Computers & Operations Research 161 (2024) 106452

Contents lists available at ScienceDirect

Computers and Operations Research

journal homepage: www.elsevier.com/locate/cor

Improved approximation algorithms for k-submodular maximization under a knapsack constraint

Dung T.K. Ha [a], Canh V. Pham [b,*], Tan D. Tran [a]

[a] VNU University of Engineering Technology, Hanoi, 11300, Viet Nam
[b] ORLab, Phenikaa University, Hanoi, 12116, Viet Nam

ARTICLE INFO

Keywords:
Combinatorial optimization
Approximation algorithm
k-submodular
Knapsack constraint

ABSTRACT

We investigate the problem of k-submodular maximization under a knapsack constraint over the ground set of size n. This problem finds many applications in various fields, such as multi-topic propagation, multi-sensor placement, cooperative games, etc. However, existing algorithms for the studied problem face challenges in practice as the size of instances increases in practical applications.

This paper introduces three deterministic and approximation algorithms for the problem that significantly improve both the approximation ratio and query complexity of existing practical algorithms. Our first algorithm, FA, returns an approximation ratio of $1/10$ within $O(nk)$ query complexity. The second one, IFA, improves the approximation ratio to $1/4-\epsilon$ in $O(nk/\epsilon)$ queries. The last one IFA+ upgrades the approximation ratio to $1/3-\epsilon$ in $O(nk\log(1/\epsilon)/\epsilon)$ query complexity, where ϵ is an accuracy parameter. Our algorithms are the first ones that provide constant approximation ratios within only $O(nk)$ query complexity, and the novel idea to achieve results lies in two components. Firstly, we divide the ground set into two appropriate subsets to find the near-optimal solution over these ones with $O(nk)$ queries. Secondly, we devise algorithmic frameworks that combine the solution of the first algorithm and the greedy threshold method to improve solution quality. In addition to the theoretical analysis, we have evaluated our proposed ones with several experiments in some instances: Influence Maximization, Information Coverage Maximization, and Sensor Placement for the problem. The results confirm that our algorithms ensure theoretical quality as the cutting-edge techniques, including streaming and non-streaming algorithms, and also significantly reduce the number of queries.

1. Introduction

The problems of constrained k-submodular function maximization have played an important role in advancing the fields of operational research, artificial intelligence and machine learning recently because of their natural usages in various domains such as influence maximization via social networks (Ohsaka and Yoshida, 2015; Rafiey and Yoshida, 2020a; Qian et al., 2018a; Nguyen and Thai, 2020; Zheng et al., 2021), sensor placement (Ohsaka and Yoshida, 2015; Rafiey and Yoshida, 2020a; Qian et al., 2018a; Zheng et al., 2021), feature selection (Singh et al., 2012) and information coverage maximization (Qian et al., 2018a), etc. Given a finite ground set V and an integer number k, we define $[k] = \{1, 2, \dots, k\}$ and $(k+1)^V = \{(V_1, V_2, \dots, V_k) | V_i \subseteq V, \forall i \in [k], V_i \cap V_j = \emptyset, \forall i \neq j\}$ as a family of k disjoint sets, called the **k-set**. A function $f : (k+1)^V \mapsto \mathbb{R}_+$ is **k-submodular** iff for any $\mathbf{x} = (X_1, X_2, \dots, X_k)$ and $\mathbf{y} = (Y_1, Y_2, \dots, Y_k) \in (k+1)^V$, we have:

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

where

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

and

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

In this paper, we investigate the k-Submodular Maximization under a Knapsack constraint (kSMK) problem, which is one of the most natural and general k-submodularity optimizations since it captures the limitation of budget, time, or size when choosing elements. Under the knapsack constraint, each element e is assigned a positive cost $c(e)$. Given a limited budget $B > 0$, the problem kSMK asks to find a k-set $\mathbf{s} = (S_1, S_2, \dots, S_k)$ with total cost $c(\mathbf{s}) = \sum_{e \in S_i, i \in [k]} c(e) \leq B$ so that $f(\mathbf{s})$ is maximized. The problem is a general model for many important applications in social networks, sensor placement, information coverage, feature selection, etc., listed below:

* Corresponding author.
E-mail address: canh.phamvan@phenikaa-uni.du.vn (C.V. Pham).

https://doi.org/10.1016/j.cor.2023.106452
Received 8 March 2023; Received in revised form 6 October 2023; Accepted 6 October 2023
Available online 10 October 2023

k-topic influence maximization. In some real situations, such as viral marketing, product recommendation, etc., a company may desire to broadly spread an advertisement campaign about k different products (or topics) to users via social networks or the Internet. The campaign may hire influential users to share their hobbies or experiments about the products with their followers, and then these followers share the news with their relationships. Hence, the information about the products gradually propagates over the network and influences many users. The mathematical nature of the campaign is k-submodular maximization under a diffusion model, which Kempe et al. (2003) first proposed with a single type of influence. Authors in Ohsaka and Yoshida (2015) generalized this model to allow $k \geq 2$ types of influence, then other authors were also attracted by k-topic influence maximization (Rafiey and Yoshida, 2020a; Qian et al., 2018a; Nguyen and Thai, 2020). In the Experiment section, we describe more detail about this instance.

k-type sensor placement. Sensor placement originates from research in practice when researchers desire to gain information about water quality, temperature, earthquakes, and air pollution observed from environments around a large area such as rivers, mountains or air, etc., to support forecasting (Krause and Guestrin, 2007, 2009; Du et al., 2014) or sensor networks provide real-time monitoring and control of network systems (Huy and Viet, 2015). These applications require multiple types of sensors to work. Suppose there are k types of sensors about temperature, wind speed, energy, etc., and a set V of n locations to place them. If only one sensor is allocated to one site, each k-tuple of pairwise disjoint subsets of V represents a k-type sensor placement plan. The plan's performance can usually be evaluated using k-submodular functions such as the entropy functions. Therefore, constrained k-submodular maximization can be used to describe a k-type sensor placement issue. We also introduce more detail about this problem in the Experiment.

k-topic information coverage maximization. In the influence maximization problem, an inactive node may be informed of information by any of its active neighbor nodes. Hence, Wang et al. (2015) proposed a new problem called information coverage maximization: maximizing the expected number of active and informed nodes. An inactive node is informed if at least one active neighbor node exists. Qian et al. (2018b) introduced k-topic information coverage maximization, which expands the k-topic information maximization.

k-class feature selection. This problem has become interesting recently (Bilbao, 2000; Singh et al., 2012; Saeys et al., 2007) in machine learning. Feature selection enhances the analysis of a vast amount of data by reducing its dimensionality. The resulting multi-class feature selection issues consist of a pool of k associated features and k uncorrelated prediction variables. The issue asks to not only find the most informative features but also classify the features with respect to the prediction variables, leading to a k-submodular maximization problem.

In the above applications, the size of the problem often increases rapidly, requiring efficient algorithms to provide suitable solutions in a reasonable computational time. We refer to the query complexity as a measure of computational time since it dominates the time running of an algorithm.

1.1. Our contribution

In this work, we design novel algorithms for the problem that respond to some requirements about maintaining performance and reducing query complexity. In particular, our work is the first one that provides a constant approximation ratio within only $O(kn)$ query complexity and can return an approximation ratio of $1/3 - \epsilon$ which is better than the current best approximation ratio $(1 - 1/e)/2$ of a deterministic and polynomial time algorithm in Tang et al. (2022). To our knowledge, our contribution is the lowest computational cost of any constant ratio approximation algorithm and plays a significant role in finding near-optimal solutions for applications, as the expense of evaluating the function f might be costly. A preliminary version of this work appears in the proceedings of the 9th IEEE International Conference on Data Science and Advanced Analytics (DSAA 2022) (Pham et al., 2022a). This paper extends the conference version by providing more material and experiment evaluation. In general, our contributions are as per the following:

- We first propose the FA algorithm (Algorithm 1), a 1/10-approximation one that needs only one pass over the ground set and $O(kn)$ query complexity. It is our work's first simple but necessary version since it bounds the optimal and provides a data division strategy to reduce query complexity to $O(nk)$.
- We next propose IFA algorithm (Algorithm 2) that achieves an approximation ratio $1/4 - \epsilon$, and requires $O(kn/\epsilon)$ query complexity where ϵ is an accuracy parameter.
- We enhance the quality of the algorithm IFA by proposing IFA+ which takes $O(kn\log(1/\epsilon)/\epsilon)$ query complexity but can provide the approximation ratio at $1/3 - \epsilon$. To the best of our knowledge, this algorithm outperforms the Greedy algorithm in Tang et al. (2022) with approximation ratio $(1 - 1/e)/2$, the current best approximation ratio for a deterministic algorithm.
- To illustrate the theoretical contributions, we conduct several comprehensive experiments in three applications of kSMK, including k-topic Influence Maximization, k-topic Information Coverage Maximization, and k-type Sensor Placement. Experimental results have shown that our algorithms not only need a much more modest number of queries than the cutting-edge non-streaming ones (mentioned in Table 1) but also return comparable solutions in terms of quality.

Table 1 compares our algorithms with some state-of-the-art algorithms for kSMK on three aspects including approximation ratio, query complexity, and whether the algorithm is deterministic or not. These fields indicate that our algorithms have both a low number of queries and valuable deterministic approximation ratios that are equivalent to or even better than the others.

Organization. The rest of the paper is organized as follows: We provide a literature review and discussions in Section 2. The notations and properties of k-submodular functions are presented in Section 3. Section 4 presents our algorithms and theoretical analysis. The extensive experiments are shown in Section 5. Finally, we conclude this work in Section 6.

2. Related works

In this section, we review related works and provide some discussions on existing applications of k-submodular function and existing algorithms

The submodular maximization problem is a type of combinatorial optimization in which the objective generally is a set function. Because of its high applicability, it has attracted in multiple domains such as economics (Amir, 2005; Milgrom and Roberts, 1990), social welfare (Vondrák, 2008), artificial intelligence (Khuller et al., 1999; Buchbinder et al., 2015; Krause et al., 2008; Mirzasoleiman et al., 2016a), data summarization (Mirzasoleiman et al., 2015, 2016b), recommendation systems (Guillory and Bilmes, 2011), and a lot of applications in operational research and management science such as sensor placement (Krause and Guestrin, 2009; Du et al., 2014; Lopes and Ramos, 2023), multi-agent systems (Lee et al., 2021), social influence (Kempe et al., 2003; Güney et al., 2021), bandwidth packing problem (Using submodularity, 2023) and facility location (Ortiz-Astorquiza et al., 2015; Dam et al., 2022), etc. However, submodularity might not be enough to capture some practical scenarios. Studying k-submodular functions arises when one desires to gather information from various sources. First, Lovász (1982) took a question about a similar but deeper theory than submodularity when working with partitioned matroid

Table 1
Algorithms for kSMK.

Reference	Approximation ratio	Query complexity	Type
FA (Alg. 1, this paper)	$1/10$	$O(kn)$	Deterministic
IFA (Alg. 2, this paper)	$1/4-\epsilon$	$O(kn/\epsilon)$	Deterministic
IFA+ (Alg. 3, this paper)	$1/3-\epsilon$	$O(kn\log(1/\epsilon)/\epsilon)$	Deterministic
Deterministic streaming (Pham et al., 2022b)	$1/4-\epsilon$	$O(kn\log(n)/\epsilon)$	Deterministic
Random streaming (Pham et al., 2022b)	$k/(4k-1)-\epsilon$	$O(kn\log(n)/\epsilon)$	Randomized
Greedy (Tang et al., 2022)	$1/2-1/(2e)$	$O(n^4k^3)$	Deterministic
Algorithm 3 in Wang and Zhou (2021)	$1/2$	$poly(n)$	Randomized

examples. After that, people considered a bisubmodular function to answer the above question, creating the foundation for k-submodularity research. Singh et al. (2012) worked with bisubmodular maximization, which meant $k = 2$. After that, more works have focused on the issue of general k.

