Non-Monotone Submodular Maximization under Matroid and Knapsack constraints

Anonymous Authors[1]

Abstract

In this work, we propose two efficient parallel algorithms, LinAst and LinAtg, which improve both the approximation ratio and query complexity of existing practical parallel algorithms for non-monotone submodular maximization over the ground-sized n under a cardinality constraint k. Specifically, our algorithms maintain the best adaptive complexity of $O(\log n)$ while significantly improving the approximation ratio from $1/6 - \epsilon$ to $0.193 - \epsilon$ and reducing the query complexity from $O(n \log(k))$ to $O(n)$. The key building block of our algorithms is LinAdapt, a constant approximation ratio within $O(\log n)$ sequence rounds and linear queries. LinAdapt can reduce the complexity of the query by providing $O(1)$ guesses of the optimal value. We further introduce the BoostAdapt algorithm that returns a better ratio of $1/4 - \epsilon$ within $O(\log(n)\log(k))$ adaptive complexity and $O(n\log(k))$ query complexity. Our BoostAdapt works in a novel staggered greedy threshold framework that alternately constructs two disjoint sets in $O(\log k)$ sequential rounds. In addition to theoretical analysis, the results of the experiments on validated benchmarks confirm the superiority of our algorithms in terms of solution quality, number of required queries, and running time over cutting-edge algorithms.

1. Introduction

Mo ta ve bai toan Submax va mot so nghien cuu gan day Maximizing a non-negative (not necessarily monotone) submodular set function under cardinality constraint is a fundamental and important problem that has a wide range of applications in the fields of artificial intelligence and machine learning, such as data summarization (Kuhnle, 2021; Lin & Bilmes, 2011; Chen & Kuhnle, 2023; Fahrbach et al., 2019a; Han et al., 2021; Mirzasoleiman et al., 2016a), revenue maximization in social networks (Kuhnle, 2021; Chen & Kuhnle, 2023), recommendation systems (Mirzasoleiman et al., 2016a) and weight cut (Chen & Kuhnle, 2023; Kuhnle, 2021). Given a finite ground set V sized n, a submodular set function $f : 2^V \mapsto \mathbb{R}^+$ and a positive number k (cardinality constraint), the problem of submodular maximization under cardinality constraint (SMC) asks to find a set $S \subseteq V$ with $|S| \leq k$ so that $f(S)$ is maximized. The problem has received much attention for finding approximation solutions with theoretical bounds (Buchbinder et al., 2014; 2015; Kuhnle, 2019; Chen & Kuhnle, 2023; Gupta et al., 2010). Tap trung vao cac bai toan co rang buoc

Mo ta ve bai toan voi rang buoc matroid va knapsack. Nho phan tich mot vi du ve bai toan nay trong thuc tien However, the problem must face with an exponentially increasing search space due to the explosion of input data. This motivates much effort to design efficient parallel algorithms that can power the parallel architectures of computer systems to obtain a good solution promptly (See Table 2 for an overview of parallelizable algorithms). In this context, the concept of *adaptive complexity* or *adaptivity* becomes an essential measure of the feasibility of any parallel algorithm. This concept evaluates the number of sequential rounds of an algorithm that can execute many independent polynomial oracle queries in parallel (Balkanski & Singer, 2018). It is noted that improving adaptive complexity from $O(\log^2(n))$ to $O(\log n)$ dramatically reduces the number of sequential iterations, thus significantly reducing the running time of algorithms in practice (Fahrbach et al., 2019a; Ene & Nguyen, 2020; Amanatidis et al., 2021; Fahrbach et al., 2023; Kuhnle, 2021; Chen & Kuhnle, 2024). Although the recent studies significantly reduce the adaptive complexity from $O(\log^2(n))$ to $O(\log n)$, they still face with many challenges, including:

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Table 1. Comparison of efficient parallel algorithms for SMC.† The algorithm of (Amanatidis et al., 2021) may not return any approximation ratio due to the technique flaws that have been pointed (Cui et al., 2025).

Reference	Ratio	Adaptivity	Query Complexity
(Gupta et al., 2010)	$1/6-\epsilon$	$O(nk)$	$O(nk)$
(Buchbinder et al., 2015)	$1/e-\epsilon \approx 0.367-\epsilon$	$O(k)$	$O(n)$
(Chen et al., 2024)	$0.385-\epsilon$	$O(kn)$	$O(kn)$
(Balkanski et al., 2018)	$1/(2e)-\epsilon \approx 0.183-\epsilon$	$O(\log^2(n))$	$O(\mathsf{opt} n \log^2(n)\log(k))$
(Chekuri & Quanrud, 2019)	$3-2\sqrt{2}-\epsilon \approx 0.171-\epsilon$	$O(\log^2(n))$	$O(nk^4\log^2(n))$
(Ene & Nguyen, 2020)	$1/e-\epsilon \approx 0.367-\epsilon$	$O(\log n)$	$\Omega(nk^2\log^2(n))$
(Fahrbach et al., 2023) (ANM)	$0.039-\epsilon$	$O(\log n)$	$O(n\log(k))$
(Amanatidis et al., 2021)†	$0.172-\epsilon$	$O(\log n)$	$O(nk\log(n)\log(k))$
(Chen & Kuhnle, 2024) (AST)	$1/6-\epsilon \approx 0.166-\epsilon$	$O(\log n)$	$O(n\log(k))$
(Chen & Kuhnle, 2024) (ATG)	$0.193-\epsilon$	$O(\log(n)\log(k))$	$O(n\log(k))$
(Cui et al., 2025) (ParSKP1)	$1/8-\epsilon$	$O(\log n)$	$O(nk\log^2(n))$
(Cui et al., 2025) (ParSKP2)	$1/4-\epsilon$	$O(\log(n)\log(k))$	$O(nk\log^2(n))$
LinAst (this work)	$1/6-\epsilon$	$O(\log n)$	$O(n)$
LinAtg (this work)	$0.193-\epsilon$	$O(\log n)$	$O(n)$
BoostAdapt (this work)	$1/4-\epsilon$	$O(\log(n)\log(k))$	$O(n\log(k))$

(1) The high query complexity made the parallel algorithm in (Ene & Nguyen, 2020) impractical for real applications. (Chen & Kuhnle, 2024) demonstrated that (Ene & Nguyen, 2020)'s algorithm was of mostly theoretical interest because it needed $\Omega(nk^2\log^2(n))$ queries to approximate the multi-linear extension of a submodular function and its gradient. Experimentally, its running on tiny instances (e.g., $n < 100$) was already prohibitive as it required more than 10^9 queries to the set function on an instance with $n = 87$;

(2) The approximation ratios of the existing practical parallel algorithms (Fahrbach et al., 2019a; 2023; Kuhnle, 2021; Chen & Kuhnle, 2024; Cui et al., 2023) are still low as they have a significant gap in the approximation ratios compared to the best non-parallel algorithm for SMC (e.g. the ratio of 0.401 in (Buchbinder & Feldman, 2024)).

This leads us to two interesting questions when solving **SMMB**: Cac van de ma bai toan nay tap trung vao giai quyet
Q1: Can we improve the query complexity of a parallel algorithm?
Q2: Can we improve the approximation ratio of a practical parallel algorithm?
Our contributions and techniques. In this work, we tackle the above questions with the following contributions:

- First, we introduce LinAdapt, the first constant approximation ratio ($1/(12+O(\epsilon))$) within near-optimal adaptivity $O(\log n)$ and linear query complexity $O(n)$, where $\epsilon > 0$ is a constant parameter. LinAdapt is used as a building block to reduce the query complexity of our latter algorithms. The key idea to obtain such improvement lies in: **(1)** constructing two sets S, S' in $O(\log n)$ sequential rounds with an interesting property: $f(X \cup S) \leq O(1) \cdot f(S')$ with high probability for any set $X \subseteq V, |X| \leq k$, and S' with $|S'| \leq k$ is the set of last elements added into S; **(2)** combining **(1)** appropriately with the unconstrained submodular maximization algorithm to get the desired ratio.

- Second, we introduce two algorithms LinAst and LinAtg, which run $O(\log n)$ adaptivity and $O(n)$ query complexity and return the approximation ratios to $1/6-\epsilon$ and $0.193-\epsilon$, respectively. Therefore, our algorithm LinAtg improves the approximation ratio from $1/6-\epsilon$ to $0.193-\epsilon$ and significantly reduces the query complexity of the current best practical algorithm of (Chen & Kuhnle, 2024). Both LinAst and LinAtg use a common framework: first adapting LinAdapt as a subroutine to get the $O(1)$ number of guesses of the optimal, then using the adaptive simple threshold (Chen & Kuhnle, 2024) and iterated greedy framework (Gupta et al., 2010; Chen & Kuhnle, 2024) to get better theoretical bounds.

- Third, we further introduce BoostAdapt that provides a considerable approximation ratio of $1/4-\epsilon$ in $O(\log(n)\log(k))$ adaptive complexity and $O(n\log(k))$ query complexity. Thus, BoostAdapt significantly improves the ratio of the ATG (Chen & Kuhnle, 2024)(the algorithm with same adaptive and query complexity) from $0.193-\epsilon$ to $1/4-\epsilon$ and also reduces the query complexity of ParSKP2 (Cui et al., 2023) (the algorithm with same approximation ratio and adaptive complexity) by a factor of $\Omega(k\log^2(n)/\log(k))$. BoostAdapt follows a novel *staggered threshold* framework: updating alternately two disjoint sets X and Y only in $O(\log k)$ iterations by adapting a threshold sampling. It must be noted that the staggered threshold is different from the Iterated Greedy (Gupta et al., 2010), which uses two threshold greedy strategies to construct candidate solutions separately. Our algorithm also differs from the twin greedy strategy of (Han et al., 2020) and Interlaced Greedy of (Kuhnle, 2019), allowing simultaneously updating two disjoint sets in each iteration of the main loops.

- Finally, to show the consistency between theory and practice, we conducted several extensive experiments on two applications: Revenue Maximization and Maximum Weighted Cut. The results show that our algorithms not only produce better solution quality and query complexity sharply but also provide comparative adaptivity to state-of-the-art algorithms.

Paper Organization. The rest of this work is structured as follows. Section 2 provides the literature review on the studied problem. Notations are presented in Section 3. Sections 4-**??** introduce our algorithms and theoretical analysis. Experimental computation is provided in Section 6. Finally, Section 7 concludes this work.

2. Related Works

Regarding Submdoular Maximzation under Matroid constraint

Regarding Submodular Maximization under Knapsack constraint

Regarding Submodular Maximization under intersection of Matroid and Knapsack constraint

Regarding Submodular Maximization under k-matroid and d-knapsack constraint

Submodular Maximization under Matroid Constraints. Submodular maximization under matroid constraints has long been a cornerstone of combinatorial optimization, driven by its wide applicability in machine learning, influence maximization, and data summarization. Classical algorithms such as the greedy method (Fisher et al., 1978) guarantee a $\frac{1}{2}$ approximation for monotone objectives, while the continuous greedy algorithm (Calinescu et al., 2011) achieves the optimal approximation $1 - 1/e$, though with significantly higher oracle complexity.

To address scalability issues in large-scale settings, recent work has focused on minimizing the number of value oracle queries. A key advancement in this direction is the *QuickSwap* algorithm (Balkanski et al., 2024), which deterministically achieves a $1/4$-approximation for monotone submodular maximization under matroid constraints using exactly n queries—one per element. Unlike classical approaches that maintain feasible solutions throughout, QuickSwap evaluates marginal gains over infeasible supersets to significantly reduce query cost. The technique generalizes to *p-matchoid* constraints via QuickSwapPM, achieving a $1/(4p)$-approximation, and extends to non-monotone settings via QuickSwapNM, which achieves an approximation ratio of $\frac{1}{6+4\sqrt{2}}$ $(\sim 1/11.66)$.

using only $2n$ oracle queries. Empirically, QuickSwap outperforms classical methods such as threshold greedy (Badanidiyuru & Vondrák, 2014) and lazy greedy, requiring fewer function evaluations while maintaining high solution quality. In parallel, a complementary line of research has focused on *streaming submodular maximization under matroid constraints*, where elements arrive sequentially and memory is strictly limited. Chakrabarti and Kale (Chakrabarti & Kale, 2015) introduced the first single-pass streaming algorithm for monotone submodular functions under matroid constraints, achieving a $1/4$-approximation. Building on this, Feldman et al. (Feldman et al., 2022) proposed two advanced methods: (i) a single-pass semi-streaming algorithm that achieves a 0.3178-approximation using only $\tilde{\mathcal{O}}(k)$ memory, where k is the matroid rank; and (ii) a multi-pass algorithm that approaches the near-optimal $(1 - 1/e - \varepsilon)$ guarantee within $\mathcal{O}(1/\varepsilon^3)$ passes. Their single-pass approach leverages a novel *fractional swap* technique that generalizes classical element-swapping logic to fractional elements, allowing for smoother updates and stronger guarantees. Under random-order streams, the number of passes can be further reduced to $\mathcal{O}(\varepsilon^{-2} \log(1/\varepsilon))$. These results were also extended to p-matchoid constraints, offering $1/(p + 1 - \varepsilon)$ approximations with logarithmic adaptivity.

Submodular Maximization under Matroid and Knapsack Constraints. In many real-world scenarios, optimization problems are constrained by the intersection of multiple combinatorial structures—most notably matroid and knapsack constraints. Submodular maximization under such combined constraints has been extensively studied due to its importance in budgeted subset selection problems. Chekuri et al. (2010) provided some of the first theoretical guarantees in this setting, but their algorithms suffered from exponential time complexity. To address scalability, FANTOM (Mirzasoleiman et al., 2016b) introduced a density-threshold-based greedy algorithm that achieves a $(1 + \varepsilon)(2k + (2 + 2/k)m) + \mathcal{O}(1)$ approximation using $\tilde{\mathcal{O}}(n^2/\varepsilon)$ oracle calls. This direction was further advanced by DENSITYSEARCHSGS (Feldman et al., 2020), which employs a simultaneous greedy strategy to improve the approximation ratio to $(1 + \varepsilon)(k + 2m) + \mathcal{O}(\sqrt{m})$, while significantly reducing the oracle complexity to $\tilde{\mathcal{O}}(n/\varepsilon)$. These methods effectively target feasible sets constrained simultaneously by a single matroid and multiple knapsacks, offering a trade-off between solution quality and runtime performance.

More recent developments have addressed even more complex scenarios where the feasible region is defined by the intersection of multiple matroids and multiple knapsack constraints. To this end, Gu, Bian, and Qian (Gu et al., 2023) proposed SPROUT, which integrates partial enumeration with a simultaneous greedy framework. SPROUT achieves a $(1+\varepsilon)(k+m+3+2\sqrt{m+1})$-approximation using $\tilde{\mathcal{O}}(n^2/\varepsilon)$ oracle queries. For monotone objectives, the bound improves to $(1+\varepsilon)(k+m+1)$. To reduce computational overhead, the authors introduced SPROUT++, which replaces full enumeration with randomized selection and smoothed binary search, achieving $\tilde{\mathcal{O}}(\text{polylog}(n)\cdot n/\varepsilon)$ query complexity under mild assumptions on the optimal solution structure. These techniques enable scalable submodular maximization in time-sensitive applications such as influence maximization, video summarization, and recommendation systems.

Non-parallel Approximation Algorithms with Low Query Complexity. The first popular approach to solving the SMC problem in practice is to design an approximation algorithm with low query complexity. The greedy algorithm was an effective approach for submodular optimization. It sequentially selected elements with the largest marginal gain and explored the diminishing return property to get performance bounds. (Gupta et al., 2010) first developed an iterated greedy algorithm with $1/6$ ratio in $O(nk)$ queries for SMC. The work of (Lee et al., 2010) later improved the ratio to $1/4-\epsilon$ by developing a local search method but this wasted an expensive query complexity of $O(n^4\log n)$. Significantly, the elegant random greedy in (Buchbinder et al., 2014) could provide the ratio of $1/e$ with $O(nk)$ query complexity. Instead of selecting an element with the best marginal gain as the naive greedy, it chose a uniformly random element from the set of k elements with the largest marginal gain. (Buchbinder et al., 2015) then improved the query complexity to $O(n\log(1/\epsilon)/\epsilon^2)$ but still kept the same $1/e$ ratio. Recently, some works have attempted to improve the approximation factor to $0.385-\epsilon$, but they required a larger number of queries of $O(n^2)$ (Chen et al., 2024; Tukan et al., 2024). To the best of our knowledge, the best ratio for SMC was 0.401 by (Buchbinder & Feldman, 2024). However, this work used the method of multi-linear extensions and used a considerably high query complexity of $O(n^5)$ (Chen & Kuhnle, 2023). Finally, it must be emphasized that the above works weren't well parallelized due to their high adaptive complexities of $\Omega(n)$.

Parallel Algorithms with Low adaptive complexity. Research on parallel algorithms was initiated by (Balkanski & Singer, 2018). In this seminal work, they introduced the concept of *adaptive complexity or adaptivity* that measures the parallelizable of an algorithm, showed a lower bound on adaptive complexity of $O(\log(n)/\log(\log n))$ to achieve a constant ratio and devised a $1/3$-approximation algorithm in $O(\log n)$ adaptive rounds for the monotone SMC problem. Since then, many works have focused on devising algorithms with low adaptivity with tighter ratios for SMC (Balkanski et al., 2019; Fahrbach et al., 2019b; Ene et al., 2019; Chen et al., 2021). The best parallel algorithm for monotone SMC had an optimal ratio of $1-1/e-\epsilon$ in $O(\log n)$ adaptive complexity and $O(n)$ queries was due to (Chen et al., 2021). However, their performance bounds heavily relied on the monotone property and did not hold for non-monotone.

For the non-monotone SMC, (Chekuri & Quanrud, 2019; Balkanski et al., 2018) first developed parallelizable algorithms with $3-2\sqrt{2}-\epsilon$ and $1/(2e)-\epsilon$ ratios, respectively; both took $O(\log^2 n)$ adaptivity. (Ene & Nguyen, 2020) improved the ratio to $(1/e-\epsilon)$ in nearly optimal adaptive complexity of $O(\log n)$. However, due to the high query complexity of $\Omega(nk^2\log^2 n)$ to approximate the multilinear extension of a submodular function and its gradient, the (Ene & Nguyen, 2020)'s algorithm and algorithms based on multilinear extension methods, in general, are still impractical in some real applications (Kuhnle, 2021; Fahrbach et al., 2019a; Chen & Kuhnle, 2024; Cui et al., 2023). To attack the above issue, (Fahrbach et al., 2019a) first reduced the query complexity to near-linear of $O(n\log k)$ yet still kept the $O(\log n)$ adaptive complexity. However, their algorithm resulted in a small ratio, $0.039-\epsilon$. (Amanatidis et al., 2021) then tried to improve the ratio to 0.172 but their ratio may not hold due to the technique flaws that have been pointed (Cui et al., 2025). In subsequent work, (Kuhnle, 2021) boosted the ratio to $1/6-\epsilon$ in $O(\log n)$ adaptive rounds with $O(n\log k)$ queries and further improved the ratio to $0.193-\epsilon$ in $O(\log^2 n)$ adaptive rounds by exploring iterated greedy framework in (Gupta et al., 2010). It must be noted that the ratios of both (Fahrbach et al., 2019a) and (Kuhnle, 2021) may not hold because they adapt the threshold sampling routine in (Fahrbach et al., 2019b), which was pointed out disable with non-monotone functions (Chen & Kuhnle, 2024). Recently, (Chen & Kuhnle, 2024; Fahrbach et al., 2023) recovered the ratios of (Fahrbach et al., 2019a; Kuhnle, 2021) by revising threshold sampling routines with the respective analysis. More recent, (Cui et al., 2023; 2025) showed two algorithms with ratios of $1/8-\epsilon$ and $1/4-\epsilon$ with $O(\log n)$ and $O(\log^2 n)$ adaptive complexities, respectively. However, they took at least $\Omega(nk\log^2 n)$ query complexity.

3. Preliminaries

Given a ground set E includes m elements, and the set function $f: 2^E \mapsto \mathbb{R}^+$ is submodular iff it satisfies the diminishing return property, i.e., $f(A\cup\{e\})-f(A)\geq f(B\cup\{e\})-f(B)$ where $A\subseteq B\subseteq E$ and $e\notin B$. For convenience, from this point forward, we denote $f(A \cup \{e\})$ as $f(A \cup e)$, the marginal gain when adding an element e in to a set A, $f(A \cup \{e\}) - f(A)$ as $f(e|A)$, and the marginal gain of a set X with respect to a set A, $f(A \cup \{X\}) - f(A)$ as $f(X|A)$. We also assume f normalized, i.e., $f(\emptyset) = 0$.

A matroid $\mathfrak{M} = (E, \mathfrak{I})$ consists of a finite ground set E and a family of independent sets $\mathfrak{I} \subseteq 2^E$ with the following properties: (1) The empty set is independent: $\emptyset \in \mathfrak{I}$; (2) The hereditary or downward-closed: If $A \subseteq B \subseteq E$, if $B \in \mathfrak{I}$ then $A \in \mathfrak{I}$; (3) The augmentation or independent set exchange: for any $A \in \mathfrak{I}, B \in \mathfrak{I}, |A| < |B|$, it exists $x \in B \setminus A$ such that $A \cup x \in \mathfrak{I}$.

In this paper, we solve the problem of **S**ubmodular **M**aximization under **M**atroid and **B**udget constraint that is defined as follows:

Definition 3.1 (Submodular Maximization under Matroid and Budget constraints - SMMB)**.** Given a positive number B (budget constraint), a cost function $c : 2^E \mapsto \mathbb{R}^+$, the SMMB problem is to determine $S = \arg\max_{S \in \mathfrak{I}, c(S) = \sum_{e \in S} c(e) \leq B} f(S)$. We denote an instance of SMMB by a tuple $(f, E, \mathfrak{I}, B)$.

O is an optimal solution with the optimal value opt $= f(O)$. In this work, we assume that there exists an oracle query that returns $f(S)$ when queried for the set S with any $S \subseteq E$.

Due to the submodularity, the objective function also keeps the following properties: (1) The diminishing returns property that means for any $A \subseteq B \subseteq E$ and any $X \subseteq E \setminus B$, we have $f(X|A) \geq f(X|B)$; (2) The doubly submodular upper bound property that means for any $T \subseteq E$ and two disjoint subsets X, Y of E, we have: $f(T) \leq f(T \cup X) + f(T \cup Y)$; (3) A consequence of diminishing returns property that means for any disjoint subsets $X_1, X_2, \ldots, X_t$ of $B \setminus A$ with any $A, B \subseteq E$, we have: $f(B|A) = \sum_{i=1}^{t} f(X_i | X_1 \cup X_2 \cup \ldots \cup X_{i-1} \cup A) \leq \sum_{i=1}^{t} f(X_i|A)$.

We also define $[k] = \{1, 2, \ldots, k\}$ for any integer k, and every equation is shorten by Eq.. r is denoted the *rank* of $\mathfrak{M}$, i.e., $r = \max\{|S| : S \in \mathfrak{I}\}$.

