Asia-Pacific Journal of Operational Research
(2025) 2550035 (27 pages)

DOI: 10.1142/S0217595925500356

k-Submodular Maximization Under Individual Knapsack Constraints: Applications and Streaming Algorithm

Tan D. Tran
ORlab, Faculty of Computer Science, PHENIKAA University, Yen Nghia Ward Ha Dong District, Hanoi 12116, Vietnam
tan.trandinh@phenikaa-uni.edu.vn

Canh V. Pham*
ORlab, Faculty of Computer Science, PHENIKAA University, Yen Nghia Ward Ha Dong District, Hanoi 12116, Vietnam
canh.phamvan@phenikaa-uni.edu.vn

Dung T. K. Ha
Faculty of Information Technology, University of Engineering and Technology Vietnam National University, Hanoi, 144 Xuan Thuy Street, Cau Giay District Hanoi, Vietnam, Hanoi 10000, Vietnam
20028008@vnu.edu.vn

Received 27 November 2024
Revised 20 May 2025
Accepted 17 June 2025
Published

The k-submodular optimization problems have been an active research area with many applications in machine learning, such as data summarization, influence maximization, and maximum revenue. In this work, we investigate the problem of k-submodular maximization under individual knapsack constraints (kSMIK) that capture various realities of such applications. We solve the problem in a streaming setting that requires two passes over the data and a reasonable amount of memory. In particular, we propose a streaming algorithm named Str which runs in $O(nk\log(B)/\epsilon)$ query complexity, takes $O(B\log(n)/\epsilon)$ space complexity, and achieves an approximation ratio of $\frac{1-\epsilon}{2(k+1)}$ for monotone objective function; $\frac{1-\epsilon}{2k+3}$ for the non-monotone objective function, where n is the size of the ground set, B is the total budget and $\epsilon \in (0,1)$ is an input parameter. We validate our algorithm in two applications of the studied problem, including revenue maximization and sensor placement on some benchmark datasets. The experimental

*Corresponding author.

results show that our streaming algorithm provides higher-quality solutions with lower query complexity than the existing baseline algorithm.

Keywords: Combinatorial optimization; approximation algorithms; streaming algorithms; k-submodular maximization; individual knapsack constraint.

1. Introduction

Maximizing a k-submodular function is attractive these days due to its wide-range applications such as influence maximization in social networks (Ohsaka and Yoshida, 2015; Rafiey and Yoshida, 2020; Nguyen and Thai, 2020), information coverage maximization (Qian *et al.*, 2018b), sensor placement (Ohsaka and Yoshida, 2015; Rafiey and Yoshida, 2020; Qian *et al.*, 2018b) and feature selection (Singh *et al.*, 2012).

For problem setting, given a finite ground set V and an integer number k, predefined $[k] = \{1, 2, \ldots, k\}$, and $(k+1)^V = \{(V_1, V_2, \ldots, V_k) \,|\, V_i \subseteq V, i \in [k], V_i \cap V_j = \emptyset, i \neq j\}$ as a family of k disjoint sets, called the *k-set*, a function $f : (k+1)^V \mapsto \mathbb{R}_+$ is *k-submodular* if and only if for any $\mathbf{x} = (X_1, X_2, \ldots, X_k)$ and $\mathbf{y} = (Y_1, Y_2, \ldots, Y_k) \in (k+1)^V$, we have

$$f(\mathbf{x}) + f(\mathbf{y}) \geq f(\mathbf{x} \sqcap \mathbf{y}) + f(\mathbf{x} \sqcup \mathbf{y}), \tag{1}$$

where

$$\mathbf{x} \sqcap \mathbf{y} = (X_1 \cap Y_1, \ldots, X_k \cap Y_k)$$

and

$$\mathbf{x} \sqcup \mathbf{y} = (Z_1, \ldots, Z_k), \quad \text{where } Z_i = X_i \cup Y_i \setminus \bigcup_{j \neq i} (X_j \cup Y_j).$$

People research k-submodular with various cases. First, it was considered without any constraint (Ward and Zivný, 2014; Iwata *et al.*, 2016). Next, others focused on adding several constrained problems, such as cardinality (Rafiey and Yoshida, 2020; Ohsaka and Yoshida, 2015; Qian *et al.*, 2018b; Nguyen and Thai, 2020), knapsack (Pham *et al.*, 2022b; Tang *et al.*, 2022; Pham *et al.*, 2022a; Ha *et al.*, 2024; Pham *et al.*, 2021), and matroid (Sakaue, 2017; Rafiey and Yoshida, 2020). Moreover, recent research has extended the study of cardinality from the total constraints to individual ones for each subset or source $i \in [k]$ (Xiao *et al.*, 2023; Ene and Nguyen, 2022; Ohsaka and Yoshida, 2015), etc.

In these constraints, the knapsack shows its general and essential role when capturing some others, such as cardinalities, and is available for some real limitations about time, human resources, budget, etc. In this work, we investigate a novel k-Submodular Maximization under the Individual Knapsack constraints (kSMIK) problem defined as follows: Assume that each element $e \in V$ is associated with a positive cost $c(e)$ pointed by every source i out of k sources. Given budgets $B_1, B_2, \ldots, B_k$, the problem requires finding a k-set $\mathbf{s} = (S_1, S_2, \ldots, S_k)$ with a cost $c(S_i) = \sum_{e \in S_i} c(e) \leq B_i, i \in [k]$, where B_i is the budget allocated to the source i, such that the objective function $f(\mathbf{s})$ is maximized. The high applicability of this problem can be shown via some instances listed as follows:

***k*-topic influence maximization.** Some real applications, such as viral marketing (Pham *et al.*, 2019), recommendation systems, information propagation and virus blocking, can be formed as k-topic influence maximization. For instance, a company plans to advertise a campaign with k distinct products over a social network. They allocate some budget sources for each type of product. They can hire famous influencers on social networks to introduce and persuade their followers to select those products. Followers then subsequently introduce them to their friends and so on. The company aims to convince the largest number of users to choose these products for this campaign. It can be modeled as a problem of k-submodular maximization under a diffusion model. Initially introduced by Kempe *et al.* (2003) for single influence scenarios, the model was later extended by Ohsaka and Yoshida (2015) to support $k \geq 2$ types of influence, thereby generalizing to k-topic influence maximization (Rafiey and Yoshida, 2020; Qian *et al.*, 2018b; Nguyen and Thai, 2020). Hence, this is an example of kSMIK.

k-topic influence maximization has been extended to *k-topic coverage maximization* recently. It's argued that a user might not be directly persuaded by an activated neighbor but can be informed about these topics. It leads to the research about k-topic coverage maximization, which maximizes the number of users who are either influenced or informed about at least one of the k topics (Qian *et al.*, 2018b). Certainly, when we consider the constraint of k budget categories, it becomes another application of kSMIK.

***k*-type product revenue maximization.** The k-type product revenue maximization problem allocates optimally k distinct product types to a group of customers to maximize a predefined objective function. The goal is to determine the most effective distribution strategy that leverages customer preferences and interactions to achieve the highest possible value of the objective function. This allocation process must account for customer demand, product compatibility, and the overall impact on revenue and other business metrics. By efficiently distributing the k product types, the objective is to enhance overall performance and achieve strategic business goals. Addressing this problem under constraints related to k budget categories exemplifies an application of kSMIK.

***k*-type sensor placement.** The task of sensor placement involves deploying various types of sensors to monitor large areas, such as environmental quality, temperature, climate conditions, and network traffic. These sensors gather data on factors like energy levels, heat, and humidity, which are essential for forecasting. Consider a system with k different types of sensors and a collection of potential installation sites in this context. Each sensor type is associated with a specific budget, and once the budget for a particular sensor type is exhausted, no additional sensors of that type can be deployed. The objective is to place one sensor at each location to maximize coverage across as many sites as possible. This deployment strategy is represented by a k-tuple of disjoint subsets of the installation sites. Some researchers (Krause and Guestrin, 2009) have employed entropy-based functions to evaluate the quality of the deployment strategy, which are inherently k-submodular.

Consequently, the problem is formulated as a constrained k-submodular maximization task with individual knapsack constraints.

Although many researchers have focused on k-submodular maximization with total cost constraint, somehow, surprisingly, the problem of maximizing the k-submodular function under an individual knapsack constraints (kSMIK) was under-explored.

Besides, one of the main challenges of this problem is the explosion of big data, which makes kSMIK become impractical due to the increasing search space exponentially. In this paper, we highly recommend the streaming fashion and propose some streaming-based algorithms to solve the kSMIK problem. Streaming is an online algorithm that scans a dataset once or a few times within acceptable time and memory consumption.

1.1. *Our contribution*

In this paper, we address the mentioned challenges by introducing 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 novel Streaming Algorithm (Algorithm 2 in this work) for the kSMIK problem. This algorithm exhibits an $O(Bk\log(n)/\epsilon)$ query complexity, $O(B\log(n)/\epsilon)$ space complexity, and achieves an approximation ratio of $\frac{1-\epsilon}{2(k+1)}$ for monotone objective functions, and $\frac{1-\epsilon}{2k+3}$ for non-monotone objective functions, where n is the size of the ground set, B is the total budget, and ϵ is an input parameter. Specially, 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 on two practical applications: k-topic influence maximization and k-type sensor placement under individual knapsack constraints, using standard datasets including Facebook, Enron, and Intel Lab datasets. Both applications are relevant to the kSMIK problem. The results highlight the superior performance of our algorithm compared to greedy approaches.

Overall, our contributions advance the understanding and solutions for the kSMIK problem, providing a novel streaming algorithm with a near-constant approximation ratio. 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 2 provides an overview of the existing literature and discussions. Section 3 presents the symbols and properties

of k-submodular functions, the foundation for our algorithmic design. Section 4 presents our proposed algorithm along with its theoretical analysis. Comprehensive experiments are detailed in Sec. 5, highlighting the performance of our algorithm in practical applications. Finally, we conclude our work in Sec. 6.