The problem of maximizing k-submodular function was first investigated without any constraint by Ward and Zivný (2014). In that work, the authors devised a deterministic Greedy algorithm that gave an approximation ratio of $1/3$. After that, the authors in Iwata et al. (2016) presented a random Greedy approach which improved the approximation ratio to $k/(2k-1)$ by introducing a probability distribution to select which element has a larger marginal gain with higher probability. Later on, Oshima (2017) eliminated the random told in Iwata et al. (2016); however, the number of queries expanded to $O(n^2k^2)$.

The constrained maximizing of the k-submodular function has been researched further. Ohsaka and Yoshida (2015) first worked with monotone k-submodular maximization with two kinds of size constraint: overall size constraint, i.e., given a total budget $B > 0$ the goal is to construct a solution $S = (S_1, S_2, \dots, S_k)$ satisfying $|\cup_{i=1}^{k} S_i| \le B$ and singular size constraint, i.e., given singular budgets $B_i > 0, i \in [k]$, the goal is to construct a solution $S = (S_1, S_2, \dots, S_k)$ satisfying $|S_i| \le B_i$. In that work, the authors demonstrated that the Greedy algorithm provided an approximation ratio of $1/2$ for the overall size constraint and an approximation ratio of $1/3$ for the singular size constraint, which was asymptotically tight given that an approximation ratio of $(k+1)/2k+\epsilon$ for any $\epsilon > 0$ needed exponential running time (Iwata et al., 2016). A multi-objective evolutionary method with an approximation ratio of $1/2$ was also proposed by Qian et al. (2018b) for the monotone k-submodular maximization problem with the overall size constraint. However, this algorithm took a high query complexity of $O(kn\log^2 B)$ in expectation. Authors Soma (2019) then proposed an online algorithm with the same approximation ratio of $1/2$ but runs in polynomial time with regret bound.

Streaming fashion is an efficient approach for k-submodular maximization with large data when it requires only a small amount of memory to store and scans one or a few times over the ground set V. Nguyen and Thai (2020) first devised two streaming algorithms for the k-submodular maximization with overall size constraint within $O(nk\log(k))$ query complexity. Their first algorithm is deterministic and returns an approximation ratio of $1/3-\epsilon$, while the second one is randomized and returns an approximation ratio of $k/(3k-1)-\epsilon$. Recently, Ene and Nguyen (2022) developed a single-pass streaming algorithm based on integer programming formulation for k-submodular maximization with singular size constraint with an approximation ratio of $0.5/(1+B(2^{1/B}-1))$ within $O(nk)$ queries, where $B = \min_{i\in[k]} B_i$.

Besides, the k-submodular maximization has been further investigated under other constraints. Authors in Sakaue (2017) adapt the Greedy algorithm in Ohsaka and Yoshida (2015) to obtain an approximation ratio of $1/2$ for the problem under a matroid constraint. Another algorithm having the same approximation ratio by using the differential private continuous Greedy method was proposed by authors in Rafiey and Yoshida (2020b).

