Improved Parallel Algorithm for Non-Monotone Submodular Maximization under Knapsack Constraint

Tan D. Tran[1] , **Canh V. Pham**[1*] , **Dung T. K. Ha**[2] , **Phuong N.H. Pham**[3]
[1]ORLab, Faculty of Computer Science, Phenikaa University, Hanoi, Vietnam
[2] Faculty of Information Technology, VNU University of Engineering and Technology, Hanoi, Vietnam
[3] Faculty of Information Technology, Ho Chi Minh City University of Industry and Trade, Ho Chi Minh, Vietnam
canh.phamvan@phenikaa-uni.edu.vn, {22027005, 20028008}@vnu.edu.vn, phuongpnh@huit.edu.vn

Abstract

This work proposes an efficient parallel algorithm for non-monotone submodular maximization under a knapsack constraint problem over the ground set of size n. Our algorithm improves the best approximation factor of the existing parallel one from $8 + \epsilon$ to $7 + \epsilon$ with $O(\log n)$ adaptive complexity. The key idea of our approach is to create a new alternate threshold algorithmic framework. This strategy alternately constructs two disjoint candidate solutions within a constant number of sequence rounds. Then, the algorithm boosts solution quality without sacrificing the adaptive complexity. Extensive experimental studies on three applications, Revenue Maximization, Image Summarization, and Maximum Weighted Cut, show that our algorithm not only significantly increases solution quality but also requires comparative adaptivity to state-of-the-art algorithms.

1 Introduction

A wide range of instances in artificial intelligence and machine learning have been modeled as a problem of Submodular Maximization under Knapsack constraint (SMK) such as maximum weighted cut [Amanatidis *et al.*, 2020; Han *et al.*, 2021], data summarization [Han *et al.*, 2021; Mirzasoleiman *et al.*, 2016], revenue maximization in social networks [Han *et al.*, 2021; Cui *et al.*, 2023a; Cui *et al.*, 2021], recommendation systems [Amanatidis *et al.*, 2021; Amanatidis *et al.*, 2020]. The attraction of this problem comes from the diversity of submodular utility functions and the generalization of the knapsack constraint. The submodular function has a high ability to gather a vast amount of information from a small subset instead of extracting a whole large set, while the knapsack constraint can represent the budget, the cardinality, or the total time limit for a resource. Hence, people are interested in proposing expensive algorithms for SMK these years [Amanatidis *et al.*, 2021; Han *et al.*, 2021; Cui *et al.*, 2023a; Pham *et al.*, 2023; Amanatidis *et al.*, 2020].

Formally, a SMK problem can be defined such as given a ground set V of size n, a budget $B > 0$, and a non-negative submodular set function (not necessary monotone) $f : 2^V \mapsto \mathbb{R}_+$. Every element $e \in V$ has its positive cost $c(e)$. The problem SMK asks to find $S \subseteq V$ subject to $c(S) = \sum_{e \in S} c(e) \leq B$ that maximizes $f(S)$.

One of the main challenges of SMK is addressing big data in which the sizes of applications can grow exponentially. The modern approach is to design approximation algorithms with low query complexity representing the total number of queries to the oracle of f. However, required oracles of f are often expensive and may take a long time to process on the machine within a single thread. Therefore, people think of designing efficient parallel algorithms that can leverage parallel computer architectures to obtain a good solution promptly. This motivates the *adaptive complexity or adaptivity* [Balkanski and Singer, 2018] to become an important measurement of parallel algorithms. It is defined as the number of sequential rounds needed if the algorithm can execute polynomial independent queries in parallel. Therefore, the lower the adaptive complexity of an algorithm is, the higher its parallelism is.

In the era of big data now, several algorithms that achieve near-optimal solutions with low adaptive complexities have been developed recently (See Table 1 for an overview of low adaptive algorithms). As can be seen, although recent studies make an outstanding contribution by significantly reducing the adaptive complexity of a constant factor approximation algorithm from $O(\log^2 n)$ to $O(\log n)$, there are two drawbacks, including (1) the high query complexities make them become impractical in some instances [Ene and Nguyen, 2020] and (2) there is a huge gap between the high approximation factors of low adaptivity algorithms, e.g. [Amanatidis *et al.*, 2021; Cui *et al.*, 2023a; Cui *et al.*, 2023b], compared to the best one, e.g. [Buchbinder and Feldman, 2019]. This raises to us an interesting question: *Is it possible to improve the factor of an approximation algorithm with near-optimal adaptive complexity of $O(\log n)$?*

Our contributions. In this work, we address the above question by introducing the AST algorithm for the non-monotone SMK problem. AST has an approximation factor of $7 + \epsilon$, within a pair of $O(\log n)$ adaptivity, $\tilde{O}(nk)$ query complexity, where ϵ is a constant input. Therefore, our algorithm improves the best factor of the near-optimal adap-

*Corresponding author

Reference	Approximation Factor	Adaptive Complexity	Query Complexity
[Buchbinder and Feldman, 2019]	2.6	$poly(n)$	$poly(n)$
[Han *et al.*, 2021]	$4+\epsilon$	$O(n \log k)$	$O(n \log k)$
[Pham *et al.*, 2023]	$4+\epsilon$	$O(n)$	$O(n)$
[Ene *et al.*, 2019]	$e+\epsilon$	$O(\log^2 n)$	$\tilde{O}(n^2)$
[Amanatidis *et al.*, 2021]	$9.465+\epsilon$	$\boldsymbol{O(\log n)}$	$\tilde{O}(n^2)$
[Cui *et al.*, 2023a] (Alg.3)	$8+\epsilon$	$\boldsymbol{O(\log n)}$	$\boldsymbol{\tilde{O}(nk)}$
[Cui *et al.*, 2023a] (Alg.5)	$5+2\sqrt{2}+\epsilon \approx 7.83+\epsilon$	$O(\log^2 n)$	$\boldsymbol{\tilde{O}(nk)}$
AST **(Algorithm 1, this work)**	$\boldsymbol{7+\epsilon}$	$\boldsymbol{O(\log n)}$	$\boldsymbol{\tilde{O}(nk)}$

Table 1: Algorithms for SMK problem. We use the $\tilde{O}$ notation throughout the paper to hide $poly(\log n)$ factors and k is the largest cardinality of any feasible solution. Bold font indicates the best result(s) in each setting.

tive complexity algorithm in [Cui *et al.*, 2023a]. We investigate the performance of our algorithm on three benchmark applications: Revenue Maximization, Image Summarization, and Maximum Weighted Cut. The results show that our algorithm not only significantly improves the solution quality but also requires comparative adaptivity to existing practical algorithms.

New technical approach. It is noted that one popular approach to designing parallel algorithms with near-optimal adaptivity of $O(\log n)$ is based on making multiple guesses of the optimal solution in parallel and adapting a threshold sampling method[1], which selects a batch of elements whose density gains, i.e., the ratio between the marginal gain of an element per its cost, are at least a given threshold within $O(\log n)$ adaptivity [Amanatidis *et al.*, 2021; Cui *et al.*, 2023a]. By making the guesses of the optimal along with calling the threshold sampling multiple times in parallel, the existing algorithms could keep the adaptive complexity of $O(\log n)$ and obtain some approximation ratios.

