Size-Constrained k-Submodular Maximization in Near-Linear Time

Guanyu Nie[1] **Yanhui Zhu**[1] **Yididiya Y. Nadew**[1] **Samik Basu**[1] **A. Pavan**[1] **Christopher John Quinn**[1]

[1]Computer Science Deptartment, Iowa State University, Ames, IA, USA

Abstract

We investigate the problems of maximizing k-submodular functions over total size constraints and over individual size constraints. k-submodularity is a generalization of submodularity beyond just picking items of a ground set, instead associating one of k types to chosen items. For sensor selection problems, for instance, this enables modeling of which type of sensor to put at a location, not simply whether to put a sensor or not. We propose and analyze threshold-greedy algorithms for both types of constraints. We prove that our proposed algorithms achieve the best known approximation ratios for both constraint types, up to a user-chosen parameter that balances computational complexity and the approximation ratio, while only using a number of function evaluations that depends linearly (up to poly-logarithmic terms) on the number of elements n, the number of types k, and the inverse of the user chosen parameter. Other algorithms that achieve the best-known deterministic approximation ratios require a number of function evaluations that depends linearly on the budget B, while our methods do not. We empirically demonstrate our algorithms' performance in applications of sensor placement with k types and influence maximization with k topics.

1 INTRODUCTION

There are a number of problems that can be abstracted as selecting a subset of items with a limit on the number of items, and for which redundancy between items can lead to diminishing returns in terms of utility. Consider the problem of monitoring a traffic network using a limited number of sensors. We want to place sensors in the most informative locations. Putting additional sensors in close proximity to each other would be redundant and result in little additional information gain. Likewise, consider the problem of selecting a subset of influencers on social media to seed an advertising campaign. Sponsoring additional influencers will improve the spread, but if the additional influencers have the same followers, the improvement in the spread may be limited. Both of these problems, which involve selecting a subset of items, and for which redundancy can lead to diminishing returns, can be modeled as submodular maximization problems [Krause and Guestrin, 2007, Kempe et al., 2003].

A set function $f : 2^V \to \mathbb{R}$ over a set V of n elements is said to be submodular if, for any $S \subseteq T \subset V$ and $e \in V \setminus T$, it satisfies the following diminishing returns property,

$$f(S \cup \{e\}) - f(S) \geq f(T \cup \{e\}) - f(T).$$

This inequality means that the marginal gain of adding e to a set is non-increasing as the set gets larger. For many problems, the function f is assumed to be monotone non-decreasing: $f(T) \geq f(S)$ for any $S \subseteq T \subseteq V$. While maximizing a monotone submodular function without constraints is trivial (the optimal solution is the whole set V), the problem of maximizing a monotone submodular function with just a cardinality constraint of B is NP-hard even to approximate with a ratio above $(1 - 1/e) \approx 0.632$ [Nemhauser et al., 1978]. Surprisingly, a simple greedy algorithm can achieve the ratio of $(1 - 1/e)$ using $\mathcal{O}(nB)$ function evaluations [Nemhauser et al., 1978]. While submodular functions have been used in a number of applications, some problems cannot be modeled well by just selecting a single set. We give two examples to illustrate this.

Influence maximization with k topics: Influence maximization involves identifying a small subset, or seed set, within a network that can achieve the greatest possible spread of information. This selection problem is frequently modeled as a submodular maximization problem in social networks, as noted by Kempe et al. [2003]. However, if the information being spread includes multiple topics with varying effects on the network, the problem becomes more complex. Specifically, due to budget constraints, we must

Accepted for the 39th Conference on Uncertainty in Artificial Intelligence (UAI 2023).

limit our seed set to a specific number of individuals, with each person being assigned a specific topic. In such cases, the standard submodular maximization approach may not be sufficient and could lead to a loss of important information.

Sensor placement with k types: Consider the case where we want to monitor a traffic network through sensors. With a limited budget, we only want to place sensors in the most informative locations. This problem can be modeled as submodular maximization [Krause and Guestrin, 2007]. However, the standard submodular maximization model fails to account for scenarios where we have multiple sensor types (e.g., temperature, humidity, illuminance) that need to be installed at each location, with only one sensor per location.

To account for variations in sensor types or sponsored messages, a richer class of functions is needed beyond submodular functions. In particular, the class of *k-submodular* functions [Cohen et al., 2006, Kolmogorov, 2011, Huber and Kolmogorov, 2012] can be used for these problems. Instead of simply including an item $e \in V$, each selected item is assigned one of k types. Marginal gains can depend on the pair $(e, i) \in V \times \{1, \ldots, k\}$. The special case of $k = 2$, *bisubmodular* functions, has been widely studied For example, Singh et al. [2012] conducted sensor placement experiments with bisubmodular function models. For general k, Ward and Živnỳ [2014] mentioned the applications of the k-submodular function on sensor placement and feature selection. Ohsaka and Yoshida [2015] proposed algorithms for k-submodular optimization and applied them to sensor placement and influence maximization problems.

In this paper, we focus on size (cardinality) constraints. Specifically, we consider *total size* (TS) constraints, where there is a limit on the total number of items from the ground set selected regardless of type, and *individual size* (IS) constraints, where each of the k types has its own limit. Neither problem is a special case of the other. For a total size constraint, $1/2$ is the best known approximation ratio. For individual size constraints, the best known approximation ratio is $1/3$. In both cases, there is room for improvement in terms of the run time. A comparison table of works related to our methods can be found in Table 1. In Section 1.2 we discuss related works in more detail.

In this work we propose algorithms for maximizing k-submodular functions under TS and IS constraints, achieving the best deterministic approximation ratios (up to a user specified $\varepsilon > 0$) while removing linear dependence on the constraint budgets in terms of value oracle complexity.

1.1 OUR CONTRIBUTIONS

The contribution is threefold. First, we propose a threshold greedy algorithm for k-submodular maximization under a total size constraint, achieving a $(1/2 - \varepsilon)$-approximation guarantee using only $\mathcal{O}(kn\varepsilon^{-1}\log(B\varepsilon^{-1}))$ function evaluations, for any user chosen $\varepsilon > 0$. kn evaluations are the minimum needed to try each item-type pair once. This is the first algorithm that achieves a deterministic, near-optimal approximation ratio under total size constraints without linear dependence on the budget B in terms of function evaluations. Since B can be as large as n, this is significant.

Second, we propose a threshold greedy algorithm for k-submodular maximization under an individual size constraint, achieving a $(1/3 - \varepsilon)$-approximation guarantee using only $\mathcal{O}(kn\varepsilon^{-1}\log(B\varepsilon^{-1}))$ function evaluations, for any user chosen $\varepsilon > 0$. This is the first algorithm that achieves nearly the best known deterministic approximation ratio under individual size constraints without linear dependence on the budget B in terms of function evaluations. It also removes quadratic dependence on the number of types k in value oracle complexity compared to a stochastic greedy method. Third, we test our method using real-world data.

