s$, stochastic monotonicity of a binomial variable in its mean and the multiplicative upper-tail bound at $p_j = \epsilon_s$ give

$$\mathbb{P}\left[\widehat{p}_j > \frac{3}{2}\epsilon_s\right] \le \exp(-q\epsilon_s/10).$$

The choice of q and a union bound over at most M positions prove equation 7 with probability at least $1 - \delta_c$.

Position one never rejects because every source element is eligible at the beginning of the level, so the returned count is positive. If no position crosses the empirical threshold, apply the first implication in equation 7 at M. Otherwise $m+1$ is the first crossing: position m does not cross and position $m+1$ does. Applying the two implications at these adjacent positions and weakening strict endpoints proves equation 8–equation 9. The empirical rejection frequencies themselves need not be monotone. □

The distribution at position j of a random ordered batch is exactly the experiment defining J_j. Summing acceptance indicators gives the accepted size. Appending one uniformly random unsampled element gives the corresponding identity for candidates that remain in the source class.

Lemma B.4. *Let U be a uniform ordered m-sample from C, and let $A =$ SCANBATCH(U, G, η). Then*

$$\mathbb{E}|A| = \sum_{j=1}^{m} J_j \ge mJ_m. \tag{10}$$

If $m < |C|$, the expected number of unsampled elements whose distorted marginal after adding A remains at least η is

$$(|C| - m)J_{m+1}. \tag{11}$$

Consequently, under equation 9, at least $\epsilon_s|C|$ elements of the source bucket are, in expectation, either sampled or fall below the source marginal class.

Proof. At position j, the tentative prefix and current element have exactly the distribution in the definition of J_j. Summing the acceptance indicators proves the equality in equation 10, and Lemma B.2 gives the inequality.

Next expose U and then choose a uniformly random $s \in C \setminus U$. The event $D(s \mid G \cup A) \geq \eta$ is exactly acceptance of the last element in a uniform ordered $(m+1)$-sample. Linearity of expectation gives equation 11. Hence the expected number that are sampled or leave the source class is

$$m + (|C| - m)(1 - J_{m+1}) \geq m + \epsilon_s(|C| - m) \geq \epsilon_s|C|.$$

□

B.2 Properties After Dynamic Updates

The sampling bounds can be used at an arbitrary update time only after checking what the maintained sets contain and how the samples are distributed. We first establish the deterministic set relations, and then condition on a history that excludes the current level's sample.

Throughout this appendix, a level is current if it has not been discarded by reconstruction. Its version lasts from one construction until reconstruction at that level or an earlier level, or until the instance is removed. Fix a completed update time t. For a current level i, $\mathcal{H}_i^{-m}(t)$ records its creation time and the state immediately before its batch-size call, the fixed updates through t, the active flags, the states of earlier levels, and $P_i, \bar{P}_i, C_i, \eta_i, z_i$. It also records that this version exists at t. It excludes the current batch-size random bits and outcome, U_i, A_i, all later-level states, output values, and reference-count information that reveals these excluded quantities. Earlier discarded versions may be included, but no information generated by the current version's excluded random choices is included. Let $\mathcal{H}_i(t)$ additionally record the current batch-size random bits and the resulting m_i. The source and historical prefix used in that call remain fixed throughout the version. Histories are used only on the event that level i exists. These are the two histories in Lemma B.6.

Immediately after construction, the frozen and expanded candidate sets coincide. Between reconstructions, the frozen set P_i remains unchanged, whereas $\bar{P}_i$ receives later insertions. The active part of the expanded sets is the object that continues to satisfy the recursive filter relation. Deletions are handled by the active-set projection on both sides of this relation, while the reconstruction thresholds bound how far the stored expanded sets can grow beyond their construction-time sets.

Lemma B.5. *After every completed update, for every current structure maintained for a fixed value H and every $i = 0, 1, \ldots, T$,*

$$\bar{P}^{\circ}_{i+1} = \textsc{Eligible}(\bar{P}_i, G_i), \tag{12}$$

$$\bar{P}^{\circ}_{T+1} = \varnothing, \tag{13}$$

where $\bar{P}_0 := P_0$. For every nonempty current level $i = 1, \ldots, T$,

$$|\bar{P}_i| < \frac{3}{2}|P_i|, \qquad z_i < \epsilon_d|C_i|. \tag{14}$$

Moreover, $\bar{P}^{\circ}_{i+1} \subseteq \bar{P}^{\circ}_i$, and, for $i = 1, \ldots, T$, every active element $e \in \bar{P}^{\circ}_i$ satisfies $D(e \mid G_{i-1}) \in [\tau, H]$.

Proof. Immediately after a suffix reconstruction, every newly built level satisfies $P_j = \bar{P}_j$, its newly formed candidate, source, and accepted sets contain only active elements, and the assignment in Algorithm 2 gives $\bar{P}^{\circ}_{j+1} = \textsc{Eligible}(\bar{P}_j, G_j)$. At the boundary immediately preceding the rebuilt suffix, the procedure replaces $\bar{P}_i$ by its active part without changing that active part, so an already valid filter relation is preserved.

Now consider an insertion of an element e. The insertion procedure propagates e through the levels in order and adds it to $\bar{P}_i$ exactly when it is active, the prefix is not full, and $D(e \mid G_{i-1}) \geq \tau$. Hence, for every boundary between consecutive levels, e belongs to $\bar{P}^{\circ}_{i+1}$ exactly when it belongs to $\textsc{Eligible}(\bar{P}_i, G_i)$. Once e fails a filter test, submodularity implies that it cannot become eligible at a later level. If a growth or terminal trigger fires, reconstruction re-establishes the same relations on the rebuilt suffix.

A deletion leaves the historical prefixes G_i unchanged and only marks the deleted element inactive. Therefore the active-set projection removes it from both sides of equation 12. If a source-deletion trigger fires, reconstruction again re-establishes the suffix. Induction over completed updates proves equation 12.

