Maximizing a k-Submodular Maximization Function under an Individual Knapsack Constraint

Tan D. Tran
Faculty of Information Technology, VNU University of Engineering and Technology
Hanoi, Vietnam
22027005@vnu.edu.vn

Canh V. Pham *
ORLab, Faculty of Computer Science, Phenikaa University
Hanoi, Vietnam
canh.phamvan@phenikaa-uni.edu.vn

Dung K.T. Ha
Faculty of Information Technology,VNU University of Engineering and Technology
Hanoi, Vietnam
20028008@vnu.edu.vn

ABSTRACT

In this work, we consider a novel problem of maximizing monotone k-submodular functions under the individual knapsack constraint (kSMIK) over the ground set V, which has been found numerous applications in machine learning, including data summarization and information propagation. We propose an approximation algorithm that has approximation ratio $\frac{1-\epsilon}{2(k+1)}$ and takes $O(nk\log(n)/\epsilon)$ query complexity, where ϵ is an input parameter. Alongside theoretical analysis, we conduct extensive experiments on our proposed algorithm via some applications, such as Influence Maximization and Sensor Placement. The experimental results demonstrate that our algorithm gives competitive solution quality with state-of-the-art techniques but significantly reduces the required queries.

CCS CONCEPTS

• **Information systems**;

KEYWORDS

Combinatorial Optimization, Approximation Algorithms, Streaming Algorithms, k-submodular maximization, individual knapsack constraint.

ACM Reference Format:
Tan D. Tran, Canh V. Pham, and Dung K.T. Ha. 2023. Maximizing a k-Submodular Maximization Function under an Individual Knapsack Constraint. In *The 12th International Symposium on Information and Communication Technology (SOICT 2023), December 07–08, 2023, Ho Chi Minh, Vietnam.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3628797.3628843

1 INTRODUCTION

Given a finite ground set V and an integer number k, we define $[k] = \{1, 2, \ldots, k\}$ and $(k+1)^V = \{(V_1, V_2, \ldots, V_k) | V_i \subseteq V, \forall i \in [k], V_i \cap V_j = \emptyset, \forall i \neq j\}$ be a family of k disjoint sets, called the k-**set**. A function $f : (k+1)^V \mapsto \mathbb{R}_+$ is k-**submodular** iff for any $\mathbf{a} = (A_1, A_2, \ldots, A_k)$ and $\mathbf{b} = (B_1, B_2, \ldots, B_k) \in (k+1)^V$, we have:

$$f(\mathbf{a}) + f(\mathbf{b}) \geq f(\mathbf{a} \sqcap \mathbf{b}) + f(\mathbf{a} \sqcup \mathbf{b}) \quad (1)$$

where

$$\mathbf{a} \sqcap \mathbf{b} = (A_1 \cap B_1, \ldots, A_k \cap B_k)$$

and

$$\mathbf{a} \sqcup \mathbf{b} = (C_1, \ldots, C_k), \text{ where } C_i = A_i \cup B_i \setminus (\cup_{j \neq i} A_j \cup B_j)$$

The problem of maximizing k-submodular functions has received significant attention recently due to its wide-range applications in various fields. For instance, it has been extensively studied in the context of influence maximization in social networks [15, 16, 21], information coverage maximization [20], sensor placement [16, 20, 21], and feature selection [24], etc.

Initially, researchers explored the problem without any constraints [9, 26]. Some others then focused on adding some types of constraints such as cardinality constraints [15, 16, 20, 21], knapsack constraints [19, 25], and matroid constraints [22, 23]. Moreover, recent research has extended the study beyond total constraints to consider individual constraints for each subset or source $i \in [k]$ [5, 17, 27].

The k-Submodular Maximization under the Individual Knapsack constraint (kSMIK) problem is subject to the constraint that each element $e \in V$ is associated with a positive cost $c(e)$ based on its source i out of k sources. The problem requires finding a k-set $\mathbf{s} = (S_1, S_2, \ldots, S_k)$ with a cost $c(S_i) = \sum_{e \in S_i} c(e) \leq B_i, \forall i \in [k]$, where B_i is the budget allocated to source i, such that the objective function $f(\mathbf{s})$ is maximized. The problem has broadly applied to social networks, sensor placement, information coverage, feature selection, and other related domains listed below:

k-topic influence maximization. In real-world scenarios like viral marketing and product recommendation, companies aim to maximize the dissemination of advertisement campaigns involving k distinct products through social networks or the Internet. Each product type incurs a specific cost. Influential users are employed to share their experiences or interests related to the products, leading to further information sharing among their followers and connections. This gradual propagation of product information influences a significant number of users. The mathematical foundation for this campaign lies in k-submodular maximization under a diffusion model. The model is first proposed by Kempe *et al.* [10] for a single influence. Subsequently, authors in [16] extended this model to accommodate $k \geq 2$ types of influence, attracting research interest in k-topic influence maximization [15, 20, 21].

*Corresponding author

SOICT 2023, December 07–08, 2023, Ho Chi Minh, Vietnam

ACM ISBN 979-8-4007-0891-6/23/12…$15.00
https://doi.org/10.1145/3628797.3628843

***k*-type sensor placement**. Sensor placement is a cover problem that originated from practical research to collect data from a large region, such as environmental quality, heat, climate, network traffic, etc. This data is crucial for forecasting problems. For instance, people set up some types of sensors to realize the attributions of energy, heat, and humidity are common in these applications. In this case, the scenario is given a k-type sensor system, a set V of potential locations in which each sensor type is limited with a specific cost B_i. Once the cost of a sensor type has been fully utilized, other sensors cannot be installed. The allocation of one sensor per location follows a strategy represented by a k-tuple of disjoint subsets of V. A plan like the above is frequently by entropy functions, which are k-submodular. The problem is a constrained k-submodular maximization problem with individual knapsack constraints.