4. TwinGreedy2$(f, E, \mathfrak{I})$ an extension of TwinGreedy algorithm of Han et.al to solve the SMMB problem

In this section, we first introduce TwinGreedy (Algorithm 1) for Submodular Maximization under Matroid constraint proposed by (Han et al., 2020).

Algorithm 1 TwinGreedy$(f, E, \mathfrak{I})$

```
 1: Input: $f, E, \mathfrak{I}$.
 2: $S_1 \leftarrow \emptyset; S_2 \leftarrow \emptyset$;
 3: repeat
 4:   $\mathfrak{M}_1 \leftarrow \{e \in E \setminus (S_1 \cup S_2) : S_1 \cup e \in \mathfrak{I}\}$
 5:   $\mathfrak{M}_2 \leftarrow \{e \in E \setminus (S_1 \cup S_2) : S_2 \cup e \in \mathfrak{I}\}$
 6:   $C \leftarrow \{j | j \in \{1, 2\}$and $\mathfrak{M}_j \neq \emptyset\}$
 7:   If $C \neq \emptyset$ then
 8:     $(i, e) \leftarrow \arg\max_{j \in C, u \in \mathfrak{M}_j} f(u|S_j)$
 9:     If $f(e|S_i) \leq 0$ then break;
10:     $S_i \leftarrow S_i \cup e$
11: until $\mathfrak{M}_1 \cup \mathfrak{M}2 = \emptyset$
12: Return $S^* \leftarrow \arg\max_{S \in \{S_1, S_2\}} f(S)$
```

TwinGreedy's idea is on the competition between elements in O and those in $S_1 \cup S_2$, such that the marginal gains of elements in O with respect to S_1 and S_2 can be upper-bouneded by S_1 and S_2's own objective function values. This is due to the correlation between the elements in S_1 and S_2.

Inspired by TwinGreedy, we introduce TwinGreedy2 (Algorithm 2) for Submodular Maximization under Matroid and Knapsack constraints problem. It must be emphasized that it's not easy to directly apply the idea of them for the **SMMB** problem due to the hardness of the knapsack constraint. The knapsack constraint makes the solution's length unpredict. The combination of matroid constraint and knapsack constraint improves the challenge of the problem that requires a careful analysis to bound the optimal range. The detailed of TwinGreedy2 is described as follows:

Algorithm 2 TwinGreedy2$(f, E, \mathfrak{I})$

```
1: Input: $(f, E, \mathfrak{I})$, budget $B > 0$.
2: $S_1 \leftarrow \emptyset; S_2 \leftarrow \emptyset;$
3: $E_1 = \{e \in E | c(e) > B/2\}; E_2 = \{e \in E | c(e) \leq B/2\}$
4: $e_m \leftarrow \arg\max_{e \in E_1} f(e)$
5: repeat
6:     $\mathfrak{M}_1 \leftarrow \{e \in E_2 \setminus (S_1 \cup S_2) : S_1 \cup e \in \mathfrak{I}\}$
7:     $\mathfrak{M}_2 \leftarrow \{e \in E_2 \setminus (S_1 \cup S_2) : S_2 \cup e \in \mathfrak{I}\}$
8:     $C \leftarrow \{i | i \in \{1, 2\}$and $\mathfrak{M}_i \neq \emptyset\}$
9:     if $C \neq \emptyset$ then
10:        $\sigma_i = \arg\max_{i \in C, S_i \cup e \in \mathfrak{M}_i} f(e|S_i)$
11:        if $\sigma_i \geq \frac{c(e)}{B} f(S_i)$ then
12:            $S_i = S_i \cup e$
13:        end if
14:    end if
15: until $\mathfrak{M}_1 \cup \mathfrak{M}2 = \emptyset$
16: $X' \leftarrow \arg\max_{S_1^j : j \leq t, c(S_1^j) \leq B} c(S_1^j)$, where $t = |S_1^j|$ and $S_1^j = \{e_{t-j+1}, e_{t-j+2}, \ldots, e_t\}$ is the last $j$ elements added into $S_1$.
17: $Y' \leftarrow \arg\max_{S_2^j : j \leq t, c(S_2^j) \leq B} c(S_2^j)$, where $t = |S_2^j|$ and $S_2^j = \{e_{t-j+1}, e_{t-j+2}, \ldots, e_t\}$ is the last $j$ elements added into $S_2$.
18: $S^* \leftarrow \arg\max_{S \in \{e_m, X', Y'\}} f(S)$
19: Return $S^*$
```

Due to the budget of B, we divide the ground set into 2 subsets, in which the first one includes every element that has a cost larger than $B/2$, and the second includes the remaining. This division is to get the optimal solution if it exists in E_1 that is $\{e_m\}$ and get the linear optimal solution in E_2 that is X' or Y'. And the best one among them will be chosen as the final solution. That is to maintain the knapsack constraint. Moreover, $\mathfrak{M}_1$ and $\mathfrak{M}_2$ are two sub-matroids of $\mathfrak{M}$. Therefore, every element in these matroids is inside matroid $\mathfrak{M}$. Therefore, X' and Y' are also in matroid $\mathfrak{M}$. Similarly, e_m is in matroid $\mathfrak{M}$. Finally, S^* is the final solution to the SMMB problem.

Next, we must indicate the solution quality obtained from TwinGreedy2. We first define the notations as follows:

- $E_1 = \{e \in E : c(e) > B/2\}, E_2 = \{e \in E : c(e) \leq B/2\}$.
- O is an optimal solution of the problem over E and the optimal value $\mathsf{opt} = f(O)$.
- $O_1' = \{e | e \in E_1\}, O_2' = \{e | e \in E_2\}$.
- O_1 is an optimal solution of the problem over E_1.
- O_2 is an optimal solution of the problem over E_2.
- $e^{(j)}$ is the j-th element added to S_i at the iteration j and $S_i^{(j)}$ is S_i at the end of the iteration j, respectively.

Due to the submodularity and $O_1 \cap O_2 = \emptyset$, we have:

$$f(O_1) + f(O2) \geq f(O_1 \cup O_2) + f(O_1 \cap O_2) \geq f(O) \tag{1}$$

$$f(O_2 \cup S_1) + f(O_2 \cup S_2) \geq f(O_2) \tag{2}$$

Moreover, we also have:

$$f(O_1) \leq f(e_m) \leq f(S^*) \tag{3}$$

Hence, we must bound the optimal range in E_2. At iteration j, supposing the element $e^{(j)}$ was added to the set S_1, it wasn't added to the set S_2. It means $\sigma_2 \leq \sigma_1$ at that time. Moreover, due to the submodularity, $f(e^{(j)}|S_2) \leq f(e^{(j)}|S_2^{(j)})$. Therefore:

$$\frac{f(e^{(j)}|S_2)}{c(e^{(j)})} \leq \frac{f(e^{(j)}|S_2^{(j)})}{c(e^{(j)})} \leq \frac{f(e^{(j)}|S_1^{(j)})}{c(e^{(j)})} \leq \frac{f(e^{(j)}|S_1^{(j-1)})}{c(e^{(j)})} \tag{4}$$

We also have:

$$\begin{aligned}
f(O2|S_2) = f(O_2 \cup S_2) - f(S_2) &\leq \sum_{e \in O_2 \setminus S_2} f(e|S_2) && (5)\\
&= \sum_{e \in O_2 \setminus (S_1 \cup S_2)} f(e|S_2) + \sum_{e \in O_2 \cap S_1} f(e|S_2) && (6)\\
&\leq \sum_{e \in O_2 \setminus (S_1 \cup S_2)} \frac{c(e)}{B} f(S_2) + \sum_{e \in O_2 \cap S_1} \frac{c(e)}{c(e)} f(e|S_2) && (7)\\
&\leq \sum_{e \in O_2 \setminus (S_1 \cup S_2)} \frac{c(e)}{B} f(S_2) + \sum_{e \in O_2 \cap S_1} c(e) \frac{f(e|S_1^{(|S_1|-1)})}{c(e)} && (8)\\
&\leq f(S_1) + f(S_2) && (9)
\end{aligned}$$

where, Eq. 5 is due to the submodularity; Eq. 7 is due to $e \in O_2 \setminus (S_1 \cup S_2)$ mean e doesn't meet the condition in the Algorithm 2; Eq. 8 is from Eq. 4; and Eq. 9 is due to e in S_1 mean e doesn't meet the condition to be added to S_2. With the similarity between S_1 and S_2, and they are in a matroid, we have:

$$f(O_2|S_1) + f(O_2|S_2) \leq 2(f(S_1) + f(S_2)) \tag{10}$$

$$f(O_2) \leq f(O_2 \cup S_1) + f(O_2 \cup S_2) \leq 3(f(S_1) + f(S_2)) \tag{11}$$

Now, we must find out the relation between S_1, S_2 and S^*. Supposing $S'^* = \arg\max_{S'^* \in \{S_1, S_2\}} f(S'^*)$. From Eq.3, we have: $f(O_2) \leq 3(f(S_1) + f(S_2) \leq 6f(S'^*)$. So what about the final?

We now prove the approximation factor of the algorithm. Denote $S_1 = \{s_1, s_2, \ldots, s_{|S_1|}\}, S_1^i = \{s_1, s_2, \ldots, s_i\}, 1 \leq i \leq |S_1|, S_1^0 = \emptyset$. Suppose $X_1 = S_1 \setminus X' = \{s_1, \ldots, s_l\}$ and $X' = \{s_{l+1}, s_{l+2}, \ldots, s_{|S_1|}\}$. Suppose the same with S_2.

Lemma 4.1. $f(X_1) \leq 2f(S_1)/3$. *Similarly,* $f(X_2) \leq 2f(S_2)/3$

Proof. If $c(S_1) < B$, $X' = S_1$ and $X_1 = \emptyset$, the lemma holds. Now we consider the case $c(S_1) \geq B$. The selection rule of elements gives:

$$f(S_1^i) = f(S_1^{i-1}) + f(s_i|S_1^{i-1}) \geq f(S_1^{i-1}) + \frac{c(s_i)}{B} f(S_1^{i-1}) \geq f(S_1^{i-1})$$

for $1 \leq i \leq |S_1|$. Therefore:

$$\begin{aligned}
f(S_1) - f(X_1) &= \sum_{i=1}^{|X'|} f(s_{l+i}|X_1 \cup \{s_{l+1}, \ldots, s_{l+i-1}\})\\
&\geq \sum_{i=1}^{|X'|} \frac{c(s_{l+i})}{B} f(X_1 \cup \{s_{l+1}, \ldots, s_{l+i-1}\})\\
&\geq \sum_{i=1}^{|X'|} \frac{c(s_{l+i})}{B} f(X_1) = \frac{c(X')}{B} f(X_1)\\
&\geq \frac{f(X_1)}{2}.
\end{aligned}$$

The last inequality is because each element in E_2 has the cost at most $B/2$, and the selection rule of X' we have

$$B \geq c(X') \geq B - B/2 = B/2.$$

Thus, $f(X_1) \leq 2f(S_1)/3$. By the submodularity of f, we obtain:

$$f(X') \geq f(S_1) - f(X_1) \geq \frac{f(S_1)}{3}.$$

Similarly, we have: $f(Y') \geq f(S_2)/3$. The proof is completed. □

Theorem 4.2. *The algorithm 2 approximates* $1/19$ *in at most* $3n$ *queries.*

Proof. From Lemma 4.1 and $f(O_2) \leq f(e_{max})$, we have:

$$\begin{aligned} f(O) &\leq f(O \cap V_1) + f(O \cap V_2) \\ &\leq f(O_1) + f(O_2) \\ &\leq 3(f(S_1) + f(S_2)) + f(e_{max}) \\ &\leq 9(f(X') + f(Y')) + f(e_{max}) \leq 19 f(S^*) \end{aligned}$$

The algorithm first scans once over E to find e_{max} (line 4), Algorithm 3), this task takes n queries. It then scans twice over $E_2 \subseteq E$ in the main loop (lines 6-7) to produce two new matroids $\mathfrak{M}_1$ and $\mathfrak{M}_2$. For each $e \in E_2$, it finds two incremental gain $f(e|S_1) = f(S_1 \cup e) - f(S_1)$ and $f(e|S_2) = f(S_2 \cup e) - f(S_2)$ to find out σ_i. $f(S_1), f(S_2)$ are obviously evaluated at the previous iterations; they can be stored in temporary memory; thus, we need only two queries for this task. Therefore, the main loop needs $2|E_2| \leq 2n$ queries and the algorithm needs at most $3n$ queries. □

5. An improvement of TwinGreedy2, algorithm ITW

We now introduce our ITW (Algorithm 3), an **I**mprovement from **TW**inGreedy2 that provides the solution with an approximation factor of $6 + \epsilon$.

An important comment when designing an approximation algorithm to maximize a submodular objective function is when we choose a solution that is good enough for the problem; other good elements still exist since we have never chosen them. Therefore, the quality of the solution can be enhanced by re-constructing the candidate solution and adding new elements that keep the solution in the matroid. The solution's total cost at that time still doesn't exceed the budget.

The key strategy is combining the properties of two disjoint sets with a greedy threshold to construct several candidate solutions to analyze the theory of the non-monotone objective function. ITW takes an instance $(f, E, \mathfrak{I}, B)$ and a parameter ϵ as inputs. It consists of two phases. First, it bounds the optimal range by calling the procedure TwinGreedy2 at lines 3. One feasible solution S_b will be returned. Also, the algorithm gets an approximate range of optimal value of $[M, 19M]$ where $M = f(S_b)$ (line 1). Moreover, the iterative decreasing greedy threshold built on density gains is set up to filter out good elements for two disjoint sets S_1 and S_2. This phase consists of multiple iterations; each scans the ground set twice to observe a new element. An element added to the candidates S_1 or S_2 depends on which one has higher density gain without violating the budget constraint, as long as the density gain is at least ρ. The range of ρ is initiated to $19\Gamma/(6\epsilon')$ and decreases by a factor of $(1-\epsilon')$ after each iteration until less than to $\Gamma(1-\epsilon')/(6B)$, where $\epsilon' = \epsilon/14$.

The second phase (the for loop of l) is to reconstruct and then enhance the quality of candidate solutions $\{S_1, S_2\}$ we get from phase 1. For convenience, denote S_i^j as a set of the first j^{th} elements added in S_i in phase 1. S_i is S_1 or S_2.

An important observation is the performance of ITW depends on the cost of S' that is $S' = \arg\max_{S^i : c(S^i) \leq B - c^*} c(S^i)$ with $o^* = \arg\max_{o \in O} c(o), c^* = c(o^*) \leq B$. We scan an upper bound of $c(S')$ from $\epsilon' B$ to B and improve the quality of S' by adding a new element $e = \arg\max_{e \in E \setminus S : c(S' \cup e) \leq B} f(S' \cup e)$ (lines 20-25).

In the algorithm, we add to the candidates every element that makes any candidate still in the matroid $\mathfrak{M}$. Therefore, S_1, S_2 at the end of phase 1 are in $\mathfrak{M}$. Phase 2 is reconstructing S_1, S_2 and adding new good ones that still make new candidates in $\mathfrak{M}$. Therefore, the last solution S^* is definitely in matroid $\mathfrak{M}$. Consequently, the theory guarantee of the algorithm is based on the upper bound of the budget (when the final solution exceeds the budget).

In this section, we use the following notations.

- $S \in \{S_1, S_2\}, S = \{s_1, s_2, \ldots s_{|S|}\}$ we denote $S^j = \{s_1, s_2, \ldots s_j\}$, and $t = \max\{j : c(S^j) + c^* \leq B\}$.

Algorithm 3 ITW Algorithm

```
1:  **Input:** An instance $(f, E, \mathfrak{I}, B)$, $\epsilon$
2:  **Output:** A solution $S$
3:  $S_b \leftarrow \mathsf{TwinGreedy2}(f, E, \mathfrak{I}, B), M \leftarrow f(S_b), \epsilon' \leftarrow \frac{\epsilon}{14}$
4:  $L \leftarrow \lceil \frac{\log(1/\epsilon')}{\epsilon'} \rceil, S_1, S_2 \leftarrow \emptyset, \rho \leftarrow 19M/(6\epsilon' B)$
5:  **while** $\rho \geq M(1-\epsilon')/(6B)$ **do**
6:      $\mathfrak{M}_1 \leftarrow \{e \in E \setminus (S_1 \cup S_2) : S_1 \cup e \in \mathfrak{I}\}$
7:      $\mathfrak{M}_2 \leftarrow \{e \in E \setminus (S_1 \cup S_2) : S_2 \cup e \in \mathfrak{I}\}$
8:      **for** each $e \in \mathfrak{M}_1 \cup \mathfrak{M}_2$ **do**
9:          $C \leftarrow \{i | i \in \{1, 2\}$and $\mathfrak{M}_i \neq \emptyset\}$
10:         **if** $C \neq \emptyset$ **then**
11:             $\sigma_i = \arg\max_{i \in C, S_i \cup e \in \mathfrak{M}_i} f(e|S_i)$
12:             **if** $\frac{\sigma_i}{c(e)} \geq \rho$ and $c(S_i \cup e) \leq B$ **then**
13:                 $S_i = S_i \cup e$
14:             **end if**
15:         **end if**
16:     **end for**
17:     $\rho \leftarrow (1-\epsilon')\rho$
18: **end while**
19: **for** $l = 0$ to $L$ **do**
20:     $X'_{(l)} \leftarrow \arg\max_{S_1^j : c(S_1^j) \leq \epsilon' B(1+\epsilon')^l, j \leq |S_1|} c(S_1^j)$
21:     $Y'_{(l)} \leftarrow \arg\max_{S_2^j : c(S_2^j) \leq \epsilon' B(1+\epsilon')^l, j \leq |S_2|} i$
22:     $e_X \leftarrow \arg\max_{e \in E \setminus S_1 : c(X'_{(l)} \cup e) \leq B, X'_{(l)} \cup e \in \mathfrak{M}} f(X'_{(l)} \cup e)$
23:     $e_Y \leftarrow \arg\max_{e \in E \setminus S_2 : c(Y'_{(l)} \cup e) \leq B, Y'_{(l)} \cup e \in \mathfrak{M}} f(Y'_{(l)} \cup e)$
24:     $X_{(l)} \leftarrow X'_{(l)} \cup e_X$,
25:     $Y_{(l)} \leftarrow Y'_{(l)} \cup e_Y$
26: **end for**
27: $S^* \leftarrow \arg\max_{T \in \{S', S_1, S_2, X_{(0)}, .., X_{(L)}, Y_{(0)}, .., Y_{(L)}\}} f(T)$
28: Return $S^*$.
```

- $S_1^{(j)}$, $S_2^{(j)}$ (or $S^{(j)}$) are S_1, S_2 (or S) after the iteration j of the first loop of the Algorithm 3, respectively.
- For $e \in S_1 \cup S_2$, we denote by $S_1^{<e}$ and $S_2^{<e}$ the set of elements in S_1 and S_2 before adding e into S_1 or S_2, respectively.
- Denote by ρ_i as ρ at the iteration i, by $\rho_{(j)}$ as ρ when the element s_j is added into S_1, and ρ_{last} is ρ at the last iteration of the first loop.
- Subsets $X' \subseteq S_1, Y' \subseteq S_2$ are symbolized by $S' \subseteq S$. It means when $S' \subseteq S$ is mentioned, it can be either $X' \subseteq S_1$ or $Y' \subseteq S_2$.

The following Lemmas give the bounds of the final solution when $c^* < (1-\epsilon')B$ and $c^* \geq (1-\epsilon')B$, respectively.

Lemma 5.1. *If $c^* < (1-\epsilon')B$, one of two things happens:*

a) $f(S^*) \geq \frac{1}{6(1+\epsilon')}\mathsf{opt}$*;*

b) *There exists a subset $S' \subseteq S$ so that $f(O \cup S') \leq 2f(S^*) + \max\{\frac{1+\epsilon'}{1-\epsilon'}f(S^*), \frac{(1-\epsilon')}{6}\mathsf{opt}\}$.*

Proof of Lemma 5.1. In this case we have $B - c^* > \epsilon' B$. We consider the following cases:
Case 1. If S_1^t is S_1 after ending the first loop. By the submodularity and the selection rule of the algorithm, each element $e \in S_2^{<s_t}$ has the density gain satisfying:

$$\frac{f(e|S_1^t)}{c(e)} \leq \frac{f(e|S_1^{<e})}{c(e)} \leq \frac{f(e|S_2^{<e})}{c(e)}. \tag{12}$$

Each element $e \in O \setminus (S_1^t \cup S_2^{<s_t})$ having the density gain with S_1^t is less than ρ_{last}, i.e., $\frac{f(e|S_1^t)}{c(e)} < \rho_{last}$, so we get:

$$\begin{aligned}
f(S_1^t \cup O) - f(S_1^t) &\leq \sum_{e \in O \setminus S_1^t} f(e|S_1) && (13)\\
&= \sum_{e \in O \cap S_2^{<s_t}} f(e|S_1^t) + \sum_{e \in O \setminus (S_1^t \cup S_2^{<s_t})} f(e|S_1) && (14)\\
&\leq \sum_{e \in O \cap S_2^{<s_t}} \frac{f(e|S_1^t)}{c(e)} c(e) + \sum_{e \in O \setminus (S_1^t \cup S_2^{<s_t})} c(e)\rho_{last} && (15)\\
&< \sum_{e \in O \cap S_2^{<s_t}} \frac{f(e|S_2^{<e})}{c(e)} c(e) + c(O)\rho_{last} && (16)\\
&\leq f(S_2^{<s_t}) + \frac{(1-\epsilon')\mathsf{opt}}{6}, && (17)
\end{aligned}$$

where Eq. (16) is due to Eq. 12 and total cost of all elements in $O \setminus S_1^t \cup S_2^{<s_t}$ can not exceed $c(O)$, and the Eq. (17) is due to the fact that $f(e|S_2^{<e})/c(e) \geq \rho > 0$ for all $e \in S_2$, $c(O) \leq B$, then $c(O)\rho_{last} \leq B\rho_{last} \leq B.M\frac{1-\epsilon'}{6B} \leq \frac{(1-\epsilon')\mathsf{opt}}{6}$. From (17) and we realize $f(S_1^t) \leq f(S_1) \leq f(S^*)$ and $f(S_2^{<s_t}) \leq f(S_2) \leq f(S^*)$ we get:

$$f(S_1^t \cup O) < f(S_1^t) + f(S_2^{<s_t}) + \frac{(1-\epsilon')\mathsf{opt}}{6} \leq 2f(S^*) + \frac{(1-\epsilon')\mathsf{opt}}{6}.$$

Case 2. $S_1^t \subset S_1$ after ending the first loop. In this case, S_1 contains at least $t+1$ elements and $c(S_1^{t+1}) > B - c^*\epsilon' B$. We consider the second loop of the Algorithm 3. Since $\epsilon' B < B - c^* \leq B$, there exists an integer number l that

$$L = (1+\epsilon')^l \epsilon' B \leq B - c^* < (1+\epsilon')^{l+1}\epsilon' B = L(1+\epsilon').$$