2. Related Work

***k*-submodular maximizations.** Initially, Singh *et al.* (2012) solved the problem of bisubmodular maximization, which corresponded to k-submodular maximization with $k = 2$. After that, researchers generalize the problem with every k. Note that when $k = 1$, the problem reduces to submodular maximization, known as NP-hard. Therefore, it follows that k-submodular maximization is also NP-hard. To solve the unconstrained problem, Ward and Zivný (2014) first proposed a Greedy deterministic algorithm with an approximation ratio of 1/3. Next, Iwata *et al.* (2016) improved a random Greedy version, which achieved an approximation ratio of $k/(2k-1)$ by combining a probability distribution with the way to select every element with larger marginal gains. Oshima (2017) later eliminated the randomness from the previous method but wasted more queries.

Constrained maximization of k-submodularity has also been explored. Ohsaka and Yoshida (2015) focused on monotone k-submodular maximization with cardinality constraint. They proposed a Greedy algorithm that provided a 1/2-approximation ratio for the case of total cardinality and a 1/3-approximation ratio for singular one. Next, Qian *et al.* (2018b) presented a multi-objective evolutionary algorithm for monotone k-submodular maximization under the total size constraint. Zheng *et al.* (2021) 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 (2017) 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 Rafiey and Yoshida (2020) and achieved an approximation ratio of 1/2 for the same problem. Recently, several works focused on k-submodular under knapsack constraint (Pham *et al.*, 2022b; Tang *et al.*, 2022; Pham *et al.*, 2022a; Ha *et al.*, 2024; Wang and Zhou, 2021) with some constant approximation ratios.

Streaming algorithms. Submodularity is a crucial property in machine learning with practical and theoretical implications. Nemhauser *et al.* (1978) highlighted its significance and demonstrated the effectiveness of Greedy algorithms for submodular set functions. However, applying Greedy algorithms directly becomes challenging in the era of big data, where data may not fit in memory, and random access is not feasible. Besides, data collected from real-time systems are also one of the

challenges when applying Greedy due to its inability to reserve its decision when selecting elements.

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.* (1996). 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 are 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 (Gomes and Krause, 2010; Badanidiyuru *et al.*, 2014; Kumar *et al.*, 2013; Kuhnle, 2021a; Pham *et al.*, 2023), knapsack (Huang *et al.*, 2020), k-set (Haba *et al.*, 2020), and matroid (Chakrabarti and Kale, 2015) 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 Ward and Zivný (2014), Iwata *et al.* (2016) were designed as single-pass streaming ones. The approach in Iwata *et al.* (2016) involved random selection, and subsequent works have improved upon it by introducing new element selection distributions based on different costs and establishing the relationships between the current and the optimal solutions.

Rafiey and Yoshida (2020) 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 achieved 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.* (2022b) 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 $k/(4\mathrm{k}-1)-\epsilon$ (in expectation).

In our view, prior research has yet to specifically address 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. Research endeavors should investigate this interesting problem theoretically and practically and devise 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 Sec. 1. We define $\text{supp}_i(\mathbf{s}) = S_i$, $\text{supp}(\mathbf{s}) = \cup_{i\in[k]} S_i$, S_i as *ith set of* $\boldsymbol{s}$ and an empty k-set $\mathbf{0} = (\emptyset, \ldots, \emptyset)$.

For $\mathbf{x} = (X_1, X_2, \ldots, X_k), \mathbf{y} = (Y_1, Y_2, \ldots, Y_k) \in (k+1)^V$, we set if $e \in X_i$ then $\mathbf{x}(e) = i$ and i is called the *position* of e, otherwise $\mathbf{x}(e) = 0$. Adding an element $e \notin \text{supp}(\mathbf{x})$ into X_i can be represented by $\mathbf{x} \sqcup (e, i)$. When $X_i = \{e\}$, and $X_j = \emptyset, \forall\, j \neq i$, $\mathbf{x}$ is denoted by (e, i). We denote by $\mathbf{x} \sqsubseteq \mathbf{y}$ if and only if $X_i \subseteq Y_i$ $\forall i \in [k]$.

The objective function. The function $f : (k+1)^V \mapsto \mathbb{R}_+$ is *k-submodular* if and only if for any $\mathbf{x} = (X_1, X_2, \ldots, X_k)$ and $\mathbf{y} = (Y_1, Y_2, \ldots, Y_k) \in (k+1)^V$, we have

$$f(\mathbf{x}) + f(\mathbf{y}) \geq f(\mathbf{x} \sqcap \mathbf{y}) + f(\mathbf{x} \sqcup \mathbf{y}), \tag{2}$$

where

$$\mathbf{x} \sqcap \mathbf{y} = (X_1 \cap Y_1, X_2 \cap Y_2, \ldots, X_k \cap Y_k),$$

$$\mathbf{x} \sqcup \mathbf{y} = (Z_1, \ldots, Z_k) \quad \text{where } Z_i = X_i \cup Y_i \setminus (\cup_{j\neq i} X_j \cup Y_j).$$

The function f is monotone if and only if for any $\mathbf{x} \in (k+1)^V, e \notin \text{supp}(\mathbf{x})$ and $i \in [k]$, we have the *marginal gain* when adding a tuple (e, i) into a set $\mathbf{x}$ is non-negative, i.e.,

$$\Delta_{(e,i)} f(\mathbf{x}) = f(X_1, \ldots, X_{i-1}, X_i \cup \{e\}, X_{i+1}, \ldots, X_k) - f(X_1, \ldots, X_k) \geq 0.$$

Example of *k*-submodular function in the *k*-IC model: According to the k-IC model (Ohsaka and Yoshida, 2015), the graph $G = (V, E)$ represents a social network, where information spreads according to the k-IC model with $k = 2$ topics. Initially, $\mathbf{x} = \{\{1, 2\}, \{3\}\}$ and $\mathbf{y} = \{\{1, 2\}, \{3, 4\}\}$, and clearly $\mathbf{x} \sqsubseteq \mathbf{y}$. Suppose, according to the k-IC model, information from $\mathbf{x}$ spreads to the nodes $\{1, 2, 3, 6, 7, 8\}$, while information from $\mathbf{y}$ spreads to the nodes $\{1, 2, 3, 4, 5, 6, 7\}$. When $u = 5$ is added to both $\mathbf{x}$ and $\mathbf{y}$, we have

- $\mathbf{x}$ spreads information to the nodes $\{1, 2, 3, 5, 6, 7, 8, 10\}$, affecting two new nodes.
- $\mathbf{y}$ spreads information to the nodes $\{1, 2, 3, 4, 5, 6, 7, 8, 10\}$, affecting only one new node.

Thus, when $u = 5$ is added, group $\mathbf{x}$ gains a higher benefit, as two new nodes are affected, while group $\mathbf{y}$ only adds one new node. This illustrates the k-submodular property of the influence function.

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

From Ward and Zivný (2014), the k-submodularity of f implies the *orthant submodularity*, i.e.,

$$\Delta_{(e,i)} f(\mathbf{x}) \geq \Delta_{(e,i)} f(\mathbf{y}) \tag{3}$$

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

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

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 is defined as $c_i(\mathbf{x}) = \sum_{e \in \text{supp}_i(\mathbf{x})} c(e)$. 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 constraints (kSMIK) problem is to find a k-set $\mathbf{s} = (S_1, S_2, \ldots, S_k)$ satisfying $c_i(\mathbf{s}) \leq B_i, \forall i \in [k]$ such that the objective function $f(\mathbf{s})$ is maximized.

In this work, we denote an instance of kSMIK as a tuple (f, V, k) and $\mathbf{o}$ as the optimal solution for the problem with the optimal value $\mathsf{opt} = f(\mathbf{o})$.

Streaming algorithm. We study the above problem in the streaming setting where the elements of the ground set V arrive one at a time in an arbitrary (adversarial) order. A streaming algorithm is designed to operate with limited memory, generally logarithmic in the size of the stream or the maximum value in the stream.

4. Proposed Algorithms

This section presents our streaming algorithm for kSMIK. We first introduce a simple version named Streaming algorithm with knowing opt (StrOpt), which assumes the optimal value opt known. Building upon StrOpt, we propose the main Streaming (Str) algorithm, which has an $O(Bk \log(n)/\epsilon)$ query complexity, $O(B \log(n)/\epsilon)$ space complexity, and achieves an approximation ratio of $\frac{1-\epsilon}{2(k+1)}$ for monotone objective functions, and $\frac{1-\epsilon}{2k+3}$ for non-monotone objective functions. We present the details and analysis of the two algorithms as follows.

4.1. *Streaming algorithm with knowing* opt (StrOpt)

StrOpt receives an instance (f, V, k), an approximation value v of opt such that $(1-\epsilon)\mathsf{opt} \leq v \leq \mathsf{opt}$, budgets $(B_1, B_2, \ldots, B_k)$ and parameters $\alpha, \epsilon \in (0, 1)$ are inputs. StrOpt follows a single-pass streaming, in which for each element received, it identifies:

- i_e: the "best" position within the k subsets with highest marginal gain (Line 5).
- $i_{\max}$: the "best" position within the k subsets based on the highest value of $f((e, i_e))$ and updates the maximal singleton pair $(e_{\max}, i_{\max})$ (Line 6).

Algorithm 1. StrOpt Algorithm.