Although numerous studies have focused on k-submodular maximization with total cost constraint, somehow, surprisingly, the problem of maximizing the k-submodular function under an individual knapsack constraint remains unexplored.

1.1 Our contribution

In this paper, we introduce our novel contributions to the field of the kSMIK problem, a streaming algorithm that achieves a constant approximation ratio for monotone kSMIK. Our key contributions are outlined as follows:

- **Proposed Algorithm**: We propose a Streaming Algorithm (Algorithm 2 in this work) for kSMIK problem, which achieves an approximation ratio of $\frac{1-\epsilon}{2(k+1)}$. The algorithm exhibits a query complexity of $O(kn\log(B)/\epsilon)$, where $\epsilon > 0$ represents the precision parameter. Notably, our algorithm is the first deterministic approach to tackle this research problem.
- **Experiments**: To validate our theoretical contributions, we conduct a comprehensive series of experiments in two practical applications: Influence Maximization and Sensor Placement, both relevant to the kSMIK problem. The results highlight the superior query-saving capabilities experimental of our algorithm compared to greedy approaches while maintaining comparable quality performance in terms of solution.

Overall, our contributions advance the understanding and solutions for the kSMIK problem, providing a novel streaming algorithm with approximately constant scaling. Through experimental evaluation, we demonstrate the practical benefits of our approach, showcasing its effectiveness in real-world scenarios.

1.2 Organization

The rest of the paper is organized as follows: Section II provides an overview of the existing literature and discussions. Section III presents the symbols and properties of k-submodular functions, the foundation for our algorithmic design. Section IV presents our proposed algorithm along with its theoretical analysis. Comprehensive experiments are detailed in Section V, highlighting the performance of our algorithm in practical applications. Finally, we conclude our work in Section VI.

2 RELATED WORK

***k*-submodular maximizations.** Initially, Singh *et al.* [24] focused on the specific case of bisubmodular maximization, which corresponds to k-submodular maximization with $k = 2$. However, researchers soon shifted their attention to the general case of k. It is worth noting that for $k = 1$, the problem reduces to submodular maximization, known as NP-hard. Therefore, it follows that k-submodular maximization is also NP-hard. Ward *et al.* [26] investigated the unconstrained maximization of k-submodular functions and proposed a Greedy deterministic algorithm with an approximation ratio of 1/3. In a subsequent study [9], an improved random Greedy approach was introduced, which achieved an approximation ratio of $\frac{k}{2k-1}$ by incorporating a probability distribution to select elements with larger marginal gains more frequently. Oshima *et al.* [18] later eliminated the randomness from the previous method but at the cost of an increased number of queries.

Constrained maximization of k-submodularity has also been explored. Oshaka *et al.* [16] focused on monotone k-submodular maximization with cardinal constraints. They proposed a Greedy algorithm that provided a 1/2-approximation ratio for the case of total size constraint and a 1/3-approximation ratio for singular size constraint. In a subsequent work [20], the authors presented a multi-objective evolutionary algorithm for monotone k-submodular maximization under the total size constraint. Zheng *et al.* [28] addressed the problem of maximizing estimated k-submodular functions subject to size constraints by introducing an approximated k-submodular function as a surrogate for the objective function.

Furthermore, researchers have investigated k-submodular maximization under more complex constraints. For instance, Sakaue *et al.* [23] demonstrated a Greedy algorithm with an approximation ratio of 1/2 for the problem under a matroid constraint. Another algorithm utilizing the distinct private continuous Greedy method was proposed in [22] and also achieved an approximation ratio of 1/2 for the same problem.

Streaming algorithms. Submodularity is a crucial property in machine learning with practical and theoretical implications. Nemhauser *et al.* [14] highlighted its significance and demonstrated the effectiveness of Greedy algorithms for submodular set functions. However, the direct application of Greedy algorithms becomes challenging in the era of big data, where data may not fit in memory and random access is not feasible.

To tackle this challenge, researchers have turned to streaming algorithms that operate on small portions of data stored in memory at a time. The concept of streaming algorithms was initially introduced by Alon *et al.* [1]. The streaming algorithms enable accessing the input data in a single pass, limiting the usage to a small, fixed amount of memory known as space complexity. This approach is particularly well-suited for scenarios where the data must be processed in just one or a few passes and when the data is presented as a continuous stream. Streaming algorithms offer the advantage of reducing memory requirements and enabling real-time analytics. They have proven to be valuable in solving submodular maximization problems under various constraints, including cardinality [2, 6, 11, 12], knapsack [8], k-set [7], and matroid [4] constraints. Nevertheless, applying streaming algorithms to k-submodular maximization problems is not straightforward due to the inherent differences between submodularity and k-submodularity.

For k-submodular objective functions, the algorithms presented in [9, 26] are designed as single-pass streaming algorithms. The

approach in [9] involves random selection, and subsequent works have improved upon it by introducing new element selection distributions based on different costs and establishing relationships between the current and optimal solutions.