The knapsack constraint is one of the most natural ones, which requires maximizing $f(\cdot)$ subject to a given budget that the total cost of a solution cannot exceed. The knapsack constraints do not allow for just enumerating elements like cardinality or matroid constraints. Hence, there can be multiple solutions with maximal costs that are not the same size. The authors Wang and Zhou (2021) proposed a multi-linear extension method with an approximation ratio of $1/2-2\epsilon$ in expectation for the kSMK. They employed a 2-step method and constructed a continuous extension of the discrete problem. After an optimization algorithm located an optimum in the continuous space, a rounding technique extracts a discrete solution from a fractional one. This work provides the best approximation ratio in expectation; however, this algorithm is impractical due to the high query complexity of a continuous extension (Balkanski et al., 2021). In contrast, our contributions propose competitive deterministic approximation algorithms with nearly-linear query complexity for the problem. Besides, Tang et al. (2022) proposed a $(1/2-1/(2e))$-approximation algorithm for the kSMK inspired by the Greedy algorithm in Sviridenko (2004). This algorithm, however, requires an expensive query complexity of $O(n^4k^3)$, and therefore it is difficult to apply to medium-sized instances even though one can compute the objective function f in $O(1)$ time. Recently, Pham et al. (2022b) have proposed two single-pass streaming algorithms for the k-submodular maximization under the budget constraint, a general of knapsack constraint within $O(nk\log(n)/\epsilon)$ queries. These algorithms returned the ratios of $1/4-\epsilon$ and $k/(4k-1)-\epsilon$ (in expectation). Our best algorithm version, IFA+, even provides the approximation ratio of $1/3-\epsilon$ within $O(kn\log(1/\epsilon)/\epsilon)$ query complexity. The characteristic of our algorithms is deterministic, challenging approximation ratio, and nearly-linear query complexity.

3. Preliminaries

Notations. Given a ground set $V = \{e_1, e_2, \dots, e_n\}$ and an integer k, we define $[k] = \{1, 2, \dots, k\}$ and let $(k+1)^V = \{(V_1, V_2, \dots, V_n) | V_i \subseteq V\ \forall i \in [k], V_i \cap V_j = \emptyset\ \forall i \ne j\}$ be a family of k disjoint subsets of V, called k-**set**.

For $\mathbf{x} = (X_1, X_2, \dots, X_k) \in (k+1)^V$, we define $supp_i(\mathbf{x}) = X_i$, $supp(\mathbf{x}) = \cup_{i\in[k]} X_i$, X_i as i**th set of** $\mathbf{x}$ and an empty k-set $\mathbf{0} = (\emptyset, \dots, \emptyset)$. We set if $e \in X_i$ then $\mathbf{x}(e) = i$ and i is called the **position** of e in $\mathbf{x}$, otherwise $\mathbf{x}(e) = 0$. Adding an element $e \notin supp(\mathbf{x})$ into X_i can be represented by $\mathbf{x} \sqcup (e, i)$. We also write $\mathbf{x} = \{(e_1, i_1), (e_2, i_2), \dots, (e_t, i_t)\}$ for $e_j \in supp(\mathbf{x}), i_j = \mathbf{x}(e_j), \forall 1 \le j \le t$. When $X_i = \{e\}$, and $X_j = \emptyset, \forall j \ne i$, $\mathbf{x}$ is denoted by (e, i).

For $\mathbf{x} = (X_1, X_2, \dots, X_k), \mathbf{y} = (Y_1, Y_2, \dots, Y_k) \in (k+1)^V$, we denote by $\mathbf{x} \sqsubseteq \mathbf{y}$ iff $X_i \subseteq Y_i\ \forall i \in [k]$. For simplicity, we assume that f is non-negative, i.e., $f(\mathbf{x}) \ge 0$ for all $\mathbf{x} \in (k+1)^V$ and normalized, i.e., $f(\mathbf{0}) = 0$.

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

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

where

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

and

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

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

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

We assume that there exists an *oracle query*, which, when queried with the k-set $\mathbf{x}$ returns the value $f(\mathbf{x})$. We recap some properties of the k-submodular function that will be used for designing our algorithms. 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 $\mathbf{x}, \mathbf{y} \in (k+1)^V, e \notin supp(\mathbf{y}), \mathbf{x} \sqsubseteq \mathbf{y}$ and $i \in [k]$; and the *pairwise monotonicity*, i.e., for any $i, j \in [k], i \neq j$:

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

The problem definition. Assuming that each element e is assigned a positive cost $c(e)$ and the total cost of a k-set $\mathbf{x}$ $c(\mathbf{x}) = \sum_{e \in supp(\mathbf{x})} c(e)$. Given a limited budget $B > 0$, we assume that every item $e \in V$ satisfies $c(e) \leq B$; otherwise, we can discard it. The k-Submodular Maximization under Knapsack constraint (kSMK) problem is to determine:

$$\arg \max_{\mathbf{x} \in (k+1)^V : c(\mathbf{x}) \leq B} f(\mathbf{x}). \tag{5}$$

In this work, we only consider $k \geq 2$ because if $k = 1$, the k-submodular function becomes the submodular function.