From another view, we introduce a novel algorithmic framework named "alternate threshold" to improve the approximation factor to $7+\epsilon$ but keep the same adaptivity and query complexity with the best one [Cui *et al.*, 2023a]. Firstly, we adapt an existing adaptive algorithm to find a near-optimal solution within $O(\log n)$ adaptivity and give a $O(1)$ number of guesses of the optimal solution. Then, the core of our framework consists of a constant number of iterations. It initiates two disjoint candidate sets and then adapts the threshold sampling to upgrade them alternately during iterations: one is updated at *odd* iterations, and another is updated at *even* iterations. Thanks to this strategy, we can find the connection between two solutions for supporting each other in evaluating the "utility loss" after each iteration. At the end of this stage, we enhance the solution quality by finding the best element to be added to each candidate's subsets (pre-fixes of i elements) without violating the budget constraint.

It must be noted that our method differs from the Twin Greedy-based algorithms [Han *et al.*, 2020; Pham *et al.*, 2023; Sun *et al.*, 2022], which update both candidate sets at the same iterations but do not allow the integration of the threshold sampling algorithm for parallelization. Besides, we carefully analyze the role of the highest cost element in the optimal solution to deserve more tightness for the problem.

[1] We refer to threshold sampling methods as ThreshSeq in [Amanatidis *et al.*, 2021] and RandBatch in [Cui *et al.*, 2023a] with $O(\log n)$ adaptivity.

2 Related Works

This section focuses on the related works for the non-monotone case of the SMK problem.

Firstly, regarding the non-adaptive algorithms, the first algorithm for the non-monotone SMK problem was due to [Lee *et al.*, 2010] with the $5+\epsilon$ factor and polynomial query complexity. Later, several works concentrated on improving both approximation factor and query complexity [Buchbinder and Feldman, 2019; Gupta *et al.*, 2010; Mirzasoleiman *et al.*, 2016; Li, 2018; Sun *et al.*, 2022; Pham *et al.*, 2023; Han *et al.*, 2021]. In this line of works, algorithm of [Buchbinder and Feldman, 2019] archived the best approximation factor of 2.6 but required a high query complexity; the fastest algorithm was proposed by [Pham *et al.*, 2023] with $4+\epsilon$ factor in linear queries. For the non-monotone Submodular Maximization under Cardinality (SMC) problem, which finds the best solution that does not exceed k elements to maximize a submodular objective value, the best factor of 2.6 of the algorithm in [Buchbinder and Feldman, 2019] still held. Besides, a few algorithmic models have been proposed for improving running time [Badanidiyuru and Vondrák, 2014; Kuhnle, 2021b; Li *et al.*, 2022; Buchbinder *et al.*, 2015]. Among them, the fastest algorithm belonged to [Buchbinder *et al.*, 2015] that provided a $e+\epsilon$ factor within $O(n\log(1/\epsilon)/\epsilon^2)$ queries. However, the above approaches couldn't be parallelized efficiently by the high adaptive complexity of $\Omega(n)$.

The adaptive complexity was first proposed by [Balkanski and Singer, 2018] for the SMC problem. Regarding adaptivity-based algorithms for non-monotone SMK, the first one belonged to [Ene and Nguyen, 2019] with $e+\epsilon$ and $O(\log^2 n)$ adaptive complexity. However, due to the high query complexity of accessing, the multi-linear extension of a submodular function and its gradient in their method becomes impractical in real applications [Amanatidis *et al.*, 2021; Fahrbach *et al.*, 2019]. After that, [Amanatidis *et al.*, 2021] devised a $(9.465+\epsilon)$-approximation algorithm within $O(\log n)$, which was optimal up to a $\Theta(\log\log(n))$ factor by adopting the lower bound in [Balkanski and Singer, 2018]. It is noted that improving the adaptive complexity of a constant factor algorithm from $O(\log^2 n)$ to $O(\log n)$ made an outstanding contribution since it greatly reduced the number of sequential rounds in practical implementation [Cui *et al.*, 2023a; Ene and Nguyen, 2020; Fahrbach *et al.*, 2019]. More recently, [Cui *et al.*, 2023a] created a big step when contributing an efficient parallel one, which resulted in a factor of $8+\epsilon$ within a pair of $O(\log n)$ adaptivity and $\tilde{O}(nk)$ query complexity. Nevertheless, this factor still has a huge gap with the best factor of 2.6 [Buchbinder and Feldman, 2019]. They also provided an enhanced version of increasing the approximation factor to $5 + 2\sqrt{2} + \epsilon \approx 7.83 + \epsilon$. However, it required a higher adaptivity of $O(\log^2 n)$. Thus, from this view, the above result of [Cui *et al.*, 2023a] is the best one until now.

People have also focused on developing parallel algorithms for non-monotone SMC these years [Kuhnle, 2021a; Ene and Nguyen, 2020; Fahrbach *et al.*, 2019], etc. The forementioned contributions of [Ene and Nguyen, 2020] was also applied for SMC to get the best approximation factor of $e+\epsilon$, however it used multi-linear extension and thus had a high query complexity. Next, [Kuhnle, 2021a] and [Fahrbach *et al.*, 2019] tried to reduce the adaptive complexity to $O(\log n)$ with $25.64+\epsilon$ and $6+\epsilon$ factors. However, [Chen and Kuhnle, 2022] claimed that both [Kuhnle, 2021a] and [Fahrbach *et al.*, 2019] had a non-trivial error because they used the same threshold sampling subroutine which did not work for the non-monotone objective function. [Chen and Kuhnle, 2022] further tried to fixed the previous work and recovered the $6+\epsilon$ factor in $O(\log n)$. Recently, [Amanatidis *et al.*, 2021] improved the factor to $5.83 + \epsilon$ in $O(\log n)$ adaptivity. Later, the work of [Cui *et al.*, 2023a] archived the factor of $8 + \epsilon$ in $O(\log n)$ adaptivity or $4 + \epsilon$ factor in $O(\log^2 n)$.

After all, our algorithm overcome the existing drawbacks by an improved parallel version with the approximation factor increasing to $7 + \epsilon$ within $O(\log n)$ rounds to parallel $\tilde{O}(nk)$ queries.

3 Preliminaries

Given a ground set $V = \{e_1, \ldots, e_n\}$ and an utility set function $f : 2^V \mapsto \mathbb{R}_+$ to measure the quality of a subset $S \subseteq V$, we use the definition of submodularity based on *the diminishing return property*: $f : 2^V \mapsto \mathbb{R}_+$. f is submodular iff for any $A \subseteq B \subseteq V$ and $e \in V \setminus B$, we have

$$f(e|A) \geq f(e|B).$$

Each element $e \in V$ is assigned a positive cost $c(e) > 0$. Let $c : 2^V \mapsto \mathbb{R}^+$ be a cost function. Assume that c is modular, i.e., $c(S) = \sum_{e\in S} c(e)$ such that $c(S) = 0$ iff $S = \emptyset$.