Nguyen *et al.* [21] were the first to propose streaming algorithms for k-submodular maximization subject to the total size constraint with noises. They introduced two streaming algorithms that achieve approximation ratios of $O(\epsilon(1-\epsilon)^{-2}B)$ for monotone functions and $O(\epsilon(1-\epsilon)^{-3}B)$ for non-monotone functions. More recently, Pham *et al.* [19] developed two single-pass streaming algorithms for k-submodular maximization under the budget constraint, with a complexity of $O(nk\log(n)/\epsilon)$. Notably, these algorithms can achieve approximation ratios of $1/4-\epsilon$ and $k/(4k-1)-\epsilon$ (in expectation), where kSMIK represents a special case of the algorithm with $\beta = 1$.

In our view, no prior research has specifically addressed the problem of k-Submodular Maximization under an Individual Knapsack Constraint. This indicates a significant gap in the current literature and highlights the need for further exploration and development of new algorithms and methodologies. Future research endeavors should focus on investigating this problem, both theoretically and practically, and devising efficient approaches to obtain optimal or approximate solutions. Bridging this research gap will contribute to advancing the field of k-submodular optimization and addressing real-world problems that involve individual knapsack constraints.

3 PRELIMINARIES

Notations. We use the following notations throughout the paper: Given a finite set V and an integer $k > 0$, as mentioned above, the definitions of $[k]$ and $(k+1)^V$ have been referred to in Section I. We define $supp_i(\mathbf{s}) = S_i$, $supp(\mathbf{s}) = \cup_{i\in[k]}S_i$, S_i as i-**th set of s** and an empty k-set $\mathbf{0} = (\emptyset, \ldots, \emptyset)$.

For $\mathbf{a} = (A_1, A_2, \ldots, A_k), \mathbf{b} = (B_1, B_2, \ldots, B_k) \in (k+1)^V$, we set if $e \in A_i$ then $\mathbf{a}(e) = i$ and i is called the **position** of e, otherwise $\mathbf{a}(e) = 0$. Adding an element $e \notin supp(\mathbf{a})$ into A_i can be represented by $\mathbf{a} \sqcup (e, i)$. When $A_i = \{e\}$, and $A_j = \emptyset, \forall j \neq i$, $\mathbf{a}$ is denoted by (e, i). We denote by $\mathbf{a} \sqsubseteq \mathbf{b}$ iff $A_i \subseteq B_i\ \forall i \in [k]$.

The objective function. The function $f : (k+1)^V \mapsto \mathbb{R}_+$ is k-**submodular** iff for any $\mathbf{a} = (A_1, A_2, \ldots, A_k)$ and $\mathbf{b} = (B_1, B_2, \ldots, B_k) \in (k+1)^V$, we have:

$$f(\mathbf{a}) + f(\mathbf{b}) \geq f(\mathbf{a} \sqcap \mathbf{b}) + f(\mathbf{a} \sqcup \mathbf{b}) \tag{2}$$

where

$$\mathbf{a} \sqcap \mathbf{b} = (A_1 \cap B_1, \ldots, A_k \cap B_k)$$

and

$$\mathbf{a} \sqcup \mathbf{b} = (C_1, \ldots, C_k), \text{ where } C_i = A_i \cup B_i \setminus (\cup_{j\neq i} A_j \cup B_j)$$

In this work, we consider f is *monotone*, i.e., for any $\mathbf{a} \in (k+1)^V, e \notin supp(\mathbf{a})$ and $i \in [k]$, we have the *marginal gain* when adding a tuple (e, i) in to a set $\mathbf{a}$:

$$\begin{aligned}\Delta_{(e,i)}f(\mathbf{a}) =& f(A_1, \ldots, A_{i-1}, A_i \cup \{e\}, A_{i+1}, \ldots, A_k) \\ &- f(A_1, \ldots, A_k) \geq 0\end{aligned}$$

In theory, it is assumed to exist an *oracle query* meaning the query for the k-set $\mathbf{x}$ returns the value $f(\mathbf{x})$, and f is normalized, i.e., $f(\mathbf{0}) = 0$.

From [26], the k-submodularity of f implies the *orthant submodularity*, i.e.,

$$\Delta_{(e,i)}f(\mathbf{a}) \geq \Delta_{(e,i)}f(\mathbf{b}) \tag{3}$$

for any $\mathbf{a}, \mathbf{b} \in (k+1)^V, e \notin supp(\mathbf{a}), \mathbf{a} \sqsubseteq \mathbf{b}$ and $i \in [k]$; and the *pairwise monotonicity*, i.e, for any $i, j \in [k], i \neq j$:

$$\Delta_{(e,i)}f(\mathbf{a}) + \Delta_{(e,j)}f(\mathbf{a}) \geq 0 \tag{4}$$

In this problem, we assume that each element $e \in V$ is associated with a positive cost $c(e)$ and the total cost of a k-set $\mathbf{x}$ for position i defined as $c_i(\mathbf{x}) = \sum_{e\in supp_i(\mathbf{x})} c(e)$. We assume that each position $i \in [k]$ has a limited budget $B_i > 0$ and for any element e, $c(e) \leq B$ otherwise, we can remove it, where $B = \sum_{i=1}^{k} B_i$.

The k-Submodular Maximization under the Individual Knapsack constraint (kSMIK) problem is to find a k-set $\mathbf{s} = (S_1, S_2, ..., S_k)$ satisfying $c_i(\mathbf{s}) \leq B_i, \forall i \in [k]$ such that the objective function $f(\mathbf{s})$ is maximized.
We denote opt as an optimal solution for the problem.

4 PROPOSED ALGORITHMS

This section presents a streaming algorithm for the kSMIK problem. We first introduce the simple version of our algorithm named Streaming Algorithm with knowing opt (SAO), which assumes the knowledge of the optimal solution. Building upon SAO, we propose Streaming Algorithm (SA) with an approximation ratio of $\frac{1-\epsilon}{2(k+1)}$ within $O(nk\log(n))$ query complexity.