1: **Input:** V, f, k, $B_1, B_2, \ldots, B_k$, an approximate value v of opt such that $(1-\epsilon)\mathsf{opt} \leq v \leq \mathsf{opt}$, parameters $\alpha, \epsilon \in (0,1)$.
2: **Output:** solution $\mathbf{s}$.
3: $\mathbf{s} \leftarrow \emptyset$, $(e_{\max}, i_{\max}) \leftarrow (null, 0)$
4: **for all** $e \in V$ **do**
5: $\quad i_e \leftarrow \arg\max_{i \in [k]} f((e,i))$
6: $\quad (e_{\max}, i_{\max}) \leftarrow \arg\max_{\mathbf{x} \in \{(e_{\max}, i_{\max}), (e, i_e)\}} f(\mathbf{x})$
7: $\quad$ **if** $c(e) + c_{i_e}(\mathbf{s}) \leq B_{i_e}$ **then**
8: $\quad\quad$ **if** $\Delta_{(e,i_e)} f(\mathbf{s}) \geq \frac{c(e)\alpha v}{B_{i_e}}$ **then**
9: $\quad\quad\quad \mathbf{s} \leftarrow \mathbf{s} \sqcup (e, i_e)$
10: $\quad\quad$ **end if**
11: $\quad$ **end if**
12: **end for**
13: $\mathbf{s} \leftarrow \arg\max_{\mathbf{s}' \in \{(e_{\max}, i_{\max}), \mathbf{s}\}} f(\mathbf{s}')$
14: **return** $\mathbf{s}$

The algorithm next checks the two conditions:

(1) the individual budget constraint $c(e) + c_{i_e}(\mathbf{s}) \leq B_{i_e}$ (Line 7)
(2) $\Delta(e, i_e) f(\mathbf{s}) \geq c(e)\alpha v / B$ (Line 8).

If both conditions are satisfied, the pair (e, i_e) is added to the candidate solution $\mathbf{s}$. The two conditions ensure that $\mathbf{s}$ is a feasible solution, and the algorithm adds a good pair into $\mathbf{s}$.

Finally, the algorithm returns the best solution between $\mathbf{s}$ and $(e_{\max}, i_{\max})$. For a comprehensive overview of the algorithm's steps, refer to Algorithm 1.

We now analyze the performance of StrOpt. First, we provide some useful notations for the algorithm analysis.

- $\mathbf{o}$ is an optimal solution of the problem over V and the optimal value $\mathsf{opt} = f(\mathbf{o})$.
- (e_j, i_j): the jth tuple added in the main loop of Algorithm 1.
- t is the number tuples added into $\mathbf{s}$ after the main loop.
- $\mathbf{s}^t = \{(e_1, i_1), \ldots, (e_t, i_t)\}$: the k-set $\mathbf{s}$ after ending the main loop, $t = |\mathrm{supp}(\mathbf{s}^t)|$.
- $\mathbf{s}^j = \{(e_1, i_1), \ldots, (e_j, i_j)\}$: the k-set $\mathbf{s}$ (in the main loop) after adding j elements $1 \leq j \leq t$, $\mathbf{s}^0 = \mathbf{0}$, $\mathbf{s}^t = \mathbf{s}$.
- $\mathbf{o}^j = (\mathbf{o} \sqcup \mathbf{s}^j) \sqcup \mathbf{s}^j$.
- v is a guessing value of opt, thus: $(1-\epsilon)\mathsf{opt} \leq v \leq \mathsf{opt}$.
- $\mathbf{o}_2^{j-1/2} = (\mathbf{o}_2 \sqcup \mathbf{s}^j) \sqcup \mathbf{s}^{j-1}$.
- $\mathbf{s}^{j-1/2}$: If $e_j \in \mathrm{supp}(\mathbf{o}_2)$, then $\mathbf{s}^{j-1/2} = \mathbf{s}^{j-1} \sqcup (e_j, \mathbf{o}_2(e_j))$. If $e_j \notin \mathrm{supp}(\mathbf{o}_2)$, $\mathbf{s}^{j-1/2} = \mathbf{s}^{j-1}$.

We first establish the relation between $f(\mathbf{o})$ and $f(\mathbf{s}^j)$ in Lemma 1.

Lemma 1. *For any $j, 0 \leq j \leq t$, we have*:

- *If f is monotone, then $f(\boldsymbol{o}) - f(\boldsymbol{o}^j) \leq f(\boldsymbol{s}^j)$.*
- *If f is non-monotone, then $f(\boldsymbol{o}) - f(\boldsymbol{o}^j) \leq 2f(\boldsymbol{s}^j)$.*

Proof. If f is monotone. We have $\mathbf{o}^0 = (\mathbf{o} \sqcup \mathbf{s}^0) \sqcup \mathbf{s}^0$ and $f(\mathbf{o}) = f(\mathbf{o}^0)$. For all $0 \leq j \leq t$, we have

$$f(\mathbf{o}) - f(\mathbf{o}^j) = \sum_{i=1}^{j} (f(\mathbf{o}^{i-1}) - f(\mathbf{o}^i)) \tag{5}$$

$$\leq \sum_{i=1}^{j} (f(\mathbf{o}^{i-1}) - f(\mathbf{o}^{i-1/2})) \tag{6}$$

$$\leq \sum_{i=1}^{j} (f(\mathbf{s}^{i-1/2}) - f(\mathbf{s}^{i-1})) \tag{7}$$

$$\leq \sum_{i=1}^{j} (f(\mathbf{s}^{i}) - f(\mathbf{s}^{i-1})) \tag{8}$$

$$\leq f(\mathbf{s}^j), \tag{9}$$

where the inequality (6) is due to the monotonicity 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.

For f is non-monotone, we need to use the pair-wise monotonicity property to carefully analyze $f(\mathbf{o}^{j-1}) - f(\mathbf{o}^j)$. Recap that (e_j, i_j) is the j-tuple added into the candidate set $\mathbf{s}$ after the loop of Algorithm 1. We have two sub-cases:

- If $e_j \notin \text{supp}(\mathbf{o})$, define an integer number $l \in [k]$ that $l \neq i_j$ and $\mathbf{o}_l^j$ as a k-set such that $\mathbf{o}_l^j(e) = \mathbf{o}^j(e), \forall e \in V \backslash \{e_j\}$ and $\mathbf{o}_l^j(e_j) = l$, we have

$$f(\mathbf{o}^{j-1}) - f(\mathbf{o}^j) = f(\mathbf{o}_l^j) - f(\mathbf{o}^{j-1}) - (f(\mathbf{o}^j) \tag{10}$$

$$+ f(\mathbf{o}_l^j) - 2f(\mathbf{o}^{j-1})) \tag{11}$$

$$\leq f(\mathbf{o}_l^j) - f(\mathbf{o}^{j-1}) \tag{12}$$

$$\leq f(\mathbf{s}_l^j) - f(\mathbf{s}^{j-1}) \tag{13}$$

$$\leq f(\mathbf{s}^j) - f(\mathbf{s}^{j-1}), \tag{14}$$

  where the inequality (12) is due to the pair-wise monotoncity of f, the inequality (13) is due to the k-submodularity of f and the inequality (14) is due to the selection of the algorithm.

- If $e_j \in \text{supp}(\mathbf{o})$. In this case, if $\mathbf{o}^{j-1}(e_j) = i_j$. Due to the pairwise-monotone property of f, there exists $i' \in [k]$ that $f(\mathbf{s}^{j-1} \sqcup (e_j, i')) \geq 0$. Therefore, $f(\mathbf{o}^j) - f(\mathbf{o}^{j-1}) = 0 \leq f(\mathbf{s}^j) - f(\mathbf{s}^{j-1})$. If $\mathbf{o}^{j-1}(e_j) \neq i_j$, we also obtain

$$\begin{aligned} f(\mathbf{o}^{j-1}) - f(\mathbf{o}^j) &= 2f(\mathbf{o}^{j-1}) - 2f(\mathbf{o}^{j-1/2}) \\ &\quad - (f(\mathbf{o}^{j-1}) + f(\mathbf{o}^j) - 2f(\mathbf{o}^{j-1/2})) \\ &\leq 2f(\mathbf{o}^{j-1}) - 2f(\mathbf{o}^{j-1/2}) \\ &\leq 2f(\mathbf{s}^j) - 2f(\mathbf{s}^{j-1}). \end{aligned}$$

Overall, we have $f(\mathbf{o}^{j-1}) - f(\mathbf{o}^j) \leq 2f(\mathbf{s}^j) - 2f(\mathbf{s}^{j-1})$. Therefore,

$$\begin{aligned} f(\mathbf{o}) - f(\mathbf{o}^t) &= \sum_{j=1}^{t} (f(\mathbf{o}^{j-1}) - f(\mathbf{o}^j)) \\ &\leq 2\sum_{j=1}^{t} (f(\mathbf{s}^j) - f(\mathbf{s}^{j-1})) \\ &\leq 2f(\mathbf{s}^t). \end{aligned}$$

The proof is completed. □

Lemma 2. *After ending the first loop, if for any pair* (e, i) $e \in \text{supp}(\boldsymbol{o}) \setminus \text{supp}(\boldsymbol{s}^t)$ *and* $i \in [k]$ *such that* $c_i(\boldsymbol{s}) + c(e) > B_i$ *and* $\Delta_{(e,i)} f(\boldsymbol{s}^t) < c(e)\alpha v B_i$*, then we have*:

- *If* f *is monotone, then* $f(\boldsymbol{s}) > v(1 - \alpha k)/2$.
- *If* f *is non-monotone, then* $f(\boldsymbol{s}) > v(1 - \alpha k)/3$.

