CONSISTENT ALGORITHMS FOR DYNAMIC NON-MONOTONE SUBMODULAR MAXIMIZATION WITH EFFICIENT UPDATES

Anonymous authors
Paper under double-blind review

ABSTRACT

Submodular maximization is a fundamental framework for subset selection with broad applications in machine learning and data mining. In many modern applications, however, the underlying data evolves over time, making it desirable to maintain not only a high-quality solution but also a stable one that changes only slightly after each update. Recent work has initiated the study of consistent algorithms for fully dynamic submodular maximization, but existing algorithms are restricted to monotone objectives, and their expected amortized query complexity suffers from an undesirable linear dependence on the maximum number n of active elements. In this paper, we address both limitations by developing CONSISTENTSAMPLE, a consistent algorithm for dynamic non-monotone submodular maximization under a cardinality constraint k. For accuracy parameter ϵ, our algorithm achieves a $(\frac{1}{3+2\sqrt{2}} - \epsilon)$-approximation with $\mathcal{O}(\log k/\epsilon^2)$ consistency and $\mathcal{O}\left(k^2 \log^4 \frac{k}{\epsilon} + \frac{k}{\epsilon^2} \log^2 \frac{k}{\epsilon}\right)$ expected amortized query complexity. To the best of our knowledge, among consistent algorithms for fully dynamic submodular maximization, ours is the first to achieve active-set-independent query complexity and the first to obtain a constant-factor approximation guarantee for non-monotone objectives. As a by-product, for monotone objectives, our framework achieves a $0.3178 - \mathcal{O}(1/k) - \epsilon$ approximation with $\mathcal{O}(\log(2/\epsilon)/\epsilon^2)$ consistency while retaining the same query complexity. We further evaluate our algorithm on the Max-Cut problem using real-world datasets, and the empirical results demonstrate its practical efficiency and stability.

1 INTRODUCTION

Submodular maximization is a fundamental framework for selecting a small, diverse, or representative subset from a large collection. It captures the well-known diminishing-returns property and has found broad applications, including data summarization for large language models (Fan et al., 2025; Renduchintala et al., 2023; Nguyen et al., 2025; Kumari et al., 2024; Renduchintala et al., 2024; Bhatt et al., 2024), influence maximization (Jin et al., 2021; Peng, 2021), feature selection (Bao et al., 2022), movie recommendation (Haba et al., 2020; Dütting et al., 2022; Avdiukhin et al., 2019; Nikolakaki et al., 2021), and related machine learning and data mining tasks.

In many of these applications, the underlying dataset evolves continuously over time, while the selected solution is repeatedly exposed to users or consumed by downstream systems. In such settings, solution quality alone is often insufficient: it is also desirable that a small change in the data does not trigger a drastic change in the maintained solution. For example, in data summarization, repeatedly replacing a large portion of the summary after minor data updates may reduce interpretability for users and introduce unnecessary instability for downstream learning models. Similarly, frequent large changes to a recommendation list may be undesirable when the underlying catalog changes only slightly. Motivated by this consideration, a recent line of work has studied *consistent submodular maximization*, where the goal is to maintain a high-quality approximate solution while limiting the number of changes between consecutive outputs (Dütting et al., 2024; 2025).

Early work on consistent submodular maximization mainly considered the insertion-only setting, where elements arrive over time but are never deleted. In many modern applications, however, elements may also expire, become unavailable, or be explicitly removed. This motivates the more

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.

general *fully dynamic* setting, in which the set of active elements changes through both insertions and deletions. Without consistency requirements, dynamic submodular maximization has been widely studied, and a series of algorithms achieve strong approximation guarantees with efficient update complexity (Lattanzi et al., 2020; Monemizadeh, 2020; Banihashem et al., 2023b;a; 2026a;b). Their expected amortized query complexity is either polylogarithmic in the maximum number n of active elements (e.g., $\mathrm{poly}(\log n, 1/\epsilon)$, where ϵ is the accuracy parameter), or entirely independent of n and bounded by $\mathrm{poly}(k, 1/\epsilon)$, where k denotes the maximum feasible solution size.

Constructing consistent submodular maximization algorithms in the fully dynamic setting is substantially more challenging. A deletion may remove an important element from the maintained solution and invalidate decisions made much earlier, while consistency prevents the algorithm from replacing a large portion of the solution at once. Dütting et al. (2026) recently made an important step in this direction by giving the first constant-factor approximation algorithms with sublinear consistency in the dynamic setting. For a cardinality constraint, their algorithms achieve a $1/2 - \mathcal{O}(\epsilon)$ approximation with $\mathcal{O}(1/\epsilon^2)$ consistency and $\mathcal{O}(n)$ expected amortized query complexity; for a rank-k matroid constraint, they obtain a $1/4 - \mathcal{O}(\epsilon)$ approximation with $\mathcal{O}(\log k/\epsilon^2)$ consistency and $\mathcal{O}(n^2/k)$ expected amortized query complexity.[1]

Despite this progress, Dütting et al. (2026) still leaves two important gaps. First, their algorithm has an undesirable expected amortized query complexity of $\mathcal{O}(n)$, which is linear in the number of active elements and does not achieve the efficient update time attained by the aforementioned dynamic algorithms. Thus, although their maintained solution changes only slightly from one update to the next, maintaining such a solution may still incur a high computational cost. More importantly, their theoretical guarantees apply only to *monotone* submodular functions, which limits the applicability of their algorithm to the broader and more general class of *non-monotone* submodular maximization problems. These gaps lead to the following open questions in the fully dynamic setting:

> ***Q1.*** *Can we design consistent algorithms for dynamic submodular maximization with efficient amortized query complexity?*
>
> ***Q2.*** *Can consistent algorithms for dynamic submodular maximization go beyond monotone objectives to general non-monotone functions?*

Our Contributions. We answer both questions affirmatively by developing novel, consistent algorithms for dynamic submodular maximization. Our main contributions are summarized as follows.

- We develop ConsistentSample, a consistent algorithm for dynamic non-monotone submodular maximization under a cardinality constraint. It achieves a $(1/(3+2\sqrt{2}) - \epsilon)$-approximation with $\mathcal{O}(\log k/\epsilon^2)$ consistency and $\mathcal{O}(k^2 \log^4 \frac{k}{\epsilon} + \frac{k}{\epsilon^2} \log^2 \frac{k}{\epsilon})$ expected amortized query complexity. To the best of our knowledge, among consistent algorithms for fully dynamic submodular maximization, ours is the first to achieve update efficiency independent of the active-set size, and the first to obtain a constant-factor approximation guarantee for non-monotone objectives.
- For monotone objectives, the same framework yields a stronger approximation guarantee of $0.3178 - \mathcal{O}(1/k) - \epsilon$ and removes the dependence on k from the consistency bound, achieving $\mathcal{O}(\log(2/\epsilon)/\epsilon^2)$ consistency while retaining the same active-set-independent expected amortized query complexity.
- We empirically evaluate our algorithm on the Max-Cut problem using real-world datasets. The results convincingly demonstrate its practical efficiency and stability.

Challenges and Techniques. Consistency, constant-factor approximation, and efficient dynamic updates have each been achieved in related settings, but achieving all three simultaneously is substantially more challenging and cannot be accomplished by directly combining existing techniques. Techniques that provide deletion robustness or gradual solution changes may require repeatedly processing large candidate sets, while algorithms for non-monotone objectives are difficult to maintain under deletions. Our algorithm resolves these conflicts through the following tightly coupled ideas.

For non-monotone submodular maximization, the sampled-exchange rule of Feldman et al. (2018) serves as a standard building block. However, under deletions, its history dependence becomes prob-

[1] At update t, X_t denotes the active set, $S_t \subseteq X_t$ the maintained solution, and OPT_t the optimal value over feasible subsets of X_t. An α-approximation satisfies $\mathbb{E}[f(S_t)] \geq \alpha\,\mathrm{OPT}_t$, while C-consistency requires $|S_t \triangle S_{t-1}| \leq C$ after every update. Formal definitions appear in Section 1.2.

lematic, since an earlier deletion may invalidate many subsequent marginal evaluations. Restarting the execution from scratch after such updates would lead to excessively high update cost. We therefore develop a dynamic compression mechanism that preserves the valid part of a randomized execution and reconstructs only the affected suffix. Elements with comparable marginal gains are grouped into large buckets and sampled uniformly. For each randomized trial, we retain enough information to recover the execution before that trial, including accepted and rejected trials. After an update, only the earliest invalid suffix is rebuilt with fresh randomness. A key amortization follows from uniform sampling: if a trial samples from a source of size m, rebuilding its suffix costs proportionally to m, whereas the probability that its sampled element is deleted is only $\mathcal{O}(\frac{1}{m})$. These factors cancel in expectation. Together with the singleton-value decomposition, this allows the dynamic maintenance cost to be independent of the size of the current active set.

Moreover, the existing deletion-robust technique proposed by (Zhang et al., 2022) and used by the recent consistent dynamic algorithm (Dütting et al., 2026) requires processing its input candidate set. Applying it directly would retain a large update cost. Thus, our dynamic compression first reduces the active instance to a residual of size $\mathrm{poly}(k, \frac{1}{\epsilon})$. We then apply the deletion-robust technique only to this residual over geometrically decreasing deletion horizons $d_0, d_1, \ldots, d_L$. At horizon d_i, the refinement leaves only $\mathcal{O}(d_i/\epsilon)$ unresolved candidates while controlling the expected loss caused by the next $\mathcal{O}(d_i)$ deletions. Consequently, finer checkpoints require less computation and a smaller change from their predecessors. This creates a "scale–distance coupling" in which the same residual bound simultaneously controls deletion robustness, update cost, and the distance of the next target.

Also note that existing fully dynamic consistency frameworks not only incur substantial update costs, but are also limited to monotone objectives. Their monotone completion directly adds residual candidates, which is invalid for non-monotone objectives because additions may decrease the value. More subtly, after dynamic reconstruction, merely preserving a legal exchange order is not enough: the randomized acceptance probabilities required by the static non-monotone approximation analysis must also remain valid. We address these difficulties by separating the internal checkpoint from the public solution and completing the remaining residual with the sampled-exchange rule rather than unconditional additions. The public solution is still changed gradually through the randomized handover schedule, while the approximation analysis applies to non-handover updates. The key technical step is to show that the reconstructed processing segments can be concatenated into a single valid sampled-exchange execution and, crucially, that the retained randomized trials preserve the acceptance and rejection identities required by the non-monotone analysis. Together, these ideas reconcile consistency, constant-factor approximation, and efficient dynamic updates.

1.1 Additional Related Work.

Dynamic submodular maximization. Fully dynamic submodular maximization was initiated by Monemizadeh (2020); Lattanzi et al. (2020) for monotone objectives under a cardinality constraint, and was later extended to more general constraints and objectives (Banihashem et al., 2023b; 2026a). Under a cardinality constraint, the best-known ratio is $1/2 - \epsilon$ for monotone objectives, with $\mathrm{poly}(\log n, \log k, 1/\epsilon)$ expected amortized query complexity (Banihashem et al., 2023a), and $0.277 - \epsilon$ for non-monotone objectives, with $\mathrm{poly}(k, 1/\epsilon)$ worst-case expected query complexity (Banihashem et al., 2026b). These algorithms optimize approximation and update efficiency but provide no non-trivial consistency guarantees.

Consistent submodular maximization. The explicit study of consistent submodular maximization is considerably more recent. It was initiated by Dütting et al. (2024) and subsequently studied by Dütting et al. (2025), both in the simpler insertion-only setting, where elements arrive but are never deleted. More recently, Dütting et al. (2026) extended this line of work to the fully dynamic setting. However, as discussed in Sect. 1, these works still have two important limitations: high update complexity and no guarantees for non-monotone objectives. Separately, the streaming algorithm of Feldman et al. (2018) for non-monotone submodular maximization implicitly achieves 2-consistency under insertion-only settings, but neither its approximation nor its consistency guarantee holds in the fully dynamic setting with arbitrary deletions.

1.2 Preliminaries

Let V be a finite ground set and $f : 2^V \to \mathbb{R}_{\geq 0}$ be normalized, i.e., $f(\emptyset) = 0$. For $S \subseteq V$ and $u \in V \setminus S$, let $f(u \mid S) := f(S \cup \{u\}) - f(S)$ denote the marginal gain. We call f submodular if

Algorithm 1 CONSISTENTSAMPLE(f, k, ϵ)

Ensure: a feasible solution $S_t \subseteq X_t$ with $|S_t| \leq k$ after every update t

set $\widehat{\epsilon} \leftarrow \min\{\epsilon, 1/8\}$, $\eta \leftarrow \widehat{\epsilon}/32$, $\vartheta \leftarrow 3\widehat{\epsilon}$, $\delta \leftarrow 4\widehat{\epsilon}$, and $b_1 \leftarrow \lceil 2/\delta \rceil$; initialize $X_0 = S_0 = \emptyset$
if $k \geq 2b_1$ **then**
  $L \leftarrow \max\{\ell \geq 1 : 2^\ell \lceil (\ell+1)/\delta \rceil \leq k\}$; $b \leftarrow \lceil (L+1)/\delta \rceil$
  set $d_i \leftarrow b2^{L-i}$ and $h_i \leftarrow 2^{L-i}$ for $i = 0, \ldots, L$; sample a global random time shift and fix the staggered handover schedule of Appendix D.2 using $\{(d_i, h_i)\}_{i=0}^{L}$
for each update (v_t, op_t) at time t **do**
  $X_t \leftarrow X_{t-1} \cup \{v_t\}$ if $\mathrm{op}_t = \text{insert}$; otherwise $X_t \leftarrow X_{t-1} \setminus \{v_t\}$
  call MAINTAINBAND$(v_t, \mathrm{op}_t, \gamma)$ for every dyadic γ satisfying $\eta\gamma/k \leq f(\{v_t\}) \leq \gamma$
  if $k < 2b_1$ **then**
    $(I, C) \leftarrow$ SELECTCHECKPOINT$(t, 2k)$; $\widehat{S}_t \leftarrow$ SAMPLEDEXCHANGE$(I, C) \cap X_t$
    output $S_t \leftarrow \arg\max_{S \in \{S_{t-1} \cap X_t, \widehat{S}_t\}} f(S)$; **continue**
  if a level-i handover starts at t **then**
    if $i = 0$ **then** $(I_t, C_t) \leftarrow$ SELECTCHECKPOINT$(t, 2d_0)$
    else
      let a be the parent level-$(i-1)$ checkpoint time specified by the schedule
      $(I_t, C_t) \leftarrow$ RESIDUALSAMPLE$(I_a, \mathcal{U}(a,t), d_i)$ where $\mathcal{U}(a,t) := (C_a \cap X_t) \cup (X_t \setminus X_a)$
    $T_t \leftarrow I_t \cap X_t$; scan $(C_t \cap X_t) \setminus T_t$ with elements in S_{t-1} first, and add a scanned element u to T_t if $|T_t| < k$ and $f(u \mid T_t) > 0$
    freeze T_t and split a feasible path from $S_{t-1} \cap X_t$ to T_t into h_i blocks
  if t lies in a handover interval **then**
    $S_t \leftarrow$ the next state of the frozen path, after removing inactive elements
  else
    let a be the most recent level-L checkpoint preceding t
    $\widehat{S}_t \leftarrow$ SAMPLEDEXCHANGE$(I_a, \mathcal{U}(a,t)) \cap X_t$, where $\mathcal{U}(a,t) := (C_a \cap X_t) \cup (X_t \setminus X_a)$;
    $S_t \leftarrow \arg\max_{S \in \{S_{t-1} \cap X_t, \widehat{S}_t\}} f(S)$
  output S_t

$f(u \mid A) \geq f(u \mid B)$ for all $A \subseteq B \subseteq V$ and $u \in V \setminus B$, and monotone if $f(A) \leq f(B)$ for all $A \subseteq B \subseteq V$. Unless stated otherwise, f need not be monotone. Consider a dynamic sequence of T single-element insertions and deletions starting from $X_0 = \emptyset$, and let $X_t \subseteq V$ be the active set after update t and $n := \max_{0 \leq t \leq T} |X_t|$. Following the standard oblivious adversarial model (e.g., Monemizadeh, 2020; Banihashem et al., 2023b; Dütting et al., 2026), the update sequence is fixed independently of the algorithm's randomness, and each element is inserted at most once.

We study *fully dynamic submodular maximization under a cardinality constraint* k, i.e., $\max\{f(S) : S \subseteq X_t,\ |S| \leq k\}$ at any time t. Let O_t be an optimal solution, $\mathrm{OPT}_t := f(O_t)$, and S_t the feasible solution maintained by algorithms after update t. An algorithm is an α-approximation if $\mathbb{E}[f(S_t)] \geq \alpha \cdot \mathrm{OPT}_t$ for every t, and is C-consistent if $|S_t \triangle S_{t-1}| \leq C$ (Dütting et al., 2026), where $\triangle$ denotes symmetric difference. As standard (e.g., Feldman et al., 2023; Lattanzi et al., 2020), we assume value-oracle access to f and measure update efficiency by the number of oracle queries. An expected amortized query complexity Q means the expected total number of oracle queries over any T updates is $\mathcal{O}(T \cdot Q)$.

2 ALGORITHM DESIGN

In this section, we propose CONSISTENTSAMPLE (Algorithm 1 in Section 2.1), our algorithm for fully dynamic non-monotone submodular maximization, designed to coordinate three requirements that pull in different directions: maintaining a high-quality solution under deletions, avoiding repeated scans of the entire active set, and limiting the change of the returned solution after each update. We organize the algorithm into several modules with complementary roles. To adhere to space limitations and keep the algorithmic presentation focused, supporting routines and lower-level technical details are deferred to Appendix A–D. This section emphasizes why each component is needed and how the components work together.