At the end of every completed update, the terminal active expanded set is empty: if an insertion reaches $T + 1$, Algorithm 3 immediately calls REBUILD$(T + 1)$ and creates a new empty terminal slot. This proves equation 13. The strict trigger bounds hold because equality or violation causes immediate reconstruction of the earliest affected suffix. Nestedness follows from equation 12. Finally, membership in $\bar{P}_i^{\circ}$ gives the lower marginal bound, while submodularity and the admission rule give $D(e \mid G_{i-1}) \le D(\{e\}) \le H$. □

Lemma B.6. *Conditional on any feasible value of $\mathcal{H}_i(t)$, the tentative order U_i is a uniform ordered m_i-sample without replacement from C_i. Conditional on any feasible value of $\mathcal{H}_i^{-m}(t)$, the batch-size estimator is accurate with probability at least $1 - \delta_c$.*

Uniformity at construction alone is insufficient. We must also account for the event that the same version of a level is still present at time t. The proof below checks that this event and the retained information do not inspect the random sample or batch-size-estimation bits being analyzed.

Proof. At the creation time of the version, the state and estimator outcome recorded in $\mathcal{H}_i(t)$ are fixed before fresh independent bits draw U_i. Hence the first claim holds at creation. Between creation and time t, the version survives exactly when no run destruction and no reconstruction at an index at most i occurs. For an insertion, every such reconstruction decision is determined by earlier prefixes, stored candidate-set sizes, and the empty last level test. At level i the decision uses $|\bar{P}_i|/|P_i|$ and does not inspect U_i or A_i. For a deletion, the decision is determined by source buckets and counters z_h for $h \le i$; in particular the level-i trigger uses C_i rather than U_i or A_i. The update identities are fixed independently of the algorithm's hidden random bits because the update sequence is oblivious. Run destruction depends only on the exact active root.

Consequently, conditional on the information retained in $\mathcal{H}_i(t)$, the event that the version survives and the evolution of every retained variable are measurable without inspecting U_i. Conditioning a uniform order on such an event preserves its law. A reconstruction at an index larger than i leaves the version unchanged; a reconstruction at an index at most i replaces it by a fresh version, returning to the creation case. Induction over updates proves the first claim.

For the second claim, repeat the same argument with m_i and the batch-size-estimator bits omitted. At creation, Lemma B.3 applies conditionally to the state before estimating the batch size. The subsequent survival decisions described above do not inspect those bits. Conditioning therefore preserves the probability bound. □

Earlier accepted sets belong in the history because they determine G_{i-1}. The current sampled and accepted sets do not. Reference counts that reveal their membership are also excluded. This distinction is necessary even though reference counting itself is only a memory-management operation.

B.3 Approximation Analysis

We first make explicit the conditioning used below. Let

$$\bar{c}_i = |C_i|^{-1} \sum_{e \in C_i} c_e, \qquad W_i(m) = m(\eta_i + \bar{c}_i). \tag{15}$$

These quantities are determined before the actual sample at level i is drawn. Recall the constants

$$a = \frac{3 + \sqrt{5}}{2}, \qquad \rho = \frac{1}{a} = \frac{3 - \sqrt{5}}{2}, \qquad (a - 1)^2 = a. \tag{16}$$

The approximation proof consists of three steps. First, we lower-bound the active output by its original gains minus the contributions assigned to deleted elements. Second, we compare the expected gain and deletion loss at each level. Third, we use a lower bound valid for every execution to combine the two stopping cases without conditioning on the final solution size.

Fix a time and an estimate satisfying $\Delta_t \le H \le (1+\epsilon_o)\Delta_t$. Assume that $F(\{e\}) < b(H)$ holds for every active e, unless the best singleton already gives the required value. At level i, define

$$Y_i = D(A_i \mid G_{i-1}), \qquad C_i^A = c(A_i), \tag{17}$$

$$H_i^A = Y_i + C_i^A = F(A_i \mid G_{i-1}), \qquad v_i = |A_i|\eta_i + C_i^A. \tag{18}$$

The acceptance rule gives

$$H_i^A \ge v_i \ge |A_i|\eta_i \ge |A_i|\tau. \tag{19}$$

For a surviving accepted element, removing deleted predecessors can only increase its marginal gain. Thus only deleted accepted elements need to be charged, and their original marginals are bounded by the upper endpoint of the source bucket plus their costs.

Lemma B.7. *For every realization,*

$$F(G_T^\circ) \ge \sum_{i=1}^{T}(H_i^A - d_i) \ge \sum_{i=1}^{T}(v_i - d_i). \tag{20}$$

Proof. Order all accepted elements first by level and then by acceptance order. For a surviving element, its active predecessors form a subset of its original historical predecessors. Since $F = f - \ell$ is submodular, its marginal with respect to the active predecessors is at least its original marginal. Thus deleting inactive accepted elements loses at most the sum of their original marginals.

Write $A_i^{<e}$ for the elements of A_i accepted before e. If $e \in A_i$, its scaled marginal when accepted is no larger than its marginal at the beginning of the level, by submodularity. Since e belongs to the source marginal class,

$$D(e \mid G_{i-1} \cup A_i^{<e}) \le D(e \mid G_{i-1}) \le (1+\epsilon_b)\eta_i.$$

Its original regularized marginal is therefore at most $(1+\epsilon_b)\eta_i + c_e$. Summing over inactive accepted elements proves the first inequality in equation 20; the second follows from equation 19. □

PROOF OF LEMMA 5.1

When the historical prefix has fewer than k elements, termination implies that no active candidate passes the filter. We bound the marginals of the optimal solution for D and then return to the monotone utility function f. This avoids assuming monotonicity of the scaled objective.

Proof. Let O_D maximize D under the cardinality constraint. We first prove that

$$D(e \mid G_T) < \tau \qquad \text{for every } e \in O_D \setminus G_T. \tag{21}$$

If $D(\{e\}) < \tau$, submodularity proves the claim. If $D(\{e\}) > H$, the feasible singleton $\{e\}$ contradicts $\Delta_t \le H$. Otherwise $D(\{e\}) \in [\tau, H]$. Since $F(\{e\}) < b(H)$, e satisfies the admission conditions for the current fixed value H. Suppose for contradiction that $D(e \mid G_T) \ge \tau$. Since $e \in O_D \subseteq V_t$ is active and is assigned to the current instance, $e \in \bar{P}_0 = P_0$. Moreover, $e \notin G_T$ implies $e \notin G_i$ for every $i \le T$, and submodularity gives $D(e \mid G_i) \ge D(e \mid G_T) \ge \tau$ for every $i < T$. Repeatedly applying equation 12 therefore yields $e \in \bar{P}_{i+1}^\circ$ for every $i = 0, \dots, T$. In particular, $e \in \bar{P}_{T+1}^\circ$, contradicting equation 13. Thus equation 21 holds.