Proof. If f is monotone, we have

$$f(\mathbf{o}^t) - f(\mathbf{s}^t) \leq \sum_{e \in \text{supp}(\mathbf{o}) \setminus \text{supp}(\mathbf{s}^t)} \Delta_{(e,\mathbf{o}(e))} f(\mathbf{s}^t) \tag{15}$$

$$= \sum_{i=1}^{k} \sum_{e \in \text{supp}(\mathbf{o}) \setminus \text{supp}(\mathbf{s}), \mathbf{o}(e) = i} \Delta_{(e,\mathbf{o}(e))} f(\mathbf{s}^t) \tag{16}$$

$$< \sum_{i=1}^{k} \frac{c_i(\mathbf{o})\alpha v)}{B_i} \tag{17}$$

$$\leq \sum_{i=1}^{k} \frac{B_i \alpha v}{B_i} \leq k\alpha v, \tag{18}$$

where the inequality (15) is due to the k-submodularity of f, the inequality (17) is due to the given condition in the lemma. In the other hand, By applying Lemma 1,

we have

$$v - f(\mathbf{s}^t) \leq f(\mathbf{o}) - f(\mathbf{s}^t) \tag{19}$$

$$= f(\mathbf{o}) - f(\mathbf{o}^t) + f(\mathbf{o}^t) - f(\mathbf{s}^t) \tag{20}$$

$$< f(\mathbf{s}^t) + k\alpha v \tag{21}$$

which implies that $f(\mathbf{s}^t) \geq v(1-\alpha k)/2$.

If f is non-monotone, by the similar argument with the monotone case, we have:

$$v - f(\mathbf{s}^t) \leq f(\mathbf{o}) - f(\mathbf{s}^t) \tag{22}$$

$$= f(\mathbf{o}) - f(\mathbf{o}^t) + f(o^t) - f(s^t) \tag{23}$$

$$\leq 2f(s^t) + \sum_{e \in \text{supp}(\mathbf{o}) \setminus \text{supp}(\mathbf{s})} (\Delta_{(e,\mathbf{o}(e))}\mathbf{s}^t) \tag{24}$$

$$\leq 2f(s^t) + \sum_{i=1}^{k} \sum_{e \in \text{supp}(\mathbf{o}) \setminus \text{supp}(\mathbf{s}), \mathbf{o}(e)=i} \Delta_{(e,\mathbf{o}(e))} f(\mathbf{s}^t) \tag{25}$$

$$< 2f(s^t) + \sum_{i=1}^{k} \frac{B_i \alpha v}{B_i} \tag{26}$$

$$\leq 2f(s^t) + k\alpha v, \tag{27}$$

where the inequality (23) is due to the Lemma 1, the inequality (26) is due to the selection of the algorithm. Thus, we have $f(\mathbf{s}^t) > v(1-\alpha k)/3$. The proof is completed. □

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

Proof. Since $\Delta_{(e,i)} f(\mathbf{s}^t) \geq c(e)\alpha v/B_i$ so we have

$$f(\mathbf{s} \sqcup (e,i)) - f(\mathbf{s}^t) \geq c(e)\alpha v/B_i \tag{28}$$

$$\Rightarrow f(\mathbf{s}^t \sqcup (e,i)) \geq f(\mathbf{s}^t) + \frac{c(e)\alpha v}{B_i} \tag{29}$$

$$\geq \frac{c_i(\mathbf{s}^t)}{B_i} + \frac{c(e)\alpha v}{B_i} \tag{30}$$

$$\geq \frac{(c_i(\mathbf{s}^t) + c(e))\alpha v}{B_i} \tag{31}$$

$$\geq \alpha v, \tag{32}$$

where inequality (30) is due to the selection rule of the algorithm. Note that, after the main loop of Algorithm 1, we have $(e_{\max}, i_{\max}) = \arg\max_{e \in V, i \in [k]} f((e,i))$, we

have

$$f(\mathbf{s}) \geq \max\{f(\mathbf{s}^t), f((e_{\max}, i_{\max}))\} \tag{33}$$

$$\geq \frac{f(\mathbf{s}^t) + f((e_{\max}, i_{\max}))}{2} \tag{34}$$

$$\geq \frac{f(\mathbf{s}^t) + f((e, i))}{2} \tag{35}$$

$$\geq \frac{f(\mathbf{s}^t \sqcup (e, i))}{2} \tag{36}$$

$$\geq \frac{\alpha v}{2}, \tag{37}$$

where inequality (37) is due to (32). This completes the proof. □

Finally, we state the performance of Algorithm 1 as follows:

Theorem 1. *For $\epsilon, \alpha \in (0, 1)$ and any approximation value v such that $(1 - \epsilon)$ $\mathsf{opt} \leq v \leq \mathsf{opt}$, Algorithm 1. takes $O(nk)$ query complexity and*

- *If f is monotone and $\alpha = \frac{1}{k+1}$, Algorithm 1 returns an approximation ratio of $\frac{1-\epsilon}{2(k+1)}$.*
- *If f is non-monotone and $\alpha = \frac{2}{2k+3}$, Algorithm 1 returns an approximation ratio of $\frac{1-\epsilon}{2k+3}$.*

Proof. The algorithm makes at most n iteration. For each iteration, it takes k queries to decide whether to add e to the $\mathbf{s}$ or not. Thus, the Algorithm takes totally $O(nk)$ query complexity. We now prove the bounds of solution. For the case f is monotone, we apply Lemmata 2, 3 in two cases as follows:

Case 1. After ending the first loop, if for any $e \in \text{supp}(\mathbf{o})\backslash\text{supp}(\mathbf{s})$ 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

$$f(\mathbf{s}) > \frac{v(1 - \alpha k)}{2} \geq \frac{v}{2(k+1)} = \frac{\mathsf{opt}(1 - \epsilon)}{2(k+1)}. \tag{38}$$

Case 2. If there exists an element $e \in \text{supp}(\mathbf{o})\backslash\text{supp}(\mathbf{s})$ 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, we also get

$$f(\mathbf{s}) \geq f(\mathbf{s}^t) \geq \frac{v}{2(k+1)} \geq \frac{\mathsf{opt}(1 - \epsilon)}{2(k+1)}.$$

For the case f is non-monotone, we get the proof by reasoning similar to that of the monotone case. □

4.2. Str: *The main streaming algorithm*

We now remove the assumption of knowing opt and introduce our main algorithm (Algorithm 2), named Str in this part.

Str first initiates a set of guesses of optimal values: $O \leftarrow \{v = (1+\epsilon)^j : M \leq (1+\epsilon)^j \leq BM\}$, where $B = \sum_{i=1}^{k} B_k$ and M is a value of maximal singleton $(e_{\max}, i_{\max})$. Str consists of two passes. At the first pass (Lines 4–15), it constructs a set of candidate solutions $\mathbf{s}_v, v \in O$ as follows: For each element e, the algorithm handles two tasks:

- Update M and the set O according to the set of received elements (Lines 7–8). The value of M will be updated with the best value of the current $(e_{\max}, i_{\max})$ pair. Based on the updated value of M, the set O will be updated accordingly.
- Add e into candidate solution $\mathbf{s}_v, v \in O$ if it has large marginal gain and adding e to $\mathbf{s}_v$ does not violate the budget constraint.

The second pass (Lines 16–23) is to improve the solution quality of candidate solutions. For each incoming element e, the algorithm finds the best position i_e (Line 18). The algorithm adds (e, i_e) into $\mathbf{s}_v$ (Lines 19–20) if the (e, i_e) has a positive marginal gain.

In the following, we analyze the theoretical guarantee of Algorithm 2. We first recap the notations as follows:

- $\mathbf{o}$ is an optimal solution of the problem over V and the optimal value $\mathsf{opt} = f(\mathbf{o})$.
- (e_j, i_j): the jth element added of the main loop of Algorithm 1.
- $\mathbf{s}^t = \{(e_1, i_1), \ldots, (e_t, i_t)\}$: the k-set $\mathbf{s}$ after ending the main loop, $t = |\text{supp}(\mathbf{s}^t)|$.
- $\mathbf{s}^j = \{(e_1, i_1), \ldots, (e_j, i_j)\}$: the k-set $\mathbf{s}$ (in the main loop) after adding j elements $1 \leq j \leq t$, $\mathbf{s}^0 = \mathbf{0}$, $\mathbf{s}^t = \mathbf{s}$.
- $\mathbf{o}^j = (\mathbf{o} \sqcup \mathbf{s}^j) \sqcup \mathbf{s}^j$.
- v is a guessing value of opt, thus: $(1-\epsilon)\mathsf{opt} \leq v \leq \mathsf{opt}$.
- $\mathbf{o}_2^{j-1/2} = (\mathbf{o}_2 \sqcup \mathbf{s}^j) \sqcup \mathbf{s}^{j-1}$.
- $\mathbf{s}^{j-1/2}$: If $e_j \in \text{supp}(\mathbf{o}_2)$, then $\mathbf{s}^{j-1/2} = \mathbf{s}^{j-1} \sqcup (e_j, \mathbf{o}_2(e_j))$. If $e_j \notin \text{supp}(\mathbf{o}_2)$, $\mathbf{s}^{j-1/2} = \mathbf{s}^{j-1}$.

We state the performance guarantee of Str in the following theorem:

Theorem 2. *Algorithm* 2 *takes two passes over the ground set, has* $O(\frac{nk\log(B)}{\epsilon})$ *query complexity, needs* $O(\frac{B\log(B)}{\epsilon})$ *space complexity. For approximation ratio*:

- *If* f *is monotone and* $\alpha = \frac{1}{k+1}$, *Algorithm* 2 *returns an approximation ratio of* $\frac{1-\epsilon}{2(k+1)}$.
- *If* f *is monotone and* $\alpha = \frac{2}{2k+3}$, *Algorithm* 2 *returns an approximation ratio of* $\frac{1-\epsilon}{2k+3}$.

Proof. The algorithm works in two passes. For each incoming element e, it takes $|O|k$ queries to find the best position to decide whether to add e into $\mathbf{s}_v, v \in O$ or

Algorithm 2. Str algorithm.