2.1 OVERALL ALGORITHM

Algorithm 1 summarizes the online flow of CONSISTENTSAMPLE. First, when $k \geq 2b_1$, it constructs $L+1$ geometrically decreasing time scales for subsequent multiscale refinement. At level i, d_i specifies how long the corresponding checkpoint is intended to remain useful, while h_i specifies

Algorithm 2 BUCKETSAMPLE(S_0, U_0, γ)

1: $S \leftarrow S_0$; $U \leftarrow U_0$; $c \leftarrow \sqrt{2}$; $q \leftarrow (2+\sqrt{2})^{-1}$; $\ell \leftarrow \eta\gamma/k$; $B \leftarrow 1 + \lceil \log_2(k/\eta) \rceil$; $M \leftarrow \lceil 32k/\vartheta \rceil$
2: **while** $U \neq \emptyset$ **do**
3: evaluate $g(u) \leftarrow f(u \mid S)$ for all $u \in U$
4: $U \leftarrow \{u \in U : g(u) \geq \ell,\ g(u) \geq (1+c)w(r(S))\}$ ▷ $w(\cdot)$ stores acceptance marginals; $r(S)$ is a minimum-weight element, or a zero-weight dummy if $|S| < k$
5: $G_a \leftarrow \{u \in U : 2^a \ell \leq g(u) < 2^{a+1}\ell\}$ for $a = 0, \ldots, B-1$
6: **if** $\max_a |G_a| < M$ **then return** (S, U)
7: **while** $\max_a |G_a| \geq M$ **do**
8: $a^* \leftarrow \arg\max_a |G_a|$; sample p uniformly from G_{a^*}
9: append a record $(I_j, F_j^0, F_j, n_j, R_j, p_j) \leftarrow (S, G_{a^*}, G_{a^*}, |G_{a^*}|, U, p)$
10: draw and store $B_j \sim \text{Bernoulli}(q)$ independently
11: $U \leftarrow U \setminus \{p_j\}$; $G_{a^*} \leftarrow G_{a^*} \setminus \{p_j\}$
12: **if** $B_j = 1$ **then**
13: $S \leftarrow (S \setminus \{r(S)\}) \cup \{p_j\}$; $w(p_j) \leftarrow g(p_j)$; **break** ▷ recompute marginal gain if S changes
14: **return** (S, U)

how many updates are available to move toward its target. Since $d_i = bh_i$, coarser checkpoints, which may induce larger target changes, receive proportionally longer handover intervals. Then a single random time shift fixes the staggered handover schedule for the entire execution (Lines 2–4).

Upon an insertion or deletion (v_t, op_t), where v_t is the affected element and $\text{op}_t \in \{\text{insert}, \text{delete}\}$, let X_t denote the active set after the update. The algorithm then modifies only the dyadic singleton-value bands containing v_t, as determined by $f(\{v_t\})$. MAINTAINBAND (Alg. 3) updates the corresponding compressed states without revisiting unrelated elements, allowing the dynamic-compression module to avoid rescanning the entire active set after an update (Lines 6–7). When k is small, the full multiscale construction is unnecessary. We simply use SELECTCHECKPOINT (Alg. 4 in Appendix A.3) to obtain a coarse internal solution and its unresolved residual, then process the residual directly with SAMPLEDEXCHANGE (Alg. 6 in Appendix A.6). At each update, we retain the better of the fresh candidate and $S_{t-1} \cap X_t$ (Lines 8–10). When $k \geq 2b_1$, new checkpoints are computed only as required by the fixed handover schedule. A level-0 checkpoint is obtained from the dynamically maintained bands via SELECTCHECKPOINT. At a finer level $i > 0$, the schedule specifies a level-$(i-1)$ parent checkpoint, and RESIDUALSAMPLE (Algorithm 5 in Appendix A.4) continues from its internal solution. Importantly, it processes only the unresolved candidates: the surviving parent residual together with later arrivals. Thus finer checkpoints continue the previous computation rather than restart from the current active set, progressively reducing the unresolved set at shorter time scales (Lines 11–15). The resulting checkpoint (I_t, C_t) is an internal optimization state, not a public solution for users. We first construct a public target T_t from the surviving part of I_t and fill any remaining capacity with positive-marginal residual elements. Elements already in the previous public solution S_{t-1} are prioritized to avoid unnecessary replacements among valid completions. The target is then frozen, and its potentially large change from the current public solution is exposed over h_i blocks rather than all at once (Lines 16–17). Finally, if the current time lies inside a handover interval, we advance by one block along the frozen path and remove elements that have become inactive. Otherwise, we start from the most recent finest-level checkpoint and apply SAMPLEDEXCHANGE only to the unresolved candidates, producing a fresh candidate $\widehat{S}_t$. The algorithm then returns the better of $\widehat{S}_t$ and $S_{t-1} \cap X_t$, avoiding unnecessary changes without reducing solution quality. Since the preceding stages have compressed the unresolved set, producing $\widehat{S}_t$ does not require reprocessing the entire active set (Lines 18–24).

2.2 DYNAMIC COMPRESSION

Recall that, after an update (v_t, op_t), CONSISTENTSAMPLE calls MAINTAINBAND only for dyadic scales satisfying $\eta\gamma/k \leq f(\{v_t\}) \leq \gamma$. This localizes each update to $\mathcal{O}(\log(k/\eta))$ band states, rather than requiring all maintained states to be revisited. The remaining question is how to maintain each affected band without reprocessing all active elements in the band after every update. We maintain a partial randomized selection process for each band. BUCKETSAMPLE compresses its unprocessed candidates into a small residual while retaining the sampled-exchange rule used in the approximation analysis, whereas MAINTAINBAND reuses this execution across updates and rebuilds only the part that becomes invalid.

Algorithm 3 MAINTAINBAND(v, op, γ)

1: $B \leftarrow 1 + \lceil \log_2(k/\eta) \rceil$; $M \leftarrow \lceil 32k/\vartheta \rceil$
2: **if** op $=$ insert **then**
3: $\quad C_{\mathrm{end}} \leftarrow C_{\mathrm{end}} \cup \{v\}$; $R_j \leftarrow R_j \cup \{v\}$ for every live record j; $J \leftarrow \{j : |R_j| > 2Bn_j\}$
4: **else**
5: $\quad C_{\mathrm{end}} \leftarrow C_{\mathrm{end}} \setminus \{v\}$; $F_j \leftarrow F_j \setminus \{v\}$ and $R_j \leftarrow R_j \setminus \{v\}$ for every live record j
6: $\quad J \leftarrow \{j : v = p_j\} \cup \{j : |F_j| < n_j/2\}$
7: **if** $J \neq \emptyset$ **then**
8: $\quad j^* \leftarrow \min J$; $(I, U) \leftarrow (I_{j^*}, R_{j^*})$; discard records $j^*, j^*+1, \ldots$;
9: $\quad (S_{\mathrm{end}}, C_{\mathrm{end}}) \leftarrow$ BUCKETSAMPLE(I, U, γ)
10: **else if** op $=$ insert and $|C_{\mathrm{end}}| = 2BM$ **then** $(S_{\mathrm{end}}, C_{\mathrm{end}}) \leftarrow$ BUCKETSAMPLE$(S_{\mathrm{end}}, C_{\mathrm{end}}, \gamma)$

BUCKETSAMPLE. Algorithm 2 gives the compression routine for each affected band. Starting from an inherited solution S_0 and a set of unprocessed candidates U_0 at the current scale γ, BUCKETSAMPLE reduces the candidate set to a small unresolved residual while retaining the sampled-exchange rule used in the approximation analysis. More specifically, apart from the minimum marginal floor $\ell = \eta\gamma/k$, it uses the same eligibility, Bernoulli-acceptance, and replacement rule as SAMPLEDEXCHANGE (Algorithm 6 in Appendix A.6). The main difference from SAMPLEDEXCHANGE is how candidates are organized and processed. After evaluating their current marginals, BUCKETSAMPLE partitions the eligible elements into dyadic marginal-gain buckets and samples uniformly from a largest bucket whose size is at least $M = \lceil 32k/\vartheta \rceil$ (Lines 4–10). If every bucket is smaller than M, the routine stops and returns the remaining candidates as the unresolved residual (Line 6). This bucketed organization serves two purposes. Marginal gains within one bucket differ by at most a factor of two, so a small set of later deletions cannot carry a disproportionate fraction of the expected stored weight. Uniform sampling from a large bucket also spreads the selection probability over many candidates. Since there are only $B = \mathcal{O}(\log(k/\eta))$ buckets and each contains fewer than M elements at termination, the residual has size $\mathcal{O}((k/\vartheta)\log(k/\eta))$, independently of $|X_t|$.

The remaining difficulty is that this compressed execution is history-dependent. Whenever an accepted replacement changes S, all later marginal tests are evaluated against the resulting solution. Thus deleting an element involved in an earlier randomized decision may invalidate not only that decision but also a suffix of the subsequent execution. Recomputing the whole band after every such update would eliminate the benefit of the compression.

MAINTAINBAND. To avoid restarting from scratch, MAINTAINBAND stores enough information to preserve the valid prefix of a previous BUCKETSAMPLE execution and rebuild only the affected suffix. For every randomized trial j, it records the solution immediately before the trial, the original sampling source F_j^0 with $n_j = |F_j^0|$, the sampled candidate p_j, its Bernoulli outcome, and the eligible suffix at that point. Both accepted and rejected trials are retained, since otherwise a rejected candidate could disappear permanently from later reconstructions while subsequent randomized decisions are retried. For each scale γ, the maintained state consists of the recorded execution history and the terminal pair $(S_{\mathrm{end}}, C_{\mathrm{end}})$. Let F_j and R_j denote the currently active source and suffix associated with record j. A record remains reusable while $|F_j| \geq n_j/2$ and $|R_j| \leq 2Bn_j$. After an update, MAINTAINBAND identifies the earliest record whose sampled candidate has been deleted or whose source or suffix no longer satisfies these conditions. It preserves the valid prefix and rebuilds only the remaining suffix (Lines 7–9). If all records remain valid but the terminal residual reaches $2BM$, the routine simply resumes BUCKETSAMPLE from the current terminal state (Line 10). The restart thresholds are chosen so that rebuilding remains cheap in expectation. Source shrinkage and suffix growth require sufficiently many intervening updates and can therefore be charged to those updates. For a record created from a source of size n_j, deletion of its sampled candidate occurs with conditional probability $\mathcal{O}(1/n_j)$, whereas rebuilding the associated suffix incurs a factor $\mathcal{O}(n_j)$ in its query cost. These factors cancel in expectation, yielding an amortized dynamic-compression cost that is independent of the active-set size.

2.3 MULTISCALE RESIDUAL REFINEMENT

The dynamically maintained band states provide an efficient coarse representation, but their residuals are still too large to support small changes of the public solution, and a state kept for too long may lose value under deletions. We therefore refine the residual over multiple deletion horizons. At a level-0 checkpoint, SELECTCHECKPOINT (Algorithm 4 in Appendix A.3) uses the active singleton values to select a relevant maintained scale without scanning the active set. Its quantile choice keeps

only $\mathcal{O}(d_0)$ heavier elements outside the selected band and ensures that the selected scale remains at most twice the optimum throughout the checkpoint lifetime. Thus, the omitted low-singleton elements incur only an $\mathcal{O}(\eta)$ loss, while the coarse residual has size $\mathcal{O}(Bk/\vartheta)$, independent of $|X_t|$.

Starting from this coarse state, each finer checkpoint processes only the surviving residual of its parent together with subsequent arrivals. RESIDUALSAMPLE (Algorithm 5 in Appendix A.4) samples an eligible candidate u with probability proportional to $1/f(u \mid S)$ and stops once fewer than $8d/\vartheta$ eligible candidates remain. The inverse-marginal rule makes the marginal contribution cancel its own sampling probability: against any fixed set of at most $2d$ subsequent deletions, the expected deleted stored weight is at most a $\vartheta/4$ fraction of the expected sampled weight. At the same time, the stopping rule leaves only $\mathcal{O}(d/\vartheta)$ unresolved candidates. Applying this refinement at geometrically decreasing horizons $d_0, d_1, \ldots, d_L$ couples checkpoint lifetime with target distance: checkpoints refreshed more frequently leave smaller residuals, whereas coarser checkpoints may move farther but are refreshed less often.

2.4 CONSISTENT OUTPUT

The checkpoint hierarchy bounds the distance between successive internal targets, but even this distance may be too large to expose in a single update. We therefore introduce each target gradually through the randomly shifted staggered handover schedule of Appendix D.2. At the start of a level-i handover, the active checkpoint is completed using positive-marginal elements from its residual, with elements already present in the previous public solution considered first; the exact completion rule is given in Appendix A.5. The resulting target is then frozen. Since a level-i target is at distance $\mathcal{O}(d_i/\vartheta)$ from the relevant previous state and its handover lasts $h_i = d_i/b$ updates, distributing the change across the handover yields only $\mathcal{O}(b/\vartheta)$ changes per update.

For a non-monotone objective, however, the intermediate sets along a handover path need not retain the value of its two endpoints. We therefore establish the approximation guarantee only outside handover intervals. There, the remaining finest-level residual is completed by SAMPLEDEXCHANGE, and the value-preserving comparison with the surviving previous solution can only improve the returned value. The independent random shift ensures that any fixed update lies in a handover interval with probability at most δ; since f is nonnegative, these handover states cost only this probability in the expected approximation analysis. Thus the public handover mechanism controls consistency without requiring high objective value at every intermediate path state.

3 THEORETICAL ANALYSIS

In this section, we analyze the guarantees of CONSISTENTSAMPLE, giving the key proof ideas here and deferring complete proofs to Appendix B. The parameters η, ϑ, and δ respectively control threshold loss, deletion loss, and handover probability. For a random shift σ, let $\mathcal{H}_\sigma$ denote the update times lying in handover intervals.

Approximation Ratio. First consider the case $k \geq 2b_1$. Fix a realization σ of the random shift and a time $t \notin \mathcal{H}_\sigma$. Let $\tau_0 < \tau_1 < \cdots < \tau_L < t$ be the checkpoint times used to construct $\widehat{S}_t$. Concatenating the realized processing orders at these checkpoints with the final SAMPLEDEXCHANGE call yields a single candidate-trial sequence. Each level $i > 0$ processes only the unresolved candidates inherited from level $i-1$ together with later arrivals, and accepted trials use the same replacement rule as SAMPLEDEXCHANGE; the only additional deterministic rejection is the minimum marginal threshold $\eta\gamma/k$ in the first segment. Appendix D.3 formalizes this concatenation. Moreover, $f(S_t) = \max\{f(\widehat{S}_t), f(S_{t-1} \cap X_t)\}$ pointwise, so it suffices to derive a lower bound on $f(\widehat{S}_t)$ to establish the approximation ratio. The following three steps summarize the argument.

First, omitting elements whose singleton values fall below the threshold incurs only a small loss. The singleton quantile selected by SELECTCHECKPOINT bounds the contribution of excluded optimal elements by $2\eta\mathrm{OPT}_t$. *Second, the static sampled-exchange approximation extends to the dynamically reconstructed execution.* The concatenated sequence satisfies the deterministic eligibility and replacement inequalities used in Appendix A.6, while Lemma D.2 preserves the acceptance and rejection identities needed for the expectation analysis after dynamic reconstruction. Together, these yield the $1/(3+2\sqrt{2})$ factor, up to the above threshold loss. *Third, deletions remove only a small fraction of the marginal contributions of accepted elements.* For each accepted element u, let $w(u)$ denote its marginal contribution at acceptance. At level 0, uniform sampling from a marginal bucket

of size at least M, and at finer levels, sampling proportional to $1/f(u \mid S)$, both ensure that the expected total $w(u)$ of accepted elements that become inactive is at most a $\vartheta/4$ fraction of the expected total acceptance marginal. Since each replacement increases the acceptance marginal along its replacement chain by a factor of at least $1 + c$, geometric summation converts this loss into a lower bound on the value of the active final solution. Combining the three bounds, for $k \geq 2b_1$ and $t \notin \mathcal{H}_\sigma$, we have $\mathbb{E}[f(S_t) \mid \sigma] \geq \left(1 - \frac{(1+c)\vartheta}{4c}\right)\left(\frac{1-2\eta}{3+2\sqrt{2}} - \frac{2c\eta}{(1+c)^2}\right)\mathrm{OPT}_t$ where $c = \sqrt{2}$. For any fixed update t, the random shift satisfies $\Pr_\sigma[t \in \mathcal{H}_\sigma] \leq \delta$. Since $f \geq 0$, omitting the event $t \in \mathcal{H}_\sigma$ from the lower bound introduces only the multiplicative factor $1 - \delta$. Substituting the parameters of Algorithm 1 therefore gives $\mathbb{E}[f(S_t)] \geq \left(1/(3 + 2\sqrt{2}) - \epsilon\right)\mathrm{OPT}_t$. When $k < 2b_1$, the checkpoint is constructed and used at the same update, with no handover schedule. Hence the same approximation bound follows without any loss from intervening deletions or handover events.

Consistency. The consistency analysis depends on the number of candidates processed between consecutive checkpoints. Let $B := 1 + \lceil \log_2(k/\eta) \rceil$. The level-0 checkpoint leaves $\mathcal{O}(Bk/\vartheta)$ unresolved candidates. After the first refinement, a checkpoint with deletion horizon d_i leaves $\mathcal{O}(d_i/\vartheta)$ unresolved candidates, and only $\mathcal{O}(d_i)$ new elements can arrive before that checkpoint is replaced (Lemma B.4). Processing one candidate changes the internal solution by at most two elements, and target completion adds at most the number of unresolved candidates. Between a parent checkpoint and its child, at most $\mathcal{O}(d_{i-1})$ deletions occur; since $\vartheta \leq 1$, this term is also upper-bounded by $\mathcal{O}(d_{i-1}/\vartheta)$. Summing over the geometrically decreasing horizons yields an $\mathcal{O}(d_i/\vartheta)$ endpoint-distance bound (Lemma B.6).