4.1 Streaming Algorithm with knowing opt (SAO)

Our algorithm follows a two-loop approach. In the first loop, we iterate through each incoming element and identify the "best" position within the k subsets based on the highest value of $f((e, i_e))$. If the condition $\Delta(e, i_e)f(s) \leq c(e)\alpha v/B$ holds and the budget constraint $c(e) + c_{i_e}(\mathbf{s}) \leq B_{i_e}$ is satisfied, we add the pair (e, i_e) to the candidate solution $\mathbf{s}$, where α is a real number parameter. Once the first loop completes, we proceed to the second loop. Here, for every element e that has not been included in set $supp(\mathbf{s})$, we search for an index i_e that maximizes the performance metric $\Delta_{(e,i_e)}f(\mathbf{s})$ while ensuring the condition $c_{i_e}(\mathbf{s}) + c(e) \leq B_{i_e}$ is satisfied. Ultimately, the algorithm returns the final solution $\mathbf{s}'$, which is the better one between (e_{max}, i_{max}) and $\mathbf{s}$. For a comprehensive overview of the algorithm's steps, refer to Algorithm 1.

In the following, we analyze the theoretical guarantee of the Algorithm 1. We first define the notations as follows:

- $\mathbf{o}$ is an optimal solution of the problem over V and the optimal value opt $= f(\mathbf{o})$.
- $\mathbf{s}^t = \{(e_1, i_1), \ldots, (e_t, i_t)\}$ the k-set $\mathbf{s}$ after ending the main loop, $t = |supp(\mathbf{s}^t)|$.
- $\mathbf{o}^j = (\mathbf{o} \sqcup \mathbf{s}^j) \sqcup \mathbf{s}^j$.
- v is a guessing value of opt, thus: $(1-\epsilon)\text{opt} \leq v \leq \text{opt}$

The following Lemmata are crucial for analyzing the theoretical bounds of the obtained solution.

Lemma 1. *$f(\boldsymbol{o}) - f(\boldsymbol{o}^j) \leq f(\boldsymbol{s}^j)$ for all $0 \leq j \leq t$.*

Algorithm 1: Streaming Algorithm with knowing opt

Input: $V, f, k, B_1, B_2, \ldots, B_k$, an approximate value v of opt such that $(1-\epsilon)\text{opt} \le v \le \text{opt}$, parameter α.
Output: A solution $\mathbf{s}$.

$\mathbf{s} \leftarrow \emptyset$.
$(e_{max}, i_{max}) \leftarrow \arg\max_{e \in V, i \in [k] \,:\, c(e) \le B_i} f((e,i))$
foreach $e \in V$ **do**
  $i_e \leftarrow \arg\max_{i \in [k]} f((e,i))$
  if $c(e) + c_{i_e}(\mathbf{s}) \le B_{i_e}$ **then**
    if $\Delta_{(e,i_e)} f(\mathbf{s}) \ge \frac{c(e)\alpha v}{B_i}$ **then**
      $\mathbf{s} \leftarrow \mathbf{s} \sqcup (e, i_e)$
      $c_{i_e}(\mathbf{s}) += c(e)$
    end
  end
end
foreach $e \in V \setminus supp(\mathbf{s})$ **do**
  $i_e \leftarrow \arg\max_{i \in [k], c_i(s)+c(e) \le B_i} \Delta_{(e,i)} f(\mathbf{s})$
  $\mathbf{s} \leftarrow \mathbf{s} \sqcup (e, i_e)$
end
$\mathbf{s} \leftarrow \arg\max_{\mathbf{s}' \in \{(e_{max}, i_{max}), \mathbf{s}\}} f(\mathbf{s}')$
return s

Proof. We have: $\mathbf{o}^0 = (\mathbf{o} \sqcup \mathbf{s}^0) \sqcup \mathbf{s}^0$. Hence, $f(\mathbf{o}) = f(\mathbf{o}^0)$. For all $0 \le j \le t$, we have:

$$f(\mathbf{o}) - f(\mathbf{o}^j) = \sum_{i=1}^{j} (f(\mathbf{o}^{i-1}) - f(\mathbf{o}^i)) \tag{5}$$
$$\le \sum_{i=1}^{j} (f(\mathbf{o}^{i-1}) - f(\mathbf{o}^{i-1/2})) \tag{6}$$
$$\le \sum_{i=1}^{j} (f(\mathbf{s}^{i-1/2}) - f(\mathbf{s}^{i-1})) \tag{7}$$
$$\le \sum_{i=1}^{j} (f(\mathbf{s}^{i}) - f(\mathbf{s}^{i-1})) \tag{8}$$
$$\le f(\mathbf{s}^j) \tag{9}$$

where the inequality (6) is due to the monotoncity of f, the inequality (7) is due to the k-submodularity of f and the inequality (8) is due to the selection of the algorithm. The proof is completed. □

Lemma 2. *After ending the first loop, if for any $e \in supp(\mathbf{o}) \setminus supp(\mathbf{s}^t)$ and $i \in [k]$ satisfying $c_i(\mathbf{s}) + c(e) > B_i$, then $\Delta_{(e,i)} f(\mathbf{s}^t) > c(e)\alpha v$. We have: $f(\mathbf{s}) \ge v(1-\alpha k)/2$*