Assuming that $X'_{(l)} = S_1^j$ for some j. By the selection of $X'_{(l)}$ in Line 11 of Algorithm 3 and $c(S_1) \geq c(S_1^{t+1}) > \epsilon' B$, we have $c(S_1^j) \leq L < c(S_1^{j+1})$, and thus:

$$c(S_1^{j+1}) > L > \frac{B - c^*}{1+\epsilon'} \geq \frac{\epsilon' B}{1+\epsilon'}. \tag{18}$$

We further consider two following sub-cases:
Case 2.1. If the algorithm obtains S_1^{j+1} at the first iteration of the first loop, we get:

$$f(S^*) \geq f(S_1^{j+1}) \geq c(S_1^{j+1})\rho_1 > \frac{\epsilon' B}{1+\epsilon'}\frac{19M}{6\epsilon' B} \geq \frac{\mathsf{opt}}{6(1+\epsilon')},$$

which implies the Lemma holds.
Case 2.2. If the algorithm obtains S_1^{j+1} at the iteration $j \geq 2$ of the first loop. For any element $e \in E \setminus (S_1^j \cup S_2^{<s_j})$, its density gain for S_1^j is smaller than the threshold at the previous iteration (in the first loop), i.e.,

$$\frac{f(e|S_1^j)}{c(e)} < \frac{\rho_{(j+1)}}{1-\epsilon'}. \tag{19}$$

On the other hand, the density gain of each element in S_1^{j+1} is greater than or equal to the threshold $\rho_{(j+1)}$, so we get:

$$\frac{f(S_1^{j+1})}{c(S_1^{j+1})} = \frac{\sum_{k=1}^{j+1} f(s_k|S_1^{k-1})}{c(S_1^{j+1})} \geq \frac{\sum_{k=1}^{j+1} \rho_{(j+1)}c(s_k)}{c(S_1^{j+1})} = \rho_{(j+1)}. \tag{20}$$

We denote $O_1 = O \cap S_2^{<s_j}$ and $O_2 = O \setminus (S_1^j \cup O_1)$. By combining the inequalities (18), (19), and (20), we get:

$$\begin{aligned} f(S_1^j \cup O) - f(S_1^j \cup o^*) &\leq \sum_{e \in O \setminus S_1^j} f(e|S_1^j \cup o^*) \leq \sum_{e \in O \setminus (S_1^j \cup o^*)} f(e|S_1^j) \\ &= \sum_{e \in O_1 \setminus o^*} f(e|S_1^j) + \sum_{e \in O_2 \setminus o^*} f(e|S_1^j) \\ &< \sum_{e \in O_1 \setminus e} f(e|S_2^{<e}) + \sum_{e \in O_2 \setminus o^*} c(e)\frac{\rho_{(j+1)}}{1-\epsilon'} \quad \text{(Due to (19))} \\ &\leq f(S_2) + (B - c^*)\frac{\rho_{(j+1)}}{1-\epsilon'} \\ &\leq f(S_2) + \frac{(B-c^*)f(S_1^{j+1})}{(1-\epsilon')c(S_1^{j+1})} \quad \text{(Due to (20))} \\ &< f(S_2) + \frac{1+\epsilon'}{1-\epsilon'}f(S_1^{j+1}) \quad \text{(Due to (18))} \end{aligned}$$

which implies that

$$f(S_1^j \cup O) < f(S_1^j \cup o^*) + f(S_2) + \frac{1+\epsilon'}{1-\epsilon'}f(S_1^{j+1}).$$

By the selection rule of e_X in line 3 of Algorithm 3, $f(S_1^j \cup o^*) \leq f(S_1^j \cup e_X) \leq f(S^*)$, so we have:

$$f(S_1^j \cup O) \leq 2f(S^*) + \frac{1+\epsilon'}{1-\epsilon'}f(S^*).$$

By combining two cases, we get the result in the Lemma. By the similarity argument, we also have the same result to a subset $Y' \subseteq S_2$. □

Lemma 5.2. *If $c^* \geq (1-\epsilon')B$, one of two things happens:*

c) *$f(S^*) \geq \frac{(1-\epsilon')^2}{6}\mathsf{opt}$;*

d) *With every $S \in \{S_1, S_2\}$, $S' \in \{X' \subseteq S_1, Y' \subseteq S_2\}$, there exists $S' \subseteq S$ so that $f(O \cup S') \leq 2f(S^*) + \max\{\frac{(1-\epsilon')\mathsf{opt}}{6}, f(S^*) + \frac{\epsilon'\mathsf{opt}}{6}\}$.*

Proof of Lemma 5.2. In this case, we have $c(O \setminus \{c^*\}) \leq \epsilon' B, c(S^t) \leq \epsilon' B$.
Case 1. If S^t is S after ending the first loop. By the similar transform from (13) to (17) of the proof of Lemma 5.1, we also get

$$f(S^t \cup O) < 2f(S^*) + \frac{(1-\epsilon')\mathsf{opt}}{6}.$$

Case 2. If $S^t \subset S$, S contains at least $t+1$ elements. Consider the first loop of the algorithm, $\rho_j = \frac{19M(1-\epsilon')^j}{6\epsilon' B} \in [\frac{(1-\epsilon')\mathsf{opt}}{6B}, \frac{19\mathsf{opt}}{6\epsilon' B}]$ for any iteration j. Since the threshold ρ alliteratively decreasing with a factor of $1-\epsilon'$ after each iteration, there exist an iteration j satisfying

$$\frac{(1-\epsilon')\mathsf{opt}}{6B} \leq \rho_j = \frac{19M(1-\epsilon')^j}{6\epsilon' B} < \frac{\mathsf{opt}}{6B}.$$

We further consider two following sub-cases:
- If $S^{t+1} \subseteq S^{(j)}$. If $c(S^{(j)}) \geq (1-\epsilon')B$, then

$$f(S^*) \geq f(S^{(j)}) \geq c(S^{(j)})\rho_j \geq \frac{(1-\epsilon')^2}{6}\mathsf{opt}.$$

If $c(S^{(j)}) < (1-\epsilon')B$. Suppose S is S_1, denote $O_1 = O \cap S_2^{(j)}$ and $O_2 = O \setminus (S_1^{(j)} \cup O_1)$. Since $c(S^{(j)}) + \max_{e\in O\setminus\{o^*\}} c(e) \leq c(S^{(j)}) + c(O\setminus\{o^*\}) < B$, we get $\frac{f(e|S^{(j)})}{c(e)} < \rho_j$ for any $e \in O_2 \setminus \{o^*\}$. Therefore:

$$\begin{aligned}
f(S^{(j)} \cup O) - f(S^{(j)} \cup o^*) &\leq \sum_{e\in O\setminus\{o^*\}} f(e|S^{(j)}) \\
&= \sum_{e\in O_1\setminus\{o^*\}} f(e|S^{(j)}) + \sum_{e\in O_2\setminus\{o^*\}} f(e|S^{(j)}) \\
&< \sum_{e\in O_1\setminus\{o^*\}} f(e|S_2^{<e}) + \sum_{e\in O_2\setminus\{o^*\}} c(e)\rho_j \\
&\leq f(S_2) + \epsilon' B\frac{\mathsf{opt}}{6B} = f(S_2) + \frac{\epsilon'}{6}\mathsf{opt}.
\end{aligned}$$

By the selection rule of the final solution and notice that $f(o^*) \leq f(e_{max}) \leq f(S') \leq f(S^*)$, we obtain

$$\begin{aligned}
f(S^{(j)} \cup O) &\leq f(S^{(j)} \cup o^*) + f(S_2) + \frac{\epsilon'\mathsf{opt}}{6} \\
&\leq f(S^{(j)}) + f(o^*) + f(S_2) + \frac{\epsilon'}{6}\mathsf{opt} \\
&\leq 3f(S^*) + \frac{\epsilon'}{6}\mathsf{opt}.
\end{aligned}$$

- If $S^{(j)} \subset S^{t+1}$. For any element $e \in E \setminus (S^t \cup S_2^{<s_t})$, its density gain to S^t is smaller than the threshold at the previous iteration (in the first loop), thus

$$\frac{f(e|S^t)}{c(e)} < \frac{\rho_{(t+1)}}{1-\epsilon'} \leq \rho_j < \frac{\mathsf{opt}}{6B}. \tag{21}$$

Denote $O_1 = O \cap S_2^{<s_t}$ and $O_2 = O \setminus (S^t \cup O_1)$. With notice that $c(O_2 \setminus \{o^*\}) < \epsilon' B$, we get

$$\begin{aligned}
f(S^t \cup O) - f(S^t \cup o^*) &\leq \sum_{e\in O\setminus\{o^*\}} f(e|S^t \cup o^*) \leq \sum_{e\in O\setminus\{o^*\}} f(e|S^t) \\
&= \sum_{e\in O_1\setminus\{o^*\}} f(e|S^t) + \sum_{e\in O_2\setminus\{o^*\}} f(e|S^t) \\
&< \sum_{e\in O_1\setminus\{o^*\}} f(e|S_2^{<e}) + \sum_{e\in O_2\setminus\{o^*\}} c(e)\rho_j \\
&\leq f(S_2^{<s_t}) + c(O_2\setminus\{o^*\})\frac{\mathsf{opt}}{6B} \\
&\leq f(S_2) + \frac{\epsilon'\mathsf{opt}}{6}.
\end{aligned}$$

which implies that

$$\begin{aligned} f(S^t \cup O) &\leq f(S^t \cup o^*) + f(S_2) + \frac{\epsilon'}{6}\mathsf{opt} \\ &\leq f(S^t) + f(o^*) + f(S_2) + \frac{\epsilon'}{6}\mathsf{opt} \\ &\leq 3f(S^*) + \frac{\epsilon'}{6}\mathsf{opt}. \end{aligned}$$

By combining two cases, we get the result in the Lemma. By the similarity argument, we also have the same result concerning $Y' \subseteq S_2$. □

Theorem 5.3. *For any* $\epsilon \in (0, 1)$*,* ITW *is a deterministic algorithm that has a query complexity* $O(n \log(1/\epsilon)/\epsilon)$ *and returns an approximation factor of* $6 + \epsilon$*.*

Proof. The query complexity of Algorithm 3 is obtained by combining the operation of Algorithm 2 and two main loops of Algorithm 3. The first and the second loops contain at most $\lceil \log(19/\epsilon')/\epsilon' \rceil + 1$ and $\lceil \log(1/\epsilon')/\epsilon' \rceil$ iterations, respectively. Each iteration of these loops takes $O(n)$ queries; thus we get the total number of queries at most:

$$3n + 2n(\lceil \frac{1}{\epsilon'} \log(\frac{19}{\epsilon'}) \rceil + 1) + n\lceil \frac{1}{\epsilon'} \log(\frac{1}{\epsilon'}) \rceil = O(\frac{n}{\epsilon} \log(\frac{1}{\epsilon})).$$

To prove the factor, we consider two following cases:
Case 1. If $c^* \geq (1 - \epsilon')B$. By using Lemma 5.2, we consider two cases:

If **a)** happens. Since $\epsilon' = \frac{\epsilon}{14} < \frac{1}{14}$ we get: $\mathsf{opt} \leq \frac{6f(S^*)}{(1-\epsilon')^2} \leq 6(1 + \frac{14}{13}\epsilon')^2 f(S) < (6 + \epsilon)f(S)$, the Theorem holds. We consider the otherwise case **c)** happens. There exist $X' \subseteq S_1$, $Y' \subseteq S_2$ and $X' \cap Y' = \emptyset$ satisfying: $\mathsf{opt} = f(O) \leq f(O \cup X') + f(O \cup Y')$

$$\leq 4f(S^* + 2\max\{(1 - \epsilon')\mathsf{opt}/6, f(S^*) + \epsilon'\mathsf{opt}/6\}. \quad (22)$$

We consider two sub-cases: If $f(S^*) \geq \frac{\mathsf{opt}}{6}$, the Theorem holds. If $f(S^*) < \frac{\mathsf{opt}}{6}$, put it back into (22) we get:
$\mathsf{opt} < 4f(S^*) + \frac{1+\epsilon'}{3}\mathsf{opt} \Rightarrow \mathsf{opt} \leq \frac{12f(S)}{2-\epsilon'} < (6 + \epsilon)f(S)$.
Case 2. If $c^* < (1 - \epsilon')B$. By applying the Lemma 5.1, we consider two cases: If **a)** happens, we get $f(S^*) \geq \frac{\mathsf{opt}}{6(1+\epsilon')} \Rightarrow$ $\mathsf{opt} \leq (6 + 6\epsilon')f(S)$ and the Theorem holds. If both **b)** and **d)** happen, there exist $X' \subseteq S_1$, $Y' \subseteq S_2$ and $X' \cap Y' = \emptyset$ satisfying: $\mathsf{opt} = f(O) \leq f(O \cup X') + f(O \cup Y')$

$$\leq 4f(S^*) + 2\max\{\frac{1 + \epsilon'}{1 - \epsilon'}f(S), \frac{(1 - \epsilon')}{6}\mathsf{opt}\}. \quad (23)$$

If $f(S^*) \geq \frac{\mathsf{opt}}{6}$, the Theorem is true. We consider the case $f(S^*) < \frac{\mathsf{opt}}{6}$, put it into (23) we get $\mathsf{opt} < 4f(S^*) + \frac{1+\epsilon'}{1-\epsilon'}\frac{\mathsf{opt}}{3}$.
$\Rightarrow \mathsf{opt} < \frac{6(1-\epsilon')}{1-2\epsilon'}f(S^*) = (6 + \frac{6\epsilon'}{1-2\epsilon'})f(S^*) < (6 + \epsilon)f(S)$. Combining two cases, we obtain the proof. □

6. Empirical Study

In this section, we experiment our LinAst, LinAtg and BoostAdapt by comparing to state-of-the-art for non-monotone SMC including IteratedGreedy (IG) (Gupta et al., 2010), FastRandomGreedy (FRG) (Buchbinder et al., 2015), AdaptiveNon-monotoneMax (ANM) (Fahrbach et al., 2019a), AdaptiveSimpleThreshold (AST) (Chen & Kuhnle, 2024), AdaptiveThresholdGreedy (ATG) (Chen & Kuhnle, 2024), ParSKP2 (Cui et al., 2025) and (FLS) (Chen et al., 2024). The comparison is about **four metrics**: object values, adaptive complexity, number of queries, and running time. We experimented with two well-known applications: Revenue Maximization (RM) and Maximum Cut (MC) (Chen & Kuhnle, 2024; Kuhnle, 2019; Amanatidis et al., 2020).
Dataset and Setting. For all algorithms, we set $\epsilon = 0.1$, except for FLS, for which we set $\epsilon = 0.01$ according to (Chen et al., 2024). Our algorithms were set $\alpha = 4.0$. Algorithms were run 20 times and averaged the results.We used ca-Astro ($n = 18,772, m = 198,110$) for RM and utilized web-Google ($n = 875,713, m = 5,105,039$), ca-GrQc ($n = 5,242, m = 14,496$), and Barabasi-Albert ($n = 968, m = 5,708$) for MC. These standard datasets were sourced

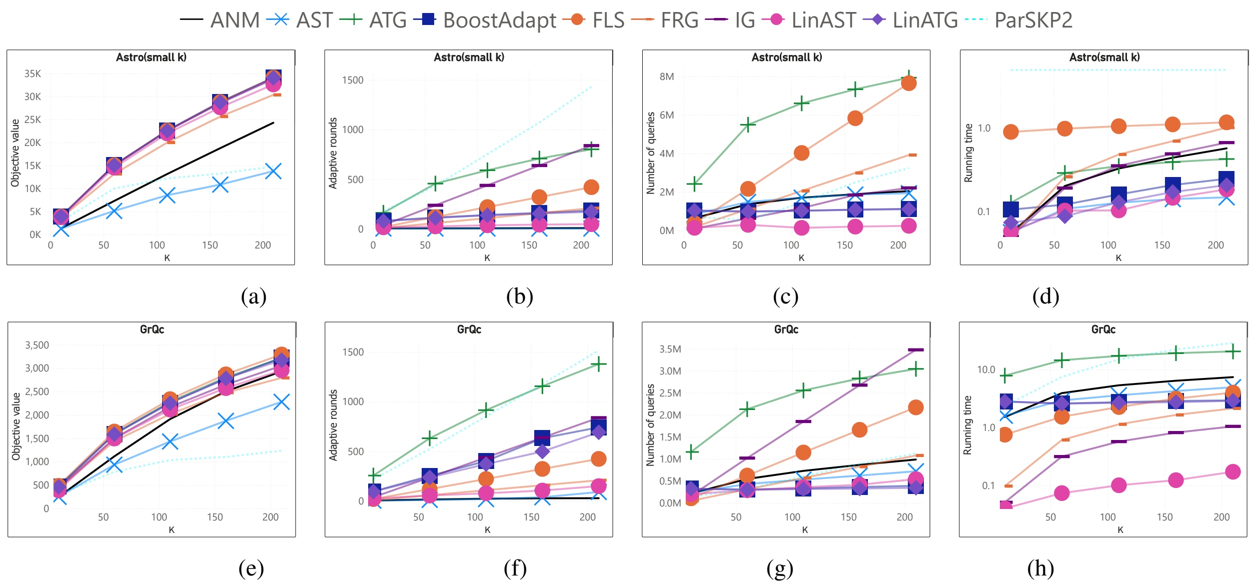

Figure 1. Performance of algorithms on Revenue Maximization (a-d) and Maximum Cut (e-h).

from SNAP[1]. Appendix H gives a more detailed setting.

Overview of Results. Figure 1 displays the performance of compared algorithms on the ca-Astro and ca-GrQC datasets for RM and MC, respectively. Additional results are presented in Appendix H.

Objective value: Figures 1(a)(e) show the objective values of algorithms. It can be observed that the lines of BoostAdapt consistently reach the highest points with every k. Equivalent to BoostAdapt is LinAtg, IG, FLS, and AST. Both LinAst and FRG mark a little lower than BoostAdapt, followed by ANM, AST, and ParSKP. Finally, our algorithms significantly improve the solution quality.

Adaptive rounds: In the MC application (Figure 1(f)), the adaptive rounds result three distinct groups. The group with low adaptivity includes ANM, AST, LinAst, and FRG while the medium group includes IG, LinAtg, FLS and BoostAdapt. The high group is the remaining. ParSKP and ATG algorithms waste the highest number of adaptive rounds while ANM hits the lowest points. With the RM application (Figure 1(b)), ParSKP, ATG and IG give high adaptivity. When k increases, their number of adaptive rounds can quickly grow to 4-10 times larger than the others. LinAtg slightly differs from the lowest ANM while LinAst and BoostAdapt are higher than ANM, but they do not exceed 200.

Number of queries: In both RM and MC (Figures 1(c) and (g)), our algorithms almost always minimize the number of queries. In RM (Figure 1(c)), LinAst has the lowest number of queries, followed by LinAtg and BoostAdapt algorithms. Meanwhile, AST is almost close to ANM and slightly higher than BoostAdapt. The remaining ATG, FRG, ParSKP, FLS and IG belong to the group with many queries. The ATG algorithm wastes the highest number of queries. For MC (Figure 1(g)), LinAtg has the lowest number of queries, followed by BoostAdapt and LinAst. The algorithms AST, ParSKP, FRG, FLS, and ANM are again at an average level. while ATG and IG typically require the highest number of queries, about 5 times greater than the lowest line, LinAtg. On the whole, all our algorithms save queries more than the others. This is consistent with (nearly) linear query complexities of our algorithms.

Time taken: In RM (Figure 1(d)), LinAst and AST have the lowest running time, followed by the group of LinAtg, BoostAdapt, ATG, but with a small gap. The rest uses more time than the others, especially ParSKP, which wastes the highest, about 10-20 times higher than the others. In MC (Figure 1(h)), LinAst wastes the lowest time, followed by IG and FRG. While BoostAdapt and LinAtg have the same running time, which is higher than IG and FRG but lower than AST, ParSKP, FLS and ANM. The gap among these considerably reduces, especially when k increases, except ParSKP. Meanwhile, the time consumption of ParSKP steadily grows along with k's growth. The above metrics show that our algorithms outperform the others when keeping the best quality solutions, wasting the lowest query numbers within acceptable low adaptive rounds.

[1]https://snap.stanford.edu/data/

7. Conclusions

Motivated by the big data challenge for non-monotone submodular maximization under cardinality constraint, in this work, we focus on parallel approximation algorithms based on the concept of adaptive complexity. In particular, we proposed efficient parallel algorithms that significantly improve both approximation ratio and query complexity but keep the near-optimal adaptive complexity of $O(\log n)$. Our algorithm also expresses superior solution quality and computation complexity compared to state-of-the-art algorithms. However, there is still a weakness in our contribution, which is about the approximation factor. It leads to an opening question: how to improve the approximation ratio for $O(\log n)$ adaptive complexity?

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Amanatidis, G., Fusco, F., Lazos, P., Leonardi, S., and Reiffenhäuser, R. Fast adaptive non-monotone submodular maximization subject to a knapsack constraint. In *Annual Conference on Neural Information Processing Systems*, 2020.

Amanatidis, G., Fusco, F., Lazos, P., Leonardi, S., Marchetti-Spaccamela, A., and Reiffenhäuser, R. Submodular maximization subject to a knapsack constraint: Combinatorial algorithms with near-optimal adaptive complexity. In *International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 231–242, 2021.

Badanidiyuru, A. and Vondrák, J. Fast algorithms for maximizing submodular functions. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 1497–1514, 2014.

Badanidiyuru, A. and Vondrák, J. Fast algorithms for maximizing submodular functions. In *Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 1497–1514, 2014. doi: 10.1137/1.9781611973402.110.

Balkanski, E. and Singer, Y. The adaptive complexity of maximizing a submodular function. In *Annual ACM SIGACT Symposium on Theory of Computing*, pp. 1138–1151, 2018.

Balkanski, E., Breuer, A., and Singer, Y. Non-monotone submodular maximization in exponentially fewer iterations. In Bengio, S., Wallach, H. M., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pp. 2359–2370, 2018.

Balkanski, E., Rubinstein, A., and Singer, Y. An exponential speedup in parallel running time for submodular maximization without loss in approximation. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pp. 283–302. SIAM, 2019.

Balkanski, E. et al. Quickswap: Linear-query submodular maximization under matroid and beyond. *In submission*, 2024.

Buchbinder, N. and Feldman, M. Constrained submodular maximization via new bounds for dr-submodular functions. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing, STOC 2024, Vancouver, BC, Canada, June 24-28, 2024*, pp. 1820–1831. ACM, 2024.

Buchbinder, N., Feldman, M., Naor, J., and Schwartz, R. A tight linear time (1/2)-approximation for unconstrained submodular maximization. In *IEEE Symposium on Foundations of Computer Science*, pp. 649–658, 2012.

Buchbinder, N., Feldman, M., Naor, J., and Schwartz, R. Submodular maximization with cardinality constraints. In *Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 1433–1452. SIAM, 2014.