Dividing this distance among $h_i = d_i/b$ updates gives $\mathcal{O}(b/\vartheta)$ changes per handover update. Outside handover intervals, Algorithm 1 compares the fresh candidate $\widehat{S}_t$ with the surviving previous solution $S_{t-1} \cap X_t$ and returns the one with larger objective value (Lines 22–24). Both choices remain within $\mathcal{O}(b/\vartheta)$ of the most recent level-L checkpoint, so this comparison also changes $\mathcal{O}(b/\vartheta)$ elements per update; the detailed inductive argument is deferred to Appendix B. Since $b = \mathcal{O}(\log k/\delta)$, the resulting consistency is $\mathcal{O}(\log k/\epsilon^2)$ for every realization of the algorithm's randomness.

Query Complexity. Within one singleton-value band, each accepted replacement increases the recorded acceptance marginal along its replacement chain by a factor of at least $1+c$. Consequently, the number of accepted trials is at most $\kappa = \mathcal{O}(k \log(k/\eta))$. Reconstructing a suffix containing r candidates therefore uses $\mathcal{O}(r\kappa)$ marginal queries. A reconstruction caused by $|R_j| > 2Bn_j$ or $|F_j| < n_j/2$ can occur only after sufficiently many insertions or deletions and is charged to those updates. If the sampled candidate p_j is deleted, then conditional on record j being retained, its deletion probability is $\mathcal{O}(1/n_j)$, whereas reconstructing the corresponding suffix costs $\mathcal{O}(Bn_j\kappa)$ queries. The factor n_j therefore cancels in conditional expectation. The number of accepted records is at most κ deterministically, and the expected number of retained records is at most κ/q by Lemma A.6. Summing these reconstruction costs over the affected records and the $\mathcal{O}(\log(k/\eta))$ bands touched by one update gives $\mathcal{O}(k^2 \log^4(k/\eta))$ expected amortized oracle queries per update (Lemma A.10).

The remaining calls operate only on the candidate sets bounded in Lemma B.4. The level-1 refinement contributes $\mathcal{O}(B^2k/\vartheta^2)$ expected amortized queries. For $i \geq 2$, the costs form a geometric sum of $\mathcal{O}(k/\vartheta^2)$. The final SAMPLEDEXCHANGE call and the value comparison at every non-handover update are lower-order terms. Combining all the above gives the following theorem.

Theorem 3.1. *For $\epsilon \in (0, 1/4)$,* CONSISTENTSAMPLE *maintains a feasible $S_t \subseteq X_t$, $|S_t| \leq k$, such that $\mathbb{E}[f(S_t)] \geq \left(1/(3 + 2\sqrt{2}) - \epsilon\right)\mathrm{OPT}_t$. It is $\mathcal{O}(\log k/\epsilon^2)$-consistent, with expected amortized query complexity $\mathcal{O}(k^2 \log^4 \frac{k}{\epsilon} + \frac{k}{\epsilon^2}\log^2 \frac{k}{\epsilon})$.*

Performance Bounds for Monotone Objectives. When f is monotone, the final residual can be exposed directly in reserved positions rather than processed through another non-monotone exchange completion. This allows the hierarchy to stop once the residual is small enough to fit in the reserved capacity. Appendix C develops this specialization and proves a $0.3178 - \mathcal{O}(1/k) - \epsilon$ approximation with $\mathcal{O}(\log(2/\epsilon)/\epsilon^2)$ consistency and the same asymptotic query bound.

4 EXPERIMENTS

We empirically evaluate CONSISTENTSAMPLE on Max-Cut, focusing on three criteria targeted by our algorithm: solution quality, oracle query efficiency, and solution stability. Accordingly, we use

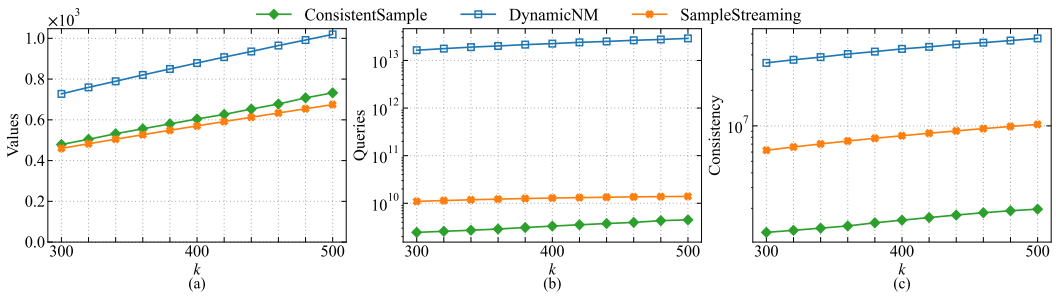

Figure 1: Panels (a)–(c) report the average objective value, total oracle queries, and cumulative solution changes, respectively. Higher is better in (a), whereas lower is better in (b)–(c).

three evaluation metrics: (1) the average objective value over the update sequence; (2) the total number of oracle queries; and (3) the cumulative number of solution changes $\sum_t |S_t \triangle S_{t-1}|$.

Baselines. We compare CONSISTENTSAMPLE against two complementary baselines. The first is SAMPLESTREAMING (Feldman et al., 2018), a classical algorithm for non-monotone submodular maximization with 2-consistency in the insertion-only setting. Following Banihashem et al. (2023b), we handle deletions by rerunning SAMPLESTREAMING on the current active set whenever a deleted element belongs to its maintained solution; otherwise, no reconstruction is needed. Our second baseline, denoted by DYNAMICNM, is the state-of-the-art fully dynamic algorithm for non-monotone submodular maximization under a cardinality constraint (Banihashem et al., 2026b). We use the 0.262-approximation variant because it provides a more practical approximation–efficiency trade-off for our comparison.

Application and Implementation. Following prior work on dynamic non-monotone submodular maximization (Banihashem et al., 2023b), we consider the Max-Cut problem on a real-world network. Given a graph $G = (V, E)$, the non-monotone submodular objective is defined as $f(S) = \sum_{\{u,v\} \in E} w_{uv} \mathbf{1}[|\{u, v\} \cap S| = 1]$. We set all edge weights to one and use the Brightkite graph from SNAP, containing $58{,}228$ vertices and $214{,}078$ edges (Leskovec & Krevl, 2014). To generate an algorithm-independent update sequence, following (Lattanzi et al., 2020; Banihashem et al., 2023b), we insert the vertices according to a uniformly random permutation and maintain at most $W = 4000$ active vertices. Once this limit is reached, the active vertex with the largest singleton value is deleted before the next insertion. After all vertices have arrived, we continue applying the same deletion rule until the active set becomes empty. Updates change only which vertices are available for selection; the graph defining f remains fixed. For all algorithms, the error parameter ϵ, when applicable, is set to the standard value 0.1. All experiments were conducted on a Windows workstation with an AMD Ryzen 9 6900HX processor at 3.30 GHz. [2]

Experimental results. Figure 1 shows that CONSISTENTSAMPLE consistently achieves the best performance on the two metrics most central to this work: solution stability and update efficiency. More specifically, compared with SAMPLESTREAMING, CONSISTENTSAMPLE attains slightly higher objective values while using only 25.89% of its oracle queries and incurring only 19.36% of its cumulative solution changes. Although DYNAMICNM achieves higher objective values, this improvement comes at substantially greater computational and replacement costs: CONSISTENTSAMPLE uses only 0.015% of its oracle queries and 3.57% of its cumulative solution changes. These results demonstrate that CONSISTENTSAMPLE substantially reduces both computational and replacement costs while preserving competitive solution quality.

5 CONCLUSION

We developed CONSISTENTSAMPLE, a consistent algorithm for fully dynamic non-monotone submodular maximization with a constant-factor approximation, logarithmic-in-k consistency, and active-set-independent update complexity. Its monotone specialization further improves the approximation and consistency guarantees. Max-Cut experiments confirm its practical efficiency and stability. A natural direction for future work is to extend the framework to more general feasibility constraints beyond cardinality.

[2] Source code is available at the anonymized repository: https://anonymous.4open.science/r/A4A7-41451

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

We provide detailed proofs of all theoretical results in the appendix. The experimental setup, datasets, baselines, evaluation metrics, and parameter settings are described in Section 4. Source code for reproducing the experiments is available in the anonymized repository linked in the paper.

REFERENCES

Dmitrii Avdiukhin, Slobodan Mitrović, Grigory Yaroslavtsev, and Samson Zhou. Adversarially robust submodular maximization under knapsack constraints. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pp. 148–156, 2019.

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)*, 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, pp. 17369–17382, 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, pp. 103930–103962, 2026a.

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)*, 2026b.

Wei-Xuan Bao, Jun-Yi Hang, and Min-Ling Zhang. Submodular feature selection for partial label learning. In *ACM Knowledge Discovery and Data Mining (SIGKDD)*, pp. 26–34, 2022.

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 *Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 6549–6560, 2024.

Niv Buchbinder, Moran Feldman, Joseph Naor, and Roy Schwartz. Submodular maximization with cardinality constraints. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 1433–1452, 2014.

Shuang Cui, Kai Han, and He Huang. Deletion-robust submodular maximization with knapsack constraints. In *AAAI Conference on Artificial Intelligence (AAAI)*, volume 38, pp. 11695–11703, 2024.

Paul Dütting, Federico Fusco, Silvio Lattanzi, Ashkan Norouzi-Fard, and Morteza Zadimoghaddam. Deletion robust submodular maximization over matroids. In *International Conference on Machine Learning (ICML)*, pp. 5671–5693, 2022.

Paul Dütting, Federico Fusco, Silvio Lattanzi, Ashkan Norouzi-Fard, and Morteza Zadimoghaddam. Consistent submodular maximization. In *International Conference on Machine Learning (ICML)*, volume 235, pp. 11979–11991, 2024.

Paul Dütting, Federico Fusco, Silvio Lattanzi, Ashkan Norouzi-Fard, Ola Svensson, and Morteza Zadimoghaddam. The cost of consistency: Submodular maximization with constant recourse. In *ACM Symposium on Theory of Computing (STOC)*, pp. 1406–1417, 2025.

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.

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)*, volume 2025, pp. 6655–6679, 2025.

Moran Feldman, Amin Karbasi, and Ehsan Kazemi. Do less, get more: Streaming submodular maximization with subsampling. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 732–742, 2018.

Moran Feldman, Christopher Harshaw, and Amin Karbasi. How do you want your greedy: Simultaneous or repeated? *Journal of Machine Learning Research (JMLR)*, 24(72):1–87, 2023.

Ran Haba, Ehsan Kazemi, Moran Feldman, and Amin Karbasi. Streaming submodular maximization under a k-set system constraint. In *International Conference on Machine Learning (ICML)*, volume 119, pp. 3939–3949, 2020.

Tianyuan Jin, Yu Yang, Renchi Yang, Jieming Shi, Keke Huang, and Xiaokui Xiao. Unconstrained submodular maximization with modular costs: Tight approximation and application to profit maximization. In *International Conference on Very Large Data Bases (VLDB)*, volume 14, pp. 1756–1768, 2021.

Lilly Kumari, Shengjie Wang, Arnav Das, Tianyi Zhou, and Jeff Bilmes. An end-to-end submodular framework for data-efficient in-context learning. In *North American Chapter of the Association for Computational Linguistics (NAACL)*, pp. 3293–3308, 2024.

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, pp. 12923–12933, 2020.

Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection, 2014. URL `https://snap.stanford.edu/data`.

Morteza Monemizadeh. Dynamic submodular maximization. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pp. 9806–9817, 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)*, volume 2025, pp. 82476–82496, 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 (SIGKDD)*, pp. 1256–1266, 2021.

Binghui Peng. Dynamic influence maximization. In *Conference on Neural Information Processing Systems (NeurIPS)*, volume 34, pp. 10718–10731, 2021.

HSVNS Kowndinya Renduchintala, Krishnateja Killamsetty, Sumit Bhatia, Milan Aggarwal, Ganesh Ramakrishnan, Rishabh Iyer, and Balaji Krishnamurthy. Ingenious: Using informative data subsets for efficient pre-training of language models. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 6690–6705, 2023.

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

Guangyi Zhang, Nikolaj Tatti, and Aristides Gionis. Coresets remembered and items forgotten: submodular maximization with deletions. In *IEEE International Conference on Data Mining (ICDM)*, pp. 676–685, 2022.

APPENDIX

A SUPPORTING ALGORITHMS AND ANALYSIS

We analyze BUCKETSAMPLE and MAINTAINBAND, specify the checkpoint and residual routines, and then prove the properties of SAMPLEDEXCHANGE used in the non-monotone analysis. The monotone algorithm is given separately in Appendix C.

State conventions. A saved solution includes its element weights at the time it is saved. Assignments and returned states preserve these weights; reconstructing a suffix restores both the saved set and its weights. Weights are local to an execution, not shared between different bands or different checkpoint continuations. A new band starts with an empty history and $S_{\mathrm{end}} = C_{\mathrm{end}} = \emptyset$. Records are ordered by creation time, and all operations on records refer to those currently retained. Saved sets are not modified by subsequent changes to the working sets from which they were obtained. Objective-value comparisons favor the surviving previous solution on ties. No particular representation of these states is required, and the complexity bounds count value-oracle queries rather than the time or space used to represent the histories.

A.1 THE BUCKETSAMPLE SUBROUTINE

Fix $k \geq 1$, $\eta, \vartheta \in (0,1)$, $c = \sqrt{2}$, and $q = (2+\sqrt{2})^{-1}$. For a dyadic scale $\gamma = 2^r$, $r \in \mathbb{Z}$, let $\ell_\gamma = \eta\gamma/k$ and $\mathrm{Band}_\gamma(X) = \{u \in X : \ell_\gamma \leq f(\{u\}) \leq \gamma\}$. Zero-singleton elements can be ignored because $f(u \mid S) \leq f(\{u\}) = 0$. Set $R = \gamma/\ell_\gamma = k/\eta$, $B = 1 + \lceil \log_2 R \rceil$, and $M = \lceil 32k/\vartheta \rceil$.

In Algorithm 2, the stored weight of an accepted element is its marginal immediately before acceptance. For a feasible state S, choose $r(S) \in \arg\min_{v \in S} w(v)$ when $|S| = k$; otherwise let $r(S) = \perp \notin V$, with $w(\perp) = 0$. Thus removing $r(S)$ removes no actual element when a free position is available.

For a candidate set U from this band, the eligible set is $P(S,U) = \{u \in U : f(u \mid S) \geq \ell_\gamma,\ f(u \mid S) \geq (1+c)w(r(S))\}$. Its buckets are $G_a(S,U) = \{u \in P(S,U) : 2^a\ell_\gamma \leq f(u \mid S) < 2^{a+1}\ell_\gamma\}$, for $a = 0, \ldots, B-1$. These buckets cover every eligible element because $f(u \mid S) \leq f(\{u\}) \leq \gamma$.

The sampling order may depend on previous trials, but the next candidate is chosen before its independent acceptance trial is drawn. Failed trials do not change S, so their marginal values can be reused. We first bound the number of accepted elements, then the residual size and the expected loss caused by subsequent deletions.

Lemma A.1. *Let $\kappa = k(1 + \lceil \log_{1+c} R \rceil)$. Every execution of Algorithm 2 from the empty state accepts at most κ elements. The same upper bound applies to the number of further acceptances in a continuation of such an execution.*

Proof. The first element placed in any of the at most k solution positions has weight at least ℓ_γ. Each subsequent replacement in that position increases its weight by a factor of at least $1+c$. Every accepted weight is at most γ by submodularity and the band restriction. Consequently, at most $1 + \lceil \log_{1+c}(\gamma/\ell_\gamma) \rceil$ elements can be accepted in each position. Summing over the positions proves the claim. □

Lemma A.2. *Given a valid inherited execution, Algorithm 2 obeys the eligibility and replacement rules of Algorithm 6, with the additional marginal threshold ℓ_γ. Its returned residual C satisfies $|C| < BM$. For an execution starting from the empty state with fresh independent acceptance trials, Lemma A.17 applies after completing the residual.*

Proof. Bucket selection and candidate selection determine only the processing order; neither inspects the candidate's acceptance trial. A candidate removed by filtering is a deterministic rejection, and every sampled candidate is processed according to the same Bernoulli and replacement rules. If the procedure returns inside the loop, each of its B buckets contains fewer than M elements, so $|C| < BM$. If it exhausts the candidates, the residual is empty. This proves both assertions. □

Lemma A.3. *Let a candidate p be uniform on a set C fixed before the draw, with $g(u) \in [\lambda, 2\lambda)$ for all $u \in C$, $\lambda > 0$, and $|C| \geq M/2$. For a fixed deletion set D, independent of the draw, with $|D| \leq 2k$, $\mathbb{E}[g(p)\mathbf{1}\{p \in D\}] \leq \frac{\vartheta}{4}\mathbb{E}[g(p)]$. The inequality also holds when both random variables are multiplied by the same independent Bernoulli acceptance indicator.*

Proof. Condition on the set and its pre-draw history. Uniformity gives $\mathbb{E}[g(p)\mathbf{1}\{p \in D\}] = |C|^{-1}\sum_{u\in C\cap D} g(u) \leq 2\lambda|C \cap D|/|C|$, while $\mathbb{E}[g(p)] \geq \lambda$. Their ratio is therefore at most $2|D|/|C| \leq 8k/M \leq \vartheta/4$. An independent acceptance indicator multiplies both expectations by q. □

A.2 The MaintainBand Subroutine

The aim is to reuse decisions that remain valid after an update. We store every randomized trial, including rejections, together with its sampling source and pre-draw state. Reconstruction begins at the earliest invalid record and leaves all earlier records unchanged; its input state is copied before the invalid records are discarded. The analysis separates pointwise validity of the retained execution from the probability identities needed for non-monotone approximation.

A *sampling record* is created immediately before the acceptance coin of a sampled candidate is revealed. These record indices are distinct from the deletion-horizon levels. For record j, let I_j be the pre-draw solution, let R_j^0 be the eligible candidates immediately before its draw, and let $F_j^0 \subseteq R_j^0$ be the maximum-cardinality bucket from which p_j is sampled. Write $n_j = |F_j^0|$, and store the independent acceptance bit B_j after it is drawn. At construction, $n_j \geq M$ and $|R_j^0| \leq Bn_j$. Here R_j^0 denotes the value of the suffix at record creation; it need not be stored separately from the state required by the algorithm. Both $B_j = 0$ and $B_j = 1$ create records. Elements discarded by a deterministic eligibility test have no coin and need no sampling record. Rejections between two successful exchanges leave the solution unchanged and therefore share the same cached marginal values.