We denote by $\mathbf{o} = \{(o_1, i_1^*), \dots, (o_m, i_m^*)\}$ an optimal solution of the problem, the optimal value $\mathsf{opt} = f(\mathbf{o})$ and $m = |supp(\mathbf{o})|$. Without loss of generality, we assume that $c(o_1) \geq c(o_2) \geq \cdots \geq c(o_m)$.

4. The proposed algorithms

In this section, we introduce three deterministic algorithms for kSMK. The first algorithm, named **Fast Approximation** (FA), has an approximation ratio of $1/10$ and takes $O(nk)$ query complexity. Although this approximation ratio is small, it is the **first one** that gives a constant approximation ratio within only $O(kn)$ queries. The approximation ratio is improved by our second algorithm, named **Improved Fast Approximation** (IFA), from $1/10$ to $1/4 - \epsilon$ by recalling the first algorithm's solution to provide a suitable range for bounding the optimal value opt. Additionally, it scans the ground set $O(1/\epsilon)$ times and integrates the decreasing threshold strategy to get the near-optimal solution. Save the best for last, we finally introduce the **Improved Approximation Plus** (IFA+) algorithm, which makes upward our contributions to $(1/3 - \epsilon)$ approximation ratio within $O(kn \log(1/\epsilon)/\epsilon)$ query complexity.

4.1. Fast Approximation algorithm

Our FA algorithm's main idea is that (1) divides the ground set V into two subsets: The elements with costs greater than $B/2$ are included in the **first subset**, while the remaining is included in the **second**, and (2) near-optimal solutions are sought and combined for the two aforementioned subsets.

In particular, the algorithm first receives an instance (V, f, k, B) of kSMK and initiates a candidate solution $\mathbf{s}$ as $\mathbf{0}$ and a tuple (e_m, i_m) as $(\emptyset, 1)$. The objective of the tuple (e_m, i_m) is to update the optimal solution for the first subset, while the objective of the candidate solution s is to locate a solution that is close to optimal for the second. For each incoming element e, the algorithm finds "the best" position i_e in terms of the set i in k sets that return the highest value $f((e, i_e))$. If its cost is greater than $B/2$, the role of (e_m, i_m) is the best solution on the current first subset (line 5).

Otherwise, the algorithm finds the position $i'_e = \arg\max_{i \in [k]} \Delta_{(e,i)} f(\mathbf{s})$ (line 6, Algorithm 1), and adds the tuple (e, i'_e) into $\mathbf{s}$ if the condition $\Delta_{(e,i'_e)} f(\mathbf{s}) \geq c(e) f(\mathbf{s})/B$ is maintained. After the main loop completes,

Algorithm 1: Fast Approximation (FA) Algorithm

Input: V, f, k, $B > 0$.
Output: A solution $\mathbf{s}$

1: $\mathbf{s} \leftarrow \mathbf{0}$; $(e_m, i_m) \leftarrow (\emptyset, 1)$
2: **foreach** $e \in V$ **do**
3: $\quad i_e \leftarrow \arg\max_{i \in [k]} f((e, i))$
4: $\quad (e_m, i_m) \leftarrow \arg\max_{(e', i') \in \{(e_m, i_m), (e, i_e)\}} f((e', i'))$
5: $\quad$ **if** $c(e) \leq B/2$ **then**
6: $\quad\quad i'_e \leftarrow \arg\max_{i \in [k]} \Delta_{(e,i)} f(\mathbf{s})$
7: $\quad\quad$ **if** $\Delta_{(e,i'_e)} f(\mathbf{s}) \geq c(e) f(\mathbf{s})/B$ **then**
8: $\quad\quad\quad \mathbf{s} \leftarrow \mathbf{s} \sqcup (e, i'_e)$
9: $\quad\quad$ **end**
10: $\quad$ **end**
11: **end**
12: $\mathbf{s}' \leftarrow \arg\max_{\mathbf{s}_j : j \leq t, c(\mathbf{s}_j) \leq B} c(\mathbf{s}_j)$, where $t = |supp(\mathbf{s})|$ and $\mathbf{s}_j = \{(e_{t-j+1}, i_{t-j+1}), (e_{t-j+2}, i_{t-j+2}), \dots, (e_t, i_t)\}$ is the last j tuples added into $\mathbf{s}$.
13: $\mathbf{s}_{final} \leftarrow \arg\max_{\mathbf{s} \in \{(e_m, i_m), \mathbf{s}'\}} f(\mathbf{s})$
14: **return** $\mathbf{s}_{final}$