Buchbinder, N., Feldman, M., and Schwartz, R. Comparing apples and oranges: Query tradeoff in submodular maximization. In *Proc. of the 26thl ACM-SIAM SODA 2015*, pp. 1149–1168, 2015.

Calinescu, G., Chekuri, C., Pál, M., and Vondrák, J. Maximizing a monotone submodular function subject to a matroid constraint. *SIAM Journal on Computing*, 40(6):1740–1766, 2011.

Chakrabarti, A. and Kale, S. Submodular maximization meets streaming: Matchings, matroids, and more. In *International Colloquium on Automata, Languages, and Programming (ICALP)*, pp. 385–396, 2015.

Chekuri, C. and Quanrud, K. Parallelizing greedy for submodular set function maximization in matroids and beyond. In Charikar, M. and Cohen, E. (eds.), *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*, pp. 78–89. ACM, 2019.

Chen, L., Feldman, M., and Karbasi, A. Unconstrained submodular maximization with constant adaptive complexity. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2019, pp. 102–113, 2019.

Chen, Y. and Kuhnle, A. Approximation algorithms for size-constrained non-monotone submodular maximization in deterministic linear time. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2023, Long Beach, CA, USA, August 6-10, 2023*, pp. 250–261. ACM, 2023.

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

Chen, Y., Dey, T., and Kuhnle, A. Best of both worlds: Practical and theoretically optimal submodular maximization in parallel. In Ranzato, M., Beygelzimer, A., Dauphin, Y. N., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 25528–25539, 2021.

Chen, Y., Nath, A., Peng, C., and Kuhnle, A. Discretely beyond 1/e: Guided combinatorial algorithms for submodular maximization. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

Cui, S., Han, K., Tang, J., Huang, H., Li, X., and Zhiyuli, A. Practical parallel algorithms for submodular maximization subject to a knapsack constraint with nearly optimal adaptivity. In *Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2023, Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence, IAAI 2023, Thirteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2023, Washington, DC, USA, February 7-14, 2023*, pp. 7261–7269. AAAI Press, 2023.

Cui, S., Han, K., Tang, J., Huang, H., Li, X., Zhiyuli, A., and Li, H. Practical parallel algorithms for non-monotone submodular maximization. *Journal of Artificial Intelligence Research*, 82, 2025.

Ene, A. and Nguyen, H. L. Parallel algorithm for non-monotone dr-submodular maximization. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pp. 2902–2911. PMLR, 2020.

Ene, A., Nguyen, H. L., and Vladu, A. Submodular maximization with matroid and packing constraints in parallel. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*, pp. 90–101. ACM, 2019.

Fahrbach, M., Mirrokni, V. S., and Zadimoghaddam, M. Non-monotone submodular maximization with nearly optimal adaptivity and query complexity. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pp. 1833–1842. PMLR, 2019a.

Fahrbach, M., Mirrokni, V. S., and Zadimoghaddam, M. Submodular maximization with nearly optimal approximation, adaptivity and query complexity. In *Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 255–273, 2019b.

Fahrbach, M., Mirrokni, V., and Zadimoghaddam, M. Non-monotone submodular maximization with nearly optimal adaptivity and query complexity. *preprint, arXiv:1808.06932*, 2023. URL https://arxiv.org/abs/1808.06932.

Feldman, M., Naor, S., and Schwartz, R. Simultaneous greedy for monotone submodular maximization under matroid and knapsack constraints. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 1201–1213, 2020.

Feldman, M., Norouzi-Fard, A., Svensson, O., and Zenklusen, R. Fractional swaps and streaming submodular maximization. *Journal of the ACM*, 2022.

Fisher, M. L., Nemhauser, G. L., and Wolsey, L. A. An analysis of approximations for maximizing submodular set functions. *Mathematical Programming*, 14(1):265–294, 1978.

Gu, Y., Bian, A., and Qian, C. Sprout and sprout++: Scalable submodular maximization with k-matroid and k-knapsack constraints. *arXiv preprint arXiv:2301.10001*, 2023.

Gupta, A., Roth, A., Schoenebeck, G., and Talwar, K. Constrained non-monotone submodular maximization: Offline and secretary algorithms. In *International Workshop on Internet and Network Economics*, 2010.

Han, K., Cao, Z., Cui, S., and Wu, B. Deterministic approximation for submodular maximization over a matroid in nearly linear time. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Han, K., Cui, S., Zhu, T., Zhang, E., Wu, B., Yin, Z., Xu, T., Tang, S., and Huang, H. Approximation algorithms for submodular data summarization with a knapsack constraint. *Proc. ACM Meas. Anal. Comput. Syst.*, 5(1):05:1–05:31, 2021.

Hartline, J. D., Mirrokni, V. S., and Sundararajan, M. Optimal marketing strategies over social networks. In *Proceedings of the 17th International Conference on World Wide Web, WWW 2008, Beijing, China, April 21-25, 2008*, pp. 189–198. ACM, 2008.

Kuhnle, A. Interlaced greedy algorithm for maximization of submodular functions in nearly linear time. In *Neural Information Processing Systems*, pp. 2371–2381, 2019.

Kuhnle, A. Nearly linear-time, parallelizable algorithms for non-monotone submodular maximization. In *Proc. of the 30th AAAI 2021*, pp. 8200–8208, 2021.

Lee, J., Mirrokni, V. S., Nagarajan, V., and Sviridenko, M. Maximizing nonmonotone submodular functions under matroid or knapsack constraints. *SIAM J. Discret. Math.*, 23(4):2053–2078, 2010.

Lin, H. and Bilmes, J. A. A class of submodular functions for document summarization. In *Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pp. 510–520, 2011.

Mirzasoleiman, B., Badanidiyuru, A., and Karbasi, A. Fast constrained submodular maximization: Personalized data summarization. In *International Conference on Machine Learning*, volume 48 of *JMLR Workshop and Conf. Proc.*, pp. 1358–1367, 2016a.

Mirzasoleiman, B., Badanidiyuru, A., Karbasi, A., Vondrák, J., and Krause, A. Fantom: Fast approximate submodular maximization over data streams. In *Advances in Neural Information Processing Systems*, pp. 1904–1912, 2016b.

Mitzenmacher, M. and Upfal, E. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, 2005. ISBN 978-0-521-83540-4.

Tukan, M., Mualem, L., and Feldman, M. Practical 0.385-approximation for submodular maximization subject to a cardinality constraint. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

Wald, A. Some generalizations of the theory of cumulative sums of random variables. *The Annals of Mathematical Statistics*, 16(3):287–293, 1945. ISSN 00034851. URL http://www.jstor.org/stable/2235707.

A. Organization of the Appendix

- Section B presents simplified version of BoostAdapt algorithms and discussion.
- Section C presents some essential Probability Lemmas and Concentration Bounds used in this work.
- Section D presents algorithms missed in Section 3.
- Section E presents the Algorithms and proofs missed of Section 4.
- Section G presents the proofs missed of Section **??**.
- Section H presents the additional experimental details and results of Section 6.

B. Simplified Version of BoostAdapt **algorithm**

To easily follow the idea of the BoostAdapt algorithm, this section presents a simplified version of Boost and some discussion on related algorithms.

This version operates in $O(\log_{\frac{1}{1-\epsilon}}(\frac{k^2}{4\epsilon}))$ iterations. It adapts greedy threshold (Badanidiyuru & Vondrák, 2014) to update two disjoint sets X and Y. In particular, it alternately selects elements with the marginal gain at least τ_X (τ_Y) into X (Y). Note that the thresholds τ_X and τ_Y are updated alternately after each iteration. Finally, the algorithm returns the best solution among X and Y.

It is noted that our algorithm operates in a different way from Iterated Greedy (Gupta et al., 2010), which works as follows: (1) it repeats greedy methods to construct two solutions: finding a feasible solution X over ground set V then finding another solution Y over new ground set $V \setminus X$; (2) it then adapts the unconstrained submodular maximization (USM) algorithm to find the solution X' over ground set X and returns the best one among X, Y, and X'. Our algorithm also differs from the twin greedy strategy (Han et al., 2020) and Interlaced Greedy (Kuhnle, 2019), allowing simultaneously updating two disjoint sets simultaneously in each iteration. In contrast to these algorithms, our algorithm allows the integration of ThreshSeq in each iteration; thus, it can be effectively parallelized.

Algorithm 4 Simplified Version of BoostAdapt Algorithm

```
1:  Input: $f, V, k, \epsilon$
2:  $M \leftarrow \max_{e \in V} f(e)$
3:  $\Delta \leftarrow \lceil \log_{\frac{1}{1-\epsilon}}(\frac{k^2}{4\epsilon}) \rceil + 1, \tau \leftarrow kM/4$
4:  $X \leftarrow \emptyset, Y \leftarrow \emptyset$
5:  for $i \leftarrow 1$ to $\Delta$ do
6:      if $i$ is odd then
7:          $\tau_X \leftarrow \tau(1-\epsilon)^i$
8:          for $e \in V \setminus (X \cup Y)$ do
9:              if $f(e|X) \geq \tau_X$ then
10:                 $X \leftarrow X \cup \{e\}$
11:             end if
12:         end for
13:     else
14:         $\tau_Y \leftarrow \tau(1-\epsilon)^i$
15:         for $e \in V \setminus (X \cup Y)$ do
16:             if $f(e|Y) \geq \tau_Y$ then
17:                 $Y \leftarrow Y \cup \{e\}$
18:             end if
19:         end for
20:     end if
21: end for
22: $S \leftarrow \arg\max_{Z \in \{X,Y\}} f(Z)$
23: Return $S$
```

We define the following notations used for analyzing the algorithm's performance guarantees

- Supposing X and Y ordered: $X=\{x_1,x_2,\ldots,x_{|X|}\}$, $Y=\{y_1,y_2,\ldots,y_{|Y|}\}$, we conduct: $X^i=\{x_1,x_2,\ldots,x_i\}$, $Y^i=\{y_1,y_2,\ldots,y_i\}$, where x_i (or y_i) is i-th element added into X (or Y).
- For $e\in X\cup Y$, we assume that x is added into X or Y at iteration $l(e)$.
- X_i and Y_i are X and Y after the iteration i of the main loop and $X_0=Y_0=\emptyset$.
- For an element $e\in X\cup Y$, we denote $X^{<e}$, $Y^{<e}$, τ^e_X and τ^e_Y are X, Y, τ_X and τ_Y right before e is selected into X or Y, respectively.
- τ^{last}_X and τ^{last}_Y are τ_X and τ_Y at the last iterations when X and Y are considered to update.

We provide the performance guarantees of Algorithm 4 in Theorem B.1.

Theorem B.1. *For any $\epsilon\in(0,1/4)$, the Algorithm 4 takes $O(\frac{n}{\epsilon}\log(\frac{k}{\epsilon}))$ query complexity and return an approximation ratio of $1/4-\epsilon$.*

Proof. The Algorithm 4 takes n queries to find M and then runs in $\Delta=O(\log_{1/(1-\epsilon)}(\frac{k^2}{4\epsilon}))=O(\frac{1}{\epsilon}\log(\frac{k}{\epsilon}))$ iterations. Each iteration takes $O(n)$ queries to update X or Y. Thus, the algorithm takes total $O(n)+O(\frac{n}{\epsilon}\log(\frac{k}{\epsilon}))=O(\frac{n}{\epsilon}\log(\frac{k}{\epsilon}))$.

For the approximation ratio. If $|X|\neq\emptyset$ or $|Y|\neq\emptyset$, i.e., it contains at least one element, we have

$$f(S)\geq\frac{(1-\epsilon)^2}{4}\max\{f(X_1),f(Y_2)\}\geq\frac{(1-\epsilon)^2kM}{4}\geq\frac{(1-\epsilon)^2\mathsf{opt}}{4}>(\frac{1}{4}-\epsilon)\mathsf{opt}. \tag{24}$$

We consider the other case when $X_1=Y_2=\emptyset$. In this case $f(X_1)=0$, so $f(X_1\cap O)=0$. We further consider the following sub-cases

- If $|X|<k$ and $|Y|<k$. Considering an element e is added into Y (at iteration $l(e)\geq 4$), it was not added into X at previous iterations. Therefore, it has marginal gain $f(e|X^{<e})<\theta^e_X=\frac{\theta_Y^{l(e)}}{1-\epsilon}\leq\frac{f(e|Y^{<e})}{1-\epsilon}$. Combining this with $\frac{kM}{4}\geq\tau\geq\frac{(1-\epsilon)\epsilon M}{k}$, we have

$$\begin{aligned}f(O\cup X)-f(X)&\leq\sum_{e\in O\setminus X}f(e|X) &(25)\\ &=\sum_{e\in O\cap Y}f(e|X)+\sum_{e\in O\setminus(X\cup Y)}f(e|X) &(26)\\ &\leq\sum_{e\in O\cap Y}\frac{f(e|Y^{<e})}{1-\epsilon}+\tau^{last}_X k &(27)\\ &<\frac{f(Y)}{1-\epsilon}+\epsilon\mathsf{opt} &(28)\end{aligned}$$

  where inequality (25) is due to the submodularity of f, inequality (27) is due to: $\tau^{last}_X\leq\epsilon M/k\leq\epsilon\mathsf{opt}/k$, $f(Y)=\sum_{e\in Y}f(e|Y^{<e})$ and Y contains elements with positive marginal gain.

  On the other hand, since $X_1=\emptyset$ any element e is added into X (at iteration $l(e)\geq 2$) having marginal gain $f(e|Y^{<e})<\theta^e_Y=\theta_X^{l(e)-1}=\frac{\theta_Y^{l(e)}}{1-\epsilon}\leq\frac{f(e|X^{<e})}{1-\epsilon}$. Thus

$$\begin{aligned}f(O\cup Y)-f(Y)&\leq\sum_{e\in O\cap X}f(e|Y)+\sum_{e\in O\setminus(X\cup Y)}f(e|X) &(29)\\ &\leq\sum_{e\in O\cap X}\frac{f(e|X^{<e})}{1-\epsilon}+\tau^{last}_Y k &(30)\\ &<\frac{f(X)}{1-\epsilon}+\epsilon\mathsf{opt}. &(31)\end{aligned}$$

By combining the submodularity of f, inequalities (28) and (31) and the selection rule of the final solution, we have:

$$\begin{aligned}
f(O) &\le f(O\cup X) + f(O \cup Y) && (32)\\
&\le \frac{f(X)+f(Y)}{1-\epsilon} + f(X) + f(Y) + 2\epsilon\mathsf{opt} && (33)\\
&\le 2\frac{2-\epsilon}{1-\epsilon} f(S) + 2\epsilon\mathsf{opt} && (34)
\end{aligned}$$

which implies that

$$f(S) \ge \frac{(1-2\epsilon)(1-\epsilon)}{2(2-\epsilon)}\mathsf{opt} > (\frac{1}{2}-\epsilon)(\frac{1}{2}-\frac{\epsilon}{2}) > (\frac{1}{4}-\epsilon)\mathsf{opt}. \quad (35)$$

- If $|X| = k$ and $|Y| = k$, then $l(x_k) \ge 3$. Any element e is not added into X at iteration $l(x_k)$ is not added into X at iteration $l(x_k) - 2$. On the other hand, since $|X| = k \ge |O|$, so $|X \setminus O| \ge |O \setminus X|$, we have:

$$\begin{aligned}
f(X\cup O) - f(X) &\le \sum_{e\in O\setminus X} f(e|X) && (36)\\
&\le \sum_{e\in O\cap Y} f(e|X) + \sum_{e\in O\setminus (X\cup Y)} f(e|X) && (37)\\
&< \frac{\sum_{e\in O\cap Y} f(e|Y^{<e})}{1-\epsilon} + |O\setminus X|\cdot \tau_X^{l(x_k)-2} && (38)\\
&\le \frac{\sum_{e\in O\cap Y} f(e|Y^{<e})}{1-\epsilon} + |X\setminus O|\cdot \frac{\tau_X^{l(x_k)}}{(1-\epsilon)^2} && (39)\\
&\le \frac{\sum_{e\in O\cap Y} f(e|Y^{<e})}{1-\epsilon} + \sum_{e\in X\setminus O} \frac{f(e|X^{<e})}{(1-\epsilon)^2}. && (40)
\end{aligned}$$

Similarly, we have

$$\begin{aligned}
f(O\cup Y) - f(Y) &\le \sum_{e\in O\cap X} f(e|Y) + \sum_{e\in O\setminus (Y\cup Y)} f(e|X) && (41)\\
&< \frac{\sum_{e\in O\cap X} f(e|X^{<e})}{1-\epsilon} + \tau_Y^{l(y_k)-2} k && (42)\\
&= \frac{\sum_{e\in O\cap X} f(e|X^{<e})}{1-\epsilon} + \frac{\tau_Y^{l(y_k)}}{(1-\epsilon)^2} k && (43)\\
&\le \frac{\sum_{e\in O\cap X} f(e|X^{<e})}{1-\epsilon} + \sum_{e\in Y\setminus O} \frac{f(e|Y^{<e})}{(1-\epsilon)^2}. && (44)
\end{aligned}$$

By combining (40) and (44), we have

$$\begin{aligned}
f(O) < &\frac{\sum_{e\in O\cap Y} f(e|Y^{<e})}{1-\epsilon} + \sum_{e\in Y\setminus O} \frac{f(e|Y^{<e})}{(1-\epsilon)^2}\\
&+ \sum_{e\in X\setminus O} \frac{f(e|X^{<e})}{(1-\epsilon)^2} + \frac{\sum_{e\in O\cap X} f(e|X^{<e})}{1-\epsilon} + f(X) + f(Y)\\
&< \frac{f(X)+f(Y)}{(1-\epsilon)^2} + 2f(S) = (2 + \frac{2}{(1-\epsilon)^2}) f(S) < \frac{4}{(1-\epsilon)^2} f(S)
\end{aligned}$$

which implies that $f(S) \ge \frac{(1-\epsilon)^2}{4}\mathsf{opt} > (\frac{1}{4}-\epsilon)\mathsf{opt}$.

- If $|X| < k$ and $|Y| = k$, by applying the transformations to (27) and (44) we have

$$f(O \cup X) - f(X) \leq \sum_{e \in O \cap Y} \frac{f(e|Y^{<e})}{1-\epsilon} + \epsilon \mathsf{opt} \tag{45}$$

  and

$$f(O \cup Y) - f(Y) \leq \frac{\sum_{e \in O \cap X} f(e|X^{<e})}{1-\epsilon} + \sum_{e \in Y \setminus O} \frac{f(e|Y^{<e})}{(1-\epsilon)^2}. \tag{46}$$

  Therefore

$$\begin{aligned} f(O) &\leq f(O \cup X) + f(O \cup Y) && (47) \\ &< \sum_{e \in O \cap Y} \frac{f(e|Y^{<e})}{1-\epsilon} + \sum_{e \in Y \setminus O} \frac{f(e|Y^{<e})}{(1-\epsilon)^2} && (48) \\ &+ \frac{\sum_{e \in O \cap X} f(e|X^{<e})}{1-\epsilon} + f(X) + f(Y) + \epsilon \mathsf{opt} && (49) \\ &\leq \frac{f(Y)}{(1-\epsilon)^2} + \frac{f(X)}{1-\epsilon} + 2f(S) + \epsilon \mathsf{opt} && (50) \\ &< \frac{4}{(1-\epsilon)^2} f(S) + \epsilon \mathsf{opt}. && (51) \end{aligned}$$

  Thus $f(S) \geq \frac{(1-\epsilon)^3}{4} \mathsf{opt} > (\frac{1}{4} - \epsilon)\mathsf{opt}$.

- If $|X| = k$ and $|Y| < k$, by applying the transformations to (30) and (40) we also have

$$f(O \cup Y) - f(Y) \leq \sum_{e \in O \cap X} \frac{f(e|X^{<e})}{1-\epsilon} + \epsilon \mathsf{opt} \tag{52}$$

  and

$$f(O \cup X) - f(X) \leq \frac{\sum_{e \in O \cap Y} f(e|Y^{<e})}{1-\epsilon} + \sum_{e \in X \setminus O} \frac{f(e|X^{<e})}{(1-\epsilon)^2}. \tag{53}$$

  Therefore

$$\begin{aligned} f(O) &\leq f(O \cup X) + f(O \cup Y) && (54) \\ &< \sum_{e \in O \cap X} \frac{f(e|X^{<e})}{1-\epsilon} + \sum_{e \in X \setminus O} \frac{f(e|X^{<e})}{(1-\epsilon)^2} && (55) \\ &+ \frac{\sum_{e \in O \cap Y} f(e|Y^{<e})}{1-\epsilon} + f(X) + f(Y) + \epsilon \mathsf{opt} && (56) \\ &\leq \frac{f(X)}{(1-\epsilon)^2} + \frac{f(Y)}{1-\epsilon} + 2f(S) + \epsilon \mathsf{opt} && (57) \\ &< \frac{4}{(1-\epsilon)^2} f(S) + \epsilon \mathsf{opt}. && (58) \end{aligned}$$

  Thus $f(S) \geq \frac{(1-\epsilon)^3}{4} \mathsf{opt} > (\frac{1}{4} - \epsilon)\mathsf{opt}$.

Combining all cases, we complete the proof. □

C. Probability Lemmas and Concentration Bounds

This section provides Probability Lemmas and Concentration Bounds that are useful for analyzing the theoretical bounds of our algorithms.

Lemma C.1 (Chernoff bounds (Mitzenmacher & Upfal, 2005))**.** *Supposing that $X_1, \ldots, X_n$ are independent binary random variables such that $\Pr[X_i = 1] = p_i$. Let $\mu = \sum_{i=1}^n p_i$, and $X = \sum_{i=1}^n X_i$. Then for any $\delta \geq 0$, we have*

$$\Pr[X \geq (1+\delta)\mu] \leq e^{\frac{-\delta^2\mu}{2+\delta}}. \tag{59}$$

For $0 \leq \delta \leq 1$, we have

$$\Pr[X \leq (1-\delta)\mu] \leq e^{\frac{-\delta^2\mu}{2}}. \tag{60}$$

Lemma C.2 (Lemma 6 in (Chen et al., 2021))**.** *Suppose there is a sequence of n Bernoulli trials: $X_1, X_2, \ldots, X_n$, where the success probability of X_i depends on the preceding trials $X_1, X_2, \ldots, X_{i-1}$. Suppose it holds that*

$$\Pr[X_i = 1 | X_1 = x_1, X_2 = x_2, \ldots, X_{i-1} = x_{i-1}] \geq \eta, \tag{61}$$

where $\eta > 0$ is a constant and $x_1, \ldots, x_{i-1}$ are arbitrary.
Then if $Y_1, \ldots, Y_n$ are independent Bernoulli trials, each with probability η of success, then

$$\Pr\left(\sum_{i=1}^n X_i \leq b\right) \leq \Pr\left(\sum_{i=1}^n Y_i \leq b\right), \tag{62}$$

where b is an arbitrary integer. Moreover, let A be the first occurrence of success in sequence X_i. Then,

$$\mathbb{E}[A] \leq 1/\eta. \tag{63}$$

Lemma C.3 (Lemma 7 in (Chen et al., 2021))**.** *Suppose there is a sequence of $n+1$ Bernoulli trials: $X_1, X_2, \ldots, X_{n+1}$, where the success probability of X_i depends on the preceding trials $X_1, X_2, \ldots, X_{i-1}$, and it decreases from 1 to 0. Let $t = \max\{i : \Pr[X_i = 1] \geq \eta\}$, where $0 < \eta < 1$ is a constant. Then, it holds that*

$$\Pr[X_i = 1 | X_1 = x_1, X_2 = x_2, \ldots, X_{i-1} = x_{i-1}, i \leq t] \geq \eta, \tag{64}$$

where $x_1, \ldots, x_{i-1}$ are arbitrary.
Then if $Y_1, \ldots, Y_{n+1}$ are independent Bernoulli trials, each with probability η of success, then

$$\Pr\left(\sum_{i=1}^t X_i \leq bt\right) \leq \Pr\left(\sum_{i=1}^t Y_i \leq bt\right), \tag{65}$$

where b is an arbitrary integer.