1.2 RELATED WORKS

We next review related works on maximizing monotone k-submodular functions, grouping works by constraint types.

Unconstrained: Unlike the $k = 1$ case, maximizing a monotone k-submodular function even without constraints is challenging. Iwata et al. [2015] showed that even achieving an approximation ratio $\alpha \in (\frac{k+1}{2k}, 1]$ is NP-hard. Ward and Živný [2014] achieved $\max\{1/3, 1/(1+a)\}$ approximation guarantee using $\mathcal{O}(kn)$ number of function evaluations for the unconstrained case, where $a = \max\{1, \sqrt{(k-1)/4}\}$. Iwata et al. [2015] improved the guarantee to $\frac{k}{2k-1}$ using the same number of oracle calls.

Size Constraints: For size constraints, two types of constraints have been considered in the literature, namely total size (TS) constraints, where the number of items selected shares a common budget, and individual size (IS) constraints, where each of the k types has a budget. Ohsaka and Yoshida [2015] analyzed the greedy algorithm and obtained $1/2$ and $1/3$ approximation guarantees for total size (TS) constraints and individual size (IS) constraints, respectively. They also proposed stochastic versions of their greedy algorithms to reduce the number of function evaluations, inspired by the $k = 1$ stochastic greedy algorithm proposed in Mirzasoleiman et al. [2015]. Those algorithms obtain the same approximation guarantee with a user-specified probability, but reduce time complexity from $\mathcal{O}(knB)$ to $\mathcal{O}(k(n-B)\log B \log \frac{B}{\delta})$ and $\mathcal{O}(k^2 n \log \frac{B}{k} \log \frac{B}{\delta})$ for total size and individual size, respectively, with δ denoting the user-specified failure probability bound.

Qian et al. [2017] proposed a multiobjective evolutionary type algorithm for total size constraint, and showed that their algorithm can find a $1/2$-approximation solution using $\mathcal{O}(knB\log^2 B)$ oracle calls. Matsuoka and Ohsaka

[2021] utilized curvature for k-submodular functions, weak k-submodularity, and approximate k-submodularity to analyze how curvature improves the approximation ratios of the standard greedy and residual random greedy algorithms. The aforementioned works are for the offline setting, which we also consider. For the streaming setting, Ene and Nguyen [2022] proposed an algorithm that achieves $\frac{1}{2(1+B_{\min}(2^{1/B_{\min}}-1))} \in (0.3, 0.25)$ approximation using only $\mathcal{O}(nk)$ number of queries, where $B_{\min} = \min_{i\in[k]} B_i$.

Other Constraints: Some recent works have considered knapsack constraints. If the cost of an item is the same across all types, Tang et al. [2022] proposed an algorithm inspired by Khuller et al. [1999], Sviridenko [2004] that achieves $\frac{1}{2}(1-\frac{1}{e})$ using $\mathcal{O}(k^4n^5)$ number of function evaluations. Chen et al. [2022] proposed a partial-enumeration algorithm inspired by Khuller et al. [1999] that achieves $\frac{1}{4}(1-\frac{1}{e})$ with time complexity $\mathcal{O}(kn^2)$. Pham et al. [2021] proposed an algorithm for the streaming setting that achieves a $\frac{1}{4}-\varepsilon$ approximation guarantee and if the cost is over more general item-type pairs, their algorithm achieves $\min\{\frac{\alpha}{2}, \frac{(1-\alpha)k}{(1+\beta)k-\beta}\} - \varepsilon$ using $\mathcal{O}(\frac{kn}{\varepsilon}\log B)$ queries where $\beta = \max_{i\neq j} \frac{c(e,i)}{c(e,j)}$, and $\alpha \in (0,1], \varepsilon \in (0,1)$ are input parameters. For matroid constraints, Sakaue [2017] proposed an algorithm that achieves a $1/2$-approximation and Matsuoka and Ohsaka [2021] proposed an algorithm that achieves a $\frac{1}{1+c}$-approximation where c is the curvature.

2 PROBLEM STATEMENT

We begin with background materials and notation. We then state the two problems we consider.

For an positive integer i let $[i] := \{1, \ldots, i\}$ denote the set of integers up to and including i. For a set S, let $|S|$ denote its cardinality. Let V denote a set of *items* (such as locations where a sensor could be placed). Let $n := |V|$ denote the number of items. Let $[k] := \{1, \ldots, k\}$ denote the set of possible *types* (such as available types of sensors). There are equivalent ways to express solutions. The notation 2^V for the power-set of V in the $k = 1$ setting can be generalized to the set $(k+1)^V$ of length-$|V|$ $(k+1)$-ary tuples to denote type assignments for items, with a 0 indicating no assignment (i.e. no sensor placed in that location). For clarity, we will mostly denote solutions by item-type pairs. Let $\mathcal{S}$ denote the set of subsets of item-type pairs corresponding to elements of $(k+1)^V$,

$$\mathcal{S} := \{ \bigcup_{\substack{j\in[n]:\\ A(j)\neq 0}} (V(j), A(j)) \mid A \in (k+1)^V \}.$$

Equivalently, V can be partitioned by type,

$$\mathcal{X} := \{ (\bigcup_{\substack{j\in[n]:\\ A(j)=1}} \{V(j)\}, \ \ldots, \ \bigcup_{\substack{V(j)\in[n]:\\ A(j)=k}} \{V(j)\}) \mid A \in (k+1)^V \}$$

denoting the sets of partitions of elements by type (among elements with an assignment). For any $S \in \mathcal{S}$, for each type $i \in [k]$, we define $U_i(S) := \{a \in V \mid \text{s.t. } (a,i) \in S\}$ to be the set of items assigned type i. We also define $U(S) := \bigcup_{i\in[k]} U_i(S)$ to denote the set of elements with some type-assignment.

We call a function $f : \mathcal{S} \to \mathbb{R}$ *monotone (non-decreasing)* if for any sets $S, S' \in \mathcal{S}$ over item-type pairs satisfying $S \subseteq S'$, $f(S) \leq f(S')$. We call a monotone function $f : \mathcal{S} \to \mathbb{R}$ *k-submodular* if for any sets $S, S' \in \mathcal{S}$ satisfying $S \subseteq S'$, and any item-type pair (e,i) with $e \notin U(S')$ (i.e. the item has no assigned type), f satisfies a diminishing returns property,

$$f(S \cup \{(e,i)\}) - f(S) \geq f(S' \cup \{(e,i)\}) - f(S').$$

We refer to such differences as marginal gains, representing them using conditioning notation $f((e,i)|S) := f(S \cup \{(e,i)\}) - f(S)$.

First, we introduce a lemma presented in Tang et al. [2022] that will be used in proofs.