Proof. We first have:

$$f(\mathbf{o}^t) - f(\mathbf{s}^t) \le \sum_{e \in supp(\mathbf{o}) \setminus supp(\mathbf{s})} \Delta_{(e,\mathbf{o}(e))} f(\mathbf{s}^t) \tag{10}$$
$$= \sum_{i=1}^{k} \sum_{e \in supp(\mathbf{o}) \setminus supp(\mathbf{s}), \mathbf{o}(e)=i} \Delta_{(e,\mathbf{o}(e))} f(\mathbf{s}^t) \tag{11}$$
$$\le \sum_{i=1}^{k} (c_i(\mathbf{o})\alpha v)/B_i) \tag{12}$$
$$\le \sum_{i=1}^{k} ((B_i \alpha v)/B_i) \tag{13}$$
$$\le k\alpha v \tag{14}$$

where the inequality (10) is due to the k-submodularity of f, the inequality (12) is due to the given condition in the Lemma. In the other hand, By applying Lemma 1, we have

$$v - f(\mathbf{s}^t) \le f(\mathbf{o}) - f(\mathbf{s}^t) \tag{15}$$
$$= f(\mathbf{o}) - f(\mathbf{o}^t) + f(\mathbf{o}^t) - f(\mathbf{s}^t) \tag{16}$$
$$\le f(\mathbf{s}^t) + k\alpha v \tag{17}$$

which implies that $f(\mathbf{s}^t) \ge v(1-\alpha k)/2$. The proof is completed. □

Lemma 3. *If there exists an element $e \in supp(\mathbf{o}) \setminus supp(\mathbf{s}^t)$ and an integer $i \in [k]$ so that $c_i(\mathbf{s}^t) + c(e) > B_i$ and $\Delta_{(e,i)} f(\mathbf{s}^t) \ge c(e)\alpha v/B_i$, then $f(\mathbf{s}) \ge \alpha v/2$.*

Proof. $\Delta_{(e,i)} f(\mathbf{s}^t) \ge c(e)\alpha v/B_i$ implies:

$$f(\mathbf{s} \sqcup (e,i)) - f(\mathbf{s}^t) \ge c(e)\alpha v/B_i \tag{18}$$
$$\Longrightarrow f(\mathbf{s}^t \sqcup (e,i)) \ge f(\mathbf{s}^t) + \frac{c(e)\alpha v}{B_i} \tag{19}$$
$$\ge \frac{c_i(\mathbf{s}^t)}{B_i} + \frac{c(e)\alpha v}{B_i} \tag{20}$$
$$\ge \frac{(c_i(\mathbf{s}^t) + c(e))\alpha v}{B_i} \tag{21}$$
$$\ge \alpha v \tag{22}$$

where inequality (20) is due to the selection of the algorithm. Recall $(e_m, i_m) = \arg\max_{e \in V, i \in [k]} f((e,i))$, we have:

$$f(\mathbf{s}) \ge \max\{f(\mathbf{s}^t), f((e_m, i_m))\} \ge \frac{f(\mathbf{s}^t) + f((e_m, i_m))}{2} \tag{23}$$
$$\ge \frac{f(\mathbf{s}^t) + f((e,i))}{2} \ge \frac{f(\mathbf{s}^t \sqcup (e,i))}{2} \tag{24}$$
$$\ge \frac{\alpha v}{2} \tag{25}$$

where inequality (25) is due to (22). The proof is completed. □

4.2 Streaming Algorithm (SA)

The primary goal of the SAO algorithm is to establish a theoretical framework for identifying the range of the optimal solution. Building upon this range, the SA algorithm aims to approximate a solution for the kSMIK problem. The defined range is $M \le (1+\epsilon)^j \le MB$, with M representing the optimal value of $f(e_{max}, i_{max})$ and $f(e, i_e)$. Within each specific range from M to

MB, the algorithm determines whether to include the element e in the solution, mirroring the process in the Algorithm 2.

Algorithm 2: Streaming Algorithm

Input: $V, f, k, \alpha, B_1, B_2, \ldots, B_k \mid B_i \geq 1(\forall i \in [k])$.
Output: A solution $\mathbf{s}$

$M \leftarrow 0$, $(e_{max}, i_{max}) \leftarrow \emptyset$
foreach $e \in V$ **do**
  $i_e \leftarrow \arg\max_{i \in [k]} f((e,i))$
  $(e_{max}, i_{max}) \leftarrow \arg\max_{\mathbf{x} \in \{(e,i_e),(e_{max},i_{max})\}} f(\mathbf{x})$
  $M \leftarrow f(\mathbf{x})$
  $O \leftarrow \{v = (1+\epsilon)^j : M \leq (1+\epsilon)^j \leq BM\}$
  foreach $v \in O$ **do**
    $i_v \leftarrow \arg\max_{i \in [k], c_i(\mathbf{s}_v)+c(e) \leq B_i} \Delta_{(e,i)} f(\mathbf{s}_v)$
    if $\Delta_{(e,i_v)} f(\mathbf{s}_v) \geq c(e)\alpha v / B_i$ **then**
      $\mathbf{s}_v \leftarrow \mathbf{s}_v \sqcup (e, i_v)$
      $c_{i_v}(s_v) += c(e)$
    end
  end
end
foreach $v \in O$ **do**
  foreach $e \in V \setminus supp(\mathbf{s}_v)$ **do**
    $i_e \leftarrow \arg\max_{i \in [k], c_i(s_v)+c(e) \leq B_i} \Delta_{(e,i)} f(\mathbf{s}_v)$
    $\mathbf{s}_v \leftarrow \mathbf{s}_v \sqcup (e, i_e)$
    $c_{i_e}(s_v) += c(e)$
  end
end
$\mathbf{s} \leftarrow \arg\max_{\mathbf{x} \in \{(e_{max},i_{max}), \mathbf{s}_v : v \in O\}} f(\mathbf{x})$
return $\mathbf{s}$