Lemma C.4 (Wald's Equation (Wald, 1945))**.** *Let $(X_n)_{n\in\mathbb{N}}$ be an infinite sequence of real-valued random variables and let N be a nonnegative integer-valued random variable. Assume that: 1) $(X_n)_{n\in\mathbb{N}}$ are integrable (finite-mean) random variables, 2) $\mathbb{E}[X_n 1_{N\geq n}] = \mathbb{E}[X_n]P(N \geq n)$ for every natural number n, 3) the infinite series satisfies $\sum_{n=1}^\infty \mathbb{E}[|X_n| 1_{\{N\geq n\}}] < \infty$. Then the random sums $S_N = \sum_{n=1}^N X_n$ and $T_N = \sum_{n=1}^N \mathbb{E}[X_n]$ are integrable and $\mathbb{E}[S_N] = \mathbb{E}[T_N]$.*

Lemma C.5 (Lemma 11 in (Chen et al., 2021))**.** *Let (Y_i) be a sequence of independent and identically distributed Bernoulli trials, where the success probability is $\beta\epsilon$. Then for a constant integer α, we have*

$$\Pr[\sum_{i=1}^\alpha Y_i > \epsilon\alpha] \leq \min\{\beta, e^{-\frac{(1-\beta)^2}{1+\beta}\epsilon\alpha}\}. \tag{66}$$

Algorithm 5 $\mathsf{ThreshSeq}(V, \epsilon, \delta)$

```
1: **Input:** $f$, $V$, $k$, $\epsilon$, $\delta$, $\tau$
2: $A \leftarrow \emptyset$, $A' \leftarrow \emptyset$, $U \leftarrow V$, $\Delta = \lceil 4(\frac{2}{\epsilon}\log(n) + \log(\frac{n}{\delta}))\rceil$
3: **for** $j \leftarrow 1$ to $\Delta$ **do**
4:     Update $U \leftarrow \{x \in U : f(x|A) \geq \tau\}$
5:     **if** $|U| = 0$ **then**
6:         **return** $A, A'$
7:     **end if**
8:     $U \leftarrow \mathbf{random-permutation}(U)$
9:     $s \leftarrow \min\{k - |A|, |V|\}$
10:    $B[1:s] \leftarrow [none, none, \ldots, none]$
11:    **for** $i \leftarrow 1$ to $s$ (**in parallel**) **do**
12:        $T_{i-1} \leftarrow \{v_1, v_2, \ldots, v_{i-1}\}$
13:        **if** $f(v_i|A \cup T_{i-1}) \geq \tau$ **then**
14:            $B[i] \leftarrow$ **true**
15:        **else**
16:            **if** $f(v_i|A \cup T_{i-1}) < 0$ **then**
17:                $B[i] \leftarrow$ **false**
18:            **end if**
19:        **end if**
20:        $i^* \leftarrow \max\{i :$ #trues in $B[1:i] \geq (1-\epsilon)i\}$
21:        $A \leftarrow A \cup T_{i^*}$
22:        $A' \leftarrow A' \cup T_{i^*}$[where B $\neq$ **false**]
23:        **if** $|A| = k$ **then**
24:            **Return** $A, A'$
25:        **end if**
26:    **end for**
27: **end for**
28: **Return** *failure*
```

D. Algorithms missed in Section 3

In this section, we recap some subroutines in our algorithms and their performance bounds.

D.1. ThreshSeq (Algorithm 2 in (Chen & Kuhnle, 2024))

D.2. Unconstrained Submodular Maximization Algorithms

In the LinAdapt algorithm, we adapt the (Fahrbach et al., 2019a)'s algorithm (USM1) that provides a factor of $1/4 - \epsilon$ with constant probability for the USM problem. The guarantees for the USM1 algorithm are sated in Lemma D.1.

Algorithm 6 USM1(V, ϵ, δ)

```
1: Input: $f$, a subset $A \subseteq V$, $\varepsilon$, $\delta$
2: Set iteration bound $t \leftarrow \lceil \log(1/\delta)/\log(1 + (4/3)\varepsilon) \rceil$
3: for $i = 1$ to $t$ in parallel do
4:     Let $R_i$ be a uniformly random subset of $A$
5: end for
6: Set $S \leftarrow \arg\max_{X \in \{R_1, \ldots, R_t\}} f(X)$
7: return $S$
```

Lemma D.1. *For any nonnegative submodular function f and subset $A \subseteq V$, Algorithm 6 outputs a set $S \subseteq V$ in one adaptive round using $O(\log(1/\delta)/\varepsilon)$ oracle queries such that with probability at least $1 - \delta$ we have $f(S) \geq (1/4 - \varepsilon)\mathsf{opt}_A$, where $\mathsf{opt}_A = \max_{T \subseteq A} f(T)$.*

We further adapt an essentially optimal algorithm of (Chen et al., 2019) for USM in our LBA which allows us to improve the approximation ratio slightly.

Theorem D.2 ((Chen et al., 2019)). *There is an algorithm that achieves a $1/(2 + \epsilon)$-approximation for USM using $O(\log(1/\epsilon)/\epsilon)$ adaptive rounds and $O(n \log^3(1/\epsilon)/\epsilon^4)$ query complexity.*

E. Proofs missed of Section 4

E.1. Proofs missed of LinBoundSet (Algorithm ??)

Lemma ??. If LinBoundSet ends successfully, then $f(S) \leq \left(1 + \frac{1+\epsilon}{(1-\epsilon)(1-2\epsilon)\alpha}\right) f(S')$.

Proof of Lemma **??**. If $S \leq k$, $S' = S$, thus the Lemma holds. We consider the remaining case $S > k$. Let $\overline{S} = S \setminus S'$. Beside the notations of T_i, T'_{λ_i} in Line **??** of Algorithm **??**, we use the following useful notations:

- S_j and T_{j,λ_j^*} are S and T_{λ^*} after iteration j.
- $T_{j,i}$ is the set of first i elements added into T_{j,λ_j^*}.
- $T'_{j,\lambda_i} = T_{j,\lambda_i} \setminus T_{j,\lambda_{i-1}}$.
- $T''_{\lambda_i} = \{v_j \in T'_{\lambda_i} : b[j] \neq \textbf{false}\}$, i.e., elements in T'_{λ_i} with non-negative marginal gain.
- $T^j = \bigcup_{\lambda_i \leq \lambda_j^*} T''_{\lambda_i}; T^j_{\lambda_i} = \bigcup_{\lambda \leq \lambda_i} T''_{\lambda}$.
- $c = \max\{c \in \mathbb{N} : S' \subseteq \bigcup_{j=c}^{\ell} T^j\}$.

By the selection rule in Line **??**, Algorithm **??**, T''_{λ_i} only contains elements with non-negative marginal gain. If T'_{λ_i} is a **good** block, it has at least $(1-\epsilon)|T'_{\lambda_i}|$ good elements and thus $|T''_{\lambda_i}| \geq (1-\epsilon)|T'_{\lambda_i}|$.
It's easy to see that $|T^j| < k$ for any $j > c$, and the size of T^c may be larger than k. Therefore, we consider two following cases:
Case 1. If $j > c$, $|T^j| \leq k$. By the selection rule of λ^* in Line **??**, Algorithm **??**, there are $\lambda_j^* - 1$ first blocks in T_{j,λ_j^*} are good and the last one is bad. For any block $T'_{j,\lambda_i} \subseteq T_{j,\lambda_j^*}, \lambda_i < \lambda_j^*$, we have

$$\begin{aligned}
f(T''_{\lambda_i}|S_{j-1} \cup T^j_{\lambda_i - 1}) &\geq f(T''_{\lambda_i}|S_{j-1} \cup T_{j,\lambda_{i-1}}) && \text{(By the submodularity and } T^j_{\lambda_i-1} \subseteq T_{j,\lambda_{i-1}}) \\
&\geq \sum_{v_l \in T''_{\lambda_i}} f(v_l|S_{j-1} \cup T_{j,l-1}) \\
&\geq (1-\epsilon)\alpha|T'_{j,\lambda_i}|\frac{M_{\lambda_{i-1}}}{k} && \text{(Since } T'_{j,\lambda_i} \text{ is a good block)} \\
&\geq (1-\epsilon)\alpha|T'_{j,\lambda_i}|\frac{f(S_{j-1})}{k} && \text{(By selection rule of } M_{\lambda_{i-1}}). \qquad (67)
\end{aligned}$$

Since the block size exponentially increases with factor of $1+\epsilon$ when $\lambda_i \leq k$, so we have $|T'_{j,\lambda_j^*}| \leq \epsilon|T_{j,\lambda_j^*}|/(1+\epsilon)$. Combining this with (67), we have:

$$\begin{aligned}
f(T^j|S_{j-1}) &= \sum_{\lambda_i \leq \lambda_j^*} f(T''_{\lambda_i}|S_{j-1} \cup T^j_{\lambda_{i-1}}) && (68) \\
&\geq \sum_{\lambda_i < \lambda_j^*} f(T''_{\lambda_i}|S_{j-1} \cup T^j_{\lambda_{i-1}}) \quad \text{(Since } b[e] \geq 0, \forall e \in T''_{\lambda_i}) && (69) \\
&\geq \sum_{\lambda_i < \lambda_j^*} (1-\epsilon)\alpha|T'_{j,\lambda_i}|\frac{f(S_{j-1})}{k} && (70) \\
&= (1-\epsilon)\alpha|T_{j,\lambda_j^*} \setminus T'_{j,\lambda_j^*}|\frac{f(S_{j-1})}{k} && (71) \\
&\geq \frac{1-\epsilon}{1+\epsilon}\alpha\frac{|T_{j,\lambda_j^*}|}{k}f(S_{j-1}) && (72) \\
&\geq \frac{1-\epsilon}{1+\epsilon}\alpha\frac{|T^j|}{k}f(S_{j-1}). && (73)
\end{aligned}$$

Case 2. If $j = c$, also by the selection rule of λ^*, the last block T'_{c,λ_c^*} is always bad. If $|T_{c,\lambda_c^*}| \leq k$, all previous blocks $T'_{c,\lambda_i}, \lambda_i < \lambda_c^*$ are good. If $|T_{c,\lambda_c^*}| > k$, some previous contiguous blocks are good and they contain at least k elements. By the selection rule of S, we can let $\lambda_r = \min\{\lambda_r \in \Lambda : \bigcup_{\lambda_r \leq \lambda_i \leq \lambda_c^*} T''_{c,\lambda_i} \subseteq S'\}$. With a note that each block T''_{c,λ_i} contains at most ϵk elements and S' is the union of some blocks, we have:

$$f(T^c|\overline{S}) = \sum_{\lambda_r \leq \lambda_i \leq \lambda_c^*} f(T''_{\lambda_i}|\overline{S} \cup T^c_{\lambda_{i-1}}) \tag{74}$$
$$\geq \sum_{\lambda_r \leq \lambda_i < \lambda_c^*} f(T''_{\lambda_i}|\overline{S} \cup T_{c,\lambda_{i-1}}) \tag{75}$$
$$\geq \sum_{\lambda_r \leq \lambda_i < \lambda_c^*} (1-\epsilon)\alpha|T'_{c,\lambda_i}|\frac{M_{\lambda_{i-1}}}{k} \tag{76}$$
$$\geq \sum_{\lambda_r \leq \lambda_i < \lambda_c^*} (1-\epsilon)\alpha|T'_{c,\lambda_i}|\frac{f(\overline{S})}{k} \tag{77}$$
$$\geq (1-\epsilon)\alpha|(T_{c,\lambda_j^*} \cap S') \setminus T'_{c,\lambda_j^*}|\frac{f(\overline{S})}{k} \tag{78}$$
$$\geq (1-\epsilon)\alpha \cdot \max\{0, |T^c \cap S'| - \epsilon k\}\frac{f(\overline{S})}{k}. \tag{79}$$

Combining two cases above with a note that $k \geq |S'| \geq (1-\epsilon)k$, we have:

$$f(S) - f(\overline{S}) = f(T^c|\overline{S}) + \sum_{j=c+1}^{\ell} f(T^j|S_{j-1}) \tag{80}$$
$$\geq (1-\epsilon)\alpha \cdot \max\{0, |T^c \cap S'| - \epsilon k\}\frac{f(\overline{S})}{k} + \sum_{j=c+1}^{\ell} \frac{1-\epsilon}{1+\epsilon}\alpha\frac{|T^j|}{k}f(S_{j-1}). \tag{81}$$
$$\geq (1-\epsilon)\alpha \cdot \max\{0, |T^c \cap S'| - \epsilon k\}\frac{f(\overline{S})}{k} + \frac{1-\epsilon}{1+\epsilon}\alpha\frac{|S' \setminus T^c|}{k}f(\overline{S}). \tag{82}$$
$$\geq \frac{1-\epsilon}{1+\epsilon}\alpha(|S'| - \epsilon k)\frac{f(\overline{S})}{k} \tag{83}$$
$$\geq \frac{1-\epsilon}{1+\epsilon}\alpha(k - 2\epsilon k)\frac{f(\overline{S})}{k} \tag{84}$$
$$\geq \frac{1-\epsilon}{1+\epsilon}(1-2\epsilon)\alpha f(\overline{S}) \tag{85}$$

which implies that $f(S) \geq (1 + \frac{(1-\epsilon)(1-2\epsilon)\alpha}{1+\epsilon})f(\overline{S})$. By the submodularity of f, we have

$$f(S) \leq f(S') + f(\overline{S}) \Longrightarrow f(S') \geq f(S) - f(\overline{S}) \geq \frac{(1-\epsilon)(1-2\epsilon)\alpha}{1+\epsilon+(1-\epsilon)(1-2\epsilon)\alpha}f(S) \tag{86}$$

which implies the proof. $\square$

Lemma ??. Pr[iteration j fails] $\leq 1/2$.

Proof of Lemma **??**. We first show that the following propositions are true.

Proposition E.1. *For each iteration in* LinBoundSet, *let* $\lambda_{(t)} = \max\{\lambda_i \in \Lambda : \lambda_i < t\}$ *and* $Q_t = \{e \in V : f(e|S \cup T_t) < \alpha M_{\lambda_{(t)}}/k\}$, *we have* $|Q_0| = 0$, $Q_{|V|} = |V|$, *and* $|Q_t| \leq |Q_{t+1}|$.

By Line **??** of Algorithm **??**, we have $f(e|S) \geq \alpha f(S)/k$ for all $e \in V$. Therefore $|Q_0| = 0$. On the other hand, since $f(e|S \cup V) = 0 < \alpha M_{\lambda_{(|V|)}}/k$ for all $e \in V$, $|Q_{|V|}| = |V|$. For the last property, by the submodularity of f and the definition of M_{λ_i}, for any element $e \in Q_t$ we have

$$f(e|S \cup T_{t+1}) \leq f(e|S \cup T_t) < \alpha \frac{M_{\lambda_{(t)}}}{k} \leq \alpha \frac{M_{\lambda_{(t+1)}}}{k}, \tag{87}$$

which implies that $e \in Q_{t+1}$. Hence $Q_t \subseteq Q_{t+1}$ and $|Q_t| \leq |Q_{t+1}|$.

Now, we provide a bound probability that $B[\lambda]$ is false in the following Proposition.

Proposition E.2. *Let $t = \min\{j \in \mathbb{N} : |Q_j| \geq \beta\epsilon|V|\}$, $\lambda_{(t)} = \max\{\lambda \in \Lambda : \lambda < t\}$ and (Y_j) be the sequence of the independent and identically distributed Bernoulli trials with the success probability is $\beta\epsilon$. For any $\lambda < \lambda_{(t)}$, we have* $\Pr[B[\lambda] = \textbf{false}] \leq \Pr[\sum_{j=1}^{|T'_\lambda|} Y_j > \epsilon|T'_\lambda|]$.

We define an element v_i that is **bad** if $f(v_i|S \cup T_{i-1}) < \alpha M_{\lambda_{(i)}}/k$ and **good** otherwise. Consider the random permutation of V as a sequence of independent Bernoulli trials, with success if the element is bad and failure otherwise.

We have $\Pr[v_i$ is $\textbf{bad}|v_1, \ldots, v_{i-1}] = |Q_{i-1}|/|V|$ thus the probability that $v_j, j < t$ is bad is less than $\beta\epsilon$. By the definition of $B[\lambda]$ in Line **??**, Algorithm **??**, the block T'_λ is bad, it must contain more than $\epsilon|T'_\lambda|$ elements. Let $X_i = 1$ if v_i is bad; and $X_i = 0$, otherwise, we have

$$\Pr(B[\lambda] = \textbf{false}) \leq \Pr(\text{the number of bad elements in } T'_\lambda > \epsilon|T'_\lambda|) \tag{88}$$

$$= \Pr\left(\sum_{v_j \in T'_\lambda} X_j > \epsilon|T'_\lambda|\right) \tag{89}$$

$$\leq \Pr\left(\sum_{j=1}^{|T'_\lambda|} Y_j > \epsilon|T'_\lambda|\right) \quad \text{(Due to appplying Lemma C.3)}. \tag{90}$$

By the operation of algorithm **??**, the **set S did not change in each iteration of the main loop (Lines 3-24)**. By using Propositions E.1, E.2 we now prove probability bound in Lemma **??**.

The probability iteration j failure is upper bounded by the probability $\lambda^* < t$. We now consider two cases. If $\lambda^* \leq k$, there is at least one block T'_λ with $\lambda \in \Lambda, \lambda < \lambda_{(t)}$ is bad. If $\lambda^* > k$, let $\lambda'_{(t)} = \max\{\lambda' \in \Lambda : \sum_{\lambda \in \Lambda, \lambda' \leq \lambda \leq \lambda_{(t)}} |T'_\lambda| \geq k\}$. Then there must be at least one integer $\lambda \in \Lambda$ between $\lambda_{(t)}$ and $\lambda'_{(t)}$ such that block T'_λ is bad. Therefore, let $B_1 = \{\lambda \in \Lambda : \lambda < k\}$, $B_2 = \{\lambda \in \Lambda : |\Lambda \cap [\lambda, \lambda_{(t)}]| \leq \lceil 1/\epsilon \rceil\}$, then we have

$$\Pr(\text{iteration } j \text{ fails}) \leq \Pr[\lambda^* < t] \leq \Pr(\exists \lambda \in B_1 \cup B_2, B[\lambda] = \textbf{false}) \leq \frac{1}{2}. \tag{91}$$

The proof of inequality (91) is based on some properties of Bernoulli trials, which is shown in detail in (Chen et al., 2021). For the sake of completeness, we write down the details of the proof below.
By Lemma C.4 we have

$$\Pr[\text{iteration } j \text{ fails}] \leq \Pr[\exists \lambda \in B_1 \cup B_2 \text{ with } B[\lambda] = \textbf{false}] \tag{92}$$

$$= \mathbb{E}[1_{\{\exists \lambda \in B_1 \cup B_2 \text{ with } B[\lambda] = \textbf{false}\}}] \tag{93}$$

$$\leq \mathbb{E}\left[\sum_{\lambda \in B_1} 1_{\{B[\lambda]=\textbf{false}\}} + \sum_{\lambda \in B_2} 1_{\{B[\lambda]=\textbf{false}\}}\right] \tag{94}$$

$$\leq \mathbb{E}\left[\sum_{\lambda \in B_1} 1_{\{B[\lambda]=\textbf{false}\}}\right] + \mathbb{E}\left[\sum_{\lambda \in B_2} 1_{\{B[\lambda]=\textbf{false}\}}\right] \tag{95}$$

$$= \mathbb{E}\left[\sum_{\lambda \in B_1} \mathbb{E}[1_{\{B[\lambda]=\textbf{false}\}}]\right] + \mathbb{E}\left[\sum_{\lambda \in B_2} \mathbb{E}[1_{\{B[\lambda]=\textbf{false}\}}]\right] \tag{96}$$

where (96) holds, since the sequence $(1_{\{B[\lambda]=\textbf{false}\}})$ and the random variable t follows the assumptions in Lemma C.4: 1) $1_{\{B[\lambda]=\textbf{false}\}}$s are all integrable random variables because they only take the values 0 and 1; 2) t is a stopping time since it only depends on the previous $t-1$ selections; 3) $\Pr[1_{\{t \geq n\}} = 0)] = 1$ for any $n \geq |V|$.