Using monotonicity and submodularity of f, together with $|O_D| \le k$, we obtain

$$\begin{aligned} f(O_D) &\le f(G_T \cup O_D) \\ &\le f(G_T) + \sum_{e \in O_D \setminus G_T} f(e \mid G_T) \\ &= f(G_T) + \sum_{e \in O_D \setminus G_T} \big(D(e \mid G_T) + a\ell_e\big) \\ &\le f(G_T) + k\tau + a\ell(O_D). \end{aligned}$$

Subtracting $a\ell(O_D)$ gives

$$f(G_T) \ge \Delta_t - k\tau \ge \frac{H}{1+\epsilon_o} - \frac{H}{a}.$$

The identity $F(S) = D(S) + c(S) = a^{-1}D(S) + (a-1)a^{-1}f(S)$ now yields

$$F(G_T) = \frac{1}{a}D(G_T) + \frac{a-1}{a}f(G_T) \geq \frac{1}{a}D(G_T) + b(H).$$

Finally, $D(G_T) = \sum_i Y_i \geq 0$ by Lemma B.1. □

To bound expected deletion loss, replace the accepted subset by the entire uniformly sampled batch. The source-deletion threshold bounds the number of deleted elements, while the cost grouping bounds their costs relative to the source average.

Lemma B.8. *For every realized count m_i,*

$$\mathbb{E}[d_i \mid \mathcal{H}_i(t)] \leq \lambda_d \epsilon_d W_i(m_i), \qquad \lambda_d = 1 + 2\epsilon_b. \tag{22}$$

Proof. Since $A_i \subseteq U_i$, we may upper-bound the charge by replacing A_i with U_i. Conditional uniformity gives

$$\mathbb{E}[d_i \mid \mathcal{H}_i(t)] \leq \frac{m_i}{|C_i|}\left(z_i(1+\epsilon_b)\eta_i + \sum_{e \in C_i \setminus V_t} c_e\right). \tag{23}$$

For an ordinary cost bucket with lower endpoint ζ, every source cost is below $(1+\epsilon_b)\zeta$, while $\bar{c}_i \geq \zeta$. Since $z_i < \epsilon_d |C_i|$, equation 23 is at most

$$(1+\epsilon_b)\epsilon_d m_i(\eta_i + \bar{c}_i) \leq \lambda_d \epsilon_d W_i(m_i).$$

For the low-cost bucket, every source cost is below $\epsilon_b \tau \leq \epsilon_b \eta_i$, so equation 23 is at most $(1 + 2\epsilon_b)\epsilon_d m_i \eta_i \leq \lambda_d \epsilon_d W_i(m_i)$. □

We now compare the gain from accepted elements with the same source-average expression. On an accurate call, the expected accepted fraction is large. Comparable costs then turn this accepted fraction into a bound on the cost part of v_i as well as its marginal-gain part.

Lemma B.9. *Let*

$$\gamma = \frac{q_0}{1+\epsilon_b}, \qquad q_0 = 1 - 2\epsilon_s. \tag{24}$$

Conditional on an accurate count call,

$$\mathbb{E}[v_i \mid \mathcal{H}_i(t),\ \textit{accurate}] \geq \gamma W_i(m_i). \tag{25}$$

Without conditioning on accuracy,

$$\mathbb{E}[v_i \mid \mathcal{H}_i(t)] \leq W_i(m_i). \tag{26}$$

Proof. On the accuracy event, Lemmas B.3 and B.4 give $\mathbb{E}|A_i| \geq q_0 m_i$. In an ordinary cost bucket with lower endpoint ζ, $c(A_i) \geq \zeta |A_i|$ and $\bar{c}_i < (1+\epsilon_b)\zeta$. Hence, in conditional expectation, both the marginal part and the cost part of v_i are at least a γ fraction of the corresponding part of $W_i(m_i)$. In the low-cost bucket, $\bar{c}_i < \epsilon_b \tau \leq \epsilon_b \eta_i$, and therefore

$$q_0 m_i \eta_i \geq \frac{q_0}{1+\epsilon_b} m_i(\eta_i + \bar{c}_i) = \gamma W_i(m_i).$$

For the upper bound, use $|A_i| \leq m_i$, $A_i \subseteq U_i$, and $\mathbb{E}[c(U_i) \mid \mathcal{H}_i(t)] = m_i \bar{c}_i$. □

PROOF OF LEMMA 5.2

Combining the preceding two lemmas gives a positive fraction of the expected level contribution after deletion loss. An inaccurate count call may choose a larger batch, so its contribution is bounded using both its probability and the deterministic range $1 \leq m_i \leq k$.

Proof. All expectations and probabilities in this proof are conditional on $\mathcal{H}_i^{-m}(t)$. Let $\mathcal{G}_i$ be the accuracy event. Conditional on a realized m_i and $\mathcal{G}_i$, Lemmas B.8 and B.9 give

$$\mathbb{E}[v_i - d_i \mid m_i, \mathcal{G}_i] \geq (\gamma - \lambda_d \epsilon_d) W_i(m_i).$$

On the complement, $v_i \geq 0$ and the deletion bound still gives

$$\mathbb{E}[v_i - d_i \mid m_i, \neg\mathcal{G}_i] \geq -\lambda_d \epsilon_d W_i(m_i).$$

Averaging these two inequalities yields

$$\mathbb{E}[v_i - d_i] \geq (\gamma - \lambda_d \epsilon_d)\mathbb{E}[W_i(m_i)] - \gamma \mathbb{E}[W_i(m_i)\mathbf{1}[\neg\mathcal{G}_i]]. \tag{27}$$

Since $1 \leq m_i \leq k$ and $\mathbb{P}[\neg\mathcal{G}_i] \leq \delta_c$ by Lemma B.6,

$$\mathbb{E}[W_i(m_i)\mathbf{1}[\neg\mathcal{G}_i]] \leq k\delta_c(\eta_i + \bar{c}_i) \leq k\delta_c \mathbb{E}[W_i(m_i)].$$

Substituting into equation 27 gives $\mathbb{E}[v_i - d_i] \geq \kappa \mathbb{E}[W_i(m_i)]$. Finally, equation 26 implies $\mathbb{E}[v_i] \leq \mathbb{E}[W_i(m_i)]$, and $\kappa \geq 0$ proves Lemma 5.2. □

Define the random relaxation

$$\Phi = \sum_i (v_i - d_i) + \kappa \left(b(H) - \sum_i v_i \right). \tag{28}$$

The expression Φ combines the stopping cases. If the sum of level lower bounds already reaches $b(H)$, its additional term is nonpositive. Otherwise the historical prefix is not full and the terminal-candidate argument supplies the missing value.