Lemma 1. *(Tang et al. [2022]) For any $S, S' \in \mathcal{S}$ with $S \subseteq S'$, we have*

$$f(S') - f(S) \leq \sum_{(e,i)\in S'\setminus S} f((e,i)|S).$$

Remark 2. We use set notation (with sets over item-type pairs) to simplify the presentation. We note that f and subsequently marginal gains are only defined over $\mathcal{S}$, for which there are no item-type pairs with the same item. For non-monotone functions, k-submodular functions are those with the above diminishing returns property (referred to as orthant submodularity) and an additional pairwise-monotonicity condition [Ward and Živnỳ, 2014].

In the following, let f be an arbitrary non-negative, monotone, k-submodular function. We further assume that $f(\emptyset) = 0$, which is without loss of generality because otherwise, we can redefine $f(S) := f(S) - f(\emptyset)$ for all $S \in \mathcal{S}$.

We next state the two problems we consider.

Problem 1. For a monotone k-submodular function f and total size constraint B, solve

$$\operatorname*{arg\,max}_{S\in\mathcal{S}:\ |S|\leq B} f(S).$$

Problem 2. For a monotone k-submodular function f and individual size constraints $\{B_i\}_{i=1}^k$, solve

$$\operatorname*{arg\,max}_{\substack{S\in\mathcal{S}:\\ |U_i(S)|\leq B_i\ \forall i\in[k]}} f(S).$$

Neither Problem 1 nor Problem 2 are special cases of the other, but both generalize unconstrained maximization, for which it was shown that it is NP-hard to even approximate with a ratio larger than $\frac{k+1}{2k} > \frac{1}{2}$ [Iwata et al., 2015].

Table 1: Table of selected related works. For unconstrained maximization, which both Problems 1 and 2 generalize, it is known that it is NP-hard to approximate better than $\frac{k+1}{2k} > \frac{1}{2}$ [Iwata et al., 2015]. *For individual size constraints, $B \leftarrow \sum_{i=1}^{k} B_i$. †This result is for the streaming case, with $B_{\min} = \min_{i\in[k]} B_i$. The approximation guarantee is at least 1/4 and achieves its best (0.2953) when $B_{\min}$ tends to infinity. ‡Curvature $c \in [0,1]$ with $c = 0$ for linear functions.

Reference	Constraint	Approximation	Time
Ohsaka and Yoshida [2015]	total size	1/2	$\mathcal{O}(knB)$
Ohsaka and Yoshida [2015]		1/2 with prob. $\geq 1-\delta$	$\mathcal{O}(kn \log B \log \frac{B}{\delta})$
Qian et al. [2017]		1/2	$\mathcal{O}(knB \log^2 B)$
This paper		$1/2 - \varepsilon$	$\mathcal{O}(kn\varepsilon^{-1} \log(B\varepsilon^{-1}))$
Ohsaka and Yoshida [2015]	individual size*	1/3	$\mathcal{O}(knB)$
Ohsaka and Yoshida [2015]		1/3 with prob. $\geq 1-\delta$	$\mathcal{O}(k^2 n \log \frac{B}{k} \log \frac{B}{\delta})$
Ene and Nguyen [2022]		$\frac{1}{2(1+B_{\min}(2^{1/B_{\min}}-1))}$ †	$\mathcal{O}(kn)$
Matsuoka and Ohsaka [2021]		$\frac{1}{1+2c}$ where c is curvature ‡	$\mathcal{O}(knB)$
This paper		$1/3 - \varepsilon$	$\mathcal{O}(kn\varepsilon^{-1} \log(B\varepsilon^{-1}))$

Algorithm 1 k-submodular Threshold Greedy-TS

Input: access to a value oracle for a monotone k-submodular function $f : (k+1)^V \to \mathbb{R}^+$, an integer budget $B \in \mathbb{Z}^+$ and a tolerance parameter $\varepsilon > 0$.
Output: a set S of item-index pairs with $|U(S)| \leq B$.
Initialize $S \leftarrow \emptyset$, $\tau \leftarrow d = \max_{e\in V, i\in[k]} f(\{(e,i)\})$
while $\tau > \frac{(1-\varepsilon)\varepsilon d}{2B}$ **do**
  for $e, i \in V \setminus U(S), [k]$ **do**
    if $|U(S)| < B$ and $f((e,i)|S) \geq \tau$ **then**
      $S \leftarrow S \cup \{(e,i)\}$.
    end if
  end for
  Update $\tau \leftarrow (1-\varepsilon)\tau$.
end while
Return S

3 THRESHOLD GREEDY - TOTAL SIZE

In this section, we present our first algorithm designed for Problem 1, maximizing a k-submodular function under a total size constraint B. The pseudo-code is presented in Algorithm 1. The algorithm design is inspired by the threshold greedy algorithm for $k = 1$ submodular maximization proposed by Badanidiyuru and Vondrák [2014]. Algorithm 1 uses a decreasing sequence of thresholds, starting from $d := \max_{e\in V, i\in[k]} f(\{(e,i)\})$, which is the largest value among any item-type pair. For each threshold τ considered, the algorithm iterates over all item-type pairs that are still feasible, in an arbitrary order. A feasible item-type pair (e,i) is added to the current solution S if its marginal gain with respect to S is above the current threshold, $f((e,i)|S) \geq \tau$. After going over all the item-type pairs, the algorithm lowers the threshold and repeats. The algorithm will terminate when the selected subset uses up the budget $|S| = B$ or the lower bound for the threshold is reached $\frac{\varepsilon d}{2B}$.

Remark 3. For implementation, we will use lazy evaluation Minoux [1978]. If the output set S does not use up the size budget B, we can pad the output S with extra feasible elements to use up the budget, such as based on their previously marginal gains with respect to earlier conditioning sets (if using lazy evaluation) or a randomly chosen set. Provided this step does not involve value queries, by monotonicity the following guarantees will still hold.

We next state our main results for Problem 1.

Theorem 4. *Algorithm 1 achieves a $(1/2 - \varepsilon)$-approximation for the problem of maximizing a monotone k-submodular function under a total size constraint using at most $\mathcal{O}(nk\varepsilon^{-1} \log(B\varepsilon^{-1}))$ function evaluations.*

Proof. **Run-time:** The run-time is dominated by function evaluations. The **for** loop takes $\mathcal{O}(nk)$ time. The outer **while** loop is called t' times, where t' is the smallest integer t' such that $(1-\varepsilon)^{t'} d \leq \frac{(1-\varepsilon)\varepsilon d}{2B}$. Let t denote the value where equality holds, so $t' = \lceil t \rceil$. Rearranging, t satisfies

$$t = 1 - \frac{\log(2B\varepsilon^{-1})}{\log(1-\varepsilon)} \leq 1 + \frac{\log(2B\varepsilon^{-1})}{\varepsilon},$$

where the last inequality follows from $\log(1-x) < -x$ for $x < 1$. Then, t' can be upper bounded by $\mathcal{O}(\varepsilon^{-1} \log(B\varepsilon^{-1}))$ by observing that $t' \leq t + 1$. Thus, with $\mathcal{O}(\varepsilon^{-1} \log(B\varepsilon^{-1}))$ calls of the outer **while** loop, the total run time is $\mathcal{O}(nk\varepsilon^{-1} \log(B\varepsilon^{-1}))$.