If record j was created at time t_j, then $F_j = F_j^0 \cap X_t$ and $R_j = (R_j^0 \cap X_t) \cup (\mathrm{Band}_\gamma(X_t) \setminus X_{t_j})$. In particular, R_j includes its own sampled candidate while active and all active band elements inserted after record creation. These containers are not reduced when later candidates are processed: they must retain the elements needed to reconstruct a suffix. For a retained record, the maintained bounds are $|R_j| \leq 2Bn_j$ and $|F_j| \geq n_j/2$, and its sampled candidate must remain active. The terminal residual has size below $2BM$. A source-shrink or suffix-growth violation, or deletion of p_j, invalidates record j regardless of its bit B_j. The algorithm reconstructs from the earliest such record, restoring I_j and its stored weights and processing the active container R_j with fresh draws and coins. If only the terminal residual reaches $2BM$, the algorithm continues from its current terminal state. No later reconstruction or terminal continuation erases an earlier record. A reconstruction leaves a residual below BM.

The set operations in Algorithm 3 are bookkeeping, not oracle queries. Our bounds concern oracle queries; they do not assert a deterministic $\mathcal{O}(\kappa)$ bound on the number of failed-trial records or on the non-oracle work of an update.

Lemma A.4. *After every update, the retained processing order and terminal residual obey the threshold-truncated replacement rules of Algorithm 6. Every active band element occurs exactly once, either in that processing order or in the residual. Each accepted element's stored weight is its marginal at its position in this execution.*

Proof. A fresh call has these properties by Lemma A.2. An insertion is appended after the retained processing order. Removing a residual element or a deterministically rejected element changes no earlier state. Removing any recorded candidate, even a rejected one, reconstructs the suffix from its saved pre-draw state. A size trigger also replaces only a suffix. Deterministic rejections before that suffix remain valid because their prefix states have not changed, while the saved container includes all active candidates that must be reconsidered. A fresh continuation processes these candidates once, with the original retained prefix. Induction gives the statement. This is a pointwise assertion; the next lemma supplies the separate probability argument. □

Initial uniformity does not, by itself, justify uniformity of a record observed later. We condition on a specified record's creation history rather than the full retained execution, which would reveal the candidate or coin we need to analyze.

Lemma A.5. *Fix the update sequence and an observation time t. Identify a record j by its creation update and its draw index within that update, and let $\mathcal{H}_j$ be the history immediately before its candidate is drawn. Conditional on $\mathcal{H}_j$ and on this record still being retained at t, whenever this event has positive probability, $p_j \sim \mathrm{Unif}(F_j^0 \cap X_t), \quad B_j \sim \mathrm{Bernoulli}(q)$, independently of one another. If $L_j(t)$ indicates retention and $a_j(p_j)$ is any integrable quantity determined by $\mathcal{H}_j$ and p_j, then*

$$\mathbb{E}[L_j(t)B_j a_j(p_j)] = q\,\mathbb{E}[L_j(t)a_j(p_j)]. \tag{1}$$

The corresponding identity for a rejection has multiplier $1-q$.

Proof. Condition on the creation event and $\mathcal{H}_j$. All metadata of earlier records, the present prefix, R_j^0, and F_j^0 are now fixed. The first time an earlier record is invalidated depends only on its already fixed candidate and reference sets and the fixed update sequence. The present record's source-shrink and suffix-growth tests also depend only on its reference sets and that sequence. None of these events depends on p_j or B_j. Later-record reconstructions and terminal continuations cannot remove j.

Consequently, for a $\{0,1\}$-valued quantity $H_j(t)$ determined by the conditioned history and fixed updates, $L_j(t) = H_j(t)\,\mathbf{1}\{p_j \in X_t\}$. The second factor accounts for deletion of its own candidate. This factorization uses identical reconstruction tests for accepted and rejected records. Conditional on retention, the initially uniform draw is uniform on $F_j^0 \cap X_t$, and its independent acceptance coin is unaffected. The moment identities follow by conditioning first on $\mathcal{H}_j, p_j$ and then averaging. For a finite update prefix there are only finitely many creation events, since each call processes each of its finitely many candidates at most once. Thus these identities can be summed over creation events; no optional-stopping or full-history independence assumption is used. □

The successful records are bounded by geometric weight growth. The preceding coin identity converts that deterministic bound into an expected bound on all retained records.

Lemma A.6. *Let N_t be the number of retained trial records after update t and N_t^+ the number of accepted ones. Then $N_t^+ \le \kappa$ for every realization, $\quad \mathbb{E}[N_t^+] = q\,\mathbb{E}[N_t], \quad \mathbb{E}[N_t] \le \kappa/q$.*

Proof. The first inequality follows from Lemmas A.4 and A.1. Write $N_t = \sum_j L_j(t)$ and $N_t^+ = \sum_j L_j(t)B_j$, summing over creation events, and apply equation 1 with $a_j = 1$. □

The same conditional distribution controls the weight lost after a checkpoint. The relevant marginals are measured against the record's frozen pre-draw prefix, not against the current terminal solution.

For a set Y of accepted elements in the execution under consideration, write $W(Y) := \sum_{u \in Y} w(u)$, using their recorded marginals in that execution.

Lemma A.7. *Fix a checkpoint time τ and a singleton band selected independently of its sampling randomness. Let D be a fixed set of at most $2k$ subsequent deletions. If A_γ is the accepted history retained at τ, then $\mathbb{E}[W(A_\gamma \cap D)] \le \frac{\vartheta}{4}\mathbb{E}[W(A_\gamma)]$.*

Proof. For each creation event, condition on its pre-draw history and retention at τ as in Lemma A.5. Its surviving source has size at least $n_j/2 \ge M/2$. The gains against its frozen prefix remain in the original factor-two bucket. Lemma A.3 therefore bounds the fraction of its expected weight in D by $\vartheta/4$; the independent coin multiplies both terms by q. Sum over retained-record contributions and average over their creation histories. In the public algorithm, fixing the independent schedule fixes τ and the singleton quantile, so its band selection satisfies the stated independence condition. □

A.2.1 Query Complexity for One Band

A reconstruction scans marginals only when an acceptance changes the solution, apart from one final scan. Thus its cost depends on the bound on successful exchanges, not on a deterministic bound on all trial records.

Lemma A.8. *Rebuilding a suffix from r active candidates uses $\mathcal{O}(r(\kappa+1))$ oracle queries.*

Proof. Each scan evaluates at most r marginals and forms the buckets. Failed trials remove already evaluated candidates without changing the solution. Lemma A.1 bounds the successful exchanges, and hence the scans, by $\kappa+1$ up to a constant factor. □

The size-triggered costs are charged to preceding updates. Candidate-deletion costs instead use the conditional uniformity of each retained creation record, including rejected records.

Lemma A.9. *For a fixed singleton scale γ, Algorithm 3 uses $\mathcal{O}(B\kappa^2/q+\kappa) = \mathcal{O}(B\kappa^2+\kappa)$ expected amortized oracle queries per stream update, where $q = (2+\sqrt{2})^{-1}$.*

Proof. At creation, record j has at most Bn_j suffix candidates. Its suffix cannot grow beyond $2Bn_j$ without at least Bn_j intervening insertions. At the trigger the reconstruction has $\mathcal{O}(Bn_j)$ candidates and costs $\mathcal{O}(Bn_j\kappa)$. Charge $\mathcal{O}(\kappa)$ credits per such insertion to this record. Likewise, reducing its surviving source below $n_j/2$ requires more than $n_j/2$ source deletions; charging $\mathcal{O}(B\kappa)$ credits per deletion pays for the corresponding reconstruction. Credits belong to a specified creation record. If an earlier reconstruction erases it, its credits may be discarded and are never spent twice.

For a fixed deletion update, condition on each possible record's creation history and retention immediately before that update. Its source has size at least $n_j/2$, so Lemma A.5 bounds the probability that the deleted identity equals its sampled candidate by $2/n_j$, regardless of its acceptance bit. The associated suffix costs at most $\mathcal{O}(Bn_j\kappa)$ to rebuild. Multiplying these bounds and summing retention probabilities over creation events gives expected work at most $\mathcal{O}(B\kappa)\,\mathbb{E}[N_{t-1}] \le \mathcal{O}(B\kappa^2/q)$. We do not condition on the complete current state, which would reveal the sampled identities. If multiple records are invalidated, rebuilding only the earliest suffix costs no more than charging the chosen trigger, or the sum of all candidate triggers as an upper bound.

An update deposits size-trigger credits into at most N_{t-1} records. Lemma A.6 bounds their expected total by the same $\mathcal{O}(B\kappa^2/q)$ per update. A terminal continuation requires the residual to grow from below BM to $2BM$, costs $\mathcal{O}(BM\kappa)$, and is paid for by BM insertions, adding $\mathcal{O}(\kappa)$ amortized queries. Summing over any finite update sequence starting from empty structures proves the claim. □

A.2.2 Query Complexity over All Bands

An inserted element of singleton value $s>0$ belongs to scale γ exactly when $s \le \gamma \le \frac{k}{\eta}s$. Thus it participates in at most $G := 2 + \left\lceil \log_2 \frac{k}{\eta} \right\rceil$ dyadic singleton-band processes. Thus a single insertion or deletion affects only logarithmically many scales.

The remaining step is to sum the one-band cost over these affected scales. The admissible singleton scales span a multiplicative range of k/η, so the total number of membership-changing band updates is only a logarithmic factor larger than the stream length.

Lemma A.10. *Maintaining all singleton-value band instances uses $\mathcal{O}(k^2\log^4(k/\eta))$ expected amortized oracle queries per update.*

Proof. Apply Lemma A.9 separately to the subsequence of membership-changing updates associated with each dyadic scale. If scale γ receives T_γ such updates, then $\sum_\gamma T_\gamma \le GT$; before its first membership-changing update, its maintained state is empty and requires no scan of the active set. Summing the per-instance total bounds and dividing by T gives $\mathcal{O}(GB\kappa^2+G\kappa)$, since q is constant. Use $B, G = \mathcal{O}(\log(k/\eta))$ and $\kappa = \mathcal{O}(k\log(k/\eta))$. □

A.3 The SelectCheckpoint Subroutine

At time t, let $P_t = \{u \in X_t : f(\{u\}) > 0\}$. Order these elements as $v_1, \dots, v_{|P_t|}$ by nonincreasing singleton value, with a fixed rule for ties. This order can be maintained using singleton values cached on insertion; rank selection and retrieval of the largest values need no further value-oracle queries. The ordering is not recomputed by making new singleton queries at each call.

Algorithm 4 SELECTCHECKPOINT(t, h)

if $|X_t| \le h$ **then**
  return $(\emptyset, X_t)$
else if $|P_t| \le h$ **then**
  return $(\emptyset, P_t)$
$\Delta \leftarrow f(\{v_{h+1}\})$; $\gamma \leftarrow 2^{\lceil \log_2 \Delta \rceil}$
$H_t \leftarrow \{v_j : 1 \le j \le h,\ f(\{v_j\}) > \gamma\}$
return $(I_t^\gamma, C_t^\gamma \cup H_t)$ ▷ copy the maintained band state

Algorithm 5 RESIDUALSAMPLE(S_0, U_0, d)

$S \leftarrow S_0$; $U \leftarrow U_0$; $m_d \leftarrow \lceil 8d/\vartheta \rceil$; $c \leftarrow \sqrt{2}$; $q \leftarrow (2+\sqrt{2})^{-1}$
while $U \neq \emptyset$ **do**
  $g(u) \leftarrow f(u \mid S)$ for all $u \in U$
  $U \leftarrow \{u \in U : g(u) > 0,\ g(u) \ge (1+c)w(r(S))\}$
  if $|U| < m_d$ **then**
    return (S, U)
  sample $u \in U$ with probability $g(u)^{-1} / \sum_{x \in U} g(x)^{-1}$
  $U \leftarrow U \setminus \{u\}$; $B_u \sim \text{Bernoulli}(q)$ independently
  if $B_u = 1$ **then**
    $S \leftarrow (S \setminus \{r(S)\}) \cup \{u\}$; $w(u) \leftarrow g(u)$
return (S, U)

Algorithm 4 returns a state from a suitable band and adds the at most h elements with larger singleton values to its residual. The scheduled calls use $h = 2d_0$, and the calls for $k < 2b_1$ use $h = 2k$. Thus $h = \mathcal{O}(k)$ in every use of this routine.

When a positive quantile is selected at time τ, write Δ_τ and γ_τ for these values. The returned residual has size at most $2BM + h = \mathcal{O}(Bk/\vartheta)$. The elements excluded from the band and the explicit set H_τ are exactly $L_\tau = \{u \in X_\tau : f(\{u\}) < \ell_{\gamma_\tau}\}$; elements that were processed and rejected inside the band remain represented by its execution history.

Lemma A.11. *Suppose* SELECTCHECKPOINT(τ, h) *selects a positive quantile* Δ_τ *and scale* γ_τ*. If at most* h *deletions occur between* τ *and* t*, then* $f(O_t \setminus L_\tau) \ge (1-2\eta)\mathrm{OPT}_t$ *and* $k\ell_{\gamma_\tau} \le 2\eta\mathrm{OPT}_t$*.*

Proof. At least one of the $h+1$ largest positive-singleton elements at time τ survives until t. Its singleton is feasible and has value at least Δ_τ, so $\mathrm{OPT}_t \ge \Delta_\tau$ and $\gamma_\tau < 2\Delta_\tau \le 2\mathrm{OPT}_t$. Writing $L = O_t \cap L_\tau$, submodularity gives $f(O_t) - f(O_t \setminus L) \le \sum_{u \in L} f(\{u\}) \le k\ell_{\gamma_\tau} = \eta\gamma_\tau \le 2\eta\mathrm{OPT}_t$. This proves both conclusions, including when $L = \emptyset$. □

If all active elements are returned, set $L_\tau = \emptyset$ and $\ell_{\gamma_\tau} = 0$ for the analysis. If only the positive-singleton elements are returned, set $L_\tau = X_\tau \setminus P_\tau$ and again use threshold zero. Removing these zero-singleton elements cannot decrease the objective, so the same conclusions hold with zero loss.

A.4 THE RESIDUALSAMPLE SUBROUTINE

RESIDUALSAMPLE continues an inherited weighted solution S_0 on the candidate set U_0. It uses inverse-marginal sampling from the deletion-robust literature (Zhang et al., 2022; Cui et al., 2024), followed by the independent acceptance trial of SAMPLEDEXCHANGE. The procedure is used only after dynamic compression has reduced its input size to a function of k and $1/\epsilon$. The inherited solution, including any elements that have become inactive, is not projected onto the active set before continuation; the projection is applied only when producing an output or a public target.

Lemma A.12. *Condition on the state before a draw from an eligible set* P*, with* $g(u) = f(u \mid S) > 0$ *for all* $u \in P$*. Let* D *be a fixed deletion set, independent of the draw, with* $|D| \le 2d$*. If* $|P| \ge 8d/\vartheta$*, then* $\mathbb{E}[g(p)\mathbf{1}\{p \in D\}] \le \frac{\vartheta}{4}\mathbb{E}[g(p)]$*. This remains true after multiplication by an independent acceptance indicator. The routine leaves fewer than* $8d/\vartheta$ *candidates and uses* $\mathcal{O}(m^2)$ *oracle queries on* m *input candidates.*

Proof. Let $Z = \sum_{x \in P} 1/g(x)$. The sampling rule gives $\mathbb{E}[g(p)\mathbf{1}\{p \in D\}] = |P \cap D|/Z$ and $\mathbb{E}[g(p)] = |P|/Z$. Their ratio is at most $2d/(8d/\vartheta) = \vartheta/4$. An independent acceptance indicator multiplies both expectations by q. At termination, $|U| < \lceil 8d/\vartheta \rceil$, and integrality implies $|U| < 8d/\vartheta$. At most m candidates are sampled, and each iteration evaluates at most m marginals, giving the query bound. □

A.5 TARGET COMPLETION

At checkpoint time τ, set $I_\tau^a = I_\tau \cap X_\tau$ and $C_\tau^a = (C_\tau \cap X_\tau) \setminus I_\tau^a$. Algorithm 1 scans $C_\tau^a \cap S_{\tau-1}$ first, then its remaining elements, with a fixed order inside each part. Its target T_τ is constructed once and frozen.

Only a few properties of the completion step enter the analysis. It must preserve feasibility and the internal state, and its distance from the active core must be bounded by the residual size. The following lemma records these properties; no optimality guarantee for the residual scan is needed.

Lemma A.13. *For every handover start τ, $I_\tau^a \subseteq T_\tau \subseteq I_\tau^a \cup C_\tau^a$, $\quad |T_\tau| \le k$, $\quad f(T_\tau) \ge f(I_\tau^a)$, and $|T_\tau \triangle I_\tau^a| \le |C_\tau^a|$. The target can be constructed with $\mathcal{O}(|C_\tau^a|)$ oracle queries and without revealing or consuming any independent Bernoulli trial used by the internal algorithm.*

Proof. The routine only adds elements from C_τ^a until capacity k is reached, and every accepted addition has positive current marginal. Feasibility, the containments, value domination, and the displacement bound follow immediately. Each residual element is evaluated at most once; after an element is rejected, subsequent additions can only decrease its marginal by submodularity. □

A.6 THE SAMPLEDEXCHANGE SUBROUTINE

SAMPLEDEXCHANGE is the non-monotone completion routine used after the preceding stages have reduced the unresolved candidate set. Its randomized replacement rule is a stored-weight variant of the sampled-exchange technique of Feldman et al. (2018) under a cardinality constraint. We include the exact routine and the guarantees needed by our dynamic analysis for completeness. In particular, the processing order may be adaptive, while the next element must be chosen before its own Bernoulli trial is exposed. We also record the small additive loss incurred when the first processing segment applies the minimum marginal threshold used by BUCKETSAMPLE.