Lemma B.10. *For every realization,*

$$F(G_T^\circ) \geq \Phi. \tag{29}$$

Proof. Let $z = b(H) - \sum_i v_i$. If $z \leq 0$, then $\kappa z \leq 0$ and Lemma B.7 gives equation 29. Suppose $z > 0$. By equation 19,

$$|G_T|\tau = \sum_i |A_i|\tau \leq \sum_i v_i < b(H) \leq k\tau,$$

so $|G_T| < k$. By Lemmas B.7 and 5.1,

$$F(G_T^\circ) \geq F(G_T) - \sum_i d_i \geq b(H) - \sum_i d_i.$$

Since $\kappa \leq 1$ and $\sum_i v_i < b(H)$,

$$\Phi = \kappa b(H) + (1 - \kappa) \sum_i v_i - \sum_i d_i \leq b(H) - \sum_i d_i.$$

□

PROOF OF THEOREM 5.3

We can now derive the approximation guarantee for a fixed estimate. Because $T \leq k$ for every execution, taking expectations requires only a finite sum. The final conversion uses $a\beta \leq 1$ to preserve coefficient one on the optimal cost.

Proof. The heavy-singleton case is immediate. Otherwise the structure for the stated grid value H is maintained. To see this, choose an active element $e^\star$ maximizing $x_e = D(\{e\})$. Normalized submodularity gives $0 < x_{e^\star} \leq \Delta_t \leq k x_{e^\star}$. Hence $x_{e^\star} \leq H \leq (1 + \epsilon_o) k x_{e^\star} \leq a k x_{e^\star}$; since $F(\{e^\star\}) < b(H)$, the registration rule assigns it to H. The rule also assigns every active element satisfying the admission conditions to this same structure. Extend the level sequence by setting $v_i = d_i = 0$ for $i > T$. The deterministic bound $T \leq k$ makes every sum below finite. Taking expectations in equation 28, applying Lemma 5.2 to each $i = 1, \ldots, k$, and using Lemma B.10, we obtain

$$\begin{aligned}
\mathbb{E}[F(Q)] &\geq \mathbb{E}[\Phi] \\
&= \sum_{i=1}^{k} \mathbb{E}[v_i - d_i] + \kappa \left(b(H) - \sum_{i=1}^{k} \mathbb{E}[v_i] \right) \\
&\geq \kappa \sum_{i=1}^{k} \mathbb{E}[v_i] + \kappa \left(b(H) - \sum_{i=1}^{k} \mathbb{E}[v_i] \right) \\
&= \kappa b(H).
\end{aligned}$$

By $H \geq \Delta_t$, $\tau = \frac{H}{ak}$ and $b(H) = \frac{a-1}{a}(\frac{H}{1+\epsilon_o} - \frac{H}{a})$,

$$b(H) \geq \frac{a-1}{a}\left(\frac{1}{1+\epsilon_o} - \frac{1}{a}\right)\Delta_t,$$

which proves $\mathbb{E}[F(Q)] \geq \kappa b(H) \geq \beta\Delta_t$. The factor $((a-1)/a)(1/(1+\epsilon_o) - 1/a)$ is at most its value at $\epsilon_o = 0$, namely $1/a$, by $(a-1)^2 = a$. Since $\kappa \leq 1$, we have $\beta \leq 1/a$.

Finally, $\Delta_t \geq D(O_t) = f(O_t) - a\ell(O_t)$, so

$$\beta\Delta_t \geq \beta f(O_t) - a\beta\ell(O_t) \geq \beta f(O_t) - \ell(O_t).$$

If $\Delta_t = 0$, then $D(O_t) \leq 0$ implies $(1/a)f(O_t) - \ell(O_t) \leq 0$, and the empty set suffices. □

B.4 The Number of Levels

For this part of the analysis, fix a time and one current instance. For $e \in \bar{P}_i^\circ$, let $p_i(e) \in \{1, \ldots, L_D\}$ be the index of its marginal-gain class with respect to G_{i-1}, with larger marginals assigned larger indices. Set $p_i(e) = 0$ outside $\bar{P}_i^\circ$, and define

$$\Psi_i = \sum_{e \in V_t} p_i(e). \tag{30}$$

Then

$$|\bar{P}_i^\circ| \leq \Psi_i \leq L_D|\bar{P}_i^\circ|, \qquad \Psi_{T+1} = 0. \tag{31}$$

By Lemma B.5, membership in $\bar{P}_i^\circ$ and the corresponding marginal-class index can only decrease with the level. The remaining capacity

$$r_i = k - |G_{i-1}| \tag{32}$$

is also nonincreasing.

Call a level good when its count call is accurate. A good level is a rank level if either $M_i = |C_i|$ or $m_i < M_i$; otherwise it is a capacity level. A capacity level has $M_i = \lceil r_i/2 \rceil < |C_i|$ and $m_i = M_i$. These terms distinguish the two cases of the proof, not additional algorithmic procedures.

At a rank level, sampled elements either leave the candidate set or fall below their original class, and the same is true for a sufficiently large expected number of unsampled elements. The source-deletion bound limits how much of this progress can be attributed to already inactive elements.

Lemma B.11. *Conditional on a feasible value of $\mathcal{H}_i(t)$ whose count call is accurate and whose level is a rank level,*

$$\mathbb{E}[\Psi_i - \Psi_{i+1} \mid \mathcal{H}_i(t)] \geq \frac{\epsilon_s}{2}|C_i| \geq \frac{\epsilon_s}{3JL_D}\Psi_i. \tag{33}$$

Proof. Suppose first that $M_i = |C_i|$ and $m_i = M_i$. Every source element is tentative. An accepted active source element leaves $\bar{P}_{i+1}^\circ$ because the filter excludes G_i, while a rejected active source element has marginal below η_i with respect to G_i by Lemma B.1 and therefore drops below its source marginal class. Hence every active source element loses at least one unit of rank. Since $z_i < \epsilon_d|C_i|$ and $\epsilon_d \leq \epsilon_s/4$, the active loss is at least $(1-\epsilon_d)|C_i| \geq (\epsilon_s/2)|C_i|$.