Approximation guarantee: For proving the approximation guarantee, we divide the problem into two cases. The first case is when we used up the budget, i.e., we have selected B items upon the execution of the algorithm is finished. Similar to the construction in [Ohsaka and Yoshida, 2015], we consider swapping one element at a time from the output of our algorithm, S, with one item in the optimal set OPT. Since $|\mathrm{OPT}| = B$, we need at most B steps to construct OPT starting from S. Then, we show that each step of the

swapping result in a small gain in the function value, and thus the total advantage of the OPT over S is also not large. The second case is when we have selected less than B items. The result from the first case also holds if the algorithm selected B items ignoring the size constraint. Then, since the gain of the items exceeding the budget is small (less than the minimum threshold by the selection rule), the total gain of those items is also small. This will indicate that the value of the selected set is also not far from the value of OPT. Let S° denote the solution output by Algorithm 1.

Case 1: When the final selected S° satisfies $|U(S^\circ)| = B$.

Let $(e_j, i_j) \in V \times [k]$ denote the j-th pair chosen by the algorithm, and let S_j denote the set S after (e_j, i_j) was added. We define $S_0 = \emptyset$. Let OPT be the optimal solution. In the following, we will compare the marginal gains of item-type pairs in S° to the marginal gains for item-type pairs in OPT. We will construct a sequence of subsets of item-type pairs, combining pairs from S° and OPT in order to show inequalities resulting in the stated approximation bound. Some care will be needed for item-type pairs $(e, i) \in S^\circ$ and $(e, i') \in$ OPT where an item e is included in both S° and OPT but with different types.

We begin by indexing the item-type pairs in the output $S^\circ = \{(e_1, i_1), \ldots, (e_B, i_B)\}$ in the order they were added to form S°. We next let S_j denote S after j elements were added, so $S_j := \{(e_1, i_1), \ldots, (e_j, i_j)\}$ for $j \in [B]$ and we set $S_0 := \emptyset$ as the initial empty set. Thus by construction $f(S_{j+1}) - f(S_j) = f((e_{j+1}, i_{j+1})|S_j)$.

We also index the item-type pairs in the optimal solution $\mathrm{OPT} = \{(e'_1, i'_1), \ldots, (e'_B, i'_B)\}$ using the same indices in OPT as we have in S° for pairs containing the same items. That is, if the item e_j in the jth selected pair (e_j, i_j) in Algorithm 1 is also in a pair (e_j, i') in OPT, the latter pair should have the same index (even though the type i_j and i' assigned to that item in S° and OPT respectively may be different). For other pairs in OPT, the indexing is arbitrary. With this alignment of indices of pairs in S° and OPT that share a common item, we construct a sequence of cardinality B sets O_0, O_1, …, O_B, by swapping elements of OPT with elements of S° in increasing order of the indexing (i.e. beginning with the first pair selected by Algorithm 1),

$$\begin{aligned}
O_0 :=& \{(e'_1, i'_1), (e'_2, i'_2), \ldots, (e'_{B-1}, i'_{B-1}), (e'_B, i'_B)\} \\
O_1 :=& \{(e_1, i_1), (e'_2, i'_2), \ldots, (e'_{B-1}, i'_{B-1}), (e'_B, i'_B)\} \\
&\vdots \\
O_{B-1} :=& \{(e_1, i_1), (e_2, i_2), \ldots, (e_{B-1}, i_{B-1}), (e'_B, i'_B)\} \\
O_B :=& \{(e_1, i_1), (e_2, i_2), \ldots, (e_{B-1}, i_{B-1}), (e_B, i_B)\}.
\end{aligned}$$

By construction, for $j \in \{0, \ldots, B\}$ we have $S_j \subseteq O_j$. Furthermore, for $j + 1 \in [B]$, we have $S_j \subseteq O_j \cap O_{j+1}$. By this construction, we also have that at the time when Algorithm 1 selected its $(j + 1)$st pair (e_{j+1}, i_{j+1}), the aligned pair in OPT, (e'_{j+1}, i'_{j+1}) was also feasible. This entails that both the item was still available,

$$e'_{j+1} \notin U(S_j) \quad \forall\, j + 1 \in [B], \tag{1}$$

and that the budget had not yet been consumed. Trivially the budget B was never violated. (1) follows by construction since either the items in the $(j + 1)$st pairs in S° and OPT match, $e'_{j+1} = e_{j+1}$, for which reason we would have aligned them in the first place, or, if they are not the same, then that item was never chosen by Algorithm 1, so $e'_{j+1} \notin U(S^\circ)$, for which the ordering of OPT is arbitrary but importantly means e'_{j+1} was always available.

We consider the difference $f(O_j) - f(O_{j+1})$. This is not a marginal gain since neither set contains the other. However, since the sets differ in exactly one index (the $(j + 1)$st) by construction, we can bound the difference.

$$\begin{aligned}
&f(O_j) - f(O_{j+1}) \\
&= \big(f(O_j \cap O_{j+1}) + f((e'_{j+1}, i'_{j+1})|O_j \cap O_{j+1})\big) \\
&\qquad - \big(f(O_j \cap O_{j+1}) + f((e_{j+1}, i_{j+1})|O_j \cap O_{j+1})\big) \\
&\le f((e'_{j+1}, i'_{j+1})|O_j \cap O_{j+1}) \qquad (2) \\
&\le f((e'_{j+1}, i'_{j+1})|S_j), \qquad (3)
\end{aligned}$$

where (2) follows from monotonicity and (3) follows from orthant submodularity with $S_j \subseteq O_j \cap O_{j+1}$. We cannot directly relate the marginal gain $f((e'_{j+1}, i'_{j+1})|S_j)$ to that achieved by the $(j + 1)$st pair added to S° in Algorithm 1, $f((e_{j+1}, i_{j+1})|S_j)$, since the ordering of the **for** loop is arbitrary. We will be able to bound the ratio of those two marginal gains based on how the threshold τ is shrunk. We will first consider the case the threshold was the initial threshold and then consider the alternative case.