The problem SMK asks to find $S \subseteq V$ subject to $c(S) = \sum_{e\in S} c(e) \leq B$ that maximizes $f(S)$. We denote by a tuple (f, V, B) an instance of SMK. Without loss of generality, f is assumed non-negative, i.e., $f(X) \geq 0$ for all $X \subseteq V$ and normalized, i.e., $f(\emptyset) = 0$. We also assume there exists an oracle query, which, when queried with the set S returns the value $f(S)$.

For convenience, we denote by $S \cup e$ as $S \cup \{e\}$. Next, we denote by O an optimal solution with the optimal value $\mathsf{OPT} = f(O)$ and $r = \arg\max_{o\in O} c(o)$. We also define the contribution gain of a set T to a set S as $f(T|S) = f(T \cup S) - f(S)$. Also, the contribution gain of an element e to a set $S \subseteq V$ is defined as $f(e|S) = f(S \cup \{e\}) - f(S)$ and $f(\{e\})$ is written as $f(e)$ for any $e \in V$.

In this paper, we design a parallel algorithm based on *Adaptive complexity* or *Adaptivity*, which is defined as follows:

Definition 1 (Adaptive complexity or Adaptivity [Balkanski and Singer, 2018])**.** *Given a value of oracle of f, the adaptivity or adaptive complexity of an algorithm is the minimum number of rounds needed such that in each round, the algorithm makes $O(poly(n))$ independent queries to the evaluation oracle.*

In the following, we recap two sub-problems which our algorithm need to solve: Unconstrained Submodular Maximization and Density Threshold.

Unconstrained Submodular Maximization (UnSubMax) This problem requires to find a subset $S \subseteq V$ that maximizes $f(S)$ without any constraint. The problem was shown NP-hard [Feige *et al.*, 2011a].
To obtain mentioned approximation factor, our algorithm adapts the low adaptivity algorithm in [Chen *et al.*, 2019] that achieves an approximation factor of $(2 + \epsilon)$ in constant adaptive rounds of $O(\log(1/\epsilon)/\epsilon)$ and linear queries of $O(n \log^3(1/\epsilon)/\epsilon^4)$.

Density Threshold (DS). The problem receives an instance (f, V, B), a fixed threshold τ and a parameter $\epsilon > 0$ as inputs, it asks to find a subset $S \subseteq V$ satisfies two conditions: (1) $f(S) \geq c(S) \cdot \tau$; (2) $\sum_{e\in V\setminus S} f(e|S) \leq \epsilon \cdot \mathsf{OPT}$.
Two algorithms in the literature satisfy the above conditions, including those in [Amanatidis *et al.*, 2021] and [Cui *et al.*, 2023a]. In this work, we adapt the RandBatch algorithm in [Cui *et al.*, 2023a]. RandBatch requires the set I, a submodular function $f(\cdot)$, and parameters ϵ, M to control the solution's accuracy and complexities. RandBatch is combined with the aforementioned density thresholds to set up sieves in parallel for SMK. Due to the space limitations, Pseudocode for RandBatch is given in the appendix.

For an instance (V, f, B) of SMK, two subsets I, M of V, a fixed threshold θ and input parameter ϵ. The performance of RandBatch is provided in the following Lemmas.

Lemma 1 (Lemma 1 in [Cui *et al.*, 2023a])**.** *The sets A, L output by* RandBatch$(\theta, I, M, \epsilon, f(\cdot), c(\cdot))$ *satisfy $\mathbb{E}[f(A)] \geq (1-\epsilon)^2\theta \cdot \mathbb{E}[c(A)]$ and $\epsilon \cdot M \cdot \sum_{u\in L} f(u|A) \leq$* OPT.

Lemma 2 (Lemma 2 in [Cui *et al.*, 2023a])**.** RandBatch *has $O(\frac{1}{\epsilon p}(\log(|I| \cdot \beta(I)) + M))$ adaptivity, and its query complexity is $O(|I|k)$ times of its adaptive complexity, where $\beta(I) = \max_{u,v} \frac{c(u)}{c(v)}$. If we use binary search in Line 10 of* RandBatch, *then it has $O(\frac{1}{\epsilon p}(\log(|I| \cdot \beta(I)) + M)\log(k))$ adaptivity, and its query complexity is $O(|I|)$ times of its adaptivity.*

Interestingly, we further explore a useful property of RandBatch when applying it to our algorithm.

Lemma 3. *The sets A, L output by* RandBatch$(\theta, I, M, \epsilon, f(\cdot), c(\cdot))$ *satisfy $\mathbb{E}[f(a_i|A_{i-1})] \geq (1-\epsilon)^2\mathbb{E}[c(a_i)]\theta$, where $A = \{a_1, a_2, \ldots, a_{|A|}\}, A_i = \{a_1, a_2, \ldots, a_i\}$.*

4 Proposed Algorithm

In this section, we introduce AST (Algorithm 1), a $(7+\epsilon)$-approximation algorithm in $O(\log n)$ adaptivity and $O(n^2 \log^2 n)$ query complexity.

AST receives an instance (f, V, B), constant parameters δ, ϵ, α as inputs. It contains two main phases. At the first phase (Lines 1-14), it first divides the ground set V into two subsets: V_0 contains elements with small costs, and V_1 contains the rest. AST then calls ParSKP1 [Cui *et al.*, 2023a] as a subroutine which returns a $(1/8 - \delta)$-approximation solution within $O(\log n)$ adaptive rounds. Based on that, the algorithm can offer $O(\log(1/\epsilon)/\epsilon)$ guesses of the optimal solution for the main loop (Lines 4-14). The main loop consists of $O(\log(1/\epsilon)/\epsilon)$ iterations; each corresponds to a guess of the optimal. It sequentially constructs two disjoint solutions, X and Y, one at odd iterations and the other at even iterations. The work of the odd and the even is the same. At the odd (or even) ones, it sets the threshold θ_X (θ_Y) and calls the RandBatch routine with the ground set I and the function $f(\cdot|X)$ ($f(\cdot|Y)$) as inputs to provide the new set A_i (B_i) (Lines 8, 12). It then updates X (Y) and I as the remaining elements (Line 8 or 12).

The second phase (Lines 15-24) is to improve the quality of solutions. If $c(X_1 \cup V_0) \le \epsilon B$, this phase first adapts UnSubMax algorithm [Chen *et al.*, 2019] for unconstrained submodular maximization over $X_1 \cup V_0$ to get a candidate solution S_1 (Lines 15-16). This step is based on an observation that X_1 is important in analyzing the algorithm's performance. It then selects the sets of the first i elements added into X and Y and finds the best elements without violating the total cost constraint (Lines 19-24). Finally, the algorithm returns the best candidate solution (Lines 25-26). The details of AST are depicted in Algorithm 1.

At the high level, AST works follow a novel framework that combines an alternate threshold greedy algorithm with the boosting phase. The term "alternate" means that candidate solutions are updated alternately with each other in multiple iterations. At each iteration, only one partial solution is updated based on two factors: one guess of the optimal solution and the remaining elements of the ground set that do not belong to the other solution.