Input: V, f, k, α, $B_1, B_2, \ldots, B_k (\forall i \in [k])$.
Output: A solution $\mathbf{s}$
$M \leftarrow 0$, $(e_{\max}, i_{\max}) \leftarrow (null, 0)$
for all $e \in V$ **do**
  $i_e \leftarrow \arg\max_{i \in [k]} f((e,i))$
  $(e_{\max}, i_{\max}) \leftarrow \arg\max_{\mathbf{x} \in \{(e,i_e),(e_{\max},i_{\max})\}} f(\mathbf{x})$
  $M \leftarrow f((e_{\max}, i_{\max}))$
  $O \leftarrow \{v = (1+\epsilon)^j : M \leq (1+\epsilon)^j \leq BM\}$
  for all $v \in O$ **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 $\Delta_{(e,i_v)} f(\mathbf{s}_v) \geq c(e)\alpha v / B_{i_e}$ **then**
      $\mathbf{s}_v \leftarrow \mathbf{s}_v \sqcup (e, i_v)$
    end if
  end for
end for
for all $e \in V \backslash \mathrm{supp}(\mathbf{s}_v)$ **do**
  for all $v \in O$ **do**
    $i_e \leftarrow \arg\max_{i \in [k], c_i(\mathbf{s}_v)+c(e) \leq B_i} \Delta_{(e,i)} f(\mathbf{s}_v)$
    if $\Delta_{(e,i)} f(\mathbf{s}_v) > 0$ **then**
      $\mathbf{s}_v \leftarrow \mathbf{s}_v \sqcup (e, i_e)$
    end if
  end for
end for
$\mathbf{s} \leftarrow \arg\max_{\mathbf{x} \in \{(e_{\max},i_{\max})\} \cup \{\mathbf{s}_v : v \in O\}} f(\mathbf{x})$
return $\mathbf{s}$

not. Therefore, the total number of queries of the algorithm is at most:

$$2|O|kn = 2\frac{nk\log(B)}{\epsilon} = O\Big(\frac{nk\log(B)}{\epsilon}\Big). \tag{39}$$

The algorithm requires $|O|$ candidate solutions, each B memories so the algorithm takes $O(\frac{B\log(B)}{\epsilon})$ space complexity.

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

$$(1-\epsilon)\mathsf{opt} \leq \frac{\mathsf{opt}}{1+\epsilon} \leq v \leq \mathsf{opt}. \tag{40}$$

Therefore, we apply Lemma 2 and obtain the approximation ratio. □

5. Experiments

In this section, we present experimental results to complement the theoretical findings provided in the previous sections.

Algorithm 3. Greedy Algorithm.

Input: V, f, k, $B_1, B_2, \ldots, B_k$.
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 if
  $\mathbf{s} \leftarrow \mathbf{s} \sqcup (e_m, i_m)$
  $V \leftarrow V \backslash e_m$
end while
return $\mathbf{s}$

5.1. *Applications and datasets*

As mentioned previously, no prior work has specifically addressed the kSMIK problem, so we developed a simple greedy algorithm (Algorithm 3) for comparison. The greedy algorithm iteratively selects every remaining element with the highest marginal gain to add to the solution at each step. Greedy Algorithm takes $O(n^2 k)$ query complexity and does not provide any approximation ratio.

We evaluate their performance of algorithms on three specific applications and five standard datasets (see Table 1).

5.1.1. *k-type Product Revenue Maximization Under Individual Knapsack Constraint* (kPMIK)

The kPMIK is defined as follows: Given $G = (V, E)$, where V represents the set of vertices corresponding to customers, and each edge $(i, j) \in E$ has a weight $w_{ij} \geq 0$ that reflects the relationship between customers i and j. The objective is to allocate the k products to the customers by determining a k-set $\mathbf{s} = (S_1, S_2, \ldots, S_k)$, where S_i is the subset of customers allocated to product i, such that the objective function $f(\mathbf{s})$ is maximized. The objective function is defined as follows:

$$f(\mathbf{s}) = \sum_{u \in V} \sum_{i=1}^{k} \left(\sum_{v \in S_i} w_{uv} \right)^{\alpha_{u,i}}, \tag{41}$$

where $\alpha_{u,i} \in (0, 1)$. The function $f(\mathbf{s})$ is a k-submodular and monotone function.

To demonstrate this, consider $\mathbf{s} \sqsubseteq \mathbf{t}$ (i.e., $S_i \subseteq T_i$ for all i) and an element $e \notin \text{supp}(\mathbf{t})$. The marginal gain when adding e to S_i is

$$\Delta_{(e,i)} f(\mathbf{s}) = \sum_{u \in V} \left[\left(\sum_{v \in S_i} w_{uv} + w_{ue} \right)^{\alpha_{u,i}} - \left(\sum_{v \in S_i} w_{uv} \right)^{\alpha_{u,i}} \right]. \tag{42}$$

Since $\alpha_{u,i} \in (0,1)$ and $w_{uv} \geq 0$, the function $h(x) = x^{\alpha_{u,i}}$ is concave and increasing on $x \geq 0$. Because $S_i \subseteq T_i$, we have $a = \sum_{v \in S_i} w_{uv} \leq b = \sum_{v \in T_i} w_{uv}$, and $c = w_{ue} \geq 0$. By the property of concave functions, we have

$$h(a+c) - h(a) \geq h(b+c) - h(b). \tag{43}$$

Therefore, $\Delta_{(e,i)} f(\mathbf{s}) \geq \Delta_{(e,i)} f(\mathbf{t})$, which shows that $f(\mathbf{s})$ is k submodular. Additionally, since the marginal gain is non-negative ($\Delta_{(e,i)} f(\mathbf{s}) \geq 0$), the function $f(\mathbf{s})$ is monotonic. Thus, $f(\mathbf{s})$ is both k-submodular and monotonic.

For kPMIK, we use three datasets including: Facebook[a] (Leskovec and Krevl, 2014), Astro[b] (Leskovec *et al.*, 2007) and, Enron[c] (Leskovec *et al.*, 2009). The Facebook dataset comprises a subgraph of the social network Facebook, featuring 4,039 nodes and 88,234 edges. Each node corresponds to a user within the social network, while edges denote user relationships. The Astro dataset represents the collaboration network from the Astro Physics section of arXiv, consisting of 18,772 nodes and 198,110 edges. Each node corresponds to an author, and an undirected edge indicates that two authors have co-authored a paper. The Enron dataset represents an email communication network consisting of 36,692 nodes and 183,831 edges. Each node corresponds to an email address, while edges signify the sending of emails between these addresses.

5.1.2. *k-type Sensor Placement Under Individual Knapsack constraint (kSPIK)*

The kSPIK is defined as follows: Given k types of sensors catering to various measures and a set V comprising n locations, each designated for only one sensor. Each sensor e incurs a positive cost $c(e)$ associated with its type i. Given a set of positive values $(B_1, B_2, \ldots, B_k)$, the objective is strategically positioning these sensors to maximize information acquisition with $c(S_i) = \sum_{e \in S_i} c(e) \leq B_i, \forall i \in [k]$, where S is a solution.

Let X_e^i be a random variable representing the observation collected from a sensor of type i. For a k-set of sensors $\mathbf{s}$, the information gained is defined as

$$f(\mathbf{s}) = H\left(\bigcup_{e \in \text{supp}(\mathbf{s})} X_e^i\right), \tag{44}$$

where H is the entropy function. The function f is monotone and k-submodular (Ohsaka and Yoshida, 2015).

[a]https://snap.stanford.edu/data/ego-Facebook.html.
[b]https://snap.stanford.edu/data/ca-AstroPh.html.
[c]https://snap.stanford.edu/data/email-Enron.html.

For kSPIK, we use the Intel Lab dataset (Bodik *et al.*, 2004). Intel Lab dataset comprising data gathered from 54 sensors was deployed at the Intel Berkeley Research lab from February 28th to April 5th, 2004. The sensors, which included Mica2Dot sensors equipped with weatherboards, recorded timestamped topology information and measurements of humidity, temperature, light, and voltage every 31 s.

5.1.3. *k-topic Influence Maximization Under Individual Knapsack Constraint* (kIMIK)

The problem is formally defined as follows: Consider a social network represented by a directed graph $G = (V, E)$, where V denotes the set of users and E represents the set of directed edges. The objective is to select a seed set $\mathbf{s} = (S_1, S_2, \ldots, S_k)$ for k distinct topics, such that the total number of active nodes across all topics is maximized, while adhering to an individual knapsack constraint for each topic.

Each edge $(u, v) \in E$ is associated with a weight $w^i(u, v)$, which quantifies the influence exerted by user u on user v for topic i. Furthermore, each node $u \in V$ has an associated influence threshold $\theta^i(u)$ for each topic i, which is randomly drawn from the interval $[0, 1]$. Given that each topic i is subject to its own budget constraint B_i, the task is to identify a set of seed nodes S_i for each topic i, such that the total cost of selecting these seed nodes, denoted by $c(S_i) = \sum_{e \in S_i} c(e)$, satisfies the individual budget constraint $c(S_i) \leq B_i$, while maximizing the total number of active nodes resulting from the information diffusion process.

Let $\sigma(\mathbf{s})$ denote the number of nodes that become active in at least one of the k topics after the information diffusion process. The function $\sigma(\cdot)$ is a monotone k-submodular function (Ohsaka and Yoshida, 2015). The problem of k-topic Influence Maximization under Individual Knapsack Constraints can be formally expressed as follows:

$$\text{Max}_{\mathbf{s} \in (k+1)^V} \quad \sigma(\mathbf{s}) = \mathbb{E}\left[\left|\bigcup_{i \in [k]} \sigma_i(S_i)\right|\right] \tag{45}$$

$$\text{Subject to} \quad c(S_i) \leq B_i \quad \text{for each} \quad i \in [k]. \tag{46}$$

For the kIMIK problem, we utilize the random ER dataset (Erdös and Rényi, 1960) with $n = 500$ and $p = 0.1$. Additionally, we employ the Independent Cascade (IC) model to simulate the information diffusion process over the social network. To accurately estimate the value of the objective function, we perform 1000 simulations for each seed set and compute the average result.