Sub-case $\tau_{j+1} = d$: Suppose the threshold τ_{j+1} when the $(j + 1)$st pair was added in Algorithm 1 was the initial threshold, $\tau_{j+1} = d$, by construction the largest of all marginal gains. Then

$$f((e_{j+1}, i_{j+1})|S_j) = \tau_{j+1} = d \ge f((e'_{j+1}, i'_{j+1})|S_j).$$

Sub-case $\tau_{j+1} < d$: Suppose the threshold $\tau_{j+1} < d$, equivalently Algorithm 1 is in the second or later execution of the outer **while** loop. The pair (e'_{j+1}, i'_{j+1}) was considered in the previous **while** loop execution but not added. (Recall the element e'_{j+1} is either e_{j+1} or an element never chosen in S°, and thus any pair containing e'_{j+1} was still feasible in the previous **while** loops.) Since the pair (e'_{j+1}, i'_{j+1}) was not added, its marginal gains with respect to the greedy set in the previous **while** loops, one of $\{S_j, S_{j-1}, \ldots, \emptyset\}$, must have been below the previous threshold $\tau_{j+1}(1-\varepsilon)^{-1}$. By orthant submodularity, marginal gains are non-increasing in the conditioning set, so

$$\begin{aligned}
f((e'_{j+1}, i'_{j+1})|S_j) &\le (1-\varepsilon)^{-1}\tau_{j+1} \qquad (4) \\
&\le (1-\varepsilon)^{-1} f((e_{j+1}, i_{j+1})|S_j), \qquad (5)
\end{aligned}$$

where (4) holds because (e'_{j+1}, i'_{j+1}) is not chosen in a previous round and (5) holds because (e_{j+1}, i_{j+1}) is chosen in this round. Thus, whether (e_{j+1}, i_{j+1}) was added during the first execution of the **while** loop or later, we have

$$\begin{aligned} f(O_j) - f(O_{j+1}) &\leq f((e'_{j+1}, i'_{j+1})|S_j) \\ &\leq (1-\varepsilon)^{-1} f((e_{j+1}, i_{j+1})|S_j) \\ &= (1-\varepsilon)^{-1}\big(f(S_{j+1}) - f(S_j)\big). \quad (6) \end{aligned}$$

Using a telescoping sum with $O_0 = \mathrm{OPT}$ and $O_B = S^\circ$,

$$\begin{aligned} f(\mathrm{OPT}) - f(S^\circ) &= \sum_{j=0}^{B-1} (f(O_j) - f(O_{j+1}) \\ &\leq \sum_{j=0}^{B-1} \frac{1}{1-\varepsilon}(f(S_{j+1}) - f(S_j)) \\ &= \frac{1}{1-\varepsilon} f(S^\circ), \end{aligned}$$

using (6) and $f(\emptyset) = 0$, which for $\varepsilon < 1$ implies

$$\begin{aligned} f(S^\circ) &\geq \frac{1-\varepsilon}{2-\varepsilon} f(\mathrm{OPT}) \\ &\geq (\frac{1}{2} - \varepsilon) f(\mathrm{OPT}). \quad (7) \end{aligned}$$

Case 2: When the final selected S° satisfies $|U(S^\circ)| < B$. Let $\ell = |U(S^\circ)| < B$ denote the number of elements added. Let $\tilde{S}$ denote a set of cardinality B that Algorithm 1 would have selected if Algorithm 1 terminated only when either (a) B pairs had been selected or (b) the marginal gains on all remaining elements evaluated as zero. Without loss of generality, we only consider (a), as we could trivially identify that event (b) was occurring and add an arbitrary feasible subset of item-type pairs so that $|U(S^\circ)| = B$ without any reduction in value, and the following inequalities will still hold (any item-type pairs in OPT that were feasible once all marginal gains reduced to zero, but not chosen to pad S°, have equal marginal gains to those that were). Thus, by construction $S^\circ \subset \tilde{S}$ and $\tilde{S}$ has $B - \ell$ extra elements.

First, since $\tilde{S}$ has B elements selected according to decreasing thresholds, the result (7) from **Case 1** holds for $\tilde{S}$,

$$f(\tilde{S}) \geq \frac{1-\varepsilon}{2-\varepsilon} f(\mathrm{OPT}). \quad (8)$$

Second, since S° only accumulated ℓ elements before the terminal threshold bound of $\frac{(1-\varepsilon)\varepsilon d}{2B}$ was reached, then the marginal gains of the remaining $B - \ell$ elements in $\tilde{S}$ can be bounded, with the largest possible value of the threshold τ in the last execution of the **while** loop being $(1-\varepsilon)^{-1}\frac{(1-\varepsilon)\varepsilon d}{2B} = \frac{\varepsilon d}{2B}$, using Lemma 1,

$$\begin{aligned} f(\tilde{S}) - f(S^\circ) &\leq \sum_{(e,i)\in\tilde{S}\setminus S^\circ} f((e,i)|S^\circ) \\ &\leq (B-\ell)\frac{\varepsilon d}{2B} \\ \iff \quad f(S^\circ) &\geq f(\tilde{S}) - (B-\ell)\frac{\varepsilon d}{2B}. \quad (9) \end{aligned}$$

We note that since by construction $S^\circ \subset \tilde{S}$, each of the item-index pairs in $\tilde{S}\setminus S^\circ$ must have contained items not in $U(S^\circ)$, so the marginal gains in the formulas above are well-defined.

With (8) and (9), monotonicity of f, and $d \leq f(\mathrm{OPT})$,

$$\begin{aligned} f(S^\circ) &\geq f(\tilde{S}) - (B-\ell)\frac{\varepsilon d}{2B} && \text{(by (9))} \\ &\geq \frac{1-\varepsilon}{2-\varepsilon} f(\mathrm{OPT}) - (B-\ell)\frac{\varepsilon d}{2B} && \text{(by (8))} \\ &\geq (\frac{1}{2} - \varepsilon) f(\mathrm{OPT}). \end{aligned}$$

□

Algorithm 1 and the proof of Theorem 4 generalizes the threshold greedy algorithm for submodular ($k = 1$) functions proposed in [Badanidiyuru and Vondrák, 2014]. The reason why a threshold algorithm can improve the greedy algorithm on time complexity is that, while the greedy algorithm considers adding one element during one pass of all the remaining elements, the threshold algorithm adds multiple elements during one pass. By utilizing similar techniques for choosing threshold sequences, we obtain the same reduction in worst-case function evaluations with the same additive reduction in the (worst-case) approximation ratio. More specifically, Badanidiyuru and Vondrák [2014] achieved a $(1 - 1/e - \varepsilon)$-approximation guarantee for the $k = 1$ case, where $1 - 1/e$ is the best possible when $k = 1$; we achieved $(1/2 - \varepsilon)$-approximation, where $1/2$ is (asymptotically) the best possible for general k [Iwata et al., 2015]. As for the time complexity, Ohsaka and Yoshida [2015] improved from order B to $\log(B)$ by considering a random set with size of order $\frac{\log(B)}{B}$ when adding each element. They showed that the aforementioned set has overlapping items with items in an optimal set that are still available with high probability. With a threshold strategy, we improved the run time from order B to $\log(B)$ by considering a sequence of $\mathcal{O}(\log(B))$ thresholds.