For the analysis, let A denote all elements that are accepted during the execution, including those subsequently replaced, and let R denote all eligible elements that are rejected by the Bernoulli trials. These sets are used only for the analysis and need not be maintained by the algorithm. When several calls are concatenated, A and R refer to the corresponding historical sets accumulated across all segments. Algorithm 6 uses $c = \sqrt{2}$ and $q = 1/(2+\sqrt{2})$; in the accounting lemmas below we keep c and q symbolic until the final substitution. For a current solution S, let $r(S) = \perp$ if $|S| < k$, with $w(\perp) = 0$, and otherwise let $r(S)$ be a minimum-weight element of S.

Algorithm 6 SAMPLEDEXCHANGE(I, U)

$S \leftarrow I$; $c \leftarrow \sqrt{2}$; $q \leftarrow (2+\sqrt{2})^{-1}$
while $U \neq \emptyset$ **do**
  choose $u \in U$; $U \leftarrow U \setminus \{u\}$ ▷ before drawing its acceptance trial
  $g_u \leftarrow f(u \mid S)$; $r_u \leftarrow r(S)$
  if $g_u > 0$ and $g_u \ge (1+c)w(r_u)$ **then**
    $B_u \sim \text{Bernoulli}(q)$ independently
    if $B_u = 1$ **then**
      $S \leftarrow (S \setminus \{r_u\}) \cup \{u\}$; $w(u) \leftarrow g_u$
return S

We begin by relating stored marginal gains to function values. This allows us to analyze replacements through their stored weights, without assuming monotonicity of the objective. The final inequality follows because weights grow geometrically along each sequence of replacements.

Lemma A.14. *For every execution from the empty set,*

$$f(S) \geq W(S), \qquad f(A) \leq W(A), \qquad W(A) \leq \frac{1+c}{c} W(S).$$

Proof. For the first inequality, order the final elements by acceptance time. The earlier final survivors form a subset of the solution present when the next survivor was accepted, so submodularity makes its telescoping marginal at least its stored weight. For the second, order all accepted elements by acceptance time; the set of previously accepted elements contains the then-current solution, so each telescoping marginal is at most the stored weight. For the third, view each of the at most k solution positions as a replacement chain. Every replacement multiplies the victim's weight by at least $1+c$. A chain ending at weight z has total historical weight at most $z \sum_{j \geq 0} (1+c)^{-j} = ((1+c)/c)z$. Sum over the terminal positions. □

We next bound the contribution of optimal elements rejected by the replacement test. Each such element can be associated with a distinct position in the maintained solution whose weight is sufficiently large. Positions assigned to accepted optimal elements must also be accounted for; subtracting their weight yields the following bound.

Lemma A.15. *Fix $O \subseteq V$, $|O| \leq k$, and let $Q^+ = \{u \in O \setminus (A \cup R) : g_u > 0\}$. For every execution,*

$$\sum_{u \in Q^+} g_u \leq (1+c)\big(W(S) - W(O \cap A)\big).$$

Proof. At each point of the execution, maintain an injection ϕ from $(O \cap A) \cup Q^+_{\text{processed}}$ into the current solution. Require $w(\phi(u)) \geq w(u)$ for an accepted element $u \in O \cap A$, and $w(\phi(u)) > g_u/(1+c)$ for $u \in Q^+_{\text{processed}}$. The domain contains at most $|O| \leq k$ elements.

When a nonoptimal element replaces an assigned element, transfer its assignment to the new element, whose weight is at least $1+c$ times the old weight. When an optimal element is accepted into a free position, assign it to that position. When it replaces an element, there are at most $k-1$ previous assignments. If the removed element was assigned, transfer that assignment to an unassigned survivor, whose weight is at least the removed minimum weight; then assign the newly accepted optimal element to itself. If $u \in O$ is rejected with $g_u > 0$ by the exchange test, the solution is full and $g_u < (1+c)w(r(S))$. An unassigned element exists and has weight at least $w(r(S))$, so it can be assigned to u. Bernoulli rejections add no assignment. Thus the required injection is maintained throughout.

At termination, summing over distinct assigned elements gives $W(O \cap A) + (1+c)^{-1} \sum_{u \in Q^+} g_u \leq W(S)$, which proves the claim. □

We use the standard submodular sampling lemma: if $h \geq 0$ is submodular and $\Pr[u \in Z] \leq q$ for every element, without any independence assumption, then $\mathbb{E}[h(Z)] \geq (1-q)h(\emptyset)$ Buchbinder et al. (2014).

The preceding two lemmas handle deterministic rejections. It remains to account for eligible elements rejected by randomization. Their expected contribution is proportional to the expected weight of accepted optimal elements. Choosing the acceptance probability so that these terms cancel gives the approximation guarantee below.

Lemma A.16. *For an execution from the empty set, if $(1-q)/q = 1+c$, then for every fixed comparator O contained in its input, $|O| \leq k$,*

$$\mathbb{E}[W(S)] \geq \frac{c(1-q)}{(1+c)^2} f(O).$$

With $c = \sqrt{2}$ and $q = 1/(2+\sqrt{2})$, the coefficient is $\rho_\star = \frac{1}{3+2\sqrt{2}}$.

Proof. Let $A_{<u}$ denote accepted elements before u. For $u \notin A$, $S_u^- \subseteq A_{<u} \subseteq A$, hence $f(u \mid A) \le g_u$. By Lemmas A.14 and A.15,

$$\begin{aligned} f(A \cup O) &\le W(A) + \sum_{u \in Q^+} g_u + \sum_{u \in O \cap R} g_u \\ &\le \frac{(1+c)^2}{c} W(S) - (1+c) W(O \cap A) + \sum_{u \in O \cap R} g_u. \end{aligned}$$

For an optimal eligible element, eligibility and g_u are fixed before its coin. Thus $\mathbb{E}\big[\sum_{u \in O \cap R} g_u\big] = \frac{1-q}{q}\mathbb{E}[W(O \cap A)] = (1+c)\mathbb{E}[W(O \cap A)]$, so the last two terms cancel in expectation. Moreover every element enters A with marginal probability at most q. Applying the sampling lemma to $h(Z) = f(O \cup Z)$ gives $\mathbb{E}[f(O \cup A)] \ge (1-q) f(O)$. Combining the inequalities yields the result. Substituting $c = \sqrt{2}$ gives $\rho_\star$. □

The coarse processing stage additionally rejects very small marginals through its minimum marginal threshold. Since an optimal solution contains at most k elements, this introduces only an additive $k\ell$ term before rescaling. The following lemma records the corresponding extension used in the main analysis.

Lemma A.17. *Consider a segmented execution from the empty set that is identical to Algorithm 6, except that in the first segment an element is also declared ineligible whenever its current marginal is below a minimum marginal threshold $\ell \ge 0$. Later segments impose no additional minimum marginal threshold. Let $K := \frac{(1+c)^2}{c}$. Then for every fixed comparator O contained in the input of the execution, $|O| \le k$,*

$$\mathbb{E}[W(S)] \ge \rho_\star f(O) - \frac{k\ell}{K}$$

for $c = \sqrt{2}$ and $q = 1/(2+\sqrt{2})$, where $\rho_\star = \frac{1}{3+2\sqrt{2}}$.

Proof. Repeat the proof of Lemma A.16. The only additional optimal elements not covered by the injection in Lemma A.15 are first-segment elements discarded solely because $0 < g_u < \ell$. Their total contribution to the upper bound on the optimum is at most $k\ell$. Therefore $\mathbb{E}[f(A \cup O)] \le K\mathbb{E}[W(S)] + k\ell$. The sampling lower bound $\mathbb{E}[f(A \cup O)] \ge (1-q) f(O)$ is unchanged. Since $\rho_\star = (1-q)/K$, rearranging proves the claim. □

B Proofs for Section 3

B.1 Approximation Ratio

Case $k \ge 2b_1$. We first analyze a fixed time outside all handover intervals. The calls along its checkpoint chain form one sampled-exchange execution; the additional public completion never changes this execution. We bound the weight lost to deletions and derive the approximation outside handover intervals.

If the relevant coarse checkpoint precedes the first update, its state is empty: take the omitted low set to be empty and the initial marginal threshold to be zero. All active elements are then processed as later arrivals, so there is no singleton-threshold loss. If the checkpoint returns only the positive-singleton elements, set $L_{\tau_0} = X_{\tau_0} \setminus P_{\tau_0}$ and use threshold zero, as in Appendix A.3; omitting these zero-singleton elements incurs no loss.

Write $K = (1+c)^2/c$ in this approximation analysis. Fix the entire schedule, independently of the internal sampling, and a time t outside its handover intervals. Let the relevant checkpoint chain be $\tau_0 < \tau_1 < \cdots < \tau_L < t$. Except for the low set L_{τ_0}, the refinement calls and the final call process every active element exactly once: the level-0 middle-band process handles the band, leaves its residual unprocessed, and all heavier elements are placed into that residual; higher levels pass only unprocessed active residuals and newly arrived elements. Elements arriving after τ_0, even with tiny singleton value, are therefore processed normally.

Concatenating the realized processing orders gives a pointwise legal replacement execution, whose first segment uses minimum marginal threshold $\ell_{\gamma_{\tau_0}}$. Lemma D.2 additionally verifies the inclusion-probability and acceptance/rejection identities required by the static expectation argument. For $i = 0, \dots, L$, let A_i be the elements accepted in the retained coarse segment or in refinement segment i, respectively, and let A_{L+1} be the elements accepted by the final completion. These sets are disjoint. All expectations in this subsection are conditional on the fixed schedule until the final averaging step.

We first control the weight of accepted elements that become inactive between their checkpoint and the current output time. The schedule bounds the number of intervening deletions, allowing the robustness bounds for the two sampling routines to be applied separately at each level.

Lemma B.1. *For every $i = 0, \dots, L$, $\mathbb{E}[W(A_i \setminus X_t)] \le \frac{\vartheta}{4}\mathbb{E}[W(A_i)]$.*

Proof. For $i = 0$, Lemma A.4 says that the retrieved band state is a legal threshold-truncated sampled-exchange prefix on the active band at checkpoint time τ_0, and Lemma A.5 supplies the conditional source distribution needed by Lemma A.7. At most $2d_0 \le 2k$ additional elements disappear between τ_0 and t, so Lemma A.7 applies directly. For $i > 0$, Algorithm 5 samples only while its pool has size at least $8d_i/\vartheta$, and at most $2d_i$ relevant deletions occur. Apply Lemma A.12 to every accepted pivot and sum by the tower property. □

The previous bound applies to all accepted elements, whereas the final solution retains only some of them. The stored-weight inequality relates these two quantities. We use it to bound the surviving final weight and then convert that weight back into objective value.

Lemma B.2. *Let $\overline{S}_t$ be the temporary final solution before removing inactive checkpoint elements. Then*

$$\mathbb{E}[W(\overline{S}_t \cap X_t)] \ge \left(1 - \frac{(1+c)\vartheta}{4c}\right)\mathbb{E}[W(\overline{S}_t)],$$

and for every realization $f(\overline{S}_t \cap X_t) \ge W(\overline{S}_t \cap X_t)$.

Proof. All elements of A_{L+1} are active at time t. With $A = \bigcup_{i=0}^{L+1} A_i$ and $\overline{S}_t \subseteq A$, Lemma B.1 gives $\mathbb{E}[W(\overline{S}_t \setminus X_t)] \le (\vartheta/4)\mathbb{E}[W(A)]$. Applying $W(A) \le ((1+c)/c)W(\overline{S}_t)$ from Lemma A.14 and subtracting from $\mathbb{E}[W(\overline{S}_t)]$ proves the first inequality. For the objective inequality, order surviving final elements by acceptance time. Earlier surviving final elements form a subset of the solution present at acceptance, so submodularity makes each surviving telescoping marginal at least its stored weight. □

We can now combine the static approximation bound, the singleton-threshold loss, and the surviving-weight bound. Let $\widehat{S}_t = \overline{S}_t \cap X_t$ denote the fresh set produced by the final residual completion before the algorithm compares it with $S_{t-1} \cap X_t$. The analysis below lower bounds $f(\widehat{S}_t)$; since the actual output satisfies $f(S_t) \ge f(\widehat{S}_t)$ by construction, the same guarantee immediately transfers to S_t.

Lemma B.3. *Suppose $k \ge 2b_1$. Fix a realization σ of the random time shift and a time $t \notin \mathcal{H}_\sigma$. Conditional on σ, we have $\mathbb{E}[f(S_t)] \ge \left(1 - \frac{(1+c)\vartheta}{4c}\right)\left(\frac{1-2\eta}{3+2\sqrt{2}} - \frac{2c\eta}{(1+c)^2}\right)\mathrm{OPT}_t$, where $c = \sqrt{2}$.*

Proof. Let $O'_t = O_t \setminus L_{\tau_0}$. By Lemma A.11, $f(O'_t) \ge (1-2\eta)\mathrm{OPT}_t$ and $k\ell_{\gamma_{\tau_0}} \le 2\eta\mathrm{OPT}_t$. Use the pointwise proof of Lemma A.17 and the probability identities of Lemma D.2 for the concatenated execution:

$$\mathbb{E}[W(\overline{S}_t)] \ge \rho_\star f(O'_t) - \frac{k\ell_{\gamma_{\tau_0}}}{K} \ge \left(\rho_\star(1-2\eta) - \frac{2\eta}{K}\right)\mathrm{OPT}_t, \text{ where } \rho_\star = \frac{1}{3+2\sqrt{2}}.$$

Now use Lemma B.2 to obtain the displayed lower bound for the fresh candidate $\widehat{S}_t$. Since $f(S_t) \ge f(\widehat{S}_t)$ because Algorithm 1 returns the better of the fresh candidate and the surviving previous solution, the same bound holds for the returned solution S_t. □

Case $k < 2b_1$. Suppose $k < 2\lceil 2/\delta \rceil$. At every update the algorithm calls SELECTCHECKPOINT$(t, 2k)$ and immediately completes its residual with SAMPLEDEXCHANGE, producing a fresh candidate $\widehat{S}_t$ before comparing it with $S_{t-1} \cap X_t$. Thus no handover event is involved, and because the checkpoint is constructed and used at the same update, there are no intervening deletions between checkpoint construction and the final completion; moreover $f(S_t) \geq f(\widehat{S}_t)$ pointwise. If the coarse checkpoint returns the whole active set, or all positive-singleton elements, the execution is simply a legal sampled-exchange run on all elements that can have positive marginal. Otherwise, let γ_t be its selected scale. The maintained band state is a legal threshold-truncated sampled-exchange prefix on the *current* active band by Lemma A.4; since the checkpoint is used immediately, every accepted prefix element is active. Appending the explicit heavy elements and then running SAMPLEDEXCHANGE on the returned residual therefore gives one legal segmented execution with first-segment floor ℓ_{γ_t}. At the current time $\gamma_t < 2\mathrm{OPT}_t$, so Lemma A.11 (with zero intervening deletions) and the argument of Lemma A.17, justified for retained trials by Lemma D.2, give $\mathbb{E}[f(\widehat{S}_t)] \geq \left(\rho_\star(1-2\eta) - \frac{2\eta}{K}\right)\mathrm{OPT}_t \geq (\rho_\star - \epsilon)\mathrm{OPT}_t$. Since $f(S_t) \geq f(\widehat{S}_t)$, the returned solution satisfies the same bound under the parameters of Algorithm 1. Thus the case $k < 2b_1$ satisfies the same approximation guarantee without using the random schedule.

Approximation part of Theorem 3.1.

Proof. The case $k < 2b_1$ was established above. For $k \geq 2b_1$ and every fixed t, the random schedule shift places t in a handover interval with probability at most δ. Since $f \geq 0$, handover states may be dropped from a lower bound. For $c = \sqrt{2}$ and $\rho_\star = \frac{1}{3+2\sqrt{2}}$, write

$$A := \frac{1+c}{4c}, \qquad C := 2\rho_\star + \frac{2}{K} = 2(\sqrt{2}-1).$$

Then Lemma B.3 and the elementary inequality $\prod_j (1-x_j) \geq 1 - \sum_j x_j$ give

$$\begin{aligned}\mathbb{E}[f(S_t)] &\geq (1-\delta)(1-A\vartheta)(\rho_\star - C\eta)\mathrm{OPT}_t \\ &\geq \big(\rho_\star - \rho_\star\delta - \rho_\star A\vartheta - C\eta\big)\mathrm{OPT}_t.\end{aligned}$$

Put $\widehat{\epsilon} := \min\{\epsilon, 1/8\}$ and set $\eta = \frac{\widehat{\epsilon}}{32}, \vartheta = 3\widehat{\epsilon}, \delta = 4\widehat{\epsilon}$. These choices satisfy $\delta \leq 1/2$ and $\vartheta \leq 3/8$, and $4\rho_\star + 3\rho_\star A + \frac{C}{32} = \frac{203-133\sqrt{2}}{16} < 0.932 < 1$. Consequently $\mathbb{E}[f(S_t)] \geq (\rho_\star - \widehat{\epsilon})\mathrm{OPT}_t \geq (\rho_\star - \epsilon)\mathrm{OPT}_t$. □

B.2 CONSISTENCY AND QUERY COMPLEXITY

The consistency and query analyses both depend on the number of candidates passed between consecutive checkpoints. The first refinement receives the larger residual returned by the dynamically maintained level-0 state. After this refinement, the stopping rule of RESIDUALSAMPLE reduces the residual size proportionally to the corresponding deletion horizon.