5.2. *Experiment settings*

In our evaluation, we set $\epsilon = 0.1$ and $k = 3$ for all experiments. For the kPMIK application, edge weights, vertex costs, and $\alpha_{u,i}$ values are randomly

Table 1. The datasets used in experiments.

Database	#Nodes	#Edges	Types
Enron (Leskovec *et al.*, 2009)	36,692	183,831	Undirected
Astro (Leskovec *et al.*, 2007)	18,772	198,110	Undirected
Facebook (Leskovec and Krevl, 2014)	4,039	88,234	Undirected
Intel Lab sensors(Bodik *et al.*, 2004)	54	—	—
ER (Erdös and Rényi, 1960)	500	—	Directed

assigned within the range $(0, 1)$ across all datasets. Additionally, we assume that all budgets $B_1, B_2, \ldots, B_k$ are equal without loss of generality. The total budget $(B = B_1 + B_2 + \cdots + B_k)$ benchmarks range from 0.03 to 0.3 of the total cost.

The experiments were conducted[d] in C++, building upon the source code from Kuhnle (2021b)[e] for kPMIK, kSPIK and from Kagan *et al.* (2024)[f] for kIMIK. All experiments were conducted on a High-Performance Computing (HPC) server cluster with the following specifications: partition = large, 16 CPU threads, 2 nodes, and 3,073 GB of maximum memory.

5.3. *Experiment results*

We evaluate three key metrics: the oracle value of the objective function, the number of queries, and the running time. Figures 1, 2, and 3 illustrate the results for kPMIK on the Facebook, Astro and Enron datasets, while Fig. 4 showcases the results for kSPIK on the Intel Lab dataset, and Fig. 5 presents the results for kIMIK on the ER dataset.

(a) **Regarding kPMIK.** Figures 1(a), 2(a) and 3(a) display the objective values. In general, the Str algorithm consistently yields higher objective values than the Greedy algorithm across all three datasets. There are a few instances where the Greedy algorithm outperforms the Str algorithm in terms of objective values; however, these differences are not significant ($B = 0.09$ and 0.15 for the Facebook set; $B = 0.06$ and 0.18 for the Astro set). Notably, the Str algorithm demonstrates a steady increase in the objective value as the B value increases. In contrast, the Greedy algorithm's objective value rises rapidly initially but tends to stagnate as the B value continues to increase. Specifically, in the Enron dataset, the Str algorithm's objective value is consistently 16% higher than that of the Greedy algorithm for B values of 0.15 and above.

[d]Our source code is available at: https://github.com/tantdhvan/kSIMK.
[e]https://gitlab.com/kuhnle/linear-submodular-stream.
[f]https://github.com/AlexanderKagan/gltm_experiments.

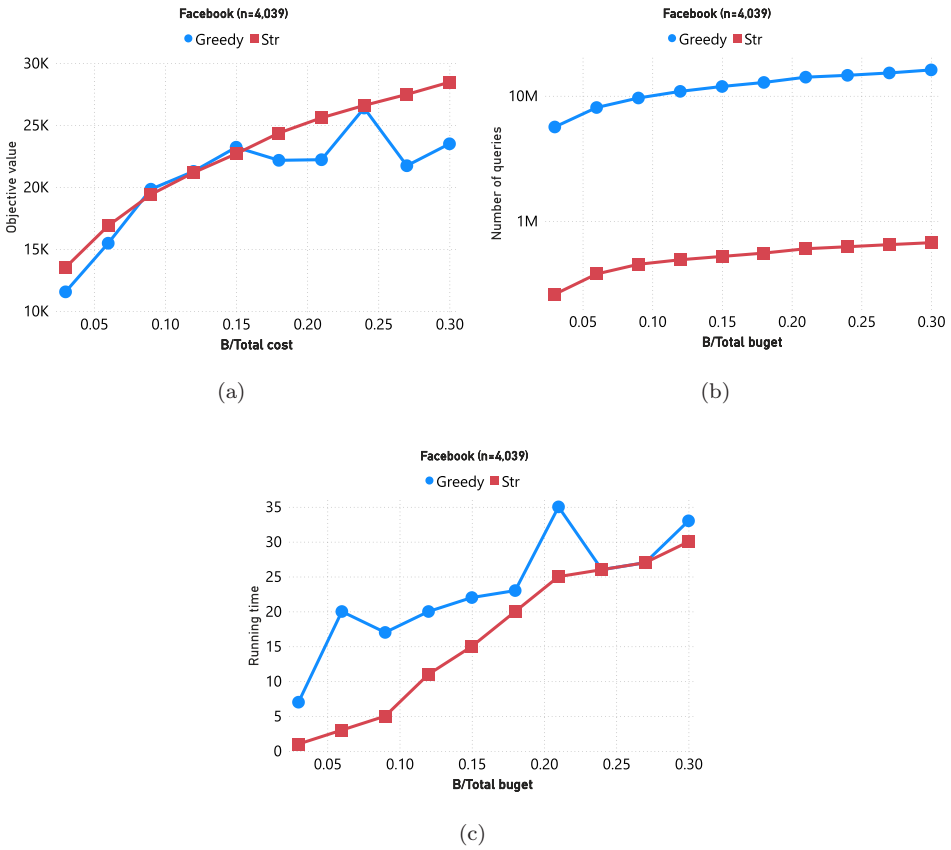

Fig. 1. Algorithm results for kPMIK on Facebook: (a) objective values, (b) number of queries, and (c) running time.

Figures 1(b), 2(b), and 3(b) illustrate the number of queries. The Str algorithm consistently requires significantly fewer queries to the function $f(\cdot)$ compared to the Greedy algorithm across all three datasets. Specifically, in the Facebook dataset, the Greedy algorithm performs 21.2 to 23.9 times more queries than the Str algorithm; in the Astro dataset, it performs 92.4 to 103.6 times more queries; and in the Enron dataset, it performs 63.5 to 130.8 times more queries. Notably, the number of queries for the Greedy algorithm increases rapidly as the B value rises, whereas the Str algorithm maintains a more stable query count.

Running time is shown in Figs. 1(c), 2(c), and 3(c). The running time of the Greedy algorithm consistently exceeds that of the Str algorithm. Specifically, in the Facebook dataset, the Greedy algorithm has a running time that is 1.1 to 7.0 times longer; in the Astro dataset, it is 1.1 to 1.5 times longer; and in the Enron dataset, it is 31.1 to 47.3 times longer. The difference is particularly evident in the Enron

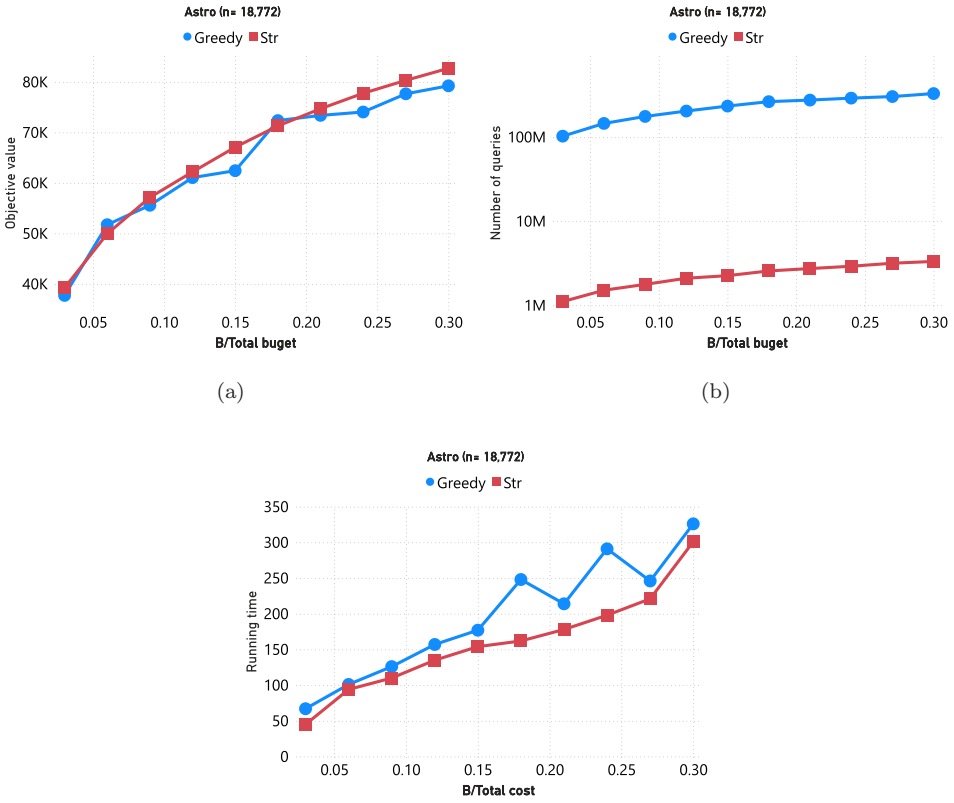

Fig. 2. Algorithm results for kPMIK on Astro: (a) objective values, (b) number of queries, and (c) running time.

dataset, which contains a large number of vertices. This demonstrates the superior performance of our Str algorithm compared to the Greedy algorithm.

(b) **Regarding kSPIK.** Figure 4(a) illustrates the objective value. It indicates that the objective values of the Str and Greedy algorithms are identical. However, the Greedy algorithm consistently yields a slightly higher objective value.

Figure 4(b) displays the number of queries. It reveals that the Greedy algorithm consistently requires more queries than the Str algorithm across all budget values. Specifically, at milestones $B_1, B_2, \ldots, B_k = 10$ and 20, the number of queries for the Greedy algorithm is 1.5 times higher than that for the Str algorithm. However, when $B_1, B_2, \ldots, B_k \geq 30$, the number of queries for the Greedy algorithm is consistently twice as high as that for the Str algorithm.