4 THRESH. GREEDY - INDIVIDUAL SIZE

In this section, we present our second algorithm, a threshold greedy algorithm for Problem 2, maximizing a k-submodular function under individual size constraints. The

Algorithm 2 k-submodular Threshold Greedy-IS

Input: access to a value oracle for a monotone k-submodular function $f : (k+1)^V \to \mathbb{R}^+$, integers $B_1, \cdots, B_k \in \mathbb{Z}^+$ and a tolerance parameter ε.
Output: an item-index pair set S with $|U_i(S)| \leq B_i$ for each $i \in [k]$.
Initialize $S \leftarrow \emptyset$, $B \leftarrow \sum_{i\in[k]} B_i$, $\tau \leftarrow d = \max_{e\in V, i\in[k]} f(\{(e,i)\})$
while $\tau > \frac{(1-\varepsilon)\varepsilon d}{3B}$ **do**
  $I \leftarrow \{i \in [k] : |U_i(S)| < B_i\}$
  for $e, i \in V \setminus U(S), I$ **do**
    if $f((e,i)|S) \geq \tau$ **then**
      $S \leftarrow S \cup \{(e,i)\}$.
    end if
  end for
  Update $\tau \leftarrow (1-\varepsilon)\tau$.
end while
Return S

pseudo-code is presented in Algorithm 2. Similar to Algorithm 1, the algorithm considers a decreasing sequence of thresholds, starting from $d = \max_{e\in V, i\in[k]} f(\{(e,i)\})$. For each threshold τ considered, the algorithm searches through all the item-type pairs that satisfy the following conditions: the item has not been chosen regardless of type, and the number of items with type i has not exceeded B_i. While searching, the algorithm includes the item-type pairs whose marginal gains are above the current threshold. After going over all such item-type pairs, the algorithm decreases the threshold and repeats the search. The algorithm will terminate when the selected subset already satisfies $|U_i(S)| = B_i$ for all $i \in [k]$ or the considered threshold is below $\frac{\varepsilon d}{3B}$, where $B = \sum_{i\in[k]} B_i$. We have the following result.

Theorem 5. *Algorithm 2 achieves a $(1/3 - \varepsilon)$-approximation for the problem of maximizing a monotone k-submodular function under individual size constraints using at most $\mathcal{O}(nk\varepsilon^{-1}\log(B\varepsilon^{-1}))$ function evaluations.*

In [Ohsaka and Yoshida, 2015], to design a stochastic version of the greedy algorithm in the individual size constraint case, unlike the total size constraint case, we need a set that overlaps with the available items in an optimal set with high probability *for each type*. This induces an additional k term compared with the total size constraint. For a threshold strategy, there is no such concern since we consider all available elements during each iteration of the **while** loop. It is still feasible to use a sequence of thresholds with size $\mathcal{O}(\log(B))$ to get the same approximation guarantee.

In the interest of space, we defer the proof of Theorem 5 to Appendix A. The general structure of the proof is similar to the proof of Theorem 4, but extra care is taken in constructing the sequence of intermediate solutions $\{O_1, \ldots, O_{B-1}\}$. In the proof of Theorem 4 for a total size constraint, when constructing O_{j+1} from O_j, we can swap types without limitation as long as the total budget is not exceeded, as there is no constraint on a specific type. With individual size constraints, however, multiple select swaps are needed going from O_j to O_{j+1} to maintain feasibility.

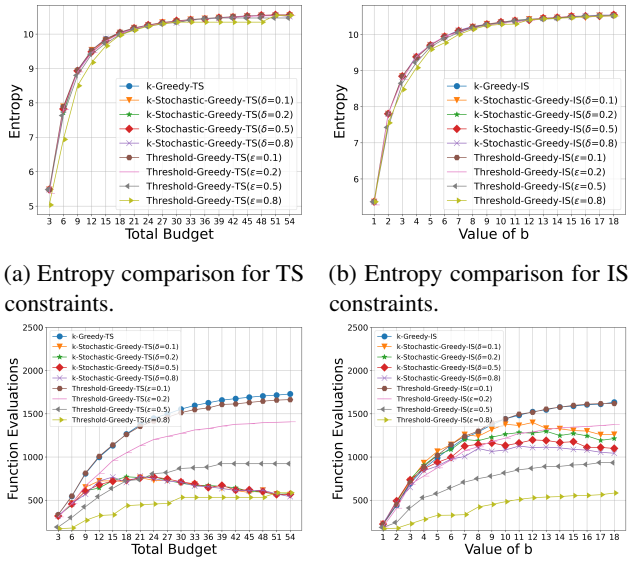
This is the first algorithm to achieve a deterministic, nearly 1/3-approximation without linear dependence on the budgets. The stochastic greedy algorithm proposed in [Ohsaka and Yoshida, 2015] achieves 1/3 with (at least) a user-specified probability, and the run time bound is slower than ours by a factor of k. In [Ene and Nguyen, 2022], a streaming algorithm with only $\mathcal{O}(kn)$ queries is proposed, but the approximation ratio is worse.

5 EXPERIMENTS

In this section, we empirically evaluate the performance of our proposed methods with applications of sensor placement with k types and influence maximization with k topics. We compare our results with baselines in terms of both the objective value achieved and oracle complexity. The code of our experiments can be found on `https://github.com/yididiyan/k_submodular/`.

Baselines: We compare our algorithms against the greedy and stochastic greedy algorithms proposed in [Ohsaka and Yoshida, 2015]. For all implementations, we use lazy evaluation. For our proposed threshold algorithms, we consider $\varepsilon = 0.1, 0.2, 0.5$ and 0.8. We note that for our threshold algorithms with approximation ratios (in the worst-case) of $\frac{1}{2} - \varepsilon$ and $\frac{1}{3} - \varepsilon$ for total size constraints and individual size constraints respectively, the worst-case bound becomes vacuous, but the algorithm could potentially work well in practice, as was shown in experiments in [Li et al., 2022] for a threshold algorithm for $k = 1$ submodular maximization with a knapsack constraint. For stochastic greedy algorithms, we use $\delta = 0.1, 0.2, 0.5$ and 0.8 (δ bounds the failure probability of achieving the stated approximation ratio) for fair comparisons with threshold greedy algorithms, although in the original paper [Ohsaka and Yoshida, 2015], only $\delta = 0.1$ was considered. We do not consider other baselines mentioned in Table 1 as there are differences in setups. We refer to Appendix B for a more detailed discussion.

Metrics: We evaluate the performance of our methods and baselines according to two criteria: the objective value and the number of function queries. We first explore how these depend on the constraint parameters, namely the total budget B for total-size constraints and the type-specific budgets $\{B_i\}$ for individual size. Then we demonstrate the main advantage of the proposed threshold greedy algorithm over the stochastic greedy algorithm under individual size constraints, namely the improvement by a factor of k in the theoretical guarantee.