It should be emphasized that the alternate threshold greedy differs from recent works [Cui *et al.*, 2023a; Amanatidis *et al.*, 2021] where two candidate solutions for each guess are constructed after only one adaptive round. Alternate threshold greedy also differs from the twin greedy method in [Han *et al.*, 2020], which allows updating both disjoint sets in each iteration. For the theoretical analysis, the key to obtaining a tighter approximation factor lines in aspects: (1) the connections between X and Y after each iteration of the first loop and (2) carefully considering the role of r to eliminate terms that worsen the approximation factor.

We now analyze the performance guarantees of AST. We consider X and Y after ending the first loop. We first introduce some notations regarding AST as follows.

- X^i, Y^i is the set of first i elements added into X and Y, respectively.

Algorithm 1: AST Algorithm

Input: An instance (f, V, B), parameters α, ϵ, δ

1: $V_0 \leftarrow \{e \in V : c(e) \le \epsilon B/n\}$, $V_1 \leftarrow V \setminus V_0$, $I \leftarrow V_1$, $p \leftarrow 1$

2: $S_0 \leftarrow$ ParSKP1$(\frac{1}{4}, \delta, f(\cdot), c(\cdot))$, $\Gamma \leftarrow \frac{8\alpha f(S_0)}{(1-8\delta)\epsilon B}$

3: $X \leftarrow \emptyset, Y \leftarrow \emptyset, \Delta \leftarrow \lceil \log_{\frac{1}{1-\epsilon}} \frac{8\alpha}{\epsilon^2(1-8\delta)} \rceil + 1$, $M \leftarrow \frac{1}{\epsilon^2}(\frac{\Delta}{2}+1)$

4: **for** $i = 1$ *to* Δ **do**

5:     **if** i ***is odd*** **then**

6:         $\theta^X \leftarrow \Gamma(1-\epsilon)^i$

7:         $(A_i, U_i, L_i) \leftarrow$ RandBatch$(\theta^X, I, M, p, \epsilon, f(\cdot|X), c(\cdot))$

8:         $X \leftarrow X \cup A_i$, $I \leftarrow I \setminus X$

9:     **else**

10:         $\theta^Y \leftarrow \Gamma(1-\epsilon)^i$

11:         $(B_i, U_i, L_i) \leftarrow$ RandBatch$(\theta^Y, I, M, p, \epsilon, f(\cdot|Y), c(\cdot))$

12:         $Y \leftarrow Y \cup B_i$, $I \leftarrow I \setminus Y$

13:     **end**

14: **end**

15: For $T \in \{X, Y\}$, define: T_i is T after the iteration i of the first loop, T^i is the set of first i elements added into T.

16: **if** $c(X_1 \cup V_0) \le \epsilon B$ **then**

17:     $S_1 \leftarrow$ UnSubMax$(X_1 \cup V_0)$

18: **end**

19: **for** $i = 1$ *to* $|X|$ **do**

20:     $a_i \leftarrow \arg\max_{e \in V: c(X^i \cup e) \le B} f(X^i \cup \{e\})$, $X'^i \leftarrow X^i \cup \{a_i\}$

21: **end**

22: **for** $i = 1$ *to* $|Y|$ **do**

23:     $b_i \leftarrow \arg\max_{e \in V: c(Y^i \cup e) \le B} f(Y^i \cup \{e\})$, $Y'^i \leftarrow Y^i \cup \{a_i\}$

24: **end**

25: $S \leftarrow \arg\max_{T \in \{X'^i\}_{i=1}^{|X|} \cup \{Y'^i\}_{i=1}^{|Y|} \cup \{X, Y, S_1\}} f(T)$

26: **return** S

- X_i and Y_i are X and Y after the iteration i of the first loop (Lines 4-14) and $X_0 = Y_0 = \emptyset$.
- O_1 is an optimal solution of SMK over instance (f, V_1, B).
- $O' = O_1 \setminus X_1, O^r = O_1 \setminus \{r\}$ and $O'^r = O' \setminus \{r\}$.
- For an element $e \in X \cup Y$, we denote: $X_{<e}$, $Y_{<e}$, θ^X_e and θ^Y_e as X, Y, θ^X and θ^Y right before e is selected into X or Y, respectively; $l(e)$ is the iteration when e is added in to X or Y.

Lemma 4 makes a connection between X and Y after each iteration.

Lemma 4. *After any iteration i of the first loop (Lines 4-14) of* AST, *we have:*

a) If $i \ge 1$, i is odd and $c(X_i) \le B - c(r)$. Let $T \subseteq Y_{i-1} \cap O_1$, we have $\sum_{e \in T} f(e|X_i) < \sum_{e \in T} \frac{\mathbb{E}[f(e|Y_{<e})]}{(1-\epsilon)^3} + \epsilon \cdot$

OPT.

b) If $i \geq 2$, i is even and $c(Y_i) \leq B - c(r)$. Let $T \subseteq X_{i-1} \cap O'$, we have $\sum_{e \in T} f(e|Y_i) < \sum_{e \in T} \frac{\mathbb{E}[f(e|X_{<e})]}{(1-\epsilon)^3} + \epsilon \cdot$ OPT.

Proof. **Prove a)** If $i = 1$, $Y_0 = \emptyset$, the Lemma holds. We consider the other case. We divide T into several subsets including $T = T_2 \cup T_4 \cup \ldots \cup T_{i-1}$, where T_j is a set of all elements in T that are added into Y_i at iteration $j \leq i-1$. Since $T_j \subseteq Y_{i-1} \cap O_1$ and $c(X_i) \leq B - c(r)$ so $c(X_{<e}) + c(e) \leq c(X_i) + c(e) \leq B$ for all $e \in T_j$. We therefore classify the elements in T_j into two disjoint sets $T_j = T_j^1 \cup T_j^2$, where $T_j^1 = \{e \in T_j : \frac{f(e|X_{<e})}{c(e)} < \theta_e^X, c(X_{<e}) + c(e) \leq B\}$ and $T_j^2 = \{e \in T_j : \frac{f(e|X_{<e})}{c(e)} \geq \theta_e^X, c(X_{<e}) + c(e) \leq B\}$. Since $(1-\epsilon)\theta_e^X = \theta_{l(e)}^Y \forall e \in T_j^1$, we have:

$$\sum_{e \in T_j} f(e|X_{<e}) = \sum_{e \in T_j^1} f(e|X_{<e}) + \sum_{e \in T_j^2} f(e|X_{<e}) \tag{1}$$

$$< \sum_{e \in T_j^1} c(e)\theta_e^X + \frac{\mathsf{OPT}}{\epsilon M} \tag{2}$$

$$= \sum_{e \in T_j^1} c(e)\frac{\theta_{l(e)}^Y}{1-\epsilon} + \frac{\mathsf{OPT}}{\epsilon M} \tag{3}$$

$$\leq \sum_{e \in T_j^1} \frac{\mathbb{E}[f(e|Y_{<e})]}{(1-\epsilon)^3} + \frac{\mathsf{OPT}}{\epsilon M} \tag{4}$$