Otherwise $m_i < M_i$. On the good event, Lemma B.4 and equation 9 imply an expected source-class loss of at least $\epsilon_s|C_i|$ before inactive items are removed. At most $z_i < \epsilon_d|C_i| \leq (\epsilon_s/4)|C_i|$ of these losses can belong to inactive source elements. The expected active loss is therefore at least $(\epsilon_s/2)|C_i|$. All other active expanded candidates have nonincreasing rank by Lemma B.5, so these source losses lower-bound $\Psi_i - \Psi_{i+1}$.

The source is a largest of at most J joint buckets of the construction-time candidate set, so $|C_i| \geq |P_i|/J$. By the growth trigger, $|\bar{P}_i^\circ| \leq |\bar{P}_i| < 3|P_i|/2$. Combining this with equation 31 yields

$$\Psi_i \leq L_D|\bar{P}_i^\circ| < \frac{3}{2}JL_D|C_i|,$$

which proves the second inequality in equation 33. □

At a capacity level, the chosen batch reaches the half-cap. Since most sampled elements are expected to be accepted, the remaining capacity decreases by a constant fraction.

Lemma B.12. *Let*

$$q_0 = 1 - 2\epsilon_s. \tag{34}$$

Conditional on a feasible value of $\mathcal{H}_i(t)$ whose count call is accurate and whose level is a capacity level, we have

$$\mathbb{E}[r_{i+1} \mid \mathcal{H}_i(t)] \le \left(1 - \frac{q_0}{2}\right) r_i. \tag{35}$$

Proof. By equation 8, $J_{M_i} \ge q_0$. Conditional uniformity and equation 10 give

$$\mathbb{E}|A_i| \ge q_0 M_i \ge \frac{q_0}{2} r_i.$$

Since $r_{i+1} = r_i - |A_i|$, the claim follows. □

We count rank levels and capacity levels separately using their respective nonincreasing quantities. Other levels cannot undo progress. Finally, a finite indicator sum accounts for inaccurate count calls, rather than assuming that all calls succeed simultaneously.

Lemma B.13. *There is a deterministic bound*

$$\bar{T} = O\left(\frac{JL_D}{\epsilon_s} \log(nL_D + 1) + \log(k+1)\right) \tag{36}$$

such that $\mathbb{E}[T] \le \bar{T}$ after every completed update for any fixed value H. The same bound holds for the conditional expected number of new levels in any suffix reconstruction, given its admissible input state.

Proof. Fix a completed update time. At each existing level i, condition separately on $\mathcal{H}_i(t)$. The current potentials Ψ_i, r_i, the accuracy event, and the level type are measurable with respect to this history, and Lemma B.6 gives the required law of U_i. Treat nonexistent levels as unmarked and extend the potentials constantly beyond the terminal slot. We use the version of Lemma B.21 that permits these conditioning histories to be nonnested; no common filtration of current level versions is assumed. Consider first only good rank levels. By Lemma B.11, they give multiplicative drift to the nonnegative integer potential Ψ, whose initial value is at most nL_D. Capacity levels and bad levels do not increase Ψ. The multiplicative-drift bound in Lemma B.21 therefore gives

$$\mathbb{E}[N_{\mathrm{rank}}^{\mathrm{good}}] = O\left(\frac{JL_D}{\epsilon_s} \log(nL_D + 1)\right).$$

On the subsequence of good capacity levels, Lemma B.12 gives constant multiplicative drift to the positive integer r, initially at most k. Other levels do not increase r, so

$$\mathbb{E}[N_{\mathrm{cap}}^{\mathrm{good}}] = O(\log(k+1)).$$

Let N_{good} and N_{bad} denote the total numbers of good and bad levels. The preceding two bounds imply

$$\mathbb{E}[N_{\mathrm{good}}] = O\left(\frac{JL_D}{\epsilon_s} \log(nL_D + 1) + \log(k+1)\right). \tag{37}$$

It remains to account for inaccurate count calls. By $T \le |G_T| \le k$, at most k levels exist. For $i = 1, \ldots, k$, let I_i indicate that level i exists and let B_i indicate that its count call is inaccurate; put $B_i = 0$ when $I_i = 0$. By Lemma B.6, conditional on the history before estimating the batch size of an existing level, $\mathbb{P}[B_i = 1] \le \delta_c$. Therefore

$$\mathbb{E}[I_i B_i] \le \delta_c \mathbb{E}[I_i].$$

Summing over the finite index set gives

$$\mathbb{E}[N_{\mathrm{bad}}] \le \delta_c \mathbb{E}[T].$$

Since $T = N_{\mathrm{good}} + N_{\mathrm{bad}}$,

$$(1 - \delta_c)\mathbb{E}[T] \le \mathbb{E}[N_{\mathrm{good}}].$$

The assumption $\delta_c \le 1/10$ and equation 37 prove equation 36. The same argument applies to a current level sequence at an arbitrary stream time because Lemma B.6 holds for surviving versions, not only for freshly built ones. □

B.5 Reconstruction and Amortized Query Complexity

Let

$$Q_c = O\left(\epsilon_s^{-1} \log \frac{k+1}{\delta_c}\right), \tag{38}$$

$$\bar{Z}_P = O(JL_D/\epsilon_s + \log(k+1)), \tag{39}$$

$$\bar{Z}_M = O(L_D/\epsilon_s + 1). \tag{40}$$

In a fresh suffix, all candidate sets contain active elements and are nested. We exploit this to count each type of work according to the actual set it processes.

The rank decrease controls the sum of source sizes over rank levels. It therefore bounds both the total candidate-set sizes, with a factor J, and the total tentative sample sizes, without that factor. Capacity levels are bounded through the remaining capacity. Inaccurate calls are handled by conditioning before their estimation bits are drawn.

Lemma B.14. *Suppose* Rebuild(i) *starts with* $N = |\bar{P}_i \cap V_t|$ *active elements. Conditional on any fixed admissible input state with this size, if* $\delta_c \le 1/10$*, the newly constructed nonempty levels satisfy*

$$\mathbb{E}\left[\sum_{j \ge i} |P_j|\right] = O(N\bar{Z}_P), \tag{41}$$

$$\mathbb{E}\left[\sum_{j \ge i} M_j\right] = O(N\bar{Z}_M). \tag{42}$$

Proof. During a fresh suffix reconstruction, no updates occur between the creation of consecutive levels, so every newly built level satisfies $P_j = \bar{P}_j = P_j^\circ = \bar{P}_j^\circ$. These candidate sets are nested and are all subsets of the N-element starting candidate set. Hence the rank potential defined using $\bar{P}_j^\circ$ coincides with the construction-time candidate sets throughout the rebuilt suffix, and $\Psi_i \le NL_D$.