Figure 4(c) presents the execution time. It shows that the running times of the Str and Greedy algorithms are equivalent, as the Intel Lab dataset comprises only

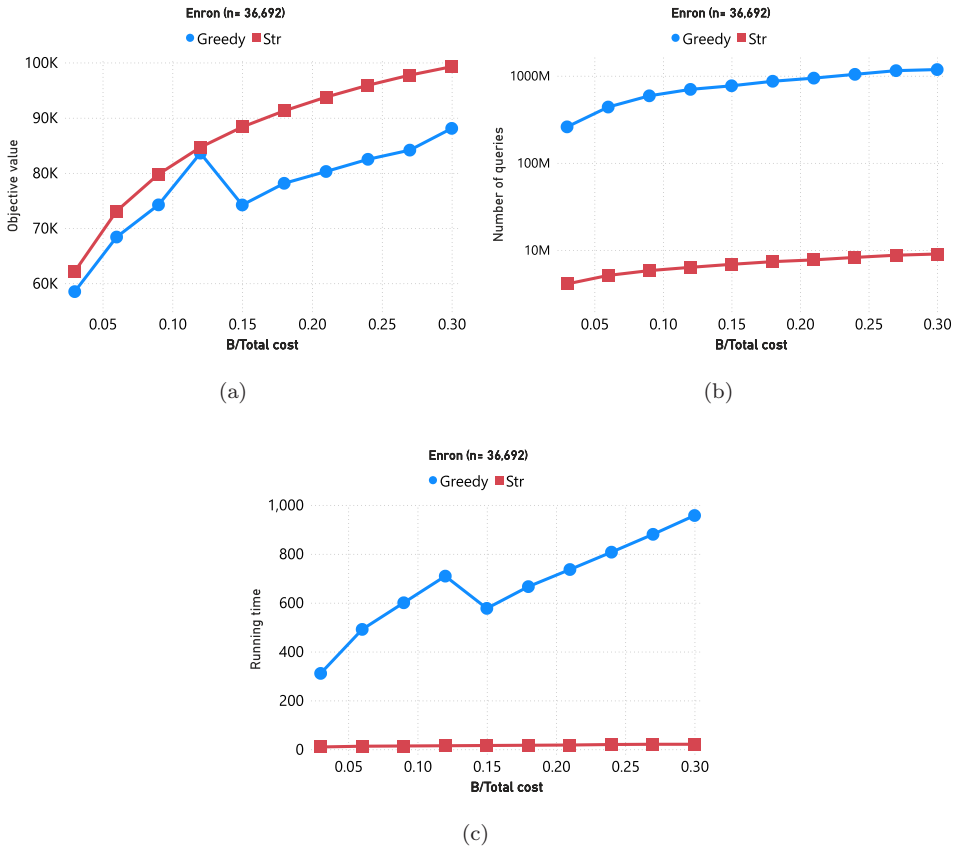

Fig. 3. Algorithm results for kPMIK on Enron: (a) objective values, (b) number of queries, and (c) running time.

54 nodes. Nevertheless, the running time of the Str algorithm consistently remains lower than that of the Greedy algorithm.

(c) **Regarding kIMIK.** Similar to the two previously discussed applications, the metrics of the objective function, the number of queries, and the running time maintain a consistent trend. Specifically, the objective function values of our algorithm and the greedy algorithm remain identical, with minor fluctuations at certain positions; however, these variations are negligible.

Regarding the number of queries, as shown in Fig. 5(b), the greedy algorithm still requires a significantly higher number of queries, ranging from two to three times more than our algorithm. Notably, the number of queries for the greedy algorithm increases rapidly as the value of B rises. In contrast, the number of queries for the function f in the Stralgorithm increases at a much slower rate.

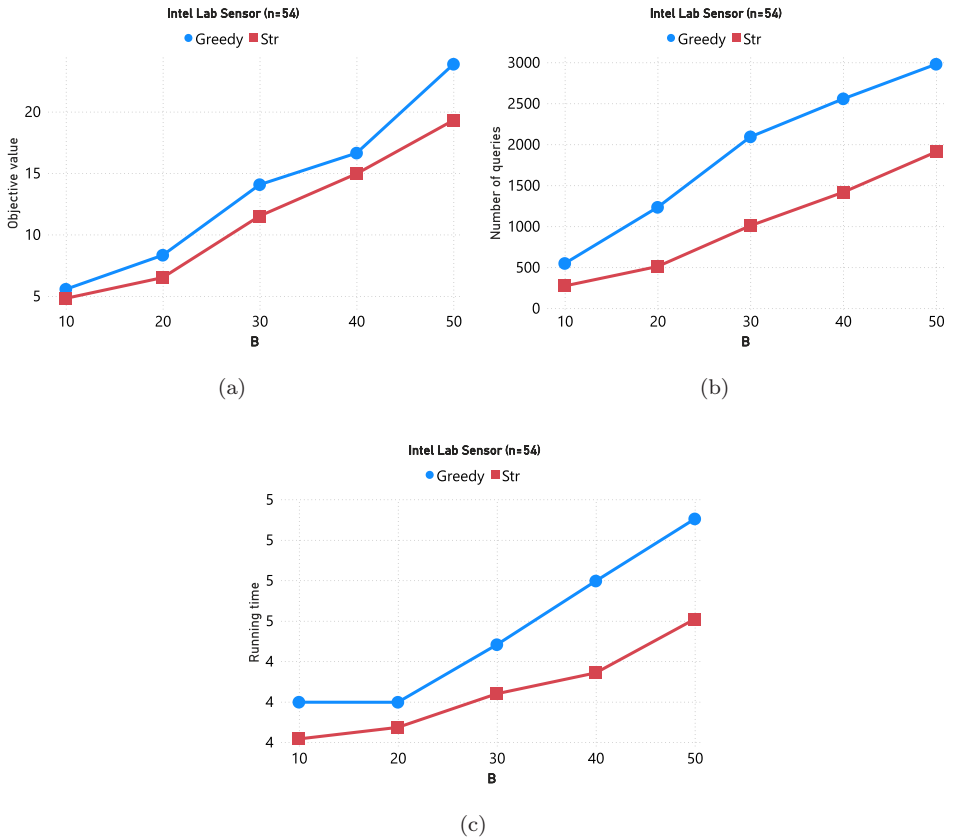

Fig. 4. Algorithm results for kSPIK on Intel Lab: (a) objective values, (b) number of queries, and (c) running time.

As for the running time of both algorithms, as illustrated in Fig. 5(c), the trend in running time mirrors the trend in the number of queries, reflecting the correlation between these two factors during execution.

(d) **Discussion.** The experimental results highlight the superiority of the Stralgorithm over the Greedy algorithm across various datasets and metrics. Our algorithm consistently yields higher objective values, especially as the budget B increases, and requires significantly fewer queries. This demonstrates better efficiency in exploring solution spaces. While the Greedy algorithm exhibits a rapid increase in query count and running time, particularly for large datasets like Enron, the Stralgorithm maintains more stable performance, showcasing its scalability and computational efficiency. These findings reinforce the effectiveness of our approach in large-scale applications.

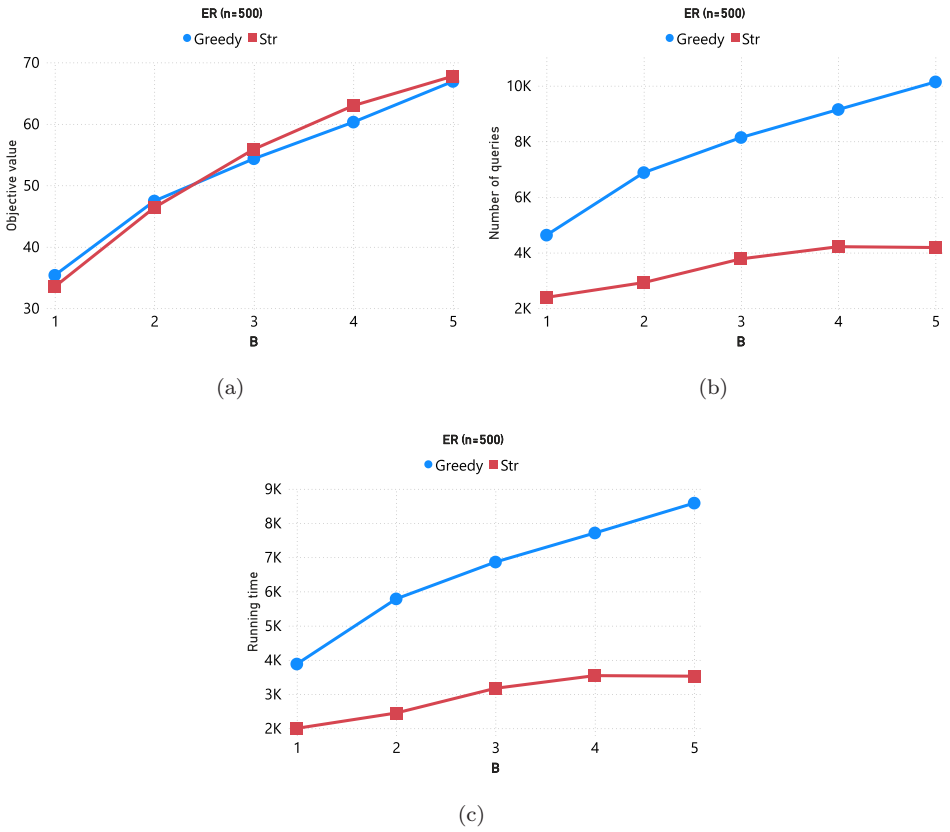

Fig. 5. Algorithm results for kIMIK on ER Dataset: (a) objective values, (b) number of queries, and (c) running time.