the algorithm selects a k-set $\mathbf{s}'$ as the set of last j tuples added into $\mathbf{s}$ with the maximum total cost nearest to B (line 11). Finally, the algorithm returns the final solution $\mathbf{s}_{final}$ as the best one between (e_m, i_m) and $\mathbf{s}'$. The details of the algorithm are fully presented in Algorithm 1.

At a high level, our algorithm resembles the "divide and conquer" strategy, which uses an appropriate subset division based on the costs of elements. The division of the ground set is productive because, during the linear time, the algorithm both finds the optimal solution on the first subset since feasible solutions have at most one element and finds the approximation solution on the second one. For the second subset, we were inspired by the suggestion from the idea of Kuhnle (2021) in which every element will be kept if its density gain, i.e., the *ratio between the marginal gain respect to the current solution and the cost of the element*, is greater or equal to the value of current solution over the limited budget, and the remaining is released. Their idea is powerful in diminishing the number of queries of a constant factor approximation algorithm. However, to deal with the cost and the k-submodular function, we need to make a non-trivial analysis to give an approximation ratio.

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

- $V_1 = \{e \in V : c(e) > B/2\}, V_2 = \{e \in V : c(e) \leq B/2\}$.
- $\mathbf{o}$ is an optimal solution of the problem over V and the optimal value $\mathsf{opt} = f(\mathbf{o})$.
- $\mathbf{o}'_1 = \{(e, \mathbf{o}(e)) : e \in V_1\}, \mathbf{o}'_2 = \{(e, \mathbf{o}(e)) : e \in V_2\}$.
- $\mathbf{o}_1$ is an optimal solution of the problem over V_1.
- $\mathbf{o}_2$ is an optimal solution of the problem over V_2.
- (e_j, i_j) as the jth element added of the main loop of Algorithm 1.
- $\mathbf{s} = \{(e_1, i_1), \dots, (e_t, i_t)\}$: the k-set $\mathbf{s}$ after ending the main loop, $t = |supp(\mathbf{s})|$.
- $\mathbf{s}^j = \{(e_1, i_1), \dots, (e_j, i_j)\}$: the k-set $\mathbf{s}$ (in the main loop) after adding j elements $1 \leq j \leq t$, $\mathbf{s}^0 = \mathbf{0}$, $\mathbf{s}^t = \mathbf{s}$.
- $\mathbf{s}_j = \{(e_{t-j+1}, i_{t-j+1}), (e_{t-j+2}, i_{t-j+2}), \dots, (e_t, i_t)\}$ is the set of last j elements added into $\mathbf{s}$.
- $\mathbf{o}_2^j = (\mathbf{o}_2 \sqcup \mathbf{s}^j) \sqcup \mathbf{s}^j$.
- $\mathbf{o}_2^{j-1/2} = (\mathbf{o}_2 \sqcup \mathbf{s}^j) \sqcup \mathbf{s}^{j-1}$.
- $\mathbf{s}^{j-1/2}$: If $e_j \in supp(\mathbf{o}_2)$, then $\mathbf{s}^{j-1/2} = \mathbf{s}^{j-1} \sqcup (e_j, \mathbf{o}_2(e_j))$. If $e_j \notin supp(\mathbf{o}_2)$, $\mathbf{s}^{j-1/2} = \mathbf{s}^{j-1}$.
- $\mathbf{u}^t = \{(u_1, i_1), (u_2, i_2), \dots, (u_r, i_r)\}$ is a set of elements that are in $\mathbf{o}'_2$ but not in $\mathbf{s}^t$, $r = |supp(\mathbf{u}^t)|$.
- $\mathbf{u}_l^t = \mathbf{s}^t \sqcup \{(u_1, i_1), (u_2, i_2), \dots, (u_l, i_l)\}, 1 \leq l \leq r$ and $\mathbf{u}_0^t = \mathbf{s}^t$.