B.2.1 CONSISTENCY

Lemma B.4. *Let τ_i be the relevant level-i checkpoint time. The level-1 refinement receives $\mathcal{O}(Bk/\vartheta)$ candidates. For $i \geq 2$, the level-i refinement receives $\mathcal{O}(d_{i-1}/\vartheta)$ candidates. At a time outside all handover intervals, the final call receives $\mathcal{O}(d_L/\vartheta) = \mathcal{O}(b/\vartheta)$ candidates.*

Proof. At a maintained level-0 checkpoint, the dynamically maintained residual satisfies $|C^{\gamma}_{\tau_0}| < 2BM$, while the explicitly added elements with singleton value above γ contribute only $\mathcal{O}(k)$ additional candidates. Since $M = \Theta(k/\vartheta)$, $|C_{\tau_0}| = \mathcal{O}(Bk/\vartheta)$. Between the level-0 checkpoint and its level-1 descendant, only $\mathcal{O}(d_0) = \mathcal{O}(k)$ new elements can arrive. Hence the level-1 refinement receives $\mathcal{O}(Bk/\vartheta)$ candidates.

For $i \geq 2$, the level-$(i-1)$ checkpoint is already the output of a residual-refinement call and therefore satisfies $|C_{\tau_{i-1}}| = \mathcal{O}(d_{i-1}/\vartheta)$. The schedule allows only $\mathcal{O}(d_{i-1})$ subsequent insertions before the level-i checkpoint is created. Thus $|\mathcal{U}(\tau_{i-1}, \tau_i)| \leq |C_{\tau_{i-1}}| + |X_{\tau_i} \setminus X_{\tau_{i-1}}| = \mathcal{O}(d_{i-1}/\vartheta)$. The same argument with the most recent level-L checkpoint gives $\mathcal{O}(d_L/\vartheta) = \mathcal{O}(b/\vartheta)$ candidates for the final SAMPLEDEXCHANGE call. □

The following invariant controls the non-handover comparison in Lines 22–23 of Algorithm 1.

Lemma B.5. *Fix a checkpoint at time a and a consecutive interval $\mathcal{Q} = \{s_0, \dots, s_1\}$ of updates outside all handover intervals, with $a < s_0$. Write $J_s = I_a \cap X_s$. Suppose that a constant $D \geq 0$ satisfies $|(S_{s_0-1} \cap X_{s_0}) \triangle J_{s_0}| \leq D$ and $|\widehat{S}_s \triangle J_s| \leq D$ for all $s \in \mathcal{Q}$. Then $|S_s \triangle J_s| \leq D$ for every $s \in \mathcal{Q}$.*

Proof. At s_0, both possible outputs satisfy the assumed bound. For $s > s_0$, the no-reinsertion assumption and the fact that all elements of I_a arrived by time a imply $I_a \cap X_s = (I_a \cap X_{s-1}) \cap X_s$. Also, for any sets A, B, X, $(A \cap X) \triangle (B \cap X) = (A \triangle B) \cap X$. Consequently, $|(S_{s-1} \cap X_s) \triangle J_s| \leq |S_{s-1} \triangle J_{s-1}| \leq D$. The fresh candidate has distance at most D by assumption, so either choice made by Algorithm 1 maintains the bound. Induction proves the claim. □

We next relate the candidate bounds above to the distance between the public solution immediately before a handover and its new target.

Lemma B.6. *Let T_τ be the target created at the start of a level-i handover and let S^- be the public solution immediately beforehand. Then* $|(S^- \cap X_\tau) \triangle T_\tau| = \begin{cases} \mathcal{O}(d_i), & i = 0, 1, \\ \mathcal{O}(d_i/\vartheta), & i \geq 2. \end{cases}$

Proof. For $i = 0, 1$, both endpoints have cardinality at most k, while $d_0, d_1 = \Theta(k)$. Hence $|(S^- \cap X_\tau) \triangle T_\tau| \leq 2k = \mathcal{O}(d_i)$.

Now fix $i \geq 2$. Let a be the parent level-$(i-1)$ checkpoint used to construct the new level-i checkpoint at time τ, and define $J := I_a \cap X_\tau$. By Lemma B.4, the level-i refinement processes $\mathcal{O}(d_{i-1}/\vartheta)$ candidates. Each processed candidate changes the internal solution by at most two elements, and the completion step adds at most $\mathcal{O}(d_i/\vartheta)$ elements. Since $d_{i-1} = 2d_i$, $|T_\tau \triangle J| = \mathcal{O}(d_i/\vartheta)$.

It remains to bound the distance between the previous public solution and the same set J. The schedule of Appendix D.2 gives two possibilities.

If τ starts the first level-i child block associated with the parent checkpoint a, it immediately follows the completed handover of the parent. The surviving parent target differs from $I_a \cap X_\tau$ only through its completion and subsequent deletions. By Lemma A.13, $|(S^- \cap X_\tau) \triangle J| \leq |C_a| = \mathcal{O}(d_{i-1}/\vartheta) = \mathcal{O}(d_i/\vartheta)$.

Suppose instead that τ starts the second level-i child block. The updates immediately preceding τ contain a non-handover interval whose checkpoints are descendants of the same parent checkpoint a. For every update s in this interval, define $J_s := I_a \cap X_s$. By Lemma B.4 and the geometric relation $d_{j-1} = 2d_j$, every fresh candidate produced in this interval differs from J_s by at most $\mathcal{O}\left(\frac{d_{i-1} + \sum_{j=i}^{L-1} d_j + d_L}{\vartheta}\right) = \mathcal{O}(d_i/\vartheta)$. The surviving endpoint of the handover preceding this non-handover interval satisfies the same bound, since it is obtained from the same parent checkpoint through the corresponding refinement chain and target completion. Applying Lemma B.5 therefore gives $|(S_{\tau-1} \cap X_\tau) \triangle J| = \mathcal{O}(d_i/\vartheta)$.

If the preceding non-handover interval intersects initialization, start the induction from $S_0 = \emptyset$; a reference checkpoint before update 1 is empty, and all subsequent arrivals are counted by the same candidate bounds. The identity $I_a \cap X_\tau = (I_a \cap X_{\tau-1}) \cap X_\tau$ ensures that projection at the current update cannot enlarge the preceding distance. In either case, the triangle inequality with $|T_\tau \triangle J| = \mathcal{O}(d_i/\vartheta)$ proves $|(S^- \cap X_\tau) \triangle T_\tau| = \mathcal{O}(d_i/\vartheta)$. □

We can now convert the endpoint-distance bound into a per-update consistency guarantee.

Theorem B.7. *For every realization of the algorithm's randomness and every update,*

$$|S_t \triangle S_{t-1}| = \mathcal{O}\left(\frac{\log k}{\epsilon^2}\right).$$

Proof. First suppose $k \geq 2b_1$ and consider an update inside a level-i handover interval. The interval contains $h_i = d_i/b$ update positions. By Lemma B.6, the total path length is $\mathcal{O}(d_i)$ for

$i = 0, 1$ and $\mathcal{O}(d_i/\vartheta)$ for $i \geq 2$. Splitting this path into h_i blocks therefore gives respectively $\mathcal{O}(b)$ and $\mathcal{O}(b/\vartheta)$ changes per update. Since $\vartheta \leq 1$, both are $\mathcal{O}(b/\vartheta)$. A deletion of the current stream element may force at most one additional removal.

It remains to consider updates outside all handover intervals. Consider one maximal consecutive non-handover interval and let a be the most recent level-L checkpoint used throughout this interval. For every update s in the interval, define $J_s := I_a \cap X_s$. By Lemma B.4, the final SAMPLEDEXCHANGE call processes only $\mathcal{O}(d_L/\vartheta) = \mathcal{O}(b/\vartheta)$ candidates. Consequently, $|\widehat{S}_s \triangle J_s| = \mathcal{O}(b/\vartheta)$ for every such update.

At the first non-handover update following the preceding level-L handover, the surviving handover endpoint also differs from J_s by only $\mathcal{O}(b/\vartheta)$ elements: its target is obtained by completing the same level-L internal solution with at most $\mathcal{O}(b/\vartheta)$ residual elements, and subsequent projection onto the active set cannot increase this distance. Hence Lemma B.5 applies with $D = \mathcal{O}(b/\vartheta)$ and gives $|S_s \triangle J_s| = \mathcal{O}(b/\vartheta)$ throughout the entire non-handover interval. If this interval starts at update 1 with a virtual preceding checkpoint, its initial state is empty and the same induction starts from $S_0 = \emptyset$.

For two consecutive updates in this interval, $S_{s-1} \cap X_s$ and S_s are therefore both within $\mathcal{O}(b/\vartheta)$ of J_s. By the triangle inequality, $|S_s \triangle (S_{s-1} \cap X_s)| = \mathcal{O}(b/\vartheta)$. The update itself removes at most one element from S_{s-1}, so $|S_s \triangle S_{s-1}| = \mathcal{O}(b/\vartheta)$. The same argument covers the boundary from the preceding level-L handover to the first non-handover update.

Since $b = \mathcal{O}\Big(\frac{\log k}{\delta}\Big)$, the case $k \geq 2b_1$ satisfies $|S_t \triangle S_{t-1}| = \mathcal{O}\Big(\frac{\log k}{\delta\vartheta}\Big) = \mathcal{O}\Big(\frac{\log k}{\epsilon^2}\Big)$, because $\delta = 4\widehat{\epsilon}$, $\vartheta = 3\widehat{\epsilon}$, and $\widehat{\epsilon} = \Theta(\epsilon)$ for $\epsilon \in (0, 1/4)$.

Finally, when $k < 2b_1 = 2\lceil 2/\delta \rceil$, both consecutive outputs have cardinality at most k, and therefore $|S_t \triangle S_{t-1}| \leq 2k = \mathcal{O}(1/\epsilon)$, which is no larger than the claimed asymptotic bound. □

B.2.2 QUERY COMPLEXITY

Finally, we add the costs of maintaining the bands, processing each deletion horizon, completing the final residual, and constructing public targets. The enlarged coarse residual affects only the first higher-level call; all remaining costs form a geometric sum.

Theorem B.8. *The expected amortized query complexity per stream update is*

$$\mathcal{O}\left(k^2 \log^4 \frac{k}{\eta} + \frac{B^2 k}{\vartheta^2} + \frac{k}{\vartheta^2}\right), \qquad B = 1 + \left\lceil \log_2 \frac{k}{\eta} \right\rceil.$$

With $\eta = \widehat{\epsilon}/32$ and $\vartheta = 3\widehat{\epsilon}$, where $\widehat{\epsilon} = \Theta(\epsilon)$, this is $\mathcal{O}\big(k^2 \log^4 \frac{k}{\epsilon} + \frac{k}{\epsilon^2} \log^2 \frac{k}{\epsilon}\big)$, independent of the active-set size and the stream length.

Proof. By Lemma A.10, the auxiliary dyadic-band structures contribute $\mathcal{O}\Big(k^2 \log^4 \frac{k}{\eta}\Big)$ queries amortized per update. Singleton values are queried once on insertion and then cached. Selecting the quantile and identifying the heavy elements requires no additional value-oracle queries once the singleton values have been obtained.

Let $N_i(T)$ be the number of level-i handovers starting among updates $1, \ldots, T$. The uniform shift gives $\mathbb{E}[N_i(T)] = T/d_i$ by equation 3, including for prefixes shorter than one period. The level-1 residual-refinement call has $m = \mathcal{O}(Bk/\vartheta)$ candidates by Lemma B.4, so each call uses at most $\mathcal{O}(B^2k^2/\vartheta^2)$ queries. Its expected total cost is therefore $\mathcal{O}(TB^2k^2/(d_1\vartheta^2)) = \mathcal{O}(TB^2k/\vartheta^2)$ because $d_1 = \Theta(k)$. For every $i \geq 2$, the residual-refinement call has $m = \mathcal{O}(d_{i-1}/\vartheta)$ candidates, costs $\mathcal{O}(d_{i-1}^2/\vartheta^2)$, and has expected count T/d_i. Since $d_{i-1} = 2d_i$, level i contributes $\mathcal{O}(d_i/\vartheta^2)$ amortized queries. The geometric sum is $\mathcal{O}(k/\vartheta^2)$. The final routine costs $\mathcal{O}(b/\vartheta)$ per update outside handover intervals and is dominated by the same term. The comparison between $\widehat{S}_t$ and $S_{t-1} \cap X_t$ is performed at every non-handover update and requires only a constant number of additional value-oracle queries, which is dominated by the displayed bound. When $k < 2b_1$, finishing the $\mathcal{O}(Bk/\vartheta)$ residual is likewise dominated by the displayed bound. The completed endpoints add only $\mathcal{O}(B/\vartheta)$ amortized queries: level 0 scans at most $\mathcal{O}(Bk/\vartheta)$ residual elements per call with expected count T/d_0, and each higher level scans at most $\mathcal{O}(d_i/\vartheta)$ elements per call with expected

count T/d_i. Summing over $\mathcal{O}(B)$ levels is dominated by the displayed bound. Summing the terms proves the theorem. □

Theorems B.7 and B.8 together with the approximation proof complete Theorem 3.1.

C THE MONOTONE ALGORITHM

This appendix presents the monotone algorithm summarized at the end of Section 3. For sets $U, Z \subseteq V$, write $f(U \mid Z) = f(Z \cup U) - f(Z)$. We state its guarantee before giving the construction and proof. Let $\beta_\star > 0$ solve $e^\beta = \beta + 2$ and define $a_k = 1 - (1 + \beta_\star/k)^{-k}$, $\alpha_k^{\mathrm{mon}} = \frac{a_k}{1+\beta_\star}$.

Theorem C.1. *Let $f : 2^V \to \mathbb{R}_{\geq 0}$ be normalized, monotone, and submodular under the same update model. For every $\epsilon \in (0, 1/4)$, there is a randomized fully dynamic algorithm maintaining $S_t \subseteq X_t$, $|S_t| \leq k$, such that*

$$\mathbb{E}[f(S_t)] \geq (\alpha_k^{\mathrm{mon}} - \epsilon)\mathrm{OPT}_t, \qquad |S_t \triangle S_{t-1}| = \mathcal{O}\left(\frac{\log(2/\epsilon)}{\epsilon^2}\right).$$

The consistency bound is realization-wise, and the expected amortized query complexity is the same as in Theorem 3.1.

C.1 ALGORITHM DESCRIPTION

Throughout this appendix, set $\eta = \vartheta = \epsilon/256$ and $\delta = \vartheta/128$. Monotonicity permits a stronger simplification of the output updates than the non-monotone construction. We first extend the analysis of the ordered tail to deletions, and then reserve a small number of positions for unprocessed candidates. At the last deletion horizon, the algorithm places the complete residual and all recent arrivals into those positions instead of processing them through another unstable completion. The approximation coefficient changes only by $\mathcal{O}(\epsilon)$, while the number of deletion horizons drops from $\Theta(\log k)$ to $\mathcal{O}(\log(1/\epsilon))$.

C.1.1 THE TAILSELECT SUBROUTINE

Let $\beta_\star > 0$ be the solution of $e^\beta = \beta + 2$. For an integer tail capacity $K \leq k$, define $a_{k,K} := 1 - \left(1 + \frac{\beta_\star}{k}\right)^{-K}$. The acceptance and tail-selection rule is based on ENCOMPASSING-SET of Dütting et al. (2024). Here it is used within the dynamic sampling structure and with a smaller tail capacity. A state consists of the sequence $Z = (z_1, \ldots, z_m)$ of accepted elements and its last K elements $T = \mathrm{tail}_K(Z)$, where $\mathrm{tail}_K(Z) = (z_{\max\{1,m-K+1\}}, \ldots, z_m)$ for $m > 0$ and is empty for $m = 0$. Function values and set operations on a sequence refer to its underlying set. The assignment $Z \leftarrow (Z, u)$ appends u to this sequence. The full accepted set need not be feasible, but $|T| \leq K$. On acceptance, $w(u)$ is set to $f(u \mid Z)$ before u is appended. Saved sequences inherit their weights as specified in Appendix A.

Algorithm 7 TAILSELECT(Z, T, U, ℓ, K)

```
while U ≠ ∅ do
    choose u ∈ U; U ← U \ {u}; g_u ← f(u | Z)
    if g_u > 0 and g_u ≥ max{ℓ, (β_⋆/k) f(Z)} then
        w(u) ← g_u; Z ← (Z, u); T ← tail_K(Z)
return (Z, T)
```

The order may depend on the entire preceding execution. The first segment uses a minimum marginal threshold $\ell \geq 0$, positive when a singleton band is selected; every later segment sets $\ell = 0$.

We first bound the optimum by the value of the full accepted set. Every unselected optimal element failed either the relative-gain test or the initial minimum threshold. Since the objective is monotone, these rejection bounds remain valid as the accepted set grows.

Algorithm 8 BUCKETTAIL$(Z_0, T_0, U_0, \gamma, K)$

$(Z, T, U) \leftarrow (Z_0, T_0, U_0)$; $\ell_\gamma \leftarrow \eta\gamma/k$; $B \leftarrow 1 + \lceil \log_2(k/\eta) \rceil$; $M \leftarrow \lceil 32k/\vartheta \rceil$
while $U \neq \emptyset$ **do**
  $g(u) \leftarrow f(u \mid Z)$ for all $u \in U$
  $U \leftarrow \{u \in U : g(u) \geq \ell_\gamma,\ g(u) \geq (\beta_\star/k) f(Z)\}$
  $G_a \leftarrow \{u \in U : 2^a \ell_\gamma \leq g(u) < 2^{a+1}\ell_\gamma\}$ for $a = 0, \ldots, B-1$
  if $\max_a |G_a| < M$ **then**
    return (Z, T, U)
  $a^* \in \arg\max_a |G_a|$; sample p uniformly from G_{a^*}
  append a record $(Z_j, T_j, F_j^0, F_j, n_j, R_j, p_j) \leftarrow (Z, T, G_{a^*}, G_{a^*}, |G_{a^*}|, U, p)$
  $U \leftarrow U \setminus \{p_j\}$; $w(p_j) \leftarrow g(p_j)$; $Z \leftarrow (Z, p_j)$; $T \leftarrow \mathrm{tail}_K(Z)$
return (Z, T, U)

Lemma C.2. *Let (Z, T) be the final state of a segmented execution of Algorithm 7, with minimum marginal threshold ℓ only in its first segment. For every O of cardinality at most k whose elements are processed except possibly first-segment elements of singleton value below ℓ, $f(O) \leq (1 + \beta_\star) f(Z) + k\ell$.*

Proof. Consider $u \in O \setminus Z$. If u was rejected because its current marginal was nonpositive, then submodularity gives $f(u \mid Z) \leq 0$. If it failed the ratio test at its processing time, then $f(u \mid Z) \leq f(u \mid Z_{<u}) < \frac{\beta_\star}{k} f(Z_{<u}) \leq \frac{\beta_\star}{k} f(Z)$, where monotonicity makes the benchmark values nondecreasing. The only remaining possibility is a first-segment threshold rejection, whose final marginal is below ℓ. Monotonicity and submodularity now give $f(O) \leq f(O \cup Z) \leq f(Z) + \sum_{u \in O \setminus Z} f(u \mid Z) \leq (1 + \beta_\star) f(Z) + k\ell$. □

The full accepted set need not satisfy the cardinality constraint. We therefore retain only its last K elements. Each acceptance increases the benchmark value multiplicatively, so these recent elements capture a controlled fraction of its total value. The final inequality quantifies the cost of reserving $k - K$ positions.