6. Conclusion

This paper has presented a streaming algorithm for maximizing a k-submodular function under an individual knapsack constraint. The algorithm achieves an approximation ratio of approximately $\frac{1-\epsilon}{2(k+1)}$ for monotone objective functions, and $\frac{1-\epsilon}{2k+3}$ for non-monotone objective functions, with $O(Bk\log(n)/\epsilon)$ query complexity, $O(B\log(n)/\epsilon)$ space complexity. The main approach employed is to restrict the optimal range. This work is the first time to deal with the k-submodular function under an individual knapsack constraint problem.

To assess the practical performance of our algorithms, experiments were conducted on two applications: k-topic Influence Maximization and k-type Sensor Placement under an individual knapsack constraint. The experimental results demonstrate that the quality of the solution of our algorithm outperforms the Greedy algorithm's. 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 kSMIK problem.

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.

Acknowledgment

The first author (Tan D. Tran) was funded by the Master, PhD Scholarship Programme of Vingroup Innovation Foundation (VINIF), code VINIF.2024.TS.069.

ORCID

Tan D. Tran https://orcid.org/0000-0001-5866-8458

Canh V. Pham https://orcid.org/0000-0002-8118-1768

Dung T. K. Ha https://orcid.org/0000-0001-8375-7866

References

Alon, N, Y Matias and M Szegedy (1996). The space complexity of approximating the frequency moments. In *Proc. 28th Annual ACM Symp. Theory of Computing (STOC)*, Philadelphia, PA, USA, pp. 20–29.

Badanidiyuru, A, B Mirzasoleiman, A Karbasi and A Krause (2014). Streaming submodular maximization: Massive data summarization on the fly. In *Proc. Int. Conf. Knowledge Discovery and Data Mining (KDD)*, New York, NY, USA, pp. 671–680.

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

Chakrabarti, A and S Kale (2015). Submodular maximization meets streaming: Matchings, matroids, and more. *Mathematical Programming,* 154(1–2), 225–247.

Ene, A and H Nguyen (2022). Streaming algorithm for monotone k-submodular maximization with cardinality constraints. In *Int. Conf. Machine Learning (ICML)*, Baltimore, MD, USA, pp. 5944–5967. PMLR.

Erdös, P and A Rényi (1960). On the evolution of random graphs. *Publication of the Mathematical Institute of the Hungarian Academy of Science*, 5, 17–61.

Gomes, R and A Krause (2010). Budgeted nonparametric learning from data streams. In *Proc. Int. Conf. Machine Learning (ICML)*, Haifa, Israel, J Fürnkranz and T Joachims (Eds.), pp. 391–398.

Ha, DT, CV Pham and TD Tran (2024). Improved approximation algorithms for k-submodular maximization under a knapsack constraint. *Computers & Operations Research*, 161, 106452.

Haba, R, E Kazemi, M Feldman and A Karbasi (2020). Streaming submodular maximization under a k-set system constraint. In *Proc. Int. Conf. Machine Learning (ICML)*, Virtual Conference, pp. 3939–3949.

Huang, C, N Kakimura and Y Yoshida (2020). Streaming algorithms for maximizing monotone submodular functions under a knapsack constraint. *Algorithmica*, 82(4), 1006–1032.

Iwata, S, S Tanigawa and Y Yoshida (2016). Improved approximation algorithms for k-submodular function maximization. In *Proc. 27th Annual ACM-SIAM Symp. Discrete Algorithms*, R Krauthgamer (Ed.), pp. 404–413. SIAM.

Kagan, A, E Levina and J Zhu (2024). General linear threshold models with application to influence maximization, arXiv:2411.09100.

Kempe, D, JM Kleinberg and É Tardos (2003). Maximizing the spread of influence through a social network. In *Proc. Int. Conf. Knowledge Discovery and Data Mining (KDD)*, Washington, DC, USA, pp. 137–146.

Krause, A and C Guestrin (2009). Optimizing sensing: From water to the web. *IEEE Computer*, 42(8), 38–45.

Kuhnle, A (2021a). Quick streaming algorithms for maximization of monotone submodular functions in linear time. In *Proc. Int. Conf. Artificial Intelligence and Statistics (AISTATS)*, Virtual Conference, pp. 1360–1368.

Kuhnle, A (2021b). Quick streaming algorithms for maximization of monotone submodular functions in linear time. In *Int. Conf. Artificial Intelligence and Statistics*, pp. 1360–1368. PMLR.

Kumar, R, B Moseley, S Vassilvitskii and A Vattani (2013). Fast greedy algorithms in MapReduce and streaming. In *Proc. Symp. Parallelism in Algorithms and Architectures (SPAA)*, Montreal, QC, Canada, pp. 1–10.

Leskovec, J, J Kleinberg and C Faloutsos (2007). Graph evolution: Densification and shrinking diameters. *ACM Transactions on Knowledge Discovery from Data*, 1(1), 2es, doi: 10.1145/1217299.1217301, https://doi.org/10.1145/1217299.1217301.

Leskovec, J and A Krevl (2014). SNAP datasets: Stanford large network dataset collection, http://snap. stanford.edu/data.

Leskovec, J, KJ Lang, A Dasgupta and MW Mahoney (2009). Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters. *Internet Mathematics*, 6(1), 29–123.

Nemhauser, GL, LA Wolsey and ML Fisher (1978). An analysis of approximations for maximizing submodular set functions — I. *Mathematical Programming*, 14(1), 265–294.

Nguyen, L and M Thai (2020). Streaming k-submodular maximization under noise subject to size constraint. In *Proc. Int. Conf. Machine Learning (ICML)*, Virtual Conference, pp. 7338–7347.

Ohsaka, N and Y Yoshida (2015). Monotone k-submodular function maximization with size constraints. In *Proc. of Annual Conf. Neural Information Processing Systems (NeurIPS)*, Montreal, QC, Canada, 28, pp. 694–702.

Oshima, H (2017). Derandomization for k-submodular maximization. In *Proc. Int. Workshop Combinatorial Algorithms (IWOCA)*, Newcastle, NSW, Australia, L Brankovic, J Ryan and WF Smyth (Eds.), pp. 88–99.

Pham, CV, HV Duong and MT Thai (2019). Importance sample-based approximation algorithm for cost-aware targeted viral marketing. In *Int. Conf. Computational Data and Social Networks*, pp. 120–132. Springer.

Pham, CV, DKT Ha, HX Hoang and TD Tran (2022a). Fast streaming algorithms for k-submodular maximization under a knapsack constraint, in *9th IEEE Int. Conf. Data Science and Advanced Analytics*, *Shenzhen*, *China*, *October 13–16*, *2022*, pp. 1–10. IEEE.

Pham, CV, TD Tran, DT Ha and MT Thai (2023). Linear query approximation algorithms for non-monotone submodular maximization under knapsack constraint. In *Proc. 32nd Int. Joint Conf. Artificial Intelligence (IJCAI)*, Macau, S.A.R., China, pp. 4127–4135.

Pham, CV, QC Vu, DK Ha, TT Nguyen and ND Le (2022b). Maximizing k-submodular functions under budget constraint: Applications and streaming algorithms. *Journal of Combinatorial Optimization*, 44(1), 723–751.

Pham, CV, QC Vu, DKT Ha and TT Nguyen (2021). Streaming algorithms for budgeted k-submodular maximization problem. In *Computational Data and Social Networks*, pp. 27–38. Cham: Springer International Publishing.

Qian, C, J Shi, K Tang and Z Zhou (2018b). Constrained monotone k-submodular function maximization using multiobjective evolutionary algorithms with theoretical guarantee. *IEEE Transactions on Evolutionary Computation*, 22(4), 595–608.

Rafiey, A and Y Yoshida (2020). Fast and private submodular and k-submodular functions maximization with matroid constraints. In *Proc. Int. Conf. Machine Learning (ICML)*, Virtual Conference, pp. 7887–7897.

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

Singh, AP, A Guillory and JA Bilmes (2012). On bisubmodular maximization. In *Proc. Int. Conf. Artificial Intelligence and Statistics (AISTATS)*, La Palma, Canary Islands, Spain, pp. 1055–1063.

Tang, Z, C Wang and H Chan (2022). On maximizing a monotone k-submodular function under a knapsack constraint. *Operations Research Letters*, 50(1), 28–31.

Wang, B and H Zhou (2021). Multilinear extension of k-submodular functions. arXiv:2107.07103.

Ward, J and S Zivný (2014). Maximizing bisubmodular and k-submodular functions. In *Proc. Symp. Discrete Algorithms (SODA)*, Portland, OR, USA, pp. 1468–1481.

Xiao, H, Q Liu, Y Zhou and M Li (2023). Non-monotone k-submodular function maximization with individual size constraints. In *11 th Int. Conf. Computational Data and Social Networks*, *Virtual Event*, *December 5–7*, *2022*, pp. 268–279. Springer.

Zheng, L, H Chan, G Loukides and M Li (2021). Maximizing approximately k-submodular functions. In *Proc. SIAM Int. Conf. Data Mining (SDM)*, Virtual Conference, pp. 414–422.

Biography

Tan D. Tran is currently a PhD candidate at the University of Engineering and Technology, Vietnam National University, Hanoi. His research interests include combinatorial optimization, submodular functions, and artificial intelligence.

Canh V. Pham is an Associate Professor and Dean of the Faculty of Computer Science at Phenikaa University, where he also leads the Operational Research and AI Laboratory (ORLab) in Hanoi, Vietnam. He earned his PhD in Computer Science from Vietnam National University – University of Engineering and Technology. His research focuses on scalable approximation algorithms for combinatorial optimization, social network analysis, submodular maximization, and machine learning–based optimization. He serves as Associate Editor for the *Journal of Combinatorial Optimization* (Springer) and as a reviewer for various IEEE/Springer journals. His academic service includes program committee roles at IJCAI 2024 and ICLR 2025.

Dung T. K. Ha received his PhD in Computer Science from the University of Engineering and Technology, Vietnam National University. His research interests include combinatorial optimization, approximation algorithms, and submodular optimization.