The first term of (96) is bounded by adapting Proposition E.2 as follows:

$$\mathbb{E}\left[\sum_{\lambda\in B_1}\mathbb{E}[1_{\{B[\lambda]=\textbf{false}\}}]\right] \leq \mathbb{E}\left[\sum_{\lambda\in B_1} \leq \Pr\left(\sum_{j=1}^{|T'_\lambda|} Y_j > \epsilon|T'_\lambda|\right)\right] \tag{97}$$

$$\leq \mathbb{E}\left[\sum_{\lambda\in\{\lfloor(1+\epsilon)^u\rfloor:u\geq 1\}} \Pr\left(\sum_{j=1}^{|T'_\lambda|} Y_j > \epsilon|T'_\lambda|\right)\right] \tag{98}$$

$$\leq \sum_{\lambda\in\{\lfloor(1+\epsilon)^u\rfloor:u\geq 1\}} \min\{\beta, e^{-\frac{(1-\epsilon)^2}{1+\beta}\epsilon|T'_\lambda|}\} \text{ (By Lemma C.5)} \tag{99}$$

$$\leq \sum_{\lambda\geq 1}\min\{\beta, e^{-\frac{\epsilon\lambda}{2}}\} \text{ (Due to } \beta<1/2, |T'_\lambda|<\lambda) \tag{100}$$

$$\leq \alpha\beta + \sum_{\lambda=a+1}^{\infty} e^{-\frac{\epsilon\lambda}{2}}, \text{ (where } a=\lfloor\frac{1}{8\beta}\rfloor = \lfloor\frac{2}{\epsilon}\log(\frac{8}{1-e^{-\epsilon/2}})\rfloor) \tag{101}$$

$$\leq \alpha\beta + \frac{e^{-\epsilon(a+1)/2}}{1-e^{-\epsilon/2}} \tag{102}$$

$$\leq \frac{1}{8}+\frac{1}{8}=\frac{1}{4}, \tag{103}$$

where (98) follows from $B_1 \subseteq \{\lfloor(1+\epsilon)^u : u\geq 1\rfloor\}$, and let $\{\lfloor(1+\epsilon)^u : u \geq 1\rfloor\}\} = \{\lambda_1,\lambda_2,\ldots\}, |T'| = \lambda_i - \lambda_{i-1}$;
For the second term in (96), from Proposition E.2, Lemma C.5 and the fact that $|B_2| < \lceil 1/\epsilon\rceil$, we have

$$\mathbb{E}\left[\sum_{\lambda\in B_2}\mathbb{E}[1_{\{B[\lambda]=\textbf{false}\}}]\right] \leq \mathbb{E}\left[\sum_{\lambda\in B_2} \leq \Pr\left(\sum_{j=1}^{|T'_\lambda|} Y_j > \epsilon|T'_\lambda|\right)\right] \leq \beta\lceil 1/\epsilon\rceil \leq \frac{1}{4} \tag{104}$$

Put it into (96), we have

$$\Pr[\text{iteration } j \text{ fails}] < \frac{1}{4}+\frac{1}{4}=\frac{1}{2} \tag{105}$$

□

Theorem ??. For any $\epsilon, \delta \in (0,1)$, the algorithm LinBoundSet runs in $O(\log(n/\delta)/\epsilon^2)$, returns two sets S and S' such that the following holds unconditionally: (1) The algorithm succeeds with a probability of at least $1-\delta/n$; (2) The algorithm takes $O(n/\epsilon^3)$ query complexity in expectation; (3) If the algorithm succeeds, it returns S, S' satisfying: for any subset $X \subseteq V, |X| \leq k$, we have $f(S\cup X) \leq \left(2+\alpha+\frac{1}{\alpha}+(1+\frac{1}{\alpha})\frac{2(2-\epsilon)}{(1-\epsilon)(1-2\epsilon)}\epsilon\right) f(S')$.

Proof of Theorem **??**. **Prove the success probability.** Since each successful iteration will remove $(\beta\epsilon)$-fraction of V thus if the LinBoundSet fails, there are no more than $\lceil\log_{1/(1-\beta\epsilon)}(n)\rceil$ successful iterations. Let X be the number of successes in the ℓ iterations. Then, By Lemma **??**, X can be regarded as a sum of dependent Bernoulli trials, where the success probability is larger than $1/2$. Let Y be a sum of independent Bernoulli trials, where the success probability is equal to $1/2$. By applying Lemma C.5, we have

$$\Pr[\text{ LinBoundSet } \textbf{fails}] \leq \Pr[X \leq \lceil\log_{1/(1-\beta\epsilon)}(n)\rceil] \tag{106}$$

$$\leq \Pr[Y \leq \lceil\log_{1/(1-\beta\epsilon)}(n)\rceil] \quad \text{(By Lemma C.5)} \tag{107}$$

$$\leq \Pr\left[Y \leq \frac{\log(n)}{\beta\epsilon}\right] \tag{108}$$

$$\leq e^{-\frac{1}{2}(\frac{(2\beta\epsilon+1)^2}{2(\beta\epsilon+1)})^2 2(1+\frac{1}{\beta\epsilon})\log(\frac{n}{\delta})} \quad \text{(By Lemma C.1)} \tag{109}$$

$$\leq (\frac{\delta}{n})^{\frac{(2\beta\epsilon+1)^2}{4\beta\epsilon(\beta\epsilon+1)}} \tag{110}$$

$$\leq \frac{\delta}{n}. \tag{111}$$

Prove the bound of $f(X \cup S)$. We use the notation in the proof of Lemma **??**. We now further prove that $f(X \cup S) \leq (\alpha+1)f(S)$. For each element $x \in X \setminus S$, define $j(x)+1$ be the iteration where x is filtered out (Line **??**, Algorithm **??**), we have $f(x|S_{j(x)}) \leq \alpha f(S_{j(x)})/k$. Therefore.

$$
\begin{aligned}
f(X \cup S) - f(S) &\leq \sum_{x \in X \setminus S} f(x|S_{j(x)}) && (112)\\
&\leq \sum_{x \in X \setminus S} \alpha \frac{f(S_{j(x)})}{k} && (113)\\
&\leq \alpha f(S_{j(x)}) && \text{(Sice } |X| \leq k) \quad (114)\\
&\leq \alpha f(S). && (115)
\end{aligned}
$$

Combine this with (86), we obtain

$$
\begin{aligned}
f(X \cup S) \leq (\alpha+1)f(S) &\leq (\alpha+1)\left(1 + \frac{1+\epsilon}{(1-\epsilon)(1-2\epsilon)\alpha}\right) f(S') && (116)\\
&= \left(2 + \alpha + \frac{1}{\alpha} + (1+\frac{1}{\alpha})\frac{2(2-\epsilon)}{(1-\epsilon)(1-2\epsilon)}\epsilon\right) f(S') && (117)
\end{aligned}
$$

Prove the adaptive and query complexities. LinBoundSet requires the oracle queries on Lines **??**, **??**, **??** of Algorithm **??**. At these times, the queries are executed in parallel, there are constant adaptive rounds in each iteration or the main loop. Thus, the algorithm needs $O(\ell) = O(\log(n/\delta)/\epsilon^3)$ adaptive complexity.

Let V_j be the set V after Line **??** at iteration j and let Y_i be the number of iterations between $(i-1)$-th success and i-th success. By Lemma C.2, we have $\mathbb{E}[Y_i] \leq 2$. At each iteration j, the algorithm takes $|V_{j-1}|+1$ queries on Line **??**, takes $|\Lambda| \leq |V_j|$ queries on Line **??** and takes total $|V_j|$ queries after the second and the third loops. Therefore, the number of queries is bounded by

$$
\begin{aligned}
\sum_{j=1}^{\ell} \mathbb{E}\left(|V_{j-1}| + 1 + 2|V_j|\right) &\leq n + \ell + \sum_{j=1}^{\ell} 3|V_j| && (118)\\
&\leq n + \ell + 3\sum_{i=1}^{\infty} [Y_i(1-\epsilon\beta)^i n] && (119)\\
&\leq n + \ell + 3\frac{n}{\beta\epsilon} && (120)\\
&= O(\frac{n}{\epsilon^3}). && (121)
\end{aligned}
$$

This completes the proof. □

F. Algorithms and Proofs missed of Section ??

LinAst constructs $O(\frac{1}{\epsilon}\log(\frac{1}{\epsilon}))$ solutions in the main loop with one adaptive round (Lines 4-9, Algorithm 7). For each iteration, the algorithm calls ThreshSeq twice sequentially to get candidate solutions A'_i, B'_i with a threshold τ_i related to the guess of optimal value.

Then, LinAst finds another solution by using USM2 algorithm of (Chen et al., 2019) for USM over the ground set A'_i. Finally, it returns the best among the obtained solutions (Line 10, Algorithm 7).

F.1. LinAst (Algorithm 7) and proof of Theorem ??

Algorithm 7 LinAst

```
1: Input: $f, V, k, \epsilon$.
2: $S_0 \leftarrow \mathsf{LinAdapt}(f, V, k, 1, \epsilon, 1/3)$, $a \leftarrow 12 + (\frac{16}{1-4\epsilon} + \frac{8(2-\epsilon)}{(1-\epsilon)(1-2\epsilon)})\epsilon$
3: $c \leftarrow 6 + \epsilon, \ell \leftarrow \lceil \log_{1-\epsilon}(\frac{1}{a}) \rceil + 1$, $\delta \leftarrow 1/3$, $M \leftarrow af(S_0)/(ck)$
4: for $i \leftarrow 1$ to $\ell$ (in parallel) do
5:     $\tau_i \leftarrow M(1-\epsilon)^i$
6:     $A_i, A'_i \leftarrow \mathsf{ThreshSeq}(f(\cdot), V, k, \tau_i, \epsilon, \delta)$
7:     $B_i, B'_i \leftarrow \mathsf{ThreshSeq}(f(\cdot), V \setminus A_i, k, \tau_i, \epsilon, \delta)$
8:     $A''_i \leftarrow \mathsf{USM2}(A'_i, \epsilon')$
9:     $C_i \leftarrow \arg\max\{f(A'_i), f(B'_i), f(A''_i)\}$
10: end for
11: $S \leftarrow \arg\max_{X \in \{C_i\}_{i=1}^{\ell} \cup \{S_0\}} f(X)$
12: Return $S$
```

Theorem ??. For any input (f, V, k, ϵ), where $\epsilon \in (0, 1)$, LinAst runs in $O(\log(n)/\epsilon^2)$ adaptive complexity and $O(n/\epsilon^3)$ query complexity and returns an approximation ratio of $1/6 - \epsilon$ in expectation with probability of at least $1 - 1/n$.

Proof of Theorem ??. LinAst calls LinAdapt to find S_0. This task takes $O(\frac{\log n}{\epsilon^2})$ adaptivity and $O(\frac{n}{\epsilon^3})$ queries. Then, the algorithm runs in $2\ell = 2(\lceil \log_{1-\epsilon}(\frac{1}{a}) \rceil + 1) = O(\frac{1}{\epsilon}\log(\frac{1}{\epsilon}))$ iterations in one adaptive round. Each iteration takes $O(\frac{n}{\epsilon})$ query complexity and $O(\frac{1}{\epsilon}\log(\frac{n}{\delta}))$ adaptive rounds to call ThreshSeq algorithm. Therefore, the adaptive complexity of LinAst is at most

$$O(\frac{\log n}{\epsilon^2}) + O(\frac{1}{\epsilon}\log(\frac{n}{\delta})) = O(\frac{\log(n)}{\epsilon^2}) \tag{122}$$

and its query complexity is

$$O(\frac{n}{\epsilon^3}) + O(\frac{1}{\epsilon}\log(\frac{1}{\epsilon})\frac{n}{\epsilon}) = O(\frac{n}{\epsilon^3}). \tag{123}$$

The probability that LinAdapt fails is at most $1/(3n)$. In the main loop of LinAst, each iteration returns sets (A_i, A'_i) (or (A_i, A'_i)) with fail probability at most $1/(3n)$. Therefore, the algorithm returns the set C as the final solution with a successful probability of at least $1 - 1/n$ by the union bound. By Theorem ??, we have $f(S_0) \leq \mathsf{opt} \leq af(S_0)$, where $a = (12 + (\frac{16}{1-4\epsilon} + \frac{8(2-\epsilon)}{(1-\epsilon)(1-2\epsilon)})\epsilon)$. Thus $\frac{f(S_0)}{ck} \leq \frac{\mathsf{opt}}{ck} \leq \frac{af(S_0)}{ck} = M$. Besides, $M(1-\epsilon)^i \in [\frac{f(S_0)}{ck}, \frac{af(S_0)}{ck}]$. Therefore, there exists an integer i so that

$$\frac{(1-\epsilon)\mathsf{opt}}{ck} \leq M(1-\epsilon)^i \leq \frac{\mathsf{opt}}{ck}. \tag{124}$$

If the algorithm succeeds, the proof of approximation ratio is similar to the proof of Theorem 7 in (Chen & Kuhnle, 2024). For the sake of completeness, we write down the details of the proof by following. Denote τ_i is the value of τ at iteration i in LinAst.

Case 1: If $|A| = k$ or $|B| = k$. By the theoretical guarantee of ThreshSeq (Theorem **??**), we have

$$f(C) \geq \max\{f(A'), f(B')\} \geq (1-\epsilon)\tau_i \tag{125}$$

$$\geq \frac{(1-\epsilon)^2}{c}\mathsf{opt} \geq (\frac{1}{6} - \epsilon)\mathsf{opt}. \tag{126}$$

Case 2: If $|A| < k$ and $|B| < k$. By Theorem **??**, for any element $e \in V$ we have $f(e|A) \leq \tau_i$ and $f(e|B) \leq \tau_i$. By the submodularity of f, we have

$$f(O \cup A) - f(A) \leq \sum_{e \in O} f(e|A) < k\tau_i \leq \frac{\mathsf{opt}}{c}. \tag{127}$$

$$f((O \setminus A) \cup B) - f(B) \leq \sum_{e \in O \setminus A} f(e|B) < k\tau_i \leq \frac{\mathsf{opt}}{c}. \tag{128}$$

Combine two inequalities (127)-(128) with the fact that $A \cap B = \emptyset$, we have

$$
\begin{aligned}
f(A') + f(B') &\geq f(A) + f(B) && (129)\\
&\geq f(O \cup A) + f((O \setminus A) \cup B) - \frac{2\mathsf{opt}}{c} && (130)\\
&\geq f(O \setminus A) + f(O \cup A \cup B) - \frac{2\mathsf{opt}}{c} \quad \text{(Due to the submodularity)} && (131)\\
&\geq f(O \setminus A) - \frac{2\mathsf{opt}}{c} \quad \text{(Due to the non-negativity)} && (132)
\end{aligned}
$$

Besides, the USM2 gives approximation ratio of $1/(2+\epsilon)$, we have $\mathbb{E}[f(A'')] \geq f(O \cup A)$. Combine this with (132), we have

$$
\begin{aligned}
\mathsf{opt} \leq f(O) &\leq f(O \cup A) + f(O \setminus A) && (133)\\
&\leq (2+\epsilon)\mathbb{E}[f(C)] + 2f(C) + \frac{2\mathsf{opt}}{c}. && (134)
\end{aligned}
$$

It follows that $\mathbb{E}[f(C)] \geq \frac{\mathsf{opt}}{c} \geq (\frac{1}{6} - \epsilon)\mathsf{opt}$. The proof is completed. ☐

F.2. LinAtg Algorithm (Algorithm 8) and proof of Theorem ??

LinAtg works in a different way from LinAst. It sequentially constructs two pairs of disjoint sets (A, B) and (A', B') in two main loops which contain at most $\ell = O(\frac{1}{\epsilon}\log(\frac{1}{\epsilon}))$ iterations (Lines 5-14, Algorithm 8). At each iteration, the algorithm calls ThreshSeq over appropriate ground sets to select two batches of elements with high marginal gain (S, S') and adds them into (A, A') (or (B, B')), respectively. It then selects a candidate solution by adapting the USM2 algorithm of (Chen et al., 2019) over the ground set A'. Finally, LinAtg returns the best one among the obtained feasible solutions (Line 16, Algorithm 8).

Algorithm 8 LinAtg

```
1: Input: $f, V, k, \epsilon$.
2: $S_0 \leftarrow \mathsf{LinAdapt}(f, V, k, 1, \epsilon, 1/3)$, $a \leftarrow 12 + (\frac{16}{1-4\epsilon} + \frac{8(2-\epsilon)}{(1-\epsilon)(1-2\epsilon)})\epsilon$
3: $c \leftarrow 8/\epsilon, \epsilon' \leftarrow (1-1/e)\epsilon/8, \ell \leftarrow \lceil \log_{1-\epsilon'}(\frac{1}{ac}) \rceil + 1, \delta \leftarrow 1/(3\ell), M \leftarrow af(S_0)/k$
4: $A \leftarrow \emptyset, A' \leftarrow \emptyset, B \leftarrow \emptyset, B' \leftarrow \emptyset$
5: for $i \leftarrow 1$ to $\ell$ do
6:    $\tau \leftarrow M(1-\epsilon')^{i-1}$
7:    $A, S' \leftarrow \mathsf{ThreshSeq}(f(A \cup \cdot), A, k - |A|, V, \epsilon', \delta, \tau)$
8:    $A \leftarrow A \cup S, A' \leftarrow A' \cup S'$
9:    If $|A| = k$ then break
10: end for
11: for $i \leftarrow 1$ to $\ell$ do
12:    $\tau \leftarrow M(1-\epsilon')^{i-1}$
13:    $S, S' \leftarrow \mathsf{ThreshSeq}(f(B \cup \cdot), V \setminus A, k - |B|, \epsilon', \delta, \tau)$
14:    $B \leftarrow B \cup S, B' \leftarrow B' \cup S'$
15:    If $|B| = k$ then break
16: end for
17: $A'' \leftarrow \mathsf{USM2}(A', \epsilon')$
18: $C \leftarrow \arg\max\{f(A'), f(B'), f(A''), f(S_0)\}$
19: Return $C$
```

Theorem ??. For any input (f, V, k, ϵ), where $\epsilon \in (0, 1)$, LinAtg runs in $O(\log(1/\epsilon)\log(n)/\epsilon^2)$ adaptive complexity and $O(n/\epsilon^3)$ query complexity, returns an approximation ratio of $0.193 - \epsilon$ in expectation with probability of at least $1 - 1/n$.

Proof of Theorem ??. **Prove the complexities.** The algorithm LinAdapt takes $O(\frac{\log n}{\epsilon^2})$ adaptivity and $O(\frac{n}{\epsilon^3})$ queries to call LinAdapt. Then, the algorithm runs two loops, each loop takes $\ell = \log_{\frac{1}{1-\epsilon'}}(ac)\rceil + 1 = O(\frac{1}{\epsilon}\log(\frac{1}{\epsilon}))$ sequential iterations. In each iteration, it takes $O(\frac{n}{\epsilon})$ query complexity and $O(\frac{1}{\epsilon}\log(\frac{n}{\delta}))$ to call ThreshSeq algorithm, where $\delta = 1/(3\ell)$. Therefore, the adaptive complexity LBA is

$$O(\frac{\log n}{\epsilon^2}) + O(\frac{1}{\epsilon}\log(\frac{1}{\epsilon})\frac{1}{\epsilon}\log(\frac{n}{\delta})) = O\Big(\frac{1}{\epsilon^2}\log(\frac{1}{\epsilon})\log(\frac{n}{\epsilon}\log(\frac{1}{\epsilon}))\Big) \tag{135}$$

$$= O\Big(\frac{1}{\epsilon^2}\log(\frac{1}{\epsilon})\big(\log(n) + \log(\frac{1}{\epsilon}\log(\frac{1}{\epsilon}))\Big) \tag{136}$$

$$= O\Big(\frac{\log(n)}{\epsilon^2}\log(\frac{1}{\epsilon})\Big). \tag{137}$$

and its query complexity is

$$O(\frac{n}{\epsilon^3}) + O(\frac{1}{\epsilon}\log(\frac{1}{\epsilon})\frac{n}{\epsilon}) = O(\frac{n}{\epsilon^3}). \tag{138}$$

The probability that LinAdapt fails is at most $1/n$, and the probability that ThreshSeq fails in each iteration is at most δ. By the union probability, the probability BoostAdapt fails is at most $1/n + 2\ell \cdot \delta = 1/n$.

Prove the approximation ratio. ATG algorithm in (Chen & Kuhnle, 2024) uses $f(e_{max})$ to give the bound $f(O)/k \leq f(e_{max}) \leq f(O)$. Our LinAtg uses $M = af(S_0)/k$ to give a bound of opt/k. By Theorem **??**, we have $f(S_0) \leq \mathsf{opt} \leq af(S_0)$, where $a = \Big(12 + (\frac{16}{1-4\epsilon} + \frac{8(2-\epsilon)}{(1-\epsilon)(1-2\epsilon)})\epsilon\Big)$. Thus

$$\frac{M}{a} = \frac{f(S_0)}{k} \leq \frac{\mathsf{opt}}{k} \leq \frac{af(S_0)}{k} = M \tag{139}$$

implying that $M(1-\epsilon')^i \in [\frac{M}{ca}, M] \supseteq [\frac{\mathsf{opt}}{ck}, \frac{\mathsf{opt}}{k}]$. From the main loops, LBA works similarly to ATG. Therefore, we follow the proof of Theorem 8 in (Chen & Kuhnle, 2024). For the sake of completeness, we write down the details of the proof below. The notations used in the proof are listed below.

- A' and A' are returned by the first loop of Algorithm 8, while B and B' are returned by the second one.
- Define A_i as A after iteration i.
- Let $\mathcal{A}'_j$ be the first j elements in A, where $1 \leq j \leq |A|$. Furthermore, for $|A'| < j \leq k$, let $\mathcal{A}'_j$ be A' combined with $k - |A'|$ dummy elements.
- Let $\{a'_j\} = \mathcal{A}'_j \setminus \mathcal{A}'_{j-1}$, a'_j be returned at iteration $i(j)$, and $A_{i(j)}$ be the set A returned at iteration $i(j)$. If $\{a'_j\}$ is dummy element, let $i(j) = \ell + 1$. Then, we likewise define B_i, $\mathcal{B}'_j$ $B_{i(j)}$.

Lemma F.1 (Lemma 9 in (Chen & Kuhnle, 2024))**.** *For $1 \leq j \leq k$, there are at least $\lceil(1-\epsilon')k\rceil$ of j such that*

$$f(\mathcal{A}'_j) - f(\mathcal{A}'_{j-1}) + \frac{M}{ck} \geq \frac{1-\epsilon}{k}(f(O \cup A_{i(j)-1}) - f(\mathcal{A}'_{j-1})). \tag{140}$$

And for any j,

$$f(\mathcal{A}'_j) \geq f(\mathcal{A}'_{j-1}). \tag{141}$$

Lemma F.2 (Lemma 10 in (Chen & Kuhnle, 2024))**.** *For $1 \leq j \leq k$, there are at least $\lceil(1-\epsilon')k\rceil$ of j such that*

$$f(\mathcal{B}'_j) - f(\mathcal{B}'_{j-1}) + \frac{M}{ck} \geq \frac{1-\epsilon}{k}(f((O \setminus A) \cup B_{i(j)-1}) - f(\mathcal{B}'_{j-1})). \tag{142}$$

And for any j,

$$f(\mathcal{B}'_j) \geq f(\mathcal{B}'_{j-1}). \tag{143}$$

Lemma F.3 (Lemma 11 in (Chen & Kuhnle, 2024)). *Let $\Gamma_u = f(\mathcal{A}'_{j(u)}) + f(\mathcal{B}'_{j(u)})$, where $j(u)$ is the u-th j which satisfies Lemma F.1 or Lemma F.2. Then, there are at least $\lceil (1-\epsilon')k \rceil$ of u follow that*

$$f(O \setminus A) - \Gamma_u - \frac{2M}{c(1-\epsilon')} \leq \left(1 - \frac{1-\epsilon'}{k}\right)\left(f(O \setminus A) - \Gamma_{u-1} - \frac{2M}{c(1-\epsilon')}\right). \tag{144}$$

Lemma F.3 yields a recurrence of the form $(b - u_{i+1}) \leq a(b - u_i)$, $u_0 = 0$, and has a solution $u_i \geq b(1-a^i)$. Consequently, we have

$$f(A') + f(B') \geq \left(1 - \left(1 - \frac{1-\epsilon'}{k}\right)^{(1-\epsilon')k}\right)\left(f(O \setminus A) - \frac{2M}{c(1-\epsilon')}\right) \tag{145}$$

$$\geq \left(1 - e^{-(1-\epsilon')^2}\right)\left(f(O \setminus A) - \frac{2M}{c(1-\epsilon')}\right). \tag{146}$$

Let $\beta = 1 - e^{-(1-\epsilon')^2}$. From the choice of C on line 16, Algorithm 8, we have $2f(C) \geq f(A') + f(B')$ and from (146), we have

$$f(O \setminus A) \leq \frac{2}{\beta} f(C) + \frac{2M}{c(1-\epsilon')^2} \tag{147}$$

$$\leq \frac{2}{\beta} f(C) + \frac{2f(O)}{c(1-\epsilon')^2}. \tag{148}$$

Assume USM2 has a ratio of $1/\rho$ for USM problem. For any A, $f(O \cap A)/\rho \leq \mathbb{E}[f(A'')|A]$; therefore

$$f(O \cap A) \leq \rho \mathbb{E}[f(C)]. \tag{149}$$

For any set A, $f(O) \leq f(O \cap A) + f(O \setminus A)$ by the submodularity and nonnegativity. Thus

$$f(O) \leq f(O \cap A) + f(O \setminus A) \tag{150}$$

$$\leq \frac{2}{\beta} f(C) + \frac{2f(O)}{c(1-\epsilon')^2} + \rho. \tag{151}$$

Therefore

$$f(C) \geq \frac{1 - \frac{2}{c(1-\epsilon')}}{\rho + \frac{2}{\beta}} f(O) \geq \left(\frac{e-1}{\rho(e-1) + 2e} - \epsilon\right) f(O). \tag{152}$$

By replacing $\rho = 2 + \epsilon$ for USM2, we obtain the ratio. □

G. Proofs missed of Section ??

We recap the following notations used for analyzing the BoostAdapt's performance guarantees

- Supposing X and Y ordered: $X = \{x_1, x_2, \ldots, x_{|X|}\}$, $Y = \{y_1, y_2, \ldots, y_{|Y|}\}$, we conduct: $X^i = \{x_1, x_2, \ldots, x_i\}$, $Y^i = \{y_1, y_2, \ldots, y_i\}$, where x_i (or y_i) is i-th element added into X (or Y).
- For $e \in X \cup Y$, we assume that e is added into X or Y at iteration i_e.
- X_i and Y_i are X and Y after the iteration i of the main loop and $X_0 = Y_0 = \emptyset$. Then, we define X'_i and Y'_i analogously.
- τ_X^i and τ_Y^i are τ_X and τ_Y after the iteration i of the main loop of BoostAdapt.
- τ_X^{last} and τ_Y^{last} are τ_X and τ_Y at the last iterations of the main loop of BoostAdapt when X and Y are considered to update.