Theorem 1. *For f is monotone and $\alpha = 1/(k+1)$, Algorithm 2 returns an approximation ratio of $\frac{1-\epsilon}{2(k+1)}$ and query complexity is $O(nklogB/\epsilon)$.*

Proof. The Algorithm 2 consists of two main loops. The first loop (outer loop) scans once over the ground set V. For each element e, it takes $|O|k$ queries to f. Therefore, the number of queries is at most

$$nk|O| = nk \log_{1+\epsilon} B \leq \frac{nk}{\epsilon} \log(B) \tag{26}$$

We now show the approximation ratio of the algorithm. Since $M\text{opt} \leq BM$, there exists $v = (1+\epsilon)^j$ satisfying

$$(1-\epsilon)\text{opt} \leq \frac{\text{opt}}{1+\epsilon} \leq v \leq \text{opt} \tag{27}$$

Therefore, we apply Lemmata 2, 3 in two following cases:
Case 1. After ending the first loop, if for any $e \in supp(\mathbf{o}) \setminus supp(\mathbf{s})$ and $i \in [k]$ so that $c_i(\mathbf{s}) + c(e) > B_i$, then $\Delta_{(e,i)} f(\mathbf{s}) < c(e)\alpha v$. By Lemma 2 and $\alpha = \frac{1}{k+1}$, we have:

$$f(\mathbf{s}) \geq f(\mathbf{s}^t) \geq \frac{v}{2(k+1)} = \frac{\text{opt}(1-\epsilon)}{2(k+1)} \tag{28}$$

Case 2. If there exists an element $e \in supp(\mathbf{o}) \setminus supp(\mathbf{s})$ and an integer $i \in [k]$ so that $c_i(\mathbf{s}) + c(e) > B_i$ and $\Delta_{(e,i)} f(\mathbf{s}) \geq c(e)\alpha v$. By Lemma 3 and $\alpha = \frac{1}{k+1}$, we also get $f(\mathbf{s}) \geq f(\mathbf{s}^t) \geq \frac{v}{2(k+1)} \geq \frac{\text{opt}(1-\epsilon)}{2(k+1)}$. The proof can be obtained by combining the two cases mentioned above. □

Algorithm 3: Greedy Algorithm

Input: $V, f, k, B_1, B_2, ..., B_k \mid B_i \geq 1(\forall i \in [k])$.
Output: A solution $\mathbf{s}$

$\mathbf{s} \leftarrow \emptyset$;
while $V \neq \emptyset$ **do**
  $(e_m, i_m) \leftarrow \arg\max_{e \in V, i \in [k] : (c_i(\mathbf{s})+c(e) \leq B_i)} \frac{\Delta_{(e,i)} f(\mathbf{s})}{c_e}$;
  if $(e_m, i_m) = \emptyset$ **then**
    break;
  end
  $\mathbf{s} \leftarrow \mathbf{s} \sqcup (e_m, i_m)$;
  $V \leftarrow V \setminus e_m$;
end

5 EXPERIMENTS

In this section, we present a comparative analysis of our algorithm and Algorithm 3 (Greedy) for the monotone kSMIK problem. As mentioned previously, no prior work has specifically addressed the kSMIK problem, so we developed a simple greedy algorithm for comparison. The greedy algorithm iteratively selects the element with the highest marginal gain to add to the solution at each step. We evaluate their performance on two specific applications: k-topic Influence Maximization under Individual Knapsack constraint (kIMIK) and k-type Sensor Placement under Individual Knapsack constraint (kSPIK). Our evaluation focuses on two key metrics: the oracle value of the objective function and the number of queries. We utilize the dataset mentioned in [15] to provide a comprehensive illustration of the performance of the compared algorithms. In all experiments, ϵ is set to 0.1. Additionally, without loss of generality, we have assumed that the budgets $B_1, B_2, \ldots, B_k$ are set to be the same.

Table 1: The dataset

Database	#Nodes	#Edges	Types
Facebook [13]	4039	88234	directed
Intel Lab sensors[3]	56	-	-

5.1 Experiment results

The objective values according to each budget were calculated, and the resulting data is presented in Figure 1 and Figure 2, effectively illustrating the experimental findings. Figure 1 illustrates the Algorithm results for kIMIK on the Facebook dataset, while Figure 2 showcases the Algorithm results for kSPIK on the Intel Lab Dataset. These results demonstrate that the SA algorithm outperforms the GREEDY algorithm in achieving higher objective values, indicating its superior performance in optimizing the given objective function for the kSMIK problem on the Facebook dataset (Figure 1(a)).