(a) Entropy comparison for TS constraints.

(b) Entropy comparison for IS constraints.

(c) Comparison of function queries for TS constraints.

(d) Comparison of function queries for IS constraints.

Figure 1: Sensor placement over k types.

5.1 SENSOR PLACEMENT

Sensor Placement with k types: In this section, we apply our algorithms for maximizing k-submodular functions with the individual size constraint to the sensor placement problem with k kinds of sensors. To formally define the problem, we need several notations from information theory. Let $\Omega = \{X_1, X_2, \ldots, X_n\}$ be a set of discrete random variables. The entropy of a subset S of Ω is defined as $H(S) = -\sum_{s \in \mathrm{dom} S} \Pr[s] \log \Pr[s]$. The conditional entropy of Ω having observed S is $H(\Omega \mid S) := H(\Omega) - H(S)$. In the sensor placement problem, we want to set the sensors so as to maximize the reduction of expected entropy, which is equivalent to finding a set S that maximizes the entropy.

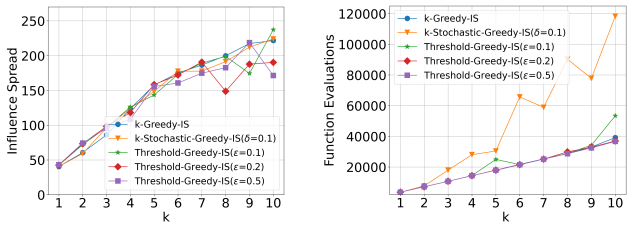
Now we formalize the sensor placement problem. There are k kinds of sensors for different measures. For total size constraints, we want to allocate B sensors to set V of n locations. For individual size constraints, we want to allocate B_i many sensors of the i-th kind for each $i \in [k]$ to set V of n locations. In both settings, each location can be instrumented with exactly one sensor. Let X_e^i be the random variable representing the observation collected from a sensor of the i-th kind if it is installed at the e-th location, and let $\Omega = \left\{X_e^i\right\}_{e \in V, i \in [k]}$. Then, the problem is to select $S \in (k+1)^V$ that maximizes $f(S) = H\left(\bigcup_{e \in U(S)} \left\{X_e^{S(e)}\right\}\right)$ subject to $|U(S)| \leq B$ for total size constraints, $|U_i(S)| \leq B_i$ for each $i \in [k]$ for individual size constraints. Ohsaka and Yoshida [2015] showed that f is monotone k-submodular.

Experiment settings: We use the publicly available Intel Lab dataset preprocessed by Ohsaka and Yoshida [2015]. This dataset contains approximately 2.3 million readings collected from 54 sensors deployed in the Intel Berkeley research lab between February 28th and April 5th, 2004. Temperature, humidity, and light values are extracted and discretized into bins of 2 degrees Celsius each, 5 points each, and 100 luxes each, respectively. Hence there are $k = 3$ kinds of sensors to be allocated to $n = 54$ locations. For total size constraints, we set the value of B to 3, 6, 9, $\ldots$, 54. For individual size constraints, we denote budgets for sensors measuring temperature, humidity, and light by B_1, B_2, and B_3 respectively. We set $B_1 = B_2 = B_3 = b$, where b is a parameter varying from 1 to 18.

Results: The results are shown in Figure 1. For total size constraints, Figure 1a shows that in terms of function values, all algorithms tested using different hyperparameter values had similar performances. In terms of the number of function evaluations (with lazy evaluations) as shown in Figure 1c, for the stochastic greedy algorithm, the number of function evaluations stays almost identical regardless of the δ parameter. One possible reason is that the δ parameter only appears inside the logarithm term of the size of the stochastic set, so varying it would not affect the size significantly. For our threshold greedy algorithm, however, increasing ε significantly reduces the number of function evaluations (since the number of thresholds considered is of order $\mathcal{O}(1/\varepsilon)$) without significant degradation in solution quality. We note there is a drop in the number of function evaluations by stochastic greedy algorithms as the budget B approaches the maximum value n (the number of locations). This phenomenon was reported in [Ohsaka and Yoshida, 2015] and may be due in part to the formula of the stochastic batch size (B appears in the denominator).

For individual size constraints, Figure 1b show that for function values, all the algorithms tested with varying hyperparameters have similar performances. For the number of function evaluations shown in Figure 1d, we can see there is a drop as δ get larger for the stochastic greedy algorithm, but the drop is not that significant compared with that of the threshold greedy algorithm when ε is varied. Again, this is due to the fact that the δ parameter only appears inside the logarithm term of the size of the stochastic set (so affects the runtime logarithmically), while the ε parameter affects the runtime of the threshold algorithm linearly. When both δ and ε are set to 0.8, it becomes apparent that the threshold algorithm outperforms the stochastic greedy algorithm in terms of the number of function evaluations. Specifically, at the largest margin, the threshold algorithm requires only one-third of the number of function evaluations compared to the stochastic greedy algorithm.

Overall, we observed that increasing the parameter δ for (baseline) stochastic greedy did not significantly impact the function values or the number of function evaluations. However, increasing ε in our proposed threshold algorithms



(a) Influence spreads. (b) Function evaluations.

Figure 2: Influence maximization over k topics under individual size constraints.

resulted in a significant reduction in the number of function evaluations with only a negligible decrease in the function values. For these experiments, our threshold algorithms enable significantly better tradeoffs in balancing accuracy and runtime compared to the baseline methods.

5.2 INFLUENCE MAXIMIZATION

Influence Maximization with k Topics: In this problem, a social network is presented as a directed graph $G = (V, E)$ where V is a set of nodes and E is the set of edges. Each edge $(u, v) \in E$ is associated with weights $\{p^i_{u,v}\}_{i\in[k]}$, where $p^i_{u,v}$ represents the strength of influence from user u to v on topic i. The objective is to maximize the number of users in the network who eventually get influenced by one of the topics. For information diffusion, we use the *k-topic independent cascade* (k-IC) model presented in Ohsaka and Yoshida [2015], which generalizes the independent cascade model [Kempe et al., 2003].

More specifically, the influence spread $\sigma : (k+1)^V \to \mathbb{R}_+$ in the k-IC model is defined as $\sigma(S) = \mathbb{E}\left[\left|\bigcup_{i\in[k]} A_i\left(U_i(S)\right)\right|\right]$, where $A_i\left(U_i(S)\right)$ is a random variable representing the set of influenced nodes in the diffusion process of the i-th topic. It is shown in [Ohsaka and Yoshida, 2015] that the influence spread function σ is monotone k-submodular. Given a directed graph $G = (V, E)$, edge probabilities $\{p^i_{u,v} \mid ((u, v) \in E, i \in [k])\}$, and a budget B for total size constraint (or B_i for $i \in [k]$ for individual size constraint), the problem is to select a seed set $S \in (k+1)^V$ that maximizes $\sigma(S)$ subject to $|U(S)| \leq B$ (or $|U_i(S)| \leq B_i$ for each $i \in [k]$).