- Define A_i^+ (res. B_i^+) as the set of elements in A_i (res. B_i) having the marginal gain at least τ_X (res. τ_Y) at iteration i, $X_i^+ = \cup_{j=1}^i A_i^+$, $Y_i^+ = \cup_{j=1}^i B_i^+$ and $X^+ = X_\Delta^+, Y^+ = Y_\Delta^+$.
- For an element $e \in X \cup Y$, we denote by $X^{<e}, Y^{<e}$ as X, Y right before e is selected into X or Y, respectively.

Lemma ??. **a)** For any iteration i, we have $f(X_i') \geq f(X_i)$ and $f(Y_i') \geq f(Y_i)$.
b) At the end of BoostAdapt, we have $f(X'') \geq (1-\epsilon) f(X')$ and $f(Y'') \geq (1-\epsilon) f(Y')$.

Proof of **??**. **a)** We prove the Lemma by the induction hypothesis. For $i = 1$, we have $f(X_1') = f(A_1') \geq f(A_1) = f(X_1)$ due to the Theorem **??**, so the Proposition holds. Assuming the proposition holds for j, we have:

$$f(X_{j+1}') = f(A_{j+1}' \cup X_j') = f(A_{j+1}'|X_j') + f(X_j') \tag{153}$$

$$\geq f(A_{j+1}'|X_j) + f(X_j) \tag{154}$$

$$\geq f(A_{j+1}|X_j) + f(X_j) = f(X_{j+1}) \tag{155}$$

where the inequality (154) is due to the submodularity of f and $X_j' \subseteq X_j$, inequality (155) is due to the theoretical bound of ThreshSeq (Theorem **??**). Therefore, by the induction hypothesis, we have $f(X_i') \geq f(X_i)$ for any iteration i.

By similar reasoning, we can prove $f(Y_i') \geq f(Y_i)$ for any iteration i, which completes the proof of **a)**.
b). If $|X'| = k$, then $X'' = X'$ the Lemma holds. If $|X'| > k$, by the setting of k' and the selection rule of X'', we get $|X''| = k \geq k'(1-\epsilon) \geq |X|(1-\epsilon) \geq |X'|(1-\epsilon)$. Therefore $|X' \setminus X''| \leq \epsilon |X'|$. Elements in $|X'' \setminus X'|$ have lowest marginal so we obtain

$$f(X') = \sum_{e \in X'} f(e|X'^{<e}) = \sum_{e \in X''} f(e|X'^{<e}) + \sum_{e \in X' \setminus X''} f(e|X'^{<e}) \tag{156}$$

$$\leq \sum_{e \in X''} f(e|X'^{<e}) + \epsilon \sum_{e \in X'} f(e|X'^{<e}) \tag{157}$$

$$\leq \sum_{e \in X''} f(e|X''^{<e}) + \epsilon f(X') \quad \text{(Due to the submodularity and } X'' \subseteq X') \tag{158}$$

$$= f(X'') + \epsilon f(X') \tag{159}$$

which implies $f(X'') \geq (1-\epsilon) f(X')$. By similar reasoning, we can prove $f(Y'') \geq (1-\epsilon) f(Y')$. This completes the proof of **b)**. ☐

Lemma ??. If BoostAdapt succeeds and $X_1 = \emptyset$, after iteration i the following properties hold:
(a) If i is odd and $|X_i| < k'$: for any sets $C \subseteq Y_{i-1}^+$ and $C' \subseteq Y_{i-1} \setminus Y_{i-1}^+$, we have

$$\sum_{x \in C} f(x|X_i) \leq \sum_{x \in C} \frac{f(e|Y^{<e})}{1-\epsilon} \quad \text{and} \quad \sum_{e \in C'} f(e|X_i) = \frac{\epsilon}{1-\epsilon} f(Y_{i-1}^+). \tag{160}$$

(b) If i is even and $|Y_i| < k'$, for any set $D \subseteq X_{i-1}^+$ and $D' \subseteq X_{i-1} \setminus X_{i-1}^+$, we have

$$\sum_{x \in D} f(x|Y_i) \leq \sum_{x \in D} \frac{f(e|X^{<e})}{1-\epsilon} \quad \text{and} \quad \sum_{e \in D'} f(e|Y_i) = \frac{\epsilon}{1-\epsilon} f(X_{i-1}^+). \tag{161}$$

Proof of Lemma **??**. **a)** For $i \geq 1$, i is odd. If $i = 1$, $X_1 = Y_0 = \emptyset$. Thus, $C = \emptyset$ and the Lemma holds. If $i \geq 3$, let $C = C_2 \cup C_4 \cup \ldots \cup C_{i-1}$, where C_j $(j \leq i-1)$ is the set of elements in C added into Y_j^+, i.e., $C_j = B_j^+ \cap C$. Each element $e \in C_j$ is selected into Y_j^+ at iteration $j \leq i-1$ and has not been added yet into X at the previous iteration. By the Theorem **??** and the submodularity of f, for any $e \in C_j$ we have

$$f(e|X_i) \leq f(e|X_{j-1}) < \tau_X^{j-1} = \frac{\tau_Y^j}{1-\epsilon} \leq \frac{f(e|Y^{<e})}{1-\epsilon}. \tag{162}$$

It follows that

$$\sum_{e \in C} f(e|X_i) = \sum_{j=2,4,\ldots,i-1} \sum_{e \in C_j} f(e|X_i) \tag{163}$$

$$< \sum_{j=2,4,\ldots,i-1} \sum_{e \in C_j} \frac{f(e|Y^{<e})}{1-\epsilon} \tag{164}$$

$$= \sum_{e \in C} \frac{f(e|Y^{<e})}{1-\epsilon}. \tag{165}$$

Each element $e \in B_j \setminus B_j^+$ also has not been added yet into X at the previous iteration, so we have

$$f(e|X_i) \leq f(e|X_{j-1}) < \tau_X^{j-1} = \frac{\tau_Y^j}{1-\epsilon}. \tag{166}$$

Since $|B_j^+| \geq (1-\epsilon)|B_j|$, $|B_j \setminus B_j^+| \leq \epsilon|B_j| \leq \frac{\epsilon}{1-\epsilon}|B_j^+|$. Besides, due to the fact that $Y_i = \bigcup_{j=1}^i B_j$ and $Y_i^+ = \bigcup_{j=1}^i B_j^+$, therefore we have:

$$\sum_{e \in C'} f(e|X_i) = \sum_{j=2,4,\ldots,i-1} \sum_{e \in C' \cap (B_j \setminus B_j^+)} f(e|X_i) \tag{167}$$

$$< \sum_{j=2,4,\ldots,i-1} \sum_{e \in B_j \setminus B_j^+} \frac{\tau_Y^j}{1-\epsilon} \tag{168}$$

$$\leq \sum_{j=2,4,\ldots,i-1} \epsilon|B_j| \frac{\tau_Y^j}{1-\epsilon} \tag{169}$$

$$\leq \sum_{j=2,4,\ldots,i-1} \frac{\epsilon}{1-\epsilon}|B_j^+| \frac{\tau_Y^j}{1-\epsilon} \tag{170}$$

$$\leq \frac{\epsilon}{1-\epsilon} \sum_{j=2,4,\ldots,i-1} \sum_{e \in B_j^+} \frac{f(e|Y^{<e})}{1-\epsilon} \tag{171}$$

$$= \frac{\epsilon}{1-\epsilon} f(Y_{i-1}^+). \tag{172}$$

b) We prove this case by the same reasoning with part **a)**
If $i \geq 2$, i is even. If $i = 2$, since $X_1 = \emptyset$, thus $D = \emptyset$ and the Lemma holds. If $i \geq 4$, let $D = D_1 \cup D_3 \cup \ldots \cup D_{i-1}$, where $D_j = A_j^+ \cap D$. Each $e \in D$ is selected into X at iteration $j, 3 \leq j \leq i-1$ and has not been added yet into Y at the previous iteration. By the Theorem **??**, for any $e \in D_j$ we have

$$f(e|Y_i) \leq f(e|Y_{j-1}) < \tau_Y^{j-1} = \frac{\tau_X^j}{1-\epsilon} \leq \frac{f(e|X^{<e})}{1-\epsilon} \tag{173}$$

Therefore

$$\sum_{e \in D} f(e|Y_i) = \sum_{j=3,5,\ldots,i-1} \sum_{e \in D_j} f(e|Y_i) \tag{174}$$

$$< \sum_{j=3,5,\ldots,i-1} \sum_{e \in D_j} \frac{f(e|X^{<e})}{1-\epsilon} \tag{175}$$

$$\leq \sum_{e \in D} \frac{f(e|X^{<e})}{1-\epsilon}. \tag{176}$$

Each element $e \in A_j \setminus A_j^+$ also has not been added yet into Y at the previous iteration, so we have $f(e|X_i) \leq f(e|X_{j-1}) <$

$\tau_X^{j-1} = \frac{\tau_Y^j}{1-\epsilon}$. We also have:

$$\sum_{e\in D'} f(e|X_i) = \sum_{j=2,4,\ldots,i-1} \sum_{e\in D'\cap(A_j\setminus A_j^+)} f(e|Y_i) \tag{177}$$
$$< \sum_{j=2,4,\ldots,i-1} \sum_{e\in A_j\setminus A_j^+} \frac{\tau_X^j}{1-\epsilon} \tag{178}$$
$$\leq \sum_{j=2,4,\ldots,i-1} \epsilon|A_j|\frac{\tau_X^j}{1-\epsilon} \tag{179}$$
$$\leq \sum_{j=2,4,\ldots,i-1} \frac{\epsilon}{1-\epsilon}|A_j^+|\frac{\tau_X^j}{1-\epsilon} \tag{180}$$
$$\leq \frac{\epsilon}{1-\epsilon}\sum_{j=2,4,\ldots,i-1} \sum_{e\in X_j^+} \frac{f(e|X^{<e})}{1-\epsilon} \tag{181}$$
$$= \frac{\epsilon}{1-\epsilon} f(X_{i-1}^+). \tag{182}$$

which completes the proof. □

Using Lemma **??** and the fact that if $|T| = k$, then $|T \setminus O| \geq |O \setminus T|$, where $T \in \{X^+, Y^+\}$, we can bound of the optimal solution in some cases related to the size of X and Y in Lemma **??**.
Lemma ?? If BoostAdapt succeeds and $X_1 = \emptyset$, at the end BoostAdapt we have
a) If $|X| < k'$ and $|Y| < k'$, then $f(O) < \frac{4f(S)}{(1-\epsilon)^2(1-2\epsilon)}$.
b) If there exists $T \in \{X, Y\}$ such that $|T| = k'$, then $f(S) \geq (1-\epsilon)^4 \frac{\mathsf{opt}}{4}$.

Proof of Lemma **??**. We consider following cases:
Case 1. If $|X| < k'$ and $|Y| < k'$, by the definition of Δ in Algorithm **??**, we have $\frac{M}{4k} \geq \tau(1-\epsilon)^i \geq \frac{\epsilon(1-\epsilon)f(S_0)}{k}$. Thus $\tau_X^{last} \leq \epsilon f(S_0)/k \leq \epsilon \mathsf{opt}/k$. Note that

By the submodularity and applying Lemma **??** and Theorem **??**, we have

$$f(X\cup O) - f(X) \leq \sum_{e\in O\setminus X} f(e|X) \tag{183}$$
$$= \sum_{e\in Y^+\cap O} f(e|X) + \sum_{e\in (Y\setminus Y^+)\cap O} f(e|X) + \sum_{e\in O\setminus(X\cup Y^+)} f(e|X) \tag{184}$$
$$< \frac{\sum_{e\in Y^+\cap O} f(e|Y^{<e})}{1-\epsilon} + \frac{\epsilon}{1-\epsilon} f(Y^+) + k\tau_X^{last} \tag{185}$$
$$\leq \frac{f(Y^+)}{1-\epsilon} + \frac{\epsilon}{1-\epsilon} f(Y^+) + k\tau_X^{last} \tag{186}$$
$$\leq \frac{1+\epsilon}{1-\epsilon} f(Y') + \epsilon\mathsf{opt} \tag{187}$$

where inequality (183) is due to the submodularity of f; inequality (185) is due to: applying Lemma **??** with $C = Y^+ \cap O$, and $C' = (Y \setminus Y^+) \cap O$ and applying Theorem **??**; inequality (187) due to Lemma **??**, i.e, $f(Y') \geq f(Y^+)$.

Similarity, since $X_1 = \emptyset$ and by applying Lemma **??** and Theorem **??** we have

$$
\begin{aligned}
f(Y \cup O) - f(Y) &\leq \sum_{e \in O \setminus Y} f(e|Y) \quad \text{(Due to the submodularity of } f\text{)} && (188)\\
&= \sum_{e \in O \cap X^+} f(e|Y) + \sum_{e \in (X \setminus X^+) \cap O} f(e|Y) + \sum_{e \in O \setminus (X^+ \cup Y)} f(e|Y) && (189)\\
&< \frac{\sum_{e \in O \cap X^+} f(e|X^{<e})}{1-\epsilon} + \frac{\epsilon}{1-\epsilon} f(X^+) + k\tau_Y^{last} && (190)\\
&\leq \frac{f(X^+)}{1-\epsilon} + \frac{\epsilon}{1-\epsilon} f(X^+) + k\tau_Y^{last} && (191)\\
&\leq \frac{1+\epsilon}{1-\epsilon} f(X') + \epsilon \mathsf{opt}. && (192)
\end{aligned}
$$

Combining two inequalities (187) and (192) and the selection rule of the final solution we have

$$
\begin{aligned}
f(O) &\leq f(O \cup X) + f(O \cup Y) \quad \text{(Due to the submodularity of } f\text{)} && (193)\\
&< \frac{1+\epsilon}{1-\epsilon}\Big(f(X') + f(Y')\Big) + f(X) + f(Y) + 2\epsilon \mathsf{opt} && (194)\\
&= \frac{2}{1-\epsilon}\Big(f(X') + f(Y')\Big) + 2\epsilon \mathsf{opt} && (195)\\
&\leq \frac{2}{(1-\epsilon)^2}\Big(f(X'') + f(Y'')\Big) + 2\epsilon \mathsf{opt} \quad \text{(Due to Lemma C.2)} && (196)\\
&\leq \frac{4f(S)}{(1-\epsilon)^2} + 2\epsilon \mathsf{opt} && (197)
\end{aligned}
$$

which implies the proof of a).

Case 2. If $|X| = k'$ and $|Y| = k'$. Since $X_1 = \emptyset$ we must have $i_{x_{k'}} > 1$. If $i_{y_{k'}} = 2$, we have

$$
f(S) \geq f(Y'_2) \geq (1-\epsilon)|Y_2|\tau_Y^2 \geq \frac{(1-\epsilon)^3 \mathsf{opt}}{4}. \tag{198}
$$

thus the Lemma holds.

If $i_{y_{k'}} > 2$, then each element e that is not added into X at iteration $i_{x_{k'}}$ also has not been added into X at iteration $i_{x_{k'}} - 2$. By applying Lemma **??** with a note that $|X_{i_{x_{k'}}-2}| < k'$, we have $f(e|X) < \tau_X^{i_{x_{k'}}-2}$. Therefore

$$
\begin{aligned}
f(X \cup O) - f(X) &\leq \sum_{e \in O \setminus X} f(e|X) && (199)\\
&= \sum_{e \in Y^+_{i_{x_{k'}}-1} \cap O} f(e|X) + \sum_{e \in (Y_{i_{x_{k'}}-1} \setminus Y^+_{i_{x_{k'}}-1}) \cap O} f(e|X) + \sum_{e \in O \setminus (X \cup Y^+_{i_{x_{k'}}-1})} f(e|X^{<e}) && (200)\\
&< \frac{\sum_{e \in Y^+_{i_{x_{k'}}-1} \cap O} f(e|Y^{<e})}{1-\epsilon} + \frac{\epsilon}{1-\epsilon} f(Y^+_{i_{x_{k'}}-1}) + |O \setminus X| \cdot \tau_X^{i_{x_{k'}}-2}. && (201)\\
&\leq \frac{\sum_{e \in Y^+ \cap O} f(e|Y^{<e})}{1-\epsilon} + \frac{\epsilon}{1-\epsilon} f(Y^+) + |O \setminus X| \cdot \tau_X^{i_{x_{k'}}-2}. && (202)
\end{aligned}
$$

where inequality (201) is due to: applying Lemma **??** with $C = Y^+_{i_{x_{k'}}-1} \cap O$, and $C' = (Y_{i_{x_{k'}}-1} \setminus Y^+_{i_{x_{k'}}-1}) \cap O$ and applying Theorem **??**.

Similarly, each element e is not added to Y at iteration $i_{y_{k'}}$ also is not added to Y at iteration $i_{y_{k'}} - 2$. By Lemma **??**, we have:

$$
\begin{aligned}
f(Y \cup O) - f(Y) &\leq \sum_{e \in O \cap X^+} f(e|Y) + \frac{\epsilon}{1-\epsilon} f(X^+) + \sum_{e \in O \setminus (Y \cup X^+_{i_{y_{k'}}-1})} f(e|Y) && (203)\\
&\leq \frac{\sum_{e \in O \cap X^+} f(e|X^{<e})}{1-\epsilon} + \frac{\epsilon}{1-\epsilon} f(X^+) + |O \setminus Y| \cdot \tau_Y^{i_{y_{k'}}-2}. && (204)
\end{aligned}
$$

On the other hand, since $|X| = k' > k \geq |O|$. Since at each iteration i, $\mathsf{ThreshSeq}$ selects at least $(1-\epsilon)|A_i|$ element with marginal gain is above the threshold, so $|A_i^+| \geq (1-\epsilon)|A_i|$. Thus $|X^+| \geq (1-\epsilon)|X| = (1-\epsilon)k'$. If $(1-\epsilon)k'$ is an integer, then $(1-\epsilon)k' = k$ and thus $|X^+| = k$. If $(1-\epsilon)k'$ is not an integer, then $(1-\epsilon)k' < k$, $\lceil (1-\epsilon)k' \rceil = k$. Since $|X^+|$ is an integer, we have $|X^+| = \lceil (1-\epsilon)k' \rceil = k$. Overall, $|X^+| = k$ which implies $|X^+ \setminus O| \geq |O \setminus X^+| \geq |O \setminus X'| \geq |O \setminus X|$. By the definition of X', X^+ we have

$$\sum_{e \in X^+ \setminus O} f(e|X^{<e}) \geq |X^+ \setminus O| \cdot \tau_X^{i_{x_{k'}}} \tag{205}$$

$$\geq (1-\epsilon)^2 |O \setminus X^+| \tau_X^{i_{x_{k'}}-2} \tag{206}$$

$$\geq (1-\epsilon)^2 |O \setminus X| \tau_X^{i_{x_{k'}}-2} \tag{207}$$

$$\Longrightarrow |O \setminus X| \tau_X^{i_{x_{k'}}-2} \leq \frac{\sum_{e \in X^+ \setminus O} f(e|X^{<e})}{(1-\epsilon)^2}. \tag{208}$$

Similarly, since $|Y| = k'$, $|Y^+| \geq k$ and $|Y^+ \setminus O| \geq |O \setminus Y^+|$ we have

$$\sum_{e \in Y^+ \setminus O} f(e|Y^{<e}) \geq (1-\epsilon)^2 |O \setminus Y| \tau_Y^{i_{y_k}-2} \tag{209}$$

$$\Longrightarrow |O \setminus Y| \tau_Y^{i_{y_k}-2} \leq \frac{\sum_{e \in Y^+ \setminus O} f(e|Y^{<e})}{(1-\epsilon)^2}. \tag{210}$$

Combining (202), (204), (208), (210), the fact that $f(X^+) \leq f(X'), f(Y^+) \leq f(Y')$ and Lemma **??** we get

$$\begin{aligned}
f(O) &\leq f(O \cup X) + f(O \cup Y) && (211)\\
&\leq \frac{\sum_{e \in Y^+ \cap O} f(e|Y^{<e})}{1-\epsilon} + \frac{\sum_{e \in Y^+ \setminus O} f(e|Y^{<e})}{(1-\epsilon)^2} && (212)\\
&\quad + \frac{\sum_{e \in O \cap X^+} f(e|X^{<e})}{1-\epsilon} + \frac{\sum_{e \in X^+ \setminus O} f(e|X^{<e})}{(1-\epsilon)^2} + \frac{\epsilon}{1-\epsilon}(f(X^+) + f(Y^+)) + f(X) + f(Y) && (213)\\
&\leq \frac{\sum_{e \in Y^+} f(e|Y^{<e})}{(1-\epsilon)^2} + \frac{\sum_{e \in X^+} f(e|X^{<e})}{(1-\epsilon)^2} + \frac{f(X') + f(Y')}{1-\epsilon} && (214)\\
&\leq \frac{\sum_{e \in Y'} f(e|Y^{<e})}{(1-\epsilon)^2} + \frac{\sum_{e \in X'} f(e|X^{<e})}{(1-\epsilon)^2} + \frac{f(X') + f(Y')}{1-\epsilon} && (215)\\
&\leq \frac{f(Y')}{(1-\epsilon)^2} + \frac{f(X')}{(1-\epsilon)^2} + \frac{f(X') + f(Y')}{1-\epsilon} && (216)\\
&< \frac{2f(Y')}{(1-\epsilon)^2} + \frac{2f(X')}{(1-\epsilon)^2} = 2\frac{f(X') + f(Y')}{(1-\epsilon)^2} && (217)\\
&\leq 2\frac{f(X'') + f(Y'')}{(1-\epsilon)^3} \quad \text{(Due to Lemma ??)}. && (218)
\end{aligned}$$