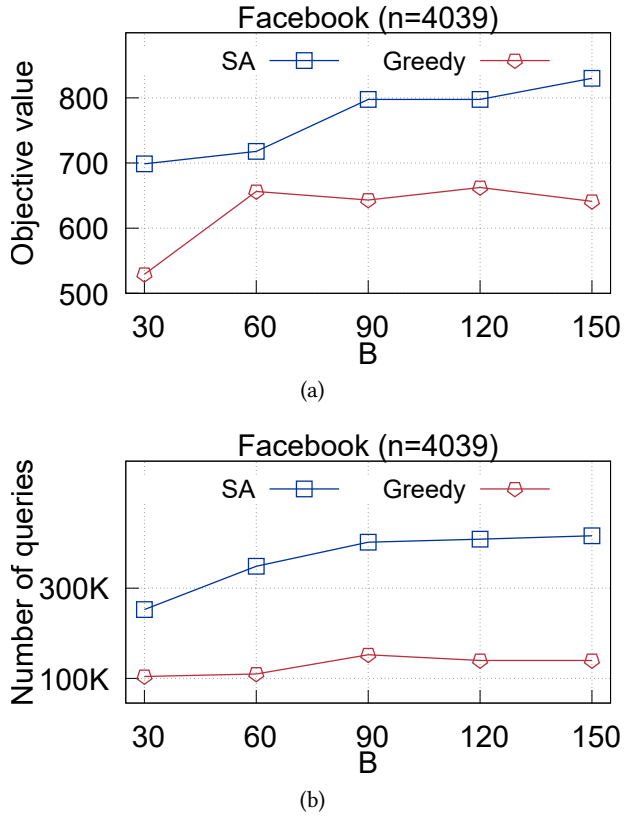

Figure 1: Algorithm results for kIMIK on Facebook: (a) The objective values, (b) The number of queries.

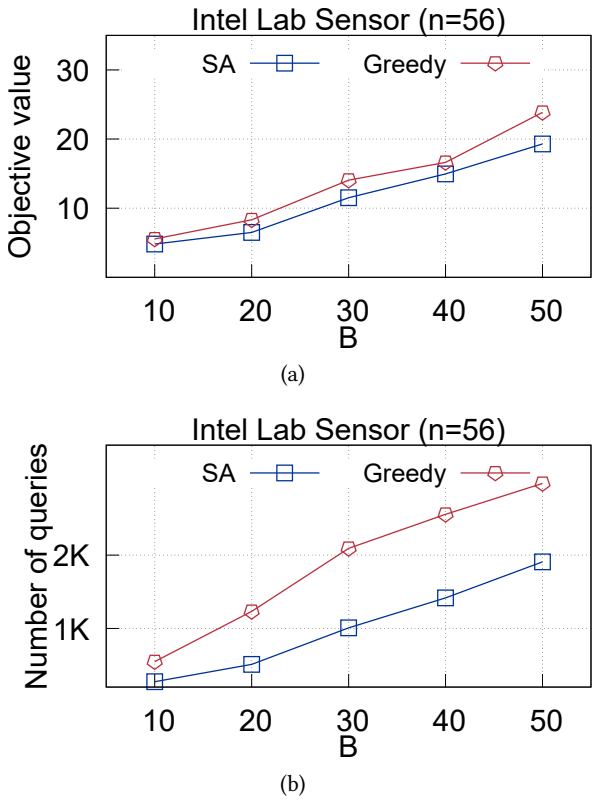

Figure 2: Algorithm results for kSPIK on Intel Lab: (a) The objective values, (b) The number of queries.

Additionally, for the GREEDY algorithm, the objective function values increase rapidly when the value of B is small. Still, as the value of B increases, the objective function values increase only slightly and tend to plateau. In contrast, with our SA algorithm, the objective function values also increase correspondingly as the value of B increases. However, it is important to note that our proposed algorithm requires nearly thrice as many queries as the GREEDY algorithm (Figure 1(b)). On the other hand, when considering the kSPIK problem on the Intel Lab dataset (Figure 2(b)), our proposed algorithm exhibits almost twice the speed compared to Greedy. It is worth mentioning that the objective functions of SA and Greedy are very similar (Figure 2(b)). For this case, the objective values of the SA and GREEDY algorithms have the same increasing trend and correspond to the value of B.

Overall, based on the experiment, our proposed algorithms are described as superior to the Greedy algorithm.

6 CONCLUSION

In conclusion, this paper has presented a streaming algorithm for maximizing a k-submodular function under an individual knapsack constraint in the monotone case. The algorithm achieves an approximation ratio of approximately $(1-\epsilon)/(2(k+1))$ with the query complexity of $O(kn \log B/\epsilon)$. The main approach employed is to restrict the range of optimal solutions.

To assess the practical performance of our algorithms, experiments were conducted on two applications: Influence Maximization and Sensor Placement. The experimental results demonstrate that our algorithms not only meet the quality requirements of solutions but also significantly reduce the number of queries compared to the greedy algorithm. Nevertheless, there remain open questions that serve as motivation for future research. These include improving the approximation ratio of approximate algorithms and addressing the linear query complexity for the non-monotone kSMIK problem.

In summary, the proposed algorithms offer efficient and effective solutions for maximizing k-submodular functions under an individual knapsack constraint. Future efforts will be directed toward refining these algorithms and tackling the remaining research challenges.

REFERENCES

[1] Noga Alon, Yossi Matias, and Mario Szegedy. 1996. The space complexity of approximating the frequency moments. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*. 20–29.

[2] Ashwinkumar Badanidiyuru, Baharan Mirzasoleiman, Amin Karbasi, and Andreas Krause. 2014. Streaming submodular maximization: massive data summarization on the fly. In *In Proc. of International Conference on Knowledge Discovery and Data Mining (KDD)*. 671–680.

[3] P. Bodik, W. Hong, C. Guestrin, S. Madden, M. Paskin, and R Thibaux. 2004. Intel Lab. (2004). http://db.csail.mit.edu/labdata/labdata.html

[4] Amit Chakrabarti and Sagar Kale. 2015. Submodular maximization meets streaming: matchings, matroids, and more. *Math. Program.* 154, 1-2 (2015), 225–247.