where inequality (2) is due to Lemma 1, inequality (4) is due to applying Lemma 3: $\mathbb{E}[f(e|Y_{<e})] \geq (1-\epsilon)^2 \mathbb{E}[c(e)]\theta_{l(e)}^Y$. It follows that

$$\sum_{e \in T} f(e|X_i) = \sum_{j=2,4,\ldots,i-1} \left(\sum_{e \in T_j} f(e|X_i) \right) \tag{5}$$

$$\leq \sum_{j=2,4,\ldots,i-1} \left(\sum_{e \in T_j} f(e|X_{<e}) \right) \tag{6}$$

$$< \sum_{j=2,4,\ldots,i-1} \left(\sum_{e \in T_j^1} \frac{\mathbb{E}[f(e|Y_{<e})]}{(1-\epsilon)^3} + \frac{\mathsf{OPT}}{\epsilon M} \right)$$

$$\leq \sum_{e \in T} \frac{\mathbb{E}[f(e|Y_{<e})]}{(1-\epsilon)^3} + (\frac{\Delta}{2}+1) \cdot \frac{\mathsf{OPT}}{\epsilon M} \tag{7}$$

$$\leq \sum_{e \in T} \frac{\mathbb{E}[f(e|Y_{<e})]}{(1-\epsilon)^3} + \epsilon \mathsf{OPT} \tag{8}$$

where inequation (6) is due to the submodularity, inequation (8) is due to setting of M.

Prove b). If $i = 2$, $X_1 \cap O = \emptyset$, the Lemma holds. If $i > 2$, we only consider set $T \subseteq X_{i-1} \cap O'$ since we can not bound the $f(e|Y_{<e})$ if $e \in X_1$. We also divide T into subsets: $T = T_3 \cup T_5 \cup \ldots \cup T_{i-1}$, where T_j is a set added into X_j at the iteration $j \leq i-1$. We also classify the elements in T_j into two sets $T_j = T_j^1 \cup T_j^2$, where $T_j^1 = \{e \in T_j : \frac{f(e|Y_{<e})}{c(e)} < \theta_{<e}^Y, c(Y_{<e}) + c(e) \leq B\}$ and $T_j^2 = \{e \in T_j : \frac{f(e|Y_{<e})}{c(e)} \geq \theta_{<e}^Y, c(Y_{<e}) + c(e) \leq B\}$. By a similar argument to the previous case, we have:

$$\sum_{e \in T_j} f(e|Y_{<e}) = \sum_{e \in T_j^1} f(e|Y_{<e}) + \sum_{e \in T_j^2} f(e|Y_{<e}) \tag{9}$$

$$< \sum_{e \in T_j^1} c(e)\theta_e^Y + \frac{\mathsf{OPT}}{\epsilon M} \tag{10}$$

$$= \sum_{e \in T_j^1} c(e)\frac{\theta_e^X}{1-\epsilon} + \frac{\mathsf{OPT}}{\epsilon M} \tag{11}$$

$$\leq \sum_{e \in T_j^1} \frac{\mathbb{E}[f(e|X_{<e})]}{(1-\epsilon)^3} + \frac{\mathsf{OPT}}{\epsilon M} \tag{12}$$

which implies that

$$\sum_{e \in T} f(e|Y_i) = \sum_{j=3,5,\ldots,i-1} \left(\sum_{e \in T_j} f(e|Y_i) \right) \tag{13}$$

$$\leq \sum_{j=3,5,\ldots,i-1} \left(\sum_{e \in T_j} f(e|Y_{<e}) \right) \tag{14}$$

$$< \sum_{j=3,5,\ldots,i-1} \left(\sum_{e \in T_j^1} \frac{\mathbb{E}[f(e|X_{<e})]}{(1-\epsilon)^3} + \frac{\mathsf{OPT}}{\epsilon M} \right)$$

$$\leq \sum_{e \in T} \frac{\mathbb{E}[f(e|X_{<e})]}{(1-\epsilon)^3} + (\frac{\Delta}{2}+1) \cdot \frac{\mathsf{OPT}}{\epsilon M} \tag{15}$$

$$\leq \sum_{e \in T} \frac{\mathbb{E}[f(e|X_{<e})]}{(1-\epsilon)^3} + \epsilon \mathsf{OPT}. \tag{16}$$

The proof is completed. □

By using Lemma 4, we further provide the bound of $f(O' \cup T)$ for T is a subset of X or Y in Lemma 5 when $c(r)$ is very large, i.e., $c(r) \geq (1-\epsilon)B$.

Lemma 5. *If $c(r) \geq (1-\epsilon)B$, one of two following propositions happens*

a) $\mathbb{E}[f(S)] \geq (1-\epsilon)^5 \alpha \mathsf{OPT}$.

b) There exists a subset $X' \subseteq X$ so that

$$f(X' \cup O') < (1+\frac{1}{(1-\epsilon)^3})\mathbb{E}[f(S)] + (2\epsilon + \frac{1}{\epsilon M})\mathsf{OPT}.$$

Similarly, one of two following propositions happens:

c) $\mathbb{E}[f(S)] \geq (1-\epsilon)^5 \alpha \mathsf{OPT}$.

d) There exists a subset $Y' \subseteq Y$ so that

$$f(Y' \cup O') < (1+\frac{1}{(1-\epsilon)^3})\mathbb{E}[f(S)] + (2\epsilon + \frac{1}{\epsilon M})\mathsf{OPT}.$$

When $c(X_1) < \epsilon B$ and $c(Y_2) < \epsilon B$, it's easy to obtain the approximation factor due to $f(S) \geq \max\{f(X_1), f(Y_2)\} \geq \epsilon B \alpha \Gamma (1-\epsilon)^2$. Otherwise, we combine Lemma 5 and the fact that $f(O') \leq f(O' \cup X) + f(O' \cup Y)$ to get the bound of $f(O')$ in Lemma 6. The proofs of them can be found in the Appendix.

Lemma 6. *If $c(X_1) < \epsilon B$ and $c(Y_2) < \epsilon B$, we have:*

- *If $c(r) < (1-\epsilon)B$, then $f(O') \leq \frac{5\mathbb{E}[f(S)]}{(1-\epsilon)^4} + \epsilon\mathsf{OPT}$*
- *If $c(r) \geq (1-\epsilon)B$, one of two things happens:*
  a) $\mathbb{E}[f(S)] \geq (1-\epsilon)^5\alpha\mathsf{OPT}$.
  b) $f(O') \leq \frac{5\mathbb{E}[f(S)]}{(1-\epsilon)^3} + 2(2\epsilon + \frac{1}{\epsilon M})\mathsf{OPT}$.

Finally, put Lemmata 4,5,6 together and divide O into appropriate subsets, we state the performance's algorithm in Theorem 4.1.