Case 3. If $|Y| = k'$ and $|X| < k'$. If $i_{y_k} = 2$, we have

$$f(S) \geq f(Y_2) \geq (1-\epsilon)^2 \tau_Y^2 \geq (1-\epsilon)^2 \frac{\mathsf{opt}}{4} \tag{219}$$

the Lemma holds. If $i_{y_k} > 2$, similar to the transformation of **Case 2** of this proof, we also have

$$f(Y \cup O) - f(Y) \leq \frac{\sum_{e \in O \cap X^+} f(e|X^{<e})}{1-\epsilon} + \frac{\epsilon}{1-\epsilon} f(X^+) + |O \setminus Y| \cdot \tau_Y^{i_{y_{k'}}-2} \tag{220}$$

$$\leq \frac{\sum_{e \in O \cap X^+} f(e|X^{<e})}{1-\epsilon} + \frac{\epsilon}{1-\epsilon} f(X^+) + \frac{\sum_{e \in Y^+ \setminus O} f(e|Y^{<e})}{(1-\epsilon)^2} \tag{221}$$

where inequality (221) is due to the fact that

$$|O \setminus Y| \tau_Y^{i_{y_k}-2} \leq \frac{\sum_{e \in Y^+ \setminus O} f(e|Y^{<e})}{(1-\epsilon)^2}. \tag{222}$$

Since $|X| < k'$, similar to the transformations of **Case 1**, we have

$$f(X \cup O) - f(X) \leq \sum_{e \in O \setminus X} f(e|X) \tag{223}$$

$$= \sum_{e \in Y^+ \cap O} f(e|X) + \sum_{e \in (Y \setminus Y^+) \cap O} f(e|X) + \sum_{e \in O \setminus (X \cup Y^+)} f(e|X) \tag{224}$$

$$< \frac{\sum_{e \in Y^+ \cap O} f(e|Y^{<e})}{1-\epsilon} + \frac{\epsilon}{1-\epsilon} f(Y^+) + k\tau_X^{last}. \tag{225}$$

$$\leq \frac{\sum_{e \in Y^+ \cap O} f(e|Y^{<e})}{1-\epsilon} + \frac{\epsilon}{1-\epsilon} f(Y') + \epsilon \mathsf{opt}. \tag{226}$$

It follows that

$$f(O) \leq f(O \cup Y) + f(O \cup X) \tag{227}$$

$$< \frac{\sum_{e \in Y^+ \cap O} f(e|Y^{<e})}{1-\epsilon} + \frac{\sum_{e \in Y^+ \setminus O} f(e|Y^{<e})}{(1-\epsilon)^2} + \frac{\sum_{e \in O \cap X^+} f(e|X^{<e})}{1-\epsilon} \tag{228}$$

$$+ \frac{\epsilon}{1-\epsilon}(f(X') + f(Y')) + f(X) + f(Y) + \epsilon \mathsf{opt} \tag{229}$$

$$\leq \frac{\sum_{e \in Y^+} f(e|Y^{<e})}{(1-\epsilon)^2} + \frac{\sum_{e \in X^+} f(e|X^{<e})}{1-\epsilon} + \frac{\epsilon}{1-\epsilon}(f(X') + f(Y')) + f(X) + f(Y) + \epsilon \mathsf{opt} \tag{230}$$

$$< \frac{f(Y')}{(1-\epsilon)^2} + \frac{f(X')}{1-\epsilon} + \frac{\epsilon}{1-\epsilon}(f(X') + f(Y')) + f(X) + f(Y) + \epsilon \mathsf{opt} \tag{231}$$

$$< 2\frac{f(Y') + f(Y')}{(1-\epsilon)^2} + \epsilon \mathsf{opt} \tag{232}$$

$$\leq 2\frac{f(X'') + f(Y'')}{(1-\epsilon)^3} + \epsilon \mathsf{opt}. \tag{233}$$

Case 4. If $|Y| < k'$ and $|X| = k'$. Since $X_1 \neq \emptyset$ we must have $i_{x_k} > 1$. Similar to the **Case 2**, we have

$$f(X \cup O) - f(X) < \frac{\sum_{e \in Y^+ \cap O} f(e|Y^{<e})}{1-\epsilon} + \frac{\epsilon}{1-\epsilon} f(Y^+) + \frac{\sum_{e \in X^+ \setminus O} f(e|X^{<e})}{(1-\epsilon)^2}. \tag{234}$$

Since $|Y| < k'$ similar the **Case 1**, we have

$$f(Y \cup O) - f(Y) \leq \frac{\sum_{e \in O \cap X^+} f(e|X^{<e})}{1-\epsilon} + \frac{\epsilon}{1-\epsilon} f(X^+) + k\tau_Y^{last}. \tag{235}$$

Similar to the transformation at the end of **Case 3**, we have

$$f(O) \leq f(O \cup X) + f(O \cup Y) \tag{236}$$

$$< \frac{\sum_{e \in Y'} f(e|Y^{<e})}{1-\epsilon} + \frac{\sum_{e \in X'} f(e|X^{<e})}{(1-\epsilon)^2} + \frac{\epsilon}{1-\epsilon}(f(X') + f(Y')) + f(X) + f(Y) + \epsilon \mathsf{opt}. \tag{237}$$

$$\leq 2\frac{f(X'') + f(Y'')}{(1-\epsilon)^3} + \epsilon \mathsf{opt}. \tag{238}$$

Combine **Case 2**, **Case 3**, **Case 4**, we have:

$$f(O) \leq 2\frac{f(X'') + f(Y'')}{(1-\epsilon)^4} \leq \frac{4f(S)}{(1-\epsilon)^4} \tag{239}$$

which completes the proof of b). □

Proof of Theorem **??**. **Prove successful probability and complexities.** BoostAdapt first calls LinAdapt to find S_0 in $O(\frac{\log n}{\epsilon^2})$ adaptivity and $O(\frac{n}{\epsilon^3})$ queries. Then, the algorithm runs in $\Delta = O(\frac{1}{\epsilon}\log(\frac{k}{\epsilon}))$ iterations; each takes $O(\frac{n}{\epsilon})$ query complexity and $O(\frac{1}{\epsilon}\log(\frac{n}{\delta}))$ adaptive rounds to call ThreshSeq algorithm. Therefore, the adaptive complexity BoostAdapt is

$$O(\frac{\log n}{\epsilon^2}) + O\Big(\frac{1}{\epsilon}\log(\frac{k}{\epsilon})\frac{1}{\epsilon}\log(\frac{n}{\delta})\Big) = O\Big(\frac{1}{\epsilon^2}\log(\frac{k}{\epsilon})\log(\frac{n}{\epsilon}\log(\frac{k}{\epsilon}))\Big) \tag{240}$$

$$= O\Big(\frac{1}{\epsilon^2}\log(\frac{k}{\epsilon})\log(\frac{n}{\epsilon}) + \frac{1}{\epsilon^2}\log(\frac{k}{\epsilon})\log(\frac{k}{\epsilon}))\Big) \tag{241}$$

$$= O\Big(\frac{1}{\epsilon^2}\log(\frac{k}{\epsilon})\log(\frac{n}{\epsilon})\Big) \tag{242}$$

and its query complexity is

$$O(\frac{n}{\epsilon^3}) + O(\frac{1}{\epsilon}\log(\frac{k}{\epsilon})\frac{n}{\epsilon}) = O(\frac{n}{\epsilon^2}\log(\frac{k}{\epsilon})) \tag{243}$$

The probability that LinAdapt fails is at most $1/(3n)$, and the probability that ThreshSeq fails in each iteration is at most δ. By the union probability, the probability BoostAdapt fails is at most $1/(3n) + \Delta \cdot \delta = 1/n$. □

H. Additional Experimental Details and Results

In this section, we elaborate on the experimental configuration and provide additional empirical findings along with some discussions.

H.1. Algorithms and Settings

We compare our algorithms LinAst, LinAtg and BoostAdapt with state-of-the-arts for non-monotone SMC including:

- **IteratedGreedy** (IG): The algorithm in (Gupta et al., 2010) achieves $1/6 - \epsilon$ approximation ratio when the $1/2$-approximation of (Buchbinder et al., 2012) is used for the unconstrained maximization subproblem. The algorithm takes $O(nk)$ query and adaptive complexities.
- **AdaptiveNonmonotoneMax** (ANM): The algorithm in (Fahrbach et al., 2023) achieves $0.039 - \epsilon$ approximation ratio with $O(\log(n))$ adaptivity complexity and $O(n\log(k))$ query complexity.
- ParSKP2: The algorithm in (Cui et al., 2025) that runs in $O(\log^2 n)$ adaptivity, $O(nk\log^2 n)$ query complexity and returns a $1/4 - \epsilon$ approximation solution in expectation.
- **AdaptiveSimpleThreshold** (AST): The algorithm in (Chen & Kuhnle, 2024) achieves $1/6 - \epsilon$ approximation ratio with $O(\log(n/\delta)/\epsilon + \log(1/\epsilon)/\epsilon)$ adaptivity complexity and $O(\log_{1-\epsilon}(1/(6k)).(n/\epsilon + n\log^3(1/\epsilon)/\epsilon^4))$ query complexity.
- **AdaptiveThresholdGreedy** (ATG): The algorithm in (Chen & Kuhnle, 2024) achieves $0.193 - \epsilon$ approximation ratio with $O(\log(n)\log(k))$ adaptivity complexity and $O(n\log(k))$ query complexity.
- ENE: The algorithm of Ene and Nguyen (Ene & Nguyen, 2020) that returns a ratio of $1/e - \epsilon$ in $O(\log n)$ adaptive rounds and $\Omega(nk^2\log n)$.
- **FastRandomGreedy** (FRG): The algorithm in (Buchbinder et al., 2015) achieves $1/e - \epsilon$ approximation ratio with $O(k)$ adaptivity complexity and $O(n)$ query complexity.
- **FASTLS+GUIDEDRG** (FLS): The algorithm in (Chen et al., 2024) achieves $0.385 - \epsilon$ approximation ratio with $O(kn/\epsilon)$ query complexity.

The comparison is about **four metrics**: object values, adaptive complexity, number of queries, and running time. We experimented with two well-known applications: Revenue Maximization (RM) and Maximum Cut (MC) (Chen & Kuhnle, 2024; Kuhnle, 2019; Amanatidis et al., 2020).

We set $\epsilon = 0.1$ for all algorithms except FLS, for which we set $\epsilon = 0.01$ following the settings in (Chen et al., 2024). All other settings are the same as those in (Chen & Kuhnle, 2024). Furthermore, we use an algorithm in (Fahrbach et al., 2019a) (USM1), which returns a ratio of $1/4 - \epsilon$ for our LinAst, LinAtg and algorithms (AST, ATG, ANM) in (Chen & Kuhnle, 2024; Fahrbach et al., 2023) and set the same setting with (Chen & Kuhnle, 2024; Fahrbach et al., 2023). In our algorithms, by setting $\alpha = 4$, we make a good balance the trade-off between the approximation ratio in LinBoundSet and the required query complexity in practice. ENE algorithm requires access to an oracle for the multilinear extension and its gradient. In the case of the Maximum Cut problem, the multilinear extension and its gradient can be computed in closed form in time linear in the size of the graph. Thus, one can evaluate it using direct oracle access to the multilinear extension and its gradient on the Maximum Cut application. However, no closed form exists for the multilinear extension of the revenue maximization objective (Chen & Kuhnle, 2024).

H.2. Applications and settings

The applications utilized in the experiments are defined as follows:

Maximum Cut Application. Given graph $G = (V, E)$, and nonnegative edge weight w_{ij} for each edge $(i, j) \in E$. For $S \subseteq V$, let

$$f(S) = \sum_{i \in V \setminus S} \sum_{j \in S} w_{ij}. \tag{244}$$

The objective of the problem is to find a solution set S such that $f(S)$ is maximized while ensuring that the cardinality of S is less than k. The function $f(\cdot)$ is a submodular and non-monotone function.

Revenue Maximization Application. Considering a graph $G = (V, E)$ representing a social network, where each edge $(i, j) \in E$ is associated with a non-negative weight w_{ij}, we adopt the concave graph model introduced by (Hartline et al., 2008). In this model, each user $i \in V$ is linked to a non-negative, a concave function $f_i : \mathbb{R}^+ \to \mathbb{R}^+$. The function $v_i(S) = f_i\left(\sum_{j \in S} w_{ij}\right)$ indicates the likelihood of user i purchasing a product if the set S adopts it. Thus, the total revenue from seeding a set S is given by

$$f(S) = \sum_{i \in V \setminus S} f_i\left(\sum_{j \in S} w_{ij}\right). \tag{245}$$

The objective of the application is to find a solution S such that $f(S)$ is maximized while ensuring that the cardinality of S is less than k. The function $f(\cdot)$ is a submodular and non-monotone function.

Table 2. Details of datasets used in the experiments.

Dataset Name	Nodes	Edges	Types	Applications	Sources\References
Barabási-Albert	968	5,708	Undirected	MC	(Chen & Kuhnle, 2024)
ca-GrQc	5,242	14,496	Undirected	MC	SNAP
ca-Astro	18,772	198,110	Undirected	RM	SNAP
web-Google	875,713	5,105,039	Undirected	MC	SNAP

Other settings. In our experimental setup, we employed OpenMP to parallelize the code written in C++. To precisely measure the execution time of these algorithms, we utilized **Chrono**, a standard component of C++. Our methodology includes marking the start and end points of each algorithm's execution using Chrono and computing the time gap between these two points. In the Maximum Cut, we uniformly set the weight of all edges to 1.0. In the Revenue Maximization, we randomly assigned weights to all edges from the interval (0, 1). Besides, we experimented on a high-performance computing (HPC) server cluster with the following parameters: partition=large, #threads(CPU)=64, node=2, max memory = 3,073 GB.

H.3. Additional results

Figure 2 shows the results of compared algorithms on Barabási-Albert and Astro with both RM and MC, while Figure 3 focuses on the largest dataset, Google, with various k on the MC application. On Google, we do not show the result of ENE and ParSKP2 because they become impractical with large size data and large k.

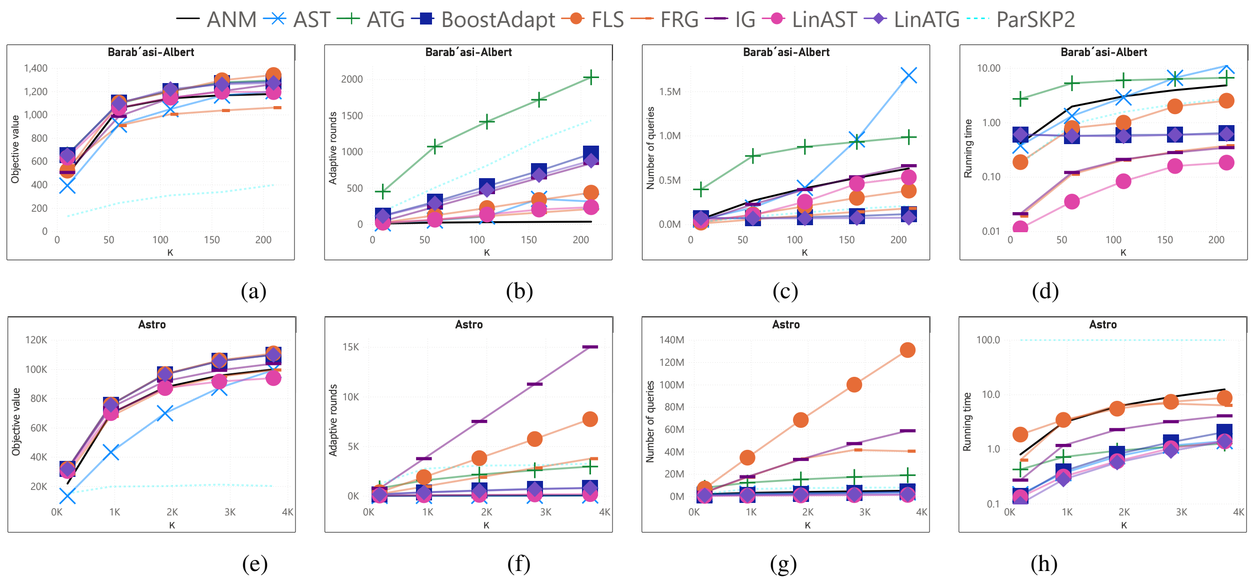

Figure 2. Performance of algorithms for Maximum Cut (a-d) and Revenue Maximization (e-h).

Objective value: Figures 2-3(a)(e) show the objective values. They demonstrate that our BoostAdapt lines consistently reach the highest points for every k, except for two points, $k(150, 200)$, in the Barabási-Albert dataset, which are slightly lower than FLS. LinAtg approximates to BoostAdapt while LinAst is a little lower than BoostAdapt. With Barabási-Albert and Astro, except AST and ParSKP2, other algorithms are also consistent with ours. AST are moderately lower than ours. Significantly, ParSKP2 always hits the lowest values. With Google, the gap between ours with FRG, AST, and ParSKP2 seems larger, especially for small k. Especially, Figure 3(a) indicates ours are much higher than FRG, AST and ParSKP2. In this case, our algorithms can be up to 1.3-1.6 times larger than FRG and AST. Finally, our algorithms significantly improve the solution quality.

Adaptive rounds: In MC (Figure 2(b)), ANM marks the lowest points, which means it saves the best adaptive rounds. Being consistent to ANM they are LinAst, FRG, FLS, and AST. Moderately higher than these lines they are LinAtg, IG and BoostAdapt. Significantly, ParSKP, ENE, and ATG are much higher than the others. In RM (Figures 2(f)), ATG, IG,FLS, and ParSKP2 always waste a sharply large number of adaptive rounds, the others can use an acceptable number of them. Especially, our LinAst always hits the lowest points while our other algorithms are almost close to it. Also, AST and FRG save more considerable rounds compared to the mentioned high adaptivity lines. As can be seen, our algorithms outperform the others in the adaptive rounds.

Number of queries: In both RM and MC of Figures 2(c)(g), our algorithms almost always minimize the number of queries. In RM Figure 2(c), LinAtg, ENE, FRG and BoostAdapt are the group of the lowest number of queries, followed by LinAst, ANM, ParSKP2, FLS, and IG algorithms. Meanwhile, AST and ATG require many queries. Nevertheless, in RM (Figure 2(g)), FLS reach highest points, followed by IG, FRG and ATG. The others are substantially low.

The change also happened with Google's experiment. With small k, ATG and IG waste much higher than the others. Our algorithms always mark the lowest points. With bigger k, LinAtg still hit the lowest points. LinAst, ATG and AST are close to LinAtg while BoostAdapt grows when k grows. With high values of k, BoostAdapt reaches to FRG. IG wastes the highest number of queries. Finally, LinAtg and LinAst always keep steadily low while BoostAdapt keep low values in most cases except Google with large k.

Time taken: In the MC (Figure 2(d)), LinAst has the lowest execution time, followed by the ENE, IG and FRG algorithms. Our algorithms BoostAdapt and LinAtg have the same running time, higher than IG and FRG but lower than the remaining. In RM (Figure 2(h)), all our algorithms have the lowest points, a little lower than ATG and IG. ANM, ENE, FLS and FRG are moderately higher than the above algorithms while ParSKP2 has essentially highest values.

On the Google dataset (Figure 3), LinAst always runs fastest while others fluctuate. With small k, our BoostAdapt and

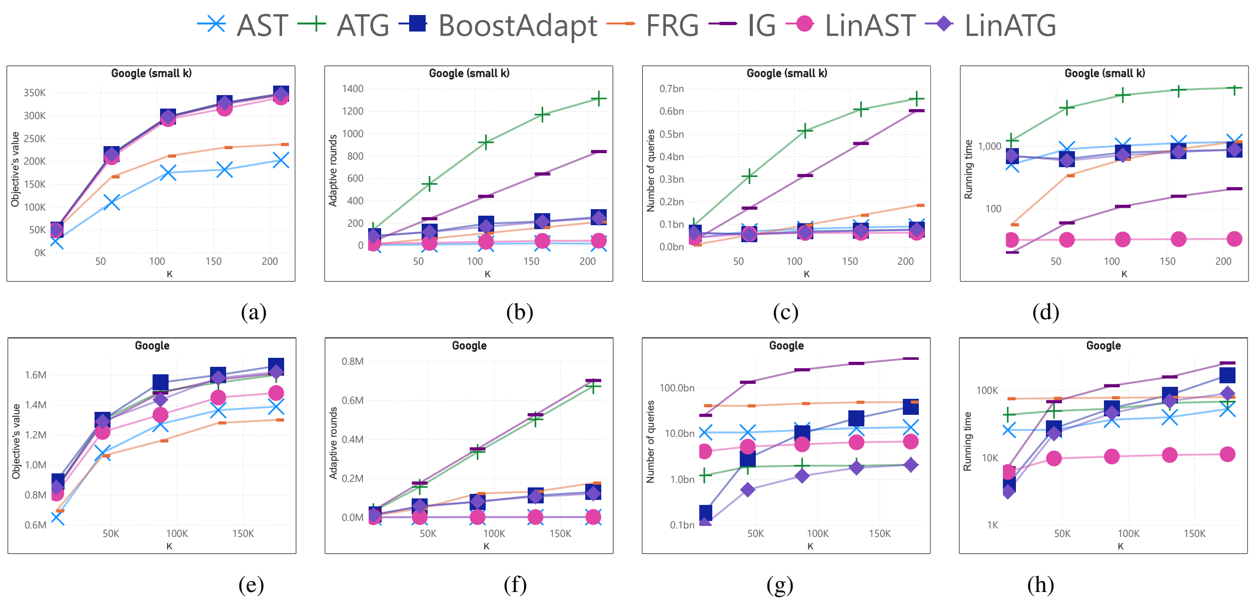

Figure 3. Performance of algorithms for Maximum Cut on large Google Dataset.

LinAtg are almost equal to AST, higher than FRG and IG, and lower than ATG. With bigger k, IG runs slower than BoostAdapt and LinAtg and LinAst and have the trend run faster than the remaining.

The above metrics show that our algorithms outperform the others when keeping the best quality solutions, wasting the lowest number of queries within acceptable query adaptivity.