First consider good rank levels. If $M_j = |C_j|$ and $m_j = M_j$, every source element loses its source rank. If $m_j < M_j$, Lemma B.4 gives expected source-class loss at least $\epsilon_s|C_j|$. Thus, conditional on the batch-size outcome that makes level j a good rank level and before drawing the actual tentative order,

$$\mathbb{E}[\Psi_j - \Psi_{j+1}] \ge \epsilon_s|C_j| \ge \frac{\epsilon_s}{J}|P_j|. \tag{43}$$

The decreases telescope even when other level types are interleaved. Since at most k levels exist, all sums are finite, and therefore

$$\mathbb{E}\left[\sum_{j:\,\text{good rank}} |P_j|\right] \le \frac{NJL_D}{\epsilon_s}. \tag{44}$$

Also $M_j \le |C_j|$, so the same argument gives

$$\mathbb{E}\left[\sum_{j:\,\text{good rank}} M_j\right] \le \frac{NL_D}{\epsilon_s}. \tag{45}$$

Every candidate set has size at most N. By Lemma B.12, the expected number of good capacity levels is $O(\log(k+1))$, and hence

$$\mathbb{E}\left[\sum_{j:\,\text{good cap}} |P_j|\right] = O(N\log(k+1)). \tag{46}$$

For the sum of tentative sample sizes, use the capacity decrease directly. At a capacity level, $M_j = \lceil r_j/2 \rceil < |C_j| \le N$, so the remaining capacity at the first such level is at most $2N$. Moreover, conditional on a good capacity count,

$$\mathbb{E}[r_j - r_{j+1}] = \mathbb{E}|A_j| \ge q_0 M_j.$$

The actual decreases over all good capacity levels sum to at most the remaining capacity at the first one. Therefore,

$$\mathbb{E}\left[\sum_{j:\,\text{good cap}} M_j\right] \le \frac{2N}{q_0} = O(N). \tag{47}$$

It remains to account for bad count calls. Pad the suffix to the deterministic range $j = i, \dots, i+k-1$ with $|P_j| = M_j = 0$ after it terminates. Before the estimation bits at an existing level are drawn, both $|P_j|$ and M_j are history-measurable, while the conditional probability of a bad call is at most δ_c. Hence, for $X_j \in \{|P_j|, M_j\}$,

$$\mathbb{E}\left[\sum_{j:\,\text{bad}} X_j\right] \le \delta_c \mathbb{E}\left[\sum_{j \ge i} X_j\right].$$

Every level is good rank, good capacity, or bad. Move the bad contribution to the left, use $\delta_c \le 1/10$, and combine equation 44–equation 47. This proves equation 41–equation 42. □

Once the two sums have been bounded, the query cost follows by charging grouping and filtering to $|P_j|$, and sampling to M_j. This avoids multiplying the full candidate-set scan by the number of estimation trials.

Lemma B.15. *Conditional on any fixed admissible input state, a call to* REBUILD(i) *with active starting candidate-set size* $N = |\bar{P}_i \cap V_t|$ *uses*

$$O\big(N\big[\bar{Z}_P + (1+Q_c)\bar{Z}_M\big]\big) \tag{48}$$

expected value-oracle queries.

Proof. At level j, bucketing and filtering use $O(|P_j|)$ queries. The shared-prefix count call and the fresh actual tentative order use $O((1+Q_c)M_j)$ queries. Sum these costs over the new suffix and apply Lemma B.14. The count routine needs only an array of M_j counters and processes its sampled orders sequentially, so its temporary memory is also linear in the current candidate set. □

To amortize this cost, consider one version of level i with construction-time candidate set P_i. If growth triggers reconstruction, the number g_i of insertions added to $\bar{P}_i$ since construction satisfies

$$g_i \ge \frac{1}{2}|P_i|. \tag{49}$$

If source deletions trigger reconstruction, their number h_i satisfies

$$h_i \ge \epsilon_d |C_i| \ge \frac{\epsilon_d}{J}|P_i|. \tag{50}$$

At either trigger, the active starting set for the reconstruction obeys

$$N \le |\bar{P}_i| \le \frac{3}{2}|P_i| + 1 \le 3|P_i|, \tag{51}$$

where $P_i \neq \varnothing$. Reconstructing the empty last level or initializing a new instance starts with one element, and its cost is assigned directly to that insertion.

The preceding bounds provide enough updates to pay for a reconstruction. The remaining issue is to avoid charging one update repeatedly through later versions of the same level. We charge an update only when it entered the current candidate set or deleted an element of the current source.

Lemma B.16. *For one fixed value H, the expected amortized value-oracle complexity per update processed by the corresponding structure is*

$$O\left(\bar{T} + \frac{J}{\epsilon_d}\bar{T}\big[\bar{Z}_P + (1+Q_c)\bar{Z}_M\big]\right). \tag{52}$$

Proof. Queries outside reconstruction arise from insertion filter tests and from refreshing the current output value. They use $O(T+1)$ queries for an insertion and $O(1)$ for a deletion, hence $O(\bar{T})$ in expectation.

Associate a nonterminal suffix reconstruction at level i with the version whose trigger ended at that update. If the trigger is caused by insertion, distribute the reconstruction cost equally among the g_i insertions counted in equation 49. If it is caused by deletion, distribute the cost equally among the h_i source deletions counted in equation 50. Conditional on the complete trigger history and the active starting candidate set, Lemma B.15 and equation 51 imply that the expected charge to any one such event is at most

$$O\left(\frac{J}{\epsilon_d}\left[\bar{Z}_P + (1+Q_c)\bar{Z}_M\right]\right). \tag{53}$$

The same asymptotic bound covers an empty-last-level reconstruction and the initial reconstruction for a newly created structure, each charged directly to its creating insertion.