Theorem 4.1. *For $\alpha = \frac{1}{7}$, $\epsilon \in (0, \frac{1}{7})$, $\delta \in (0, \frac{1}{8})$, Algorithm 1 needs $O(\log n)$ adaptive complexity and $O(nk\log^2 n)$ query complexity and returns a solution S satisfying $\mathbb{E}[f(S)] \geq (1/7 - \epsilon)\mathsf{OPT}$.*

Proof. AST first calls ParSKP1 to find a candidate solution S_0. This task takes $O(\frac{1}{\delta}\log(\frac{n}{\delta}))$ adaptivity and $O(nk\log^2 n)$ query complexity [Cui *et al.*, 2023b]. For the first loop, it calls ThreshSeq $\Delta = O(\frac{1}{\epsilon}\log(\frac{1}{\epsilon}))$ times. Each time, RandBatch needs $O(\frac{1}{\epsilon}\log(\frac{n}{\epsilon}) + M) = O(\frac{1}{\epsilon}\log(\frac{n}{\epsilon}) + \frac{1}{\epsilon^3}\log(\frac{1}{\epsilon})) = O(\log n)$ adaptive complexity and $O(\frac{nk}{\epsilon}\log(\frac{n}{\epsilon}) + \frac{nk}{\epsilon^3}\log(\frac{1}{\epsilon})) = O(nk\log n)$ query complexity. In the second phase, the algorithm may need $O(\frac{1}{\epsilon}\log(\frac{1}{\epsilon}))$ adaptivity and $O(\frac{n}{\epsilon^4}\log^3(\frac{1}{\epsilon}))$ query complexity to call UnSubMax algorithm of [Chen *et al.*, 2019] (Lines 16-17). Then, it only has two adaptive rounds and takes $O(kn)$ query complexity to find X'^i and Y'^i (Lines 19-24). Therefore, the adaptive complexity of the algorithm is $O(\frac{1}{\delta}\log\frac{n}{\delta}) + O(\frac{1}{\epsilon}\log(\frac{1}{\epsilon})\log n) + O(\frac{1}{\epsilon}\log(\frac{1}{\epsilon})) + 2 = O(\log n)$ and its query complexity is $O(nk\log^2 n) + O(nk\log n) + O(\frac{n}{\epsilon^4}\log^3(\frac{1}{\epsilon})) + O(nk) = O(nk\log^2 n)$.
For the approximation factor, we consider the following cases:
Case 1. If $c(X_1) \geq \epsilon B$ or $c(Y_2) \geq \epsilon B$, we have

$$\begin{aligned} f(S) &\geq \max\{f(X_1), f(Y_2)\} \geq \epsilon B\alpha\Gamma(1-\epsilon)^2 \\ &> \frac{(1-\epsilon)^2}{7}\mathsf{OPT} > (\frac{1}{7}-\epsilon)\mathsf{OPT}. \end{aligned}$$

Case 2. If $c(X_1) < \epsilon B$ and $c(Y_2) < \epsilon B$, we have

$$\begin{aligned} f(O) &\leq f(O\cap(V_1\setminus X_1)) + f(O\cap(V_0\cup X_1)) && (17)\\ &= f(O') + f(O\cap(V_2\cup X_1)) && (18)\\ &\leq f(O') + (2+\epsilon)\mathbb{E}[f(S)] && (19) \end{aligned}$$

where inequality (17) is due to the submodularity of f and $(V_1\setminus X_1)\cap(V_0\cap X_1) = \emptyset$, equality (18) is due to the definition of O' and inequality (19) is due to applying Algorithm in [Chen *et al.*, 2019]. We now apply Lemma 6 to bound $f(O')$. If $c(r) < (1-\epsilon)B$, then

$$\begin{aligned} f(O) &\leq \frac{5\mathbb{E}[f(S)]}{(1-\epsilon)^4} + \epsilon\mathsf{OPT} + (2+\epsilon)\mathbb{E}[f(S)] && (20)\\ &\leq \frac{7\mathbb{E}[f(S)]}{(1-\epsilon)^4} + \epsilon\mathsf{OPT}. && (21) \end{aligned}$$

Therefore

$$\mathbb{E}[f(S)] \geq \frac{(1-\epsilon)^5}{7}\mathsf{OPT} > \frac{1-5\epsilon}{7}\mathsf{OPT} > (\frac{1}{7}-\epsilon)\mathsf{OPT}.$$

If $c(r) \geq (1-\epsilon)B$, we consider two cases:
- If **a)** in Lemma 6 happens, then

$$\mathbb{E}[f(S)] \geq \frac{(1-\epsilon)^5}{7}\mathsf{OPT} > (\frac{1}{7}-\epsilon)\mathsf{OPT}.$$

- If **b)** in Lemma 6 happens, then

$$\begin{aligned} f(O) &\leq \frac{5\mathbb{E}[f(S)]}{(1-\epsilon)^3} + (4\epsilon + \frac{2}{\epsilon M})\mathsf{OPT} + (2+\epsilon)\mathbb{E}[f(S)] \\ &< \frac{7\mathbb{E}[f(S)]}{(1-\epsilon)^3} + \frac{29}{7}\epsilon\mathsf{OPT}. && (22) \end{aligned}$$

where the inequality (22) is due to $\epsilon M = \frac{1}{\epsilon}(\frac{\Delta}{2}+1) > \frac{14}{\epsilon}$ for $\epsilon \in (0,\frac{1}{7}), \delta \in (0, \frac{1}{8})$. It follows that

$$\mathbb{E}[f(S)] \geq \frac{1}{7}(1-\epsilon)^3(1-\frac{29}{7}\epsilon)\mathsf{OPT} > (\frac{1}{7}-\epsilon)\mathsf{OPT}$$

which completes the proof. □

5 Experimental Evaluation

This section evaluates our AST's performance by comparing our algorithm with state-of-the-art algorithms for non-monotone SMK including:

- ParSKP1: The parallel algorithm in [Cui *et al.*, 2023a] that runs in $O(\log n)$ adaptivity and returns a solution S satisfying $\mathbb{E}[f(S)] \geq (1/8-\epsilon)\mathsf{OPT}$.
- ParSKP2: The algorithm in [Cui *et al.*, 2023a] that runs in $O(\log^2 n)$ adaptivity and returns a solution of $\mathbb{E}[f(S)] \geq \left(1/(5+2\sqrt{2}) - \epsilon\right)\mathsf{OPT}$.
- ParKnapsack: The parallel algorithm in [Amanatidis *et al.*, 2021] achieves an approximation factor of $(9.465 + \varepsilon)$ within $O(\log n)$.
- SmkRanAcc: The non-adaptive algorithm in [Han *et al.*, 2021] that achieves an approximation factor of $4+\epsilon$ in query complexity of $O(n\log(k/\epsilon)/\epsilon)$.
- RLA: The non-adaptive algorithm in [Pham *et al.*, 2023] with a factor of $4+\epsilon$ in linear query complexity of $O(n\log(1/\epsilon)/\epsilon)$.

We experimented with the following three applications:

Revenue Maximization (RM). Given a network $G = (V, E)$ where V is a set of nodes and E is a set of edges. Each edge (u, v) in E is assigned a positive weight $w(u, v)$ sampled uniformly in $[0, 1]$ and each node is assigned a positive cost $c(u)$ defined as $c(u) = 1 - e^{\sqrt{\sum_{(u,v)\in E} w(u,v)}}$. The revenue of any subset $S \subseteq \mathcal{V}$ is defined as $f(S) = \sum_{v\in\mathcal{V}\setminus S}\sqrt{\sum_{u\in S} w_{u,v}}$. Given a budget of B, the goal of the problem is to select a set S with the cost at most B to maximize $f(S)$. As in the prior work, [Amanatidis *et al.*, 2021], we can construct the graph G using a YouTube community network dataset [Han *et al.*, 2021] with 39,841 nodes and 224,235 edges.