Lemma C.3. *For every execution of Algorithm 7 from the empty state, or continuation of such an execution, $W(T) \geq a_{k,K} f(Z)$, $\quad f(T) \geq W(T)$. Moreover, if $s = k - K$, then $a_{k,K} \geq a_k - \frac{\beta_\star s}{k}$.*

Proof. If at most K elements are accepted, then $T = Z$ and $W(T) = f(Z)$. Otherwise let P be the benchmark immediately before the last K acceptances. Every one of those acceptances satisfies $f(Z^+) \geq \left(1 + \frac{\beta_\star}{k}\right) f(Z^-)$. Consequently $f(P) \leq (1 + \beta_\star/k)^{-K} f(Z)$, while telescoping stored marginals gives $W(T) = f(Z) - f(P) \geq a_{k,K} f(Z)$. Ordering the elements of T by acceptance time, their predecessors inside T are subsets of their benchmark predecessors, so submodularity makes each telescoping marginal in $f(T)$ at least its stored weight.

For the final claim, put $r = 1 + \beta_\star/k$. Then $a_k - a_{k,K} = r^{-K} - r^{-k} \leq 1 - r^{-s} \leq s \log r \leq \frac{\beta_\star s}{k}$, where $1 - e^{-x} \leq x$ and $\log(1 + x) \leq x$. □

The choice of $\beta_\star$ optimizes the asymptotic coefficient with $K = k$: $a_k \to 1 - e^{-\beta}$ and the first-order condition for $(1 - e^{-\beta})/(1 + \beta)$ is $e^\beta = \beta + 2$.

C.1.2 THE BUCKETTAIL SUBROUTINE

For a singleton band $\mathrm{Band}_\gamma(X)$, set $\ell_\gamma = \eta\gamma/k$, $B = 1 + \lceil \log_2(k/\eta) \rceil$, and $M = \lceil 32k/\vartheta \rceil$. Given a benchmark Z and unprocessed band elements U, define $P_{\mathrm{mon}}(Z, U) = \left\{u \in U : f(u \mid Z) \geq \ell_\gamma,\ f(u \mid Z) \geq \frac{\beta_\star}{k} f(Z)\right\}$. Partition this eligible set into the same dyadic current-marginal buckets. The coarse monotone routine repeatedly chooses a maximum-cardinality bucket of size at least M, samples one pivot uniformly, accepts it into Z, updates the last-K tail, and rescans all current marginals. It stops when every bucket has fewer than M elements.

As in the non-monotone algorithm, the dynamic cost depends on a bound on accepted elements. Here the benchmark value, rather than a replacement weight, grows geometrically after each acceptance. The singleton band bounds the last possible benchmark value.

Lemma C.4. *Let* $\kappa_{\mathrm{mon}} := 1 + \left\lceil \log_{1+\beta_\star/k} \frac{(1+k/\beta_\star)\gamma}{\ell_\gamma} \right\rceil$. *Every fresh band execution and every suffix continuation of Algorithm 8 accepts at most* $\kappa_{\mathrm{mon}} = \mathcal{O}(k \log(k/\eta))$ *elements. Its terminal residual has size below* BM.

Proof. The residual claim is the stopping rule. If no element is accepted there is nothing to prove. The first accepted marginal is at least ℓ_γ. Every later acceptance multiplies the current benchmark value by at least $1 + \beta_\star/k$. Moreover, immediately before any acceptance, $f(Z) \le \frac{k}{\beta_\star} f(u \mid Z) \le \frac{k\gamma}{\beta_\star}$, so immediately afterwards $f(Z) \le (1 + k/\beta_\star)\gamma$. Comparing this upper bound with the multiplicative growth from the first accepted element proves the result. □

In this appendix, MAINTAINBAND denotes the following adaptation of Algorithm 3. The records are those created in Algorithm 8, and the terminal state is $(Z_{\mathrm{end}}, T_{\mathrm{end}}, C_{\mathrm{end}})$, initially empty. Insertions, deletions, and the tests defining J are unchanged. If $j^* = \min J$, copy $(Z_0, T_0, U_0) \leftarrow (Z_{j^*}, T_{j^*}, R_{j^*})$ before discarding the invalid suffix, and set $(Z_{\mathrm{end}}, T_{\mathrm{end}}, C_{\mathrm{end}}) \leftarrow$ BUCKETTAIL$(Z_0, T_0, U_0, \gamma, K)$. For a terminal-residual trigger, make the same call from $(Z_{\mathrm{end}}, T_{\mathrm{end}}, C_{\mathrm{end}})$. Every sampled candidate is accepted, so there are no Bernoulli rejections. The reference set F_j^0 is fixed, while F_j and R_j are updated exactly as in Appendix A.2.

The update rules and source-size tests are unchanged, but the preserved state now consists of the benchmark and its ordered tail. We record the corresponding execution, distribution, deletion-loss, and query bounds together, since they are obtained from the same maintenance argument.

Lemma C.5. *After every update, the monotone band data structure satisfies all of the following.*

(i) Its records form a legal partial execution of Algorithm 7 on the current active band, with minimum marginal threshold ℓ_γ *and the prescribed tail capacity* K.

(ii) Conditional on a specified record's pre-draw creation history and retention, its pivot is uniform on its surviving source.

(iii) For every fixed set D *of at most* $2k$ *later deletions, if* A_γ *is the accepted band history of a checkpoint, then* $\mathbb{E}[W(A_\gamma \cap D)] \le \frac{\vartheta}{4}\mathbb{E}[W(A_\gamma)]$.

(iv) Maintaining all monotone band instances uses $\mathcal{O}\left(k^2 \log^4 \frac{k}{\eta}\right)$ *expected amortized oracle queries per update.*

Proof. For (i), insertions are appended to the unprocessed suffix, deletion of an unprocessed or rejected element simply removes it, and deletion of an accepted pivot reconstructs from the state immediately before that acceptance. Every deterministic trigger replaces only an affected suffix by a fresh legal suffix. This is the proof of Lemma A.4 with the exchange state replaced by (Z, T).

For (ii), a source is frozen before its uniform pivot draw. Later liveness tests inspect only source and suffix cardinalities. Against an oblivious stream, conditioning on the fixed deleted subset and on the pivot still being alive leaves it uniform over the surviving source; pivot deletion starts a new record. This is the creation-history argument of Lemma A.5, with acceptance probability one.

For (iii), condition as in (ii). Source marginals against the frozen pre-draw benchmark differ by a factor below two, every live source has size at least $M/2$, and $|D| \le 2k$. The calculation in Lemma A.3 applies without the Bernoulli multiplier.

For (iv), a suffix with r candidates is rescanned only after an acceptance and therefore costs $\mathcal{O}(r(\kappa_{\mathrm{mon}} + 1))$. The growth, source-shrink, and pivot-deletion charges in Lemma A.9 use only source uniformity, the ratio $|R_j^0| \le B|F_j^0|$, and the number of accepted records. Replacing κ by κ_{mon} gives $\mathcal{O}(B\kappa_{\mathrm{mon}}^2 + \kappa_{\mathrm{mon}})$ per band. Each element belongs to $\mathcal{O}(\log(k/\eta))$ bands, proving the displayed bound. □

C.1.3 THE RESIDUALTAIL SUBROUTINE

Above the coarse level we use inverse-marginal sampling only on parameterized candidate sets. Given a state (Z, T), candidates U, deletion scale d, and tail capacity K, define the eligible pool $P =$

Algorithm 9 RESIDUALTAIL(Z, T, U, d, K)

$m_d \leftarrow \lceil 8d/\vartheta \rceil$
while $U \neq \emptyset$ **do**
  $g(u) \leftarrow f(u \mid Z)$ for all $u \in U$
  $U \leftarrow \{u \in U : g(u) > 0,\ g(u) \geq (\beta_\star/k) f(Z)\}$
  if $|U| < m_d$ **then**
    return (Z, T, U)
  sample $u \in U$ with probability $g(u)^{-1} / \sum_{x \in U} g(x)^{-1}$
  $U \leftarrow U \setminus \{u\}$; $w(u) \leftarrow g(u)$; $Z \leftarrow (Z, u)$; $T \leftarrow \mathrm{tail}_K(Z)$
return (Z, T, U)

$\left\{u \in U : f(u \mid Z) > 0, f(u \mid Z) \geq \frac{\beta_\star}{k} f(Z)\right\}$. While $|P| \geq \lceil 8d/\vartheta \rceil$, sample u with probability proportional to $1/f(u \mid Z)$, accept it into Z, update the tail T, and recompute eligibility. Stop with the remaining pool as residual.

The inverse-marginal calculation in Lemma A.12 gives, for every pivot and every fixed set of at most $2d$ relevant deletions, $\mathbb{E}[w(p)\mathbf{1}\{p \in D\}] \leq \frac{\vartheta}{4}\mathbb{E}[w(p)]$. The residual has size less than $8d/\vartheta + 1$, and a direct implementation on m candidates costs $\mathcal{O}(m^2)$ queries.

C.1.4 THE CONSISTENTTAIL ALGORITHM

We reserve $s = \lceil \vartheta k/8 \rceil$ positions and use tail capacity $K = k - s$ when k is large enough to stop the hierarchy at a deletion horizon proportional to $\vartheta^2 k$. Let $L_{\mathrm{mon}} = \lceil \log_2(2^{14}/\vartheta^2) \rceil$ and $b_{\mathrm{mon}} = 2^{\lceil \log_2(128(L_{\mathrm{mon}}+1)/\vartheta) \rceil}$. With $d_0 = 2^{\lfloor \log_2 k \rfloor}$, this choice is used precisely when $d_0/2^{L_{\mathrm{mon}}} \geq b_{\mathrm{mon}}$. Then $L = L_{\mathrm{mon}}$, $b = b_{\mathrm{mon}}$, and $d_i = d_0/2^i$; every $h_i = d_i/b$ is a positive integer. The handover probability is at most $(L_{\mathrm{mon}} + 1)/b_{\mathrm{mon}} \leq \vartheta/128$.

Otherwise we set $s = 0$, $K = k$, and use the full hierarchy with $\delta = \vartheta/128$ when $k \geq 2b_1$, or no schedule when $k < 2b_1$. The failed inequality implies $k < 2^{L_{\mathrm{mon}}+1} b_{\mathrm{mon}}$, and therefore $\log k = \mathcal{O}(\log(2/\vartheta))$ in this case. The condition $s > 0$ implies $k \geq 2b_1$, since $k \geq 2^{L_{\mathrm{mon}}} b_{\mathrm{mon}}$ and $2b_1 \leq 512/\vartheta + 2$. Algorithm 10 includes all these cases.

SELECTCHECKPOINT is used with the same quantile and the same explicit set of elements above the selected scale as in Algorithm 4, but returns the monotone triple $(Z_t^\gamma, T_t^\gamma, C_t^\gamma \cup H_t)$. When the active or positive-singleton set is small, it returns empty Z, T and the corresponding candidate set. All band states use the capacity K fixed at initialization. Inherited benchmarks retain inactive accepted elements; only the public output is intersected with X_t. As before, define $\mathcal{U}(a, t) = (C_a \cap X_t) \cup (X_t \setminus X_a)$.

If $s > 0$, let τ be the most recent level-L checkpoint. At times outside handover intervals, set $U_t = \mathcal{U}(\tau, t)$ and $S_t = (T_\tau \cap X_t) \cup U_t$. At a level-$L$ handover start τ, the frozen target is $(T_\tau \cap X_\tau) \cup (C_\tau \cap X_\tau)$, and the output during the handover additionally includes all active arrivals $X_t \setminus X_\tau$. At earlier levels the target is just the active tail. If $s = 0$, all targets are active tails and the final candidates are processed with TAILSELECT before comparing the resulting tail with the surviving previous solution.

Ties in the two comparisons may be broken in favor of the surviving previous solution. The empty-state convention in Appendix D.2 applies before update 1; in particular, a last-level handover beginning before initialization has empty target and uses $X_\tau = \emptyset$ in the rule for subsequent arrivals.

Lemma C.6. *When $s > 0$, Algorithm 10 satisfies $|S_t| \leq k$ after every update, including updates inside the final-level handover intervals.*

Proof. Write $d = d_L$. The final refinement leaves fewer than $8d/\vartheta + 1$ candidates, and at most $2d$ later arrivals need to be included before that checkpoint is replaced. Since $d \leq \vartheta^2 k/2^{14}$ and $d \geq b_{\mathrm{mon}} \geq 1$, we have $(8/\vartheta + 3)d + 1 \leq 12\vartheta k/2^{14} \leq \vartheta k/8 \leq s$. Thus the tail of size at most $K = k - s$, its residual, and all relevant arrivals fit in the available capacity.

For a final-level handover, both the surviving previous output and the new target have size at most $K + 8d/\vartheta + 1 + 2d$. The delete-first, insert-second path never exceeds the larger endpoint size, and at most $h_L \leq d$ additional arrivals are included during the handover. The preceding inequality

Algorithm 10 CONSISTENTTAIL(f, k, ϵ)

$\eta \leftarrow \epsilon/256$; $\vartheta \leftarrow \epsilon/256$; $\delta \leftarrow \vartheta/128$; $b_1 \leftarrow \lceil 2/\delta \rceil$; $X_0 \leftarrow \emptyset$; $S_0 \leftarrow \emptyset$
$d_0 \leftarrow 2^{\lfloor \log_2 k \rfloor}$; $L_{\mathrm{mon}} \leftarrow \lceil \log_2(2^{14}/\vartheta^2) \rceil$; $b_{\mathrm{mon}} \leftarrow 2^{\lceil \log_2(128(L_{\mathrm{mon}}+1)/\vartheta) \rceil}$
$s \leftarrow \lceil \vartheta k/8 \rceil$ if $d_0/2^{L_{\mathrm{mon}}} \geq b_{\mathrm{mon}}$; otherwise $s \leftarrow 0$; $K \leftarrow k - s$
if $k \geq 2b_1$ **then**
    if $s > 0$ **then**
        $L \leftarrow L_{\mathrm{mon}}$; $b \leftarrow b_{\mathrm{mon}}$
    else
        $L \leftarrow \max\{\ell \geq 1 : 2^\ell \lceil (\ell+1)/\delta \rceil \leq k\}$; $b \leftarrow \lceil (L+1)/\delta \rceil$; $d_0 \leftarrow b2^L$
    $d_i \leftarrow d_0/2^i$; $h_i \leftarrow d_i/b$ for $i = 0, \ldots, L$; initialize the schedule of Appendix D.2 with an independent uniform shift
for each update (v_t, op_t) at time t **do**
    $X_t \leftarrow X_{t-1} \cup \{v_t\}$ if $\mathrm{op}_t = \mathrm{insert}$; otherwise $X_t \leftarrow X_{t-1} \setminus \{v_t\}$
    call the monotone MAINTAINBAND$(v_t, \mathrm{op}_t, \gamma)$ for all dyadic γ with $\eta\gamma/k \leq f(\{v_t\}) \leq \gamma$
    if $k < 2b_1$ **then**
        $(Z, T, C) \leftarrow$ SELECTCHECKPOINT$(t, 2k)$; $(Z', T') \leftarrow$ TAILSELECT$(Z, T, C, 0, k)$; $\widehat{S}_t \leftarrow T' \cap X_t$
        $S_t \in \arg\max_{A \in \{\widehat{S}_t, S_{t-1} \cap X_t\}} f(A)$; **output** S_t; **continue**
    if a level-i handover starts at t **then**
        if $i = 0$ **then**
            $(Z_t, T_t, C_t) \leftarrow$ SELECTCHECKPOINT$(t, 2d_0)$
        else
            let a be the parent level-$(i-1)$ checkpoint time specified by the schedule
            $(Z_t, T_t, C_t) \leftarrow$ RESIDUALTAIL$(Z_a, T_a, \mathcal{U}(a,t), d_i, K)$
        $Y_t \leftarrow T_t \cap X_t$; if $s > 0$ and $i = L$, set $Y_t \leftarrow Y_t \cup (C_t \cap X_t)$
        freeze Y_t and partition the delete-first, insert-second path from $S_{t-1} \cap X_t$ to Y_t into h_i nearly equal blocks
    if t lies in a level-i handover starting at τ **then**
        $S_t \leftarrow$ the next frozen path state intersected with X_t
        if $s > 0$ and $i = L$ **then**
            $S_t \leftarrow S_t \cup (X_t \setminus X_\tau)$
    else
        let a be the most recent level-L checkpoint preceding t
        if $s > 0$ **then**
            $S_t \leftarrow (T_a \cap X_t) \cup \mathcal{U}(a,t)$
        else
            $(Z', T') \leftarrow$ TAILSELECT$(Z_a, T_a, \mathcal{U}(a,t), 0, k)$; $\widehat{S}_t \leftarrow T' \cap X_t$
            $S_t \in \arg\max_{A \in \{\widehat{S}_t, S_{t-1} \cap X_t\}} f(A)$
    output S_t

therefore proves feasibility at every intermediate update. At other levels the target is a tail of size at most K; the same path construction between feasible endpoints is feasible without adding arrivals. Initialization is empty, so induction covers all updates. □

The shorter hierarchy immediately reduces the number of handover scales. We next translate this reduction into recourse, while checking that the dynamic band maintenance and the remaining residual-refinement calls retain the same query bound.