Fix a finite update prefix and one update u in it. Charge only reconstructions completed within that prefix; versions that survive beyond it incur no reconstruction charge yet. If a reconstruction charges u, the corresponding version of the level was present when u occurred: an insertion was added to its expanded candidate set $\bar{P}_i$, or a deletion removed an element of its fixed source. A version constructed after u cannot charge it. Hence u receives at most one charge per version present at that time, plus the direct charge for an insertion reaching the empty last level. There are at most $T_{u^-}+1$ such charges, where T_{u^-} is the number of nonempty levels immediately before processing update u. Taking conditional expectations in equation 53 and applying Lemma B.13 yields the reconstruction term in equation 52. Adding the queries outside reconstruction completes the proof. □

B.6 The Complete Algorithm

For a target error $\epsilon \in (0, 1/10]$, use the parameter values in equation 2. These values satisfy equation 3. We finally justify the estimates maintained by Algorithm 4 and combine the bounds for individual values of H. The first observation locates Δ_t within a factor k of a singleton value, without assuming that D is nonnegative.

A best positive singleton is feasible, giving one direction. In the other direction, normalized submodularity bounds the value of a set by the sum of its singleton values.

Lemma B.17. *If $\Delta_t > 0$ and $M_t = \max_{e\in V_t} D(\{e\})_+$, then*

$$M_t \le \Delta_t \le kM_t.$$

Proof. The first inequality holds because every singleton is feasible. Normalized submodularity gives $D(S) \le \sum_{e\in S} D(\{e\})$ for every S. Apply this inequality to a maximizer of D and use $|S| \le k$. □

Choose the smallest geometric estimate not below Δ_t. The preceding singleton bound ensures that some element is registered for this value of H, unless the global best singleton is already sufficient. The same registration rule includes every active element needed in the fixed-estimate analysis.

Lemma B.18. *If $\Delta_t > 0$, some grid point satisfies $\Delta_t \le H_p \le (1+\epsilon_o)\Delta_t$. For this grid point, either the global best singleton has value at least $b(H_p)$, or the structure for H_p is maintained and contains every active element satisfying $\tau \le D(\{e\}) \le H_p$ and $F(\{e\}) < b(H_p)$ required by the fixed-guess analysis.*

Proof. Take the smallest grid point no smaller than Δ_t. Let $e^\star$ maximize $D(\{e\})_+$ over active elements. By Lemma B.17, $x_{e^\star} \le \Delta_t \le kx_{e^\star}$. Hence

$$x_{e^\star} \le H_p \le (1+\epsilon_o)\Delta_t \le (1+\epsilon_o)kx_{e^\star} \le akx_{e^\star},$$

where the last inequality uses $\epsilon_o \le a-1$. If $F(\{e^\star\}) \ge b(H_p)$, the singleton alternative holds. Otherwise $e^\star$ is registered, so the structure for H_p is maintained.

Now consider any active element e satisfying $\tau \le D(\{e\}) \le H_p$ and $F(\{e\}) < b(H_p)$ for this value H_p. Its singleton value satisfies $H_p/(ak) = \tau \le x_e \le H_p$, and therefore $x_e \le H_p \le akx_e$. Together with $F(\{e\}) < b(H_p)$, this is exactly the registration rule. Since singleton values and costs are immutable, e has been registered for H_p throughout its active lifetime. □

Each global update affects at most G_H maintained values of H, where

$$G_H = 1 + \lceil \log_{1+\epsilon_o}(ak) \rceil. \tag{54}$$

Summing their costs gives the query bound. We also expand the parameters to make their dependence explicit.

Theorem B.19. *Use the parameter values in equation 2. The complete algorithm uses*

$$O\left(G_H \left[\bar{T} + \frac{J}{\epsilon_d}\bar{T}\big(\bar{Z}_P + (1+Q_c)\bar{Z}_M\big)\right]\right) = \mathrm{poly}(\log n, \log k, 1/\epsilon) \tag{55}$$

expected amortized value-oracle queries per global update.

More explicitly, put

$$A = 1 + \log(ak), \qquad B = 1 + \log(k/\epsilon), \qquad N_{\log} = 1 + \log(nA/\epsilon). \tag{56}$$

Then a conservative expansion of equation 55 is

$$O\big(\epsilon^{-12} A^6 B^3 N_{\log}\big). \tag{57}$$

Proof. Each inserted item is registered with at most G_H runs. Apply Lemma B.16 to the update subsequence of each run and sum over runs. Over any global update sequence, every update appears in at most G_H such subsequences. The singleton evaluation and one output-value refresh per affected run are absorbed by equation 55.

For the explicit expansion, equation 4, equation 5, equation 38, equation 36, equation 39, equation 40, and equation 54 give

$$\begin{aligned} L_D &= O(\epsilon^{-1}A), & L_c &= O(\epsilon^{-1}B), & J &= O(\epsilon^{-2}AB), \\ Q_c &= O(\epsilon^{-1}B), & \bar{T} &= O(\epsilon^{-4}A^2 B N_{\log}), & G_H &= O(\epsilon^{-1}A), \\ \bar{Z}_P &= O(\epsilon^{-4}A^2 B), & \bar{Z}_M &= O(\epsilon^{-2}A). \end{aligned}$$

Here $\log(1/\delta_c) = O(B)$, and constants absorb the fixed relation $\alpha = 2\epsilon/5$. Substitution into equation 55 gives equation 57; the candidate-scanning term is dominant. □

PROOF OF THEOREM 5.4

The global output is at least as valuable as the candidate associated with the appropriate fixed value of H. It remains to check that the chosen internal error parameters lose at most ϵ in the approximation coefficient. The expected amortized query-complexity bound follows from the preceding theorem.

Proof. Fix a time. If $\Delta_t > 0$, Lemma B.18 supplies a suitable fixed value H whose structure is maintained unless the global singleton already has value at least $b(H)$. The global output is pointwise at least the corresponding candidate, so Theorem 5.3 applies with coefficient $\beta = \kappa\frac{a-1}{a}\left(\frac{1}{1+\epsilon_o} - \frac{1}{a}\right) \le \frac{1}{a}$. We verify that the parameter choice equation 2 makes $\beta \ge \rho - \epsilon$.