Maximum Weighted Cut (MWC). Consider a graph $G = (V, E)$ where each edge $(u, v) \in E$ has a non-negative weight $w(u, v)$. For a node subset $S \subset V$, define the weighted cut function $f(S) = \sum_{u\in V\setminus S} \sum_{v\in S} w(u, v)$. The maximum weighted cut problem seeks a subset $S \subseteq V$ that maximizes $f(S)$. As in recent work [Amanatidis *et al.*, 2020], we generate an Erdős-Rényi random graph with $5,000$ nodes and an edge probability of 0.2. The node costs $c(u)$ are randomly uniformly sampled from $(0, 1)$.

Image Summarization (IS). Consider a graph $G = (V, E)$ where each node $u \in V$ represents an image, and each edge $(u, v) \in E$ has a weight $w(u, v)$ showing the similarity between images u and v. Let $c(u)$ be the cost to acquire image u. The goal is to find a representative subset $S \subseteq V$ under the budget B that maximizes a value $f(S) = \sum_{u\in V} \max_{v\in S} w_{u,v} - \frac{1}{|V|} \sum_{u\in V} \sum_{v\in S} w_{u,v}$ [Mirzasoleiman *et al.*, 2016; Han *et al.*, 2021]. As in recent works [Han *et al.*, 2021; Mirzasoleiman *et al.*, 2016], we create an instance as follows: First, randomly sample 500 images from the CIFAR dataset [Krizhevsky, 2019] of 10,000 images, then measure the similarity between images u and v using the cosine similarity of their 3,072-dimensional pixel vectors.

Experiment setting. In our experiments, we set the accuracy parameter $\epsilon = 0.1$ for all algorithms evaluated, and for AST, we set $\delta = 0.12$. We used OpenMP to program with C++ language. Besides, we experimented on a high-performance computing (HPC) server cluster with the following parameters: partition=large, #threads(CPU)=128, node=4, max memory = 3,073 GB. For UnSubMax, we use setting of previous works [Amanatidis *et al.*, 2021; Cui *et al.*, 2023a], i.e, we adapt Algorithm in [Feige *et al.*, 2011b] returning $1/4-\epsilon$ ratio in one adaptive round and $O(n)$ query complexity.

Experimental Result. In Figures 1(a), (c), and (e), we compare the objective values between different algorithms. The results show that our AST achieves the best objective values for both the RM and MWC applications. In RM, the objective values achieved by RLA, SmkRanAcc, and ParSKP1 are the same, ParKnapsack attains lower objective values while ParSKP2 hits the lowest objective values among the algorithms. Especially, our one marks the highest value when $B = 0.015$, about 1.3 times higher than the others in RM. In IS, ParKnapsack reaches the highest values while ParSKP2 hits the lowest. Our algorithm results in best values at some points and drops at others. As shown in Figure 1 (c), most algorithms fluctuate widely. The variation in the quality of these algorithms might be due to the characteristics of this dataset.

In Figures 1(b), (d), and (f), we make the comparisons about the number of adaptive rounds. The results show that SmkRanAcc and RLA always require the highest number of adaptive rounds across all three applications. For the AST, the number of adaptive rounds is equivalent to ParSKP1 for RM and MWC. Besides, for IS, the adaptive number of rounds for AST is higher than that of ParSKP1, which has the lowest number of rounds. However, the higher number in this case is insignificant. Overall, our algorithm outperforms the others in both solution performance and the quantities of adaptivity.

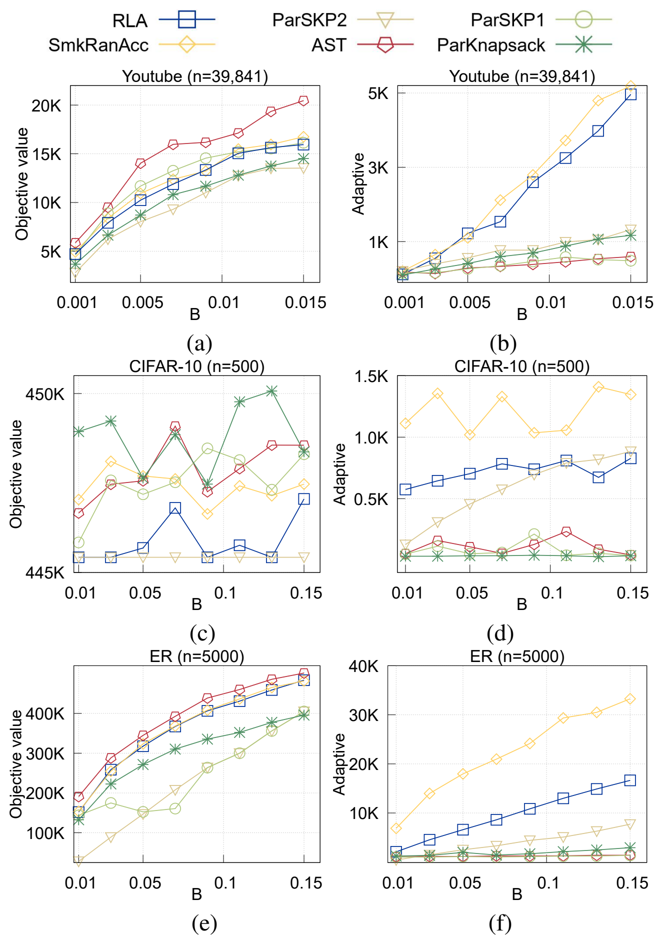

Figure 1: Performance of algorithms for non-monotone SMK on three instances: (a), (b) Revenue Maximization; (c), (d) Image Summarization and (e), (f) Maximum Weighted Cut. The budget values represent fractions of the total cost of all elements.

6 Conclusions

Motivated by the challenge of the large scale of input data, in this work, we focus on parallel approximation algorithms based on the concept of adaptive complexity. Moreover, the requirement of improving the approximation factor while decreasing the adaptivity down to $\log(n)$ motivates us to propose a competitive new algorithm. We have proposed an efficient parallel algorithm AST based on a novel alternate threshold greedy strategy. To our knowledge, our AST algorithm is the first to achieve a constant factor approximation of $7 + \epsilon$ for the above problem in the aforementioned adaptivity. Our algorithm also expresses the superiority in solution quality and computation complexity compared to state-of-the-art algorithms via some illustrations in the experiment in three real-world applications. In the future, we will address another valuable question: can we reduce the query complexity of parallelized algorithms for the SMK problem?

Acknowledgments

The first author (Tan D. Tran) was funded by the Master, PhD Scholarship Programme of Vingroup Innovation Foundation (VINIF), code VINIF.2023.TS.105. This work has been carried out partly at the Vietnam Institute for Advanced Study in Mathematics (VIASM). The second author (Canh V. Pham) would like to thank VIASM for its hospitality and financial support.