[5] Alina Ene and Huy Nguyen. 2022. Streaming Algorithm for Monotone k-Submodular Maximization with Cardinality Constraints. In *International Conference on Machine Learning*. PMLR, 5944–5967.

[6] Ryan Gomes and Andreas Krause. 2010. Budgeted Nonparametric Learning from Data Streams. In *In Proc. of the International Conference on Machine Learning (ICML)*, Johannes Fürnkranz and Thorsten Joachims (Eds.). 391–398.

[7] Ran Haba, Ehsan Kazemi, Moran Feldman, and Amin Karbasi. 2020. Streaming Submodular Maximization under a k-Set System Constraint. In *In Proc. of the International Conference on Machine Learning (ICML)*. 3939–3949.

[8] Chien-Chung Huang, Naonori Kakimura, and Yuichi Yoshida. 2020. Streaming Algorithms for Maximizing Monotone Submodular Functions Under a Knapsack Constraint. *Algorithmica* 82, 4 (2020), 1006–1032.

[9] Satoru Iwata, Shin-ichi Tanigawa, and Yuichi Yoshida. 2016. Improved Approximation Algorithms for k-Submodular Function Maximization. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016*, Robert Krauthgamer (Ed.). SIAM, 404–413.

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

[11] Alan Kuhnle. 2021. Quick Streaming Algorithms for Maximization of Monotone Submodular Functions in Linear Time. In *In Proc. of the International Conference on Artificial Intelligence and Statistics (AISTATS)*. 1360–1368.

[12] Ravi Kumar, Benjamin Moseley, Sergei Vassilvitskii, and Andrea Vattani. 2013. Fast greedy algorithms in mapreduce and streaming. In *In Proc. of Symposium on Parallelism in Algorithms and Architectures, (SPAA)*. 1–10.

[13] J. Leskovec and Krevl. 2014. A. SNAP Datasets: Stanford large network dataset collection. (2014). http://snap.stanford.edu/data

[14] George L. Nemhauser, Laurence A. Wolsey, and Marshall L. Fisher. 1978. An analysis of approximations for maximizing submodular set functions - I. *Math. Program.* 14, 1 (1978), 265–294.

[15] Lan Nguyen and My Thai. 2020. Streaming k-Submodular Maximization under Noise subject to Size Constraint. In *In Proc. of the International Conference on Machine Learning (ICML)*. 7338–7347.

[16] Naoto Ohsaka and Yuichi Yoshida. 2015. Monotone k-Submodular Function Maximization with Size Constraints. In *In Proc. of Annual Conference on Neural Information Processing Systems (NIPS)*. 694–702.

[17] Naoto Ohsaka and Yuichi Yoshida. 2015. Monotone k-submodular function maximization with size constraints. *Advances in Neural Information Processing Systems* 28 (2015).

[18] Hiroki Oshima. 2017. Derandomization for k-Submodular Maximization. In *In Proc. of International Workshop Combinatorial Algorithms (IWOCA)*, Ljiljana Brankovic, Joe Ryan, and William F. Smyth (Eds.). 88–99.

[19] Canh V. Pham, Quang C. Vu, Dung K. Ha, Tai T. Nguyen, and Nguyen D. Le. 2022. Maximizing k-submodular functions under budget constraint: applications and streaming algorithms. *J. Comb. Optim.* 44, 1 (2022), 723–751.

[20] Chao Qian, Jing-Cheng Shi, Ke Tang, and Zhi-Hua Zhou. 2018. Constrained Monotone k-Submodular Function Maximization Using Multiobjective Evolutionary Algorithms With Theoretical Guarantee. *IEEE Trans. Evol. Comput.* 22, 4 (2018), 595–608.

[21] Akbar Rafiey and Yuichi Yoshida. 2020. Fast and Private Submodular and k-Submodular Functions Maximization with Matroid Constraints. In *In Proc. of the International Conference on Machine Learning (ICML)*. 7887–7897.

[22] Akbar Rafiey and Yuichi Yoshida. 2020. Fast and Private Submodular and k-Submodular Functions Maximization with Matroid Constraints. In *In Proc. of International Conference on Machine Learning (ICML)*. 7887–7897.

[23] Shinsaku Sakaue. 2017. On maximizing a monotone k-submodular function subject to a matroid constraint. *Discret. Optim.* 23 (2017), 105–113.

[24] Ajit P. Singh, Andrew Guillory, and Jeff A. Bilmes. 2012. On Bisubmodular Maximization. In *In Proc. of the International Conference on Artificial Intelligence and Statistics (AISTATS)*. 1055–1063.

[25] Zhongzheng Tang, Chenhao Wang, and Hau Chan. 2022. On maximizing a monotone k-submodular function under a knapsack constraint. *Operations Research Letters* 50, 1 (2022), 28–31.

[26] Justin Ward and Stanislav Zivný. 2014. Maximizing Bisubmodular and k-Submodular Functions. In *In Proc. of Symposium on Discrete Algorithms (SODA)*. 1468–1481.

[27] Hao Xiao, Qian Liu, Yang Zhou, and Min Li. 2023. Non-monotone k-Submodular Function Maximization with Individual Size Constraints. In *Computational Data and Social Networks: 11th International Conference, CSoNet 2022, Virtual Event, December 5–7, 2022, Proceedings*. Springer, 268–279.

[28] Leqian Zheng, Hau Chan, Grigorios Loukides, and Minming Li. 2021. Maximizing Approximately k-Submodular Functions. In *In Proc. of the 2021 SIAM International Conference on Data Mining (SDM)*. 414–422.