Let

$$b_\alpha = \frac{a-1}{a}\left(\frac{1}{1+\alpha} - \frac{1}{a}\right).$$

Since $b_0 = \rho$ and

$$\rho - b_\alpha = \frac{a-1}{a}\left(1 - \frac{1}{1+\alpha}\right) \le \alpha,$$

we have $b_\alpha \ge \rho - \alpha$. With the batch-size bound,

$$1 - \gamma = \frac{(1+\alpha) - (1-2\alpha)}{1+\alpha} = \frac{3\alpha}{1+\alpha} \le 3\alpha.$$

For $\alpha \le 1/25$,

$$\lambda_d \epsilon_d = (1+2\alpha)\frac{\alpha}{4} \le \frac{27}{100}\alpha, \qquad \gamma k \delta_c \le \frac{1}{100}\alpha.$$

Thus $\kappa \geq 1 - (82/25)\alpha$. Since $\rho < 2/5$,

$$\beta = \kappa b_\alpha \geq \left(1 - \frac{82}{25}\alpha\right)(\rho - \alpha) \geq \rho - \frac{289}{125}\alpha \geq \rho - \epsilon,$$

where the last inequality uses $\alpha = 2\epsilon/5$. The approximation guarantee $\mathbb{E}[F(S_t)] \geq \left((3-\sqrt{5})/2 - \epsilon\right) f(O_t) - \ell(O_t)$ follows from $\mathbb{E}[F(Q)] \geq \beta f(O_t) - \ell(O_t)$. If $\Delta_t = 0$, the empty set meets the same benchmark, as noted in the proof of Theorem 5.3. Feasibility follows from construction, and the expected amortized query-complexity bound follows from Theorem B.19. □

B.7 A Lower-Bound Observation

The zero-cost specialization shows that the regularized problem includes ordinary monotone maximization. Thus the existing dynamic lower bound above $1/2$ also applies here; it is not a new lower bound specifically for the regularized coefficient.

Setting $\ell \equiv 0$ removes the distinction between utility and regularized value. Any algorithm exceeding $1/2$ in the present guarantee would then exceed $1/2$ for the corresponding dynamic monotone problem.

Corollary B.20. *Suppose a fully dynamic regularized algorithm guaranteed, for some constant $\xi > 0$,*

$$\mathbb{E}[f(S_t) - \ell(S_t)] \geq \left(\frac{1}{2} + \xi\right) f(O_t) - \ell(O_t)$$

on every instance in the present oracle model. Setting $\ell \equiv 0$ would yield a $(1/2 + \xi)$-approximation for fully dynamic monotone cardinality maximization. Therefore the lower bound of Chen & Peng (2022) also applies to this special case, with the parameter ranges and update-sequence-length convention of that result.

Proof. With zero costs, $F = f$, the approximation benchmark becomes the ordinary multiplicative benchmark, and every dynamic monotone cardinality instance is a special case. The claim follows by restriction. □

B.8 An Auxiliary Bound

For completeness, we state the elementary decrease argument used to bound the number of levels. Only the marked steps need a multiplicative expected decrease; intervening steps are allowed as long as they do not increase the quantity.

We partition the positive values into dyadic ranges. Within a range, the assumed multiplicative decrease implies a fixed additive decrease per marked step, bounding the expected number of such steps spent in that range.

Lemma B.21. *Let (X_j) be a nonnegative, integer-valued, nonincreasing random sequence with $X_0 \leq U$. For each j, let $\mathcal{F}_j$ be a sigma-algebra with respect to which X_j and the marked-step indicator $I_j \in \{0, 1\}$ are measurable; the sigma-algebras need not be nested. Suppose that on $\{I_j = 1, X_j > 0\}$, $\mathbb{E}[X_j - X_{j+1} \mid \mathcal{F}_j] \geq \alpha X_j$ for some $\alpha > 0$. Then the expected number of marked steps with $X_j > 0$ is $O(\alpha^{-1} \log(U + 1))$. Unmarked steps need only be nonincreasing, and the sequence need not reach zero.*

Proof. The claim is immediate when $U = 0$. Otherwise partition the positive integers into the singleton band $\{1\}$ and the bands $(2^{h-1}, 2^h]$ for $h = 1, \ldots, \lceil \log_2 U \rceil$. For each band, let B_j indicate that step j is marked and X_j lies in that band. This indicator is $\mathcal{F}_j$-measurable. In a band with upper endpoint 2^h, the drift assumption gives $\mathbb{E}[B_j(X_j - X_{j+1})] \geq \alpha 2^{h-1}\mathbb{E}[B_j]$; the same weaker inequality also holds for $\{1\}$ with $h = 0$. Monotonicity implies pathwise that the sum of these decreases over all steps in the band is at most 2^h. Summing first over any finite prefix and taking expectations bounds the expected number of its marked steps in the band by $2/\alpha$. Summing over the bands and applying monotone convergence proves the claim. The argument uses conditional expectation separately at each step, so no nesting of the sigma-algebras is required. □

C Implementation Details

C.1 Boundary Cases

We consider $k \geq 1$ and atomic updates starting from the empty active set. No stream update is interleaved with a reconstruction. The structural invariants are asserted after completed updates, not during intermediate assignments. The marginal and cost groups use the intervals specified in Section 4.1; the final upper endpoint is included when needed. The acceptance test uses $\geq \eta$. A deletion reconstruction is performed as soon as the deleted source fraction reaches ϵ_d, so every current level satisfies the strict inequality in equation 14. If $k = 1$ or a source has one element, BatchSize returns one, and that element is accepted. If $\Delta_t = 0$, the empty set meets the approximation guarantee.

C.2 Stored Sets and Random Choices

An insertion creates a new record with a distinct identifier. Its deletion permanently marks that record inactive. A later insertion of the same underlying item receives a different identifier, so old sources and prefixes do not become active again. This convention does not restrict the original set problem: for any finite update prefix, let π map records to underlying items, define $f^{\#}(X) = f(\pi(X))$, and give each record its item's cost. The function $f^{\#}$ is normalized, monotone, and submodular. For $X \subseteq Y$, either the new record's item is already in $\pi(Y)$, in which case its marginal on Y is zero and its marginal on X is nonnegative, or diminishing returns follows from that of f. At most one record for each underlying item is active, so feasible active solutions and their objective values are unchanged. Each query to $f^{\#}$ is one query to f on $\pi(X)$. Costs are additive over records, including historical inactive records. We suppress the record notation elsewhere.

Prefixes are represented by pointers to their accepted blocks. A value query on a prefix or on the output is supplied with the union of the relevant blocks; this representation does not duplicate element records. The count routine finishes before the actual order U_i is sampled, and these stages use independent random choices. Trials in BatchSize are independent of each other, although observations at different positions of one trial may be dependent. Source buckets remain fixed until their levels are discarded. Reference-count updates and memory reclamation never influence reconstruction triggers, samples, batch-size estimates, or output values.