Experiment settings: We use the preprocessed data of Digg network from [Ohsaka and Yoshida, 2015], where we have 3,523 users, 90,244 links, and $k = 10$ topics. Recall that for individual size constraints, the number of function evaluations for the stochastic greedy algorithm is $\tilde{\mathcal{O}}(k^2 n)$, and $\tilde{\mathcal{O}}(kn)$ for the threshold greedy algorithm. We set individual size $b = 2$ and compare both the function values and the number of function evaluations when k is varied. Similar to [Ohsaka and Yoshida, 2015], during the process of the algorithms, the influence spread was approximated by simulating the diffusion process 100 times. When the algorithms terminate, we simulated the diffusion process 10,000 times to obtain sufficiently accurate estimates of the spread.

Results: For this set of experiments, the results are shown in Figure 2. Regarding influence spread, we can observe from Figure 2a that all methods perform comparably, except for some variability in larger values of k due to the randomness of diffusions. However, when considering the number of function evaluations, it becomes apparent in Figure 2b that the stochastic greedy algorithm significantly underperforms compared to all other methods. For the threshold greedy algorithms, there is some (though not significant) performance advantage when the ε is set to 0.5.

Interestingly, the greedy algorithm outperforms the stochastic greedy algorithm in terms of the number of function evaluations by a large margin. This is due to the fact that the greedy algorithm allows for a more efficient implementation using lazy evaluations. This phenomenon is also reported in [Ene and Nguyen, 2022]. From this, we can infer that our threshold greedy algorithms also allow for the implementation of lazy evaluations in a similarly efficient manner.

6 CONCLUSION

In this work, we proposed algorithms for the problem of maximizing monotone k-submodular functions under size constraints. We showed that algorithms employing a threshold-greedy strategy improve the run-time among deterministic approximation algorithms with the best-known approximation ratios, and for independent size constraints could even improve on the run-time of a stochastic greedy algorithm. There are a number of important future directions, including investigating if similar strategies could improve run-time for more complicated constraints such as matroids, knapsacks, etc., application specific adaptations, and investigating what approximation ratios are achievable for these and related problems.

Acknowledgements

This material is based upon work supported by the National Science Foundation under Grants No. 2130536 and 2149617.

References

Ashwinkumar Badanidiyuru and Jan Vondrák. Fast algorithms for maximizing submodular functions. In *ACM-SIAM Symposium on Discrete Algorithms*, 2014.

Jingwen Chen, Zhongzheng Tang, and Chenhao Wang. Monotone k-submodular knapsack maximization: An analysis of the greedy+singleton algorithm. In Qiufen

Ni and Weili Wu, editors, *Algorithmic Aspects in Information and Management*, pages 144–155, Cham, 2022. Springer International Publishing. ISBN 978-3-031-16081-3.

David A Cohen, Martin C Cooper, Peter G Jeavons, and Andrei A Krokhin. The complexity of soft constraint satisfaction. *Artificial Intelligence*, 170(11):983–1016, 2006.

Alina Ene and Huy Nguyen. Streaming algorithm for monotone k-submodular maximization with cardinality constraints. In *International Conference on Machine Learning*, pages 5944–5967. PMLR, 2022.

Anna Huber and Vladimir Kolmogorov. Towards minimizing k-submodular functions. In *Combinatorial Optimization: Second International Symposium, ISCO 2012, Athens, Greece, April 19-21, 2012, Revised Selected Papers 2*, pages 451–462. Springer, 2012.

Satoru Iwata, Shin-ichi Tanigawa, and Yuichi Yoshida. Improved approximation algorithms for k-submodular function maximization. In *ACM-SIAM Symposium on Discrete Algorithms*, 2015.

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

Samir Khuller, Anna Moss, and Joseph Seffi Naor. The budgeted maximum coverage problem. *Information Processing Letters*, 70(1):39–45, 1999.

Vladimir Kolmogorov. Submodularity on a tree: Unifying $L^\natural$-convex and bisubmodular functions. In *Mathematical Foundations of Computer Science*, volume 11, pages 400–411. Springer, 2011.

Andreas Krause and Carlos Guestrin. Near-optimal observation selection using submodular functions. In *AAAI Conference on Artificial Intelligence*, 2007.

Wenxin Li, Moran Feldman, Ehsan Kazemi, and Amin Karbasi. Submodular maximization in clean linear time. In *Advances in Neural Information Processing Systems*, 2022.

Tatsuya Matsuoka and Naoto Ohsaka. Maximization of monotone k-submodular functions with bounded curvature and non-k-submodular functions. In *Asian Conference on Machine Learning*, pages 1707–1722. PMLR, 2021.

Michel Minoux. Accelerated greedy algorithms for maximizing submodular set functions. In J. Stoer, editor, *Optimization Techniques*, pages 234–243, Berlin, Heidelberg, 1978. Springer Berlin Heidelberg.

Baharan Mirzasoleiman, Ashwinkumar Badanidiyuru, Amin Karbasi, Jan Vondrák, and Andreas Krause. Lazier than lazy greedy. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29, 2015.

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

Naoto Ohsaka and Yuichi Yoshida. Monotone k-submodular function maximization with size constraints. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

Canh V Pham, Quang C Vu, Dung KT Ha, and Tai T Nguyen. Streaming algorithms for budgeted k-submodular maximization problem. In *Computational Data and Social Networks: 10th International Conference, Proceedings*, pages 27–38. Springer, 2021.

Chao Qian, Jing-Cheng Shi, Ke Tang, and Zhi-Hua Zhou. Constrained monotone k-submodular function maximization using multiobjective evolutionary algorithms with theoretical guarantee. *IEEE Transactions on Evolutionary Computation*, 22(4):595–608, 2017.

Shinsaku Sakaue. On maximizing a monotone k-submodular function subject to a matroid constraint. *Discrete Optimization*, 23:105–113, 2017.

Ajit Singh, Andrew Guillory, and Jeff Bilmes. On bisubmodular maximization. In *Artificial Intelligence and Statistics*, pages 1055–1063. PMLR, 2012.

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

Zhongzheng Tang, Chenhao Wang, and Hau Chan. On maximizing a monotone k-submodular function under a knapsack constraint. *Operations Research Letters*, 50: 28–31, 2022.

Justin Ward and Stanislav Živnỳ. Maximizing bisubmodular and k-submodular functions. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1468–1481. SIAM, 2014.

Justin Ward and Stanislav Živný. Maximizing k-submodular functions and beyond. *ACM Transactions on Algorithms*, 12:1 – 26, 2014.