Lemma C.7. *The monotone algorithm is $\mathcal{O}\Big(\frac{\log(2/\epsilon)}{\epsilon^2}\Big)$ consistent for every realization, and its expected amortized query complexity is $\mathcal{O}\big(k^2 \log^4 \frac{k}{\epsilon} + \frac{k}{\epsilon^2} \log^2 \frac{k}{\epsilon}\big)$.*

Proof. When $s > 0$, $L_{\mathrm{mon}} = \mathcal{O}(\log(2/\vartheta))$ and hence $b_{\mathrm{mon}} = \mathcal{O}(\log(2/\vartheta)/\vartheta)$. For every nonfinal level, the target-distance bounds of Lemma B.6 continue to hold with the tail capacity reduced from k to K. At the final level, the residual contributes only $\mathcal{O}(d_{L_{\mathrm{mon}}}/\vartheta)$ additional elements, so the total path length is still $\mathcal{O}(d_{L_{\mathrm{mon}}}/\vartheta)$. Dividing by a handover interval of length d_i/b_{mon} gives $\mathcal{O}(b_{\mathrm{mon}}/\vartheta)$ changes per handover update. Between updates outside handover intervals, the final checkpoint is fixed and U_t changes by at most the single stream update. The last-level handover accumulates $X_t \setminus X_\tau$, which changes by at most one input element per update. At its end, the output is exactly $(T_\tau \cap X_t) \cup (C_\tau \cap X_t) \cup (X_t \setminus X_\tau)$, the formula used immediately afterwards; hence no uncharged batch of arrivals is inserted at the boundary. Thus recourse is $\mathcal{O}(\log(2/\vartheta)/\vartheta^2)$.

The level-0 band structures have exactly the cost in Lemma C.5. Here $d_0, d_1 = \Theta(k)$, so the first refinement has the same $\mathcal{O}(B^2k/\vartheta^2)$ expected amortized query bound as in Theorem B.8. The later levels again contribute the geometric sum $\mathcal{O}(k/\vartheta^2)$, and the final completion is no longer run. This gives the displayed update bound. When $s = 0$ and $k \geq 2b_1$, every accepted candidate changes the ordered tail by at most two elements. The candidate and endpoint bounds therefore follow by the same common-parent argument as Lemma B.6. Applying Lemma B.5 with I_a replaced by T_a proves that comparing the fresh tail with the surviving previous output preserves $\mathcal{O}(\log k/\vartheta^2)$ recourse; the query sum is unchanged. When $k < 2b_1$, recourse is at most $2k$ and each residual completion is linear in its candidate count. Finally, the failed test $2^{\lfloor \log_2 k \rfloor - L_{\mathrm{mon}}} < b_{\mathrm{mon}}$ implies $k < 2^{L_{\mathrm{mon}}+1} b_{\mathrm{mon}}$, so $\log k = \mathcal{O}(\log(2/\vartheta))$. □

C.2 Approximation Analysis

We prove the approximation by comparing two quantities: an upper bound on the current optimum and a lower bound on the returned solution. Both bounds retain the marginal contribution of the unprocessed residual. This is important because that residual is included directly in the output, rather than rejected by the selection rule. We then account for deleted accepted elements and the independent handover schedule.

Fix the entire independent schedule and a time t outside its handover intervals, assuming $s > 0$. All expectations below are conditional on that schedule. Concatenate the relevant refinement segments, and let Z, T be the resulting historical benchmark and tail of capacity K. Let $U = U_t$ be the active residual and recent-arrival set in Algorithm 10. All active elements outside L_{τ_0} outside U are processed exactly once by the concatenated execution. Stored marginals telescope exactly: $W(Z) = f(Z)$.

Unlike the non-monotone algorithm, the last residual is left unprocessed. The upper bound on the optimum must therefore retain its marginal contribution explicitly instead of treating these elements as rejected. This gives the additional term $f(U \mid Z)$ below.

Lemma C.8. *Let O' be a size-k comparator after removing the low set selected by the level-0 quantile. Then, pointwise,*

$$f(O') \leq (1 + \beta_\star) f(Z) + f(U \mid Z) + k\ell. \tag{2}$$

Proof. Every element of $O' \setminus (Z \cup U)$ was processed and rejected. As in Lemma C.2, its final marginal to Z is at most $(\beta_\star/k) f(Z)$, except for the first-segment threshold contribution ℓ. Since adding U can only decrease these marginals, $f(O') \leq f(O' \cup Z \cup U) \leq f(Z \cup U) + \sum_{u \in O' \setminus (Z \cup U)} f(u \mid Z \cup U) \leq f(Z) + f(U \mid Z) + \beta_\star f(Z) + k\ell$. □

Some accepted benchmark elements may no longer be active. Their expected stored weight is bounded using the same per-level deletion argument. Since the benchmark only adds elements, its stored weights telescope exactly to its function value.

Lemma C.9. *At every time outside all handover intervals, $\mathbb{E}[W(Z \setminus X_t)] \leq \frac{\vartheta}{4} \mathbb{E}[f(Z)]$.*

Proof. For the coarse segment, apply part (iii) of Lemma C.5. For every higher refinement segment, apply Lemma A.12 without the acceptance indicator; at most $2d_i$ relevant deletions occur before time t. Sum over the disjoint acceptance histories and use $W(Z) = f(Z)$. □

We now lower bound the value of the returned solution. The surviving tail contributes the retained benchmark weight; the unprocessed elements contribute at least their marginal to the larger benchmark. Keeping both terms allows the residual term in the preceding upper bound to be accounted for.

Lemma C.10. *At every time outside all handover intervals,*

$$\mathbb{E}[f((T \cap X_t) \cup U)] \geq \left(a_{k,K} - \frac{\vartheta}{4} \right) \mathbb{E}[f(Z)] + \mathbb{E}[f(U \mid Z)].$$

Proof. For every realization, Lemma C.3 gives $W(T) \geq a_{k,K} f(Z)$. Hence $W(T \cap X_t) \geq a_{k,K} f(Z) - W(Z \setminus X_t)$. Because $T \cap X_t \subseteq Z$, monotonicity and submodularity give $f((T \cap X_t) \cup U) = f(T \cap X_t) + f(U \mid T \cap X_t) \geq W(T \cap X_t) + f(U \mid Z)$. Take expectations and apply Lemma C.9. □

The upper bound on the optimum and the lower bound on the exposed set now contain the same residual term. Combining them, and then applying the singleton-threshold loss, gives the following approximation guarantee.

Lemma C.11. *At every time outside all handover intervals when $s > 0$,*

$$\mathbb{E}[f(S_t)] \geq \frac{a_{k,K} - \vartheta/4}{1 + \beta_\star}(1 - 4\eta)\mathrm{OPT}_t.$$

Proof. Let $O'_t = O_t \setminus L_{\tau_0}$. The quantile lemma gives $f(O'_t) \geq (1 - 2\eta)\mathrm{OPT}_t$, $k\ell_{\gamma_{\tau_0}} \leq 2\eta\mathrm{OPT}_t$. Put $\alpha' = \frac{a_{k,K} - \vartheta/4}{1+\beta_\star}$. The condition $d_{L_{\mathrm{mon}}} \geq b_{\mathrm{mon}}$ gives $K \geq k/2$, so $a_{k,K} > \vartheta/4$ and $0 < \alpha' \leq 1$. Since $f(U \mid Z) \geq 0$, Lemma C.10 yields $\mathbb{E}[f(S_t)] \geq \alpha' \mathbb{E}[(1 + \beta_\star) f(Z) + f(U \mid Z)]$. Apply Lemma C.8 pointwise and then the two quantile inequalities. □

Proof of Theorem C.1.

Proof. First consider $s > 0$. For a fixed time t, the random schedule shift places t in a handover interval with probability at most $\vartheta/128$. Since $f \geq 0$, handover states may be omitted from a lower bound. By Lemma C.11,

$$\mathbb{E}[f(S_t)] \geq \left(1 - \frac{\vartheta}{128}\right) \frac{a_{k,K} - \vartheta/4}{1 + \beta_\star}(1 - 4\eta)\mathrm{OPT}_t.$$

The condition $d_{L_{\mathrm{mon}}} \geq b_{\mathrm{mon}}$ together with $d_{L_{\mathrm{mon}}} \leq \vartheta^2 k/2^{14}$ give $1/k \leq \vartheta/8$. Hence $s/k \leq \vartheta/4$, and $\beta_\star < 2$ together with Lemma C.3 gives $a_{k,K} \geq a_k - \vartheta/2$. Since the remaining factors are in $[0, 1]$, the coefficient above is at least $\alpha_k^{\mathrm{mon}} - 4\eta - \vartheta$. Substituting $\eta = \vartheta = \epsilon/256$ proves the stated approximation.

When $s = 0$ and $k \geq 2b_1$, the final TAILSELECT call makes $U = \emptyset$ and uses $K = k$. The same benchmark and deletion-loss arguments apply to its fresh tail, and the comparison in Algorithm 10 cannot reduce the returned value. The handover probability is at most $\delta = \vartheta/128$, so the same estimate holds without the reserved-capacity loss. In the case $k < 2b_1$, the current coarse checkpoint is completed immediately by TAILSELECT; hence no accepted checkpoint element becomes inactive between constructing the checkpoint and returning the completed solution and there is no handover loss. The same benchmark/tail argument with $U = \emptyset$, together with the current-time singleton-threshold bound, therefore yields at least the full-hierarchy coefficient. The resource guarantees follow from Lemma C.7. □

D ADDITIONAL DETAILS OF CONSISTENTSAMPLE

This appendix specifies the handover schedule and justifies concatenating processing segments created at different times. The latter requires both a valid realized processing order and probability identities for dynamically retained trials; these are separate assertions. No particular data-structure implementation is prescribed.

D.1 FEASIBLE HANDOVER PATHS

For two feasible sets A, B, retain $A \cap B$, remove the elements of $A \setminus B$, and then insert the elements of $B \setminus A$, using fixed orders. This gives a path $P_0 = A, \ldots, P_m = B$ with $m = |A \triangle B|$, $|P_j| \leq \max\{|A|, |B|\}$, and exactly one insertion or deletion between consecutive path states. For a handover of $h \geq 1$ updates, use the states $P_{\lfloor rm/h \rfloor}$, $r = 1, \ldots, h$; empty blocks are allowed when $m < h$. Each block has at most $\lceil m/h \rceil$ changes. The actual output is intersected with the current active set. Since no identity is reinserted, a stream update introduces at most one additional forced

removal, giving at most $\lceil m/h \rceil + 1$ changes. A path starting from $S_{\tau-1} \cap X_\tau$ uses only elements already inserted by its start τ. The extra-arrival rule for the monotone algorithm is specified separately in Algorithm 10.

D.2 Properties of the Handover Schedule

For the deletion horizons $d_i = d_0/2^i$ and positive integral handover lengths $h_i = d_i/b$ used in the algorithms, let $b > L + 1$. Building on the staggered scheduling scheme of Dütting et al. (2026), we use a periodic construction tailored to our deletion horizons and handover lengths. A level-i block has length $D_i = (b-i)h_i$ and begins with a handover of length h_i. If $i < L$, its remaining positions form two consecutive child blocks of length $D_{i+1} = (D_i - h_i)/2$. Thus, for a parent starting at s, the children start at $s + h_i$ and $s + h_i + D_{i+1}$. At level L, the remaining positions are idle. Repeat the level-0 block with period $D_0 = d_0$, and shift the entire pattern by a uniform $\sigma \in \{0, \dots, d_0 - 1\}$, independently of all candidate-sampling randomness. All lengths are integral because $h_i = 2h_{i+1}$ for $i < L$.

Disjointness and frequency. Children start after their parent handover, and sibling blocks are disjoint, so no two handover intervals overlap. One period contains 2^i level-i handovers, of total length $2^i h_i = d_0/b$. Hence the probability that any fixed update lies in a handover is $(L + 1)/b$. In the full hierarchy, $b = \lceil (L + 1)/\delta \rceil$, so this probability is at most δ. The monotone algorithm with $s > 0$ uses the same construction with $L_{\mathrm{mon}}, b_{\mathrm{mon}}$, giving probability at most $\vartheta/128$.

The shift also controls costs over arbitrarily short update prefixes. If $N_i(T)$ is the number of level-i handovers starting among updates $1, \dots, T$, every fixed update has probability $2^i/d_0 = 1/d_i$ of being such a start. Linearity of expectation gives

$$\mathbb{E}_\sigma[N_i(T)] = T/d_i \qquad \text{for every integer } T \geq 1. \tag{3}$$

Thus the query analysis uses an expected count valid for every prefix, not merely a long-run frequency estimate.

Checkpoint ages and predecessor relations. Outside all handovers, the current position belongs to one relevant block at each level; its level-i checkpoint is the start of that block. Its age is less than $D_i \leq d_i$, and in particular satisfies the $2d_i$ bound used in the proofs. The first child starts immediately after the parent handover. The second child starts after the first child block has finished. Since $b > L + 1$, every leaf has a nonempty idle suffix, so the update immediately preceding a second child is outside all handovers. These are the two predecessor cases used in Lemma B.6. Every maximal consecutive run of handovers ends with a level-L handover.

The periodic pattern is defined on all integer positions. Before update 1, interpret the active set, checkpoints, and path states as empty. This convention handles a partially elapsed first period without processing virtual updates or making oracle queries. Later arrivals are included by the same residual rule as at any other time. When $k \geq 2b_1$, maximality of L gives $k < 2^{L+1}\lceil (L + 2)/\delta \rceil$; together with $d_0 = 2^L \lceil (L + 1)/\delta \rceil \leq k$, this yields $d_0, d_1 = \Theta(k)$ and $d_L = b$ as used by the query analysis. When $k < 2b_1$, no handover schedule is used.

D.3 Validity of the Segmented Processing

Fix the oblivious input sequence and the entire independent schedule. We first establish the pointwise relation between the retained coarse history and its subsequent continuations.

Lemma D.1. *Along the checkpoint chain used at a time outside handover intervals, the retained coarse processing order, the selected residual-refinement calls, and the final continuation obey one sequence of replacement decisions from the empty set. Each identity occurs at most once, every active element outside the stated low-singleton set is represented, and only the coarse segment imposes the additional marginal floor.*

Proof. Lemma A.4 supplies the coarse processing order. Heavy elements are appended to its residual without being tested. At a child checkpoint, the candidate set is $X_{\tau_i} \setminus (X_{\tau_{i-1}} \setminus C_{\tau_{i-1}}) = (C_{\tau_{i-1}} \cap X_{\tau_i}) \cup (X_{\tau_i} \setminus X_{\tau_{i-1}})$, namely the surviving unprocessed residual and subsequent arrivals.

No identity is reinserted, so a processed identity cannot return to a later candidate set. Inactive processed elements may remain in the conceptual historical order, but are not made available to later calls. Each segment inherits its predecessor's solution and stored weights; the final call processes the remaining active residual. Concatenation therefore preserves the replacement rules and covers all relevant active elements. When $k < 2b_1$, the same construction applies without intermediate refinement segments. □

Pointwise validity alone does not imply that a retained history has the distribution of a fresh static execution. The approximation proof instead uses the following two moment statements.

Lemma D.2. *For the completed chain, let A be its accepted history, R its coin-rejected eligible elements, and g_u the gain at an eligible element's trial. For every fixed comparator O and every identity u,*

$$\Pr[u \in A] \le q, \qquad \mathbb{E}\left[\sum_{u \in O \cap R} g_u\right] = \frac{1-q}{q}\,\mathbb{E}[W(O \cap A)].$$

Expectations are over internal sampling conditional on the fixed schedule. The statement also holds when $k < 2b_1$.

Proof. The coarse checkpoint time, singleton quantile, and selected band depend only on the fixed input and schedule. For a coarse creation record j, apply equation 1 with $a_j(p_j) = f(p_j \mid I_j)\mathbf{1}\{p_j \in O\}$, and apply its rejection counterpart. Its accepted and rejected contributions have factors q and $1-q$ multiplying the same expected retained pre-trial contribution. The identical invalidation tests for both trial outcomes are essential here.

In a selected refinement or final call, the input may depend on preceding trials, but the next candidate and its gain are fixed before its own fresh coin is revealed. The same two identities hold conditional on each pre-trial history. The selected checkpoint times are fixed by the schedule, and later acceptances or deletions do not erase trials from these frozen segments. Summing over retained coarse creation events and all trials of the selected continuations proves the weighted identity. Using indicator contributions instead of gains gives $\Pr[u \in A] = q$ times the expected number of eligible trials of u represented in the chain. This number is at most one by Lemma D.1. When $k < 2b_1$, the current checkpoint is determined by the input and the final continuation is fresh, so the argument is unchanged. □

The pointwise inequalities of Lemmas A.14 and A.15, including the $k\ell$ floor contribution, apply to this order. Lemma D.2 supplies the acceptance/rejection cancellation and the marginal-inclusion premise of the submodular sampling lemma used in Lemma A.17. Thus its bound applies without assuming equality in distribution with a fresh execution. The comparator $O_t \setminus L_{\tau_0}$ is fixed under the conditioned input and schedule.

For the monotone specialization, the same set identity concatenates the benchmark-and-tail segments. The full ordered benchmark and its tail are inherited. At the final level when $s > 0$, the surviving residual and later arrivals are deliberately left unprocessed and returned in reserved positions; every other active element outside L_{τ_0} is processed once. The acceptance rule is deterministic, so no Bernoulli cancellation is needed. The pointwise benchmark bound and the source-distribution deletion bounds apply directly.