References

[Amanatidis *et al.*, 2020] Georgios Amanatidis, Federico Fusco, Philip Lazos, Stefano Leonardi, and Rebecca Reiffenhäuser. Fast adaptive non-monotone submodular maximization subject to a knapsack constraint. In *Annual Conference on Neural Information Processing Systems*, 2020.

[Amanatidis *et al.*, 2021] Georgios Amanatidis, Federico Fusco, Philip Lazos, Stefano Leonardi, Alberto Marchetti-Spaccamela, and Rebecca Reiffenhäuser. 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*, pages 231–242, 2021.

[Badanidiyuru and Vondrák, 2014] Ashwinkumar Badanidiyuru and Jan Vondrák. Fast algorithms for maximizing submodular functions. In *Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1497–1514, 2014.

[Balkanski and Singer, 2018] Eric Balkanski and Yaron Singer. The adaptive complexity of maximizing a submodular function. In *Annual ACM SIGACT Symposium on Theory of Computing*, pages 1138–1151, 2018.

[Buchbinder and Feldman, 2019] Niv Buchbinder and Moran Feldman. Constrained submodular maximization via a nonsymmetric technique. *Mathematical Operations Research*, 44(3):988–1005, 2019.

[Buchbinder *et al.*, 2015] Niv Buchbinder, Moran Feldman, and Roy Schwartz. Comparing apples and oranges: Query tradeoff in submodular maximization. In *Proceedings of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms 2015*, pages 1149–1168, 2015.

[Chen and Kuhnle, 2022] Yixin Chen and Alan Kuhnle. Practical and parallelizable algorithms for non-monotone submodular maximization with size constraint. *https://arxiv.org/abs/2009.01947*, 2022.

[Chen *et al.*, 2019] Lin Chen, Moran Feldman, and Amin Karbasi. Unconstrained submodular maximization with constant adaptive complexity. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2019, page 102–113, 2019.

[Cui *et al.*, 2021] Shuang Cui, Kai Han, Jing Tang, He Huang, Xueying Li, and Zhiyu Li. Streaming algorithms for constrained submodular maximization. *Proceedings of the ACM SIGMETRICS conference on Measurement and Analysis of Computer Systems*, 6(3):54:1–54:32, 2021.

[Cui *et al.*, 2023a] Shuang Cui, Kai Han, Jing Tang, He Huang, Xueying Li, and Aakas Zhiyuli. 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*, pages 7261–7269. AAAI Press, 2023.

[Cui *et al.*, 2023b] Shuang Cui, Kai Han, Jing Tang, He Huang, Xueying Li, Aakas Zhiyuli, and Hanxiao Li. Practical parallel algorithms for non-monotone submodular maximization. *CoRR*, abs/2308.10656, 2023.

[Ene and Nguyen, 2019] Alina Ene and Huy L. Nguyen. Submodular maximization with nearly-optimal approximation and adaptivity in nearly-linear time. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 274–282. SIAM, 2019.

[Ene and Nguyen, 2020] Alina Ene and Huy L. Nguyen. 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*, pages 2902–2911. PMLR, 2020.

[Ene *et al.*, 2019] Alina Ene, Huy L. Nguyen, and Adrian Vladu. 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*, pages 90–101. ACM, 2019.

[Fahrbach *et al.*, 2019] Matthew Fahrbach, Vahab S. Mirrokni, and Morteza Zadimoghaddam. Non-monotone submodular maximization with nearly optimal adaptivity and query complexity. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *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*, pages 1833–1842. PMLR, 2019.

[Feige *et al.*, 2011a] Uriel Feige, Vahab S. Mirrokni, and Jan Vondrák. Maximizing non-monotone submodular functions. *SIAM Journal on Computing*, 40(4):1133–1153, 2011.

[Feige *et al.*, 2011b] Uriel Feige, Vahab S. Mirrokni, and Jan Vondrák. Maximizing non-monotone submodular functions. *SIAM Journal on Computing*, 40(4):1133–1153, 2011.

[Gupta *et al.*, 2010] Anupam Gupta, Aaron Roth, Grant Schoenebeck, and Kunal Talwar. Constrained non-monotone submodular maximization: Offline and secretary algorithms. In *International Workshop on Internet and Network Economics*, 2010.

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

[Han *et al.*, 2021] Kai Han, Shuang Cui, Tianshuai Zhu, Enpei Zhang, Benwei Wu, Zhizhuo Yin, Tong Xu, Shaojie Tang, and He Huang. Approximation algorithms for submodular data summarization with a knapsack constraint. *Proceedings of the ACM SIGMETRICS conference on Measurement and Analysis of Computer Systems*, 5(1):05:1–05:31, 2021.

[Krizhevsky, 2019] Alex Krizhevsky. Learning multiple layers of features from tiny images. *Technical Reports, University of Toronto*, 2019.

[Kuhnle, 2021a] Alan Kuhnle. Nearly linear-time, parallelizable algorithms for non-monotone submodular maximization. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence 2021*, pages 8200–8208, 2021.

[Kuhnle, 2021b] Alan Kuhnle. Quick streaming algorithms for maximization of monotone submodular functions in linear time. In *Proceedings of the 24th International Conference on Artificial Intelligence and Statistics 2021*, volume 130 of *Proceedings of Machine Learning Research*, pages 1360–1368, 2021.

[Lee *et al.*, 2010] Jon Lee, Vahab S. Mirrokni, Viswanath Nagarajan, and Maxim Sviridenko. Maximizing nonmonotone submodular functions under matroid or knapsack constraints. *SIAM Journal on Discrete Mathematics*, 23(4):2053–2078, 2010.

[Li *et al.*, 2022] Wenxin Li, Moran Feldman, Ehsan Kazemi, and Amin Karbasi. Submodular maximization in clean linear time. In *Advances in Neural Information Processing Systems*, pages 7887–7897, 2022.

[Li, 2018] Wenxin Li. Nearly linear time algorithms and lower bound for submodular maximization. *preprint, arXiv:1804.08178*, 2018.

[Mirzasoleiman *et al.*, 2016] Baharan Mirzasoleiman, Ashwinkumar Badanidiyuru, and Amin Karbasi. Fast constrained submodular maximization: Personalized data summarization. In *International Conference on Machine Learning*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 1358–1367, 2016.

[Pham *et al.*, 2023] Canh V. Pham, Tan D. Tran, Dung T. K. Ha, and My T. Thai. Linear query approximation algorithms for non-monotone submodular maximization under knapsack constraint. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI 2023, 19th-25th August 2023, Macao, SAR, China*, pages 4127–4135. ijcai.org, 2023.

[Sun *et al.*, 2022] Xiaoming Sun, Jialin Zhang, Shuo Zhang, and Zhijie Zhang. Improved deterministic algorithms for non-monotone submodular maximization. In Yong Zhang, Dongjing Miao, and Rolf H. Möhring, editors, *Computing and Combinatorics - 28th International Conference, COCOON 2022 Proceedings*, volume 13595 of *Lecture Notes in Computer Science*, pages 496–507. Springer, 2022.