Supposing that $\mathbf{s}'$ gets T last tuples in $\mathbf{s}$, i.e., $\mathbf{s}' = \mathbf{s}_T$. Denote $Q = t-T$, we have $\mathbf{s} = \mathbf{s}^Q \sqcup \mathbf{s}'$. The following Lemmas connect the candidate solution $\mathbf{s}$ with $\mathbf{o}_2$.

Lemma 1. *$f(\mathbf{o}_2) - f(\mathbf{o}_2^j) \le f(\mathbf{s}^j)$ for all $0 \le j \le t$.*

Proof. For all $0 \le j \le t$, we have:

$$f(\mathbf{o}_2) - f(\mathbf{o}_2^j) = \sum_{i=1}^{j}(f(\mathbf{o}_2^{i-1}) - f(\mathbf{o}_2^i)) \tag{6}$$

$$\le \sum_{i=1}^{j}(f(\mathbf{o}_2^{i-1}) - f(\mathbf{o}_2^{i-1/2})) \tag{7}$$

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

$$\le \sum_{i=1}^{j}(f(\mathbf{s}^{i}) - f(\mathbf{s}^{i-1})) \tag{9}$$

$$\le f(\mathbf{s}^j) \tag{10}$$

where the inequality (7) is due to the monotonicity of f, the inequality (8) is due to the k-submodularity of f, and the inequality (9) is due to the selection rule of the algorithm. The proof is completed. □

Lemma 2. *$f(\mathbf{s}') \ge f(\mathbf{s})/3$.*

Proof. If $c(\mathbf{s}) \le B$, $\mathbf{s}' = \mathbf{s}$ and the Lemma holds. Therefore, we must consider the case $c(\mathbf{s}) > B$. We get:

$$f(\mathbf{s}) - f(\mathbf{s}^Q) = \sum_{j=Q+1}^{t} \Delta_{(e_j,i_j)} f(\mathbf{s}^{j-1}) \tag{11}$$

$$\ge \sum_{j=Q+1}^{t} c(e_j)\frac{f(\mathbf{s}^{j-1})}{B} \tag{12}$$

$$\ge \sum_{j=Q+1}^{t} c(e_j)\frac{f(\mathbf{s}^{Q})}{B} \tag{13}$$

$$\ge c(\mathbf{s}')\frac{f(\mathbf{s}^{Q})}{B} \tag{14}$$

where the inequality (12) is due to the selection rule of a tuple (e, i'_e) into $\mathbf{s}$ in Line 7 of Algorithm 1 and the inequality (13) is due to the monotonicity of f.

Since $\mathbf{s}'$ is chosen from $\mathbf{s}$ so that its total cost is closest to B and each element $e \in supp(\mathbf{s})$ has the cost at most $B/2$, thus,

$$c(\mathbf{s}') > B - \frac{B}{2} \ge \frac{B}{2}.$$

It implies that $f(\mathbf{s}) - f(\mathbf{s}^Q) \ge f(\mathbf{s}^Q)/2$. Hence $f(\mathbf{s}^Q) \le 2f(\mathbf{s}^t)/3$. In the other hand, due to the k-submodularity of f we have $f(\mathbf{s}^t) \le f(\mathbf{s}^Q) + f(\mathbf{s}')$. Thus,

$$f(\mathbf{s}') \ge f(\mathbf{s}^t) - f(\mathbf{s}^Q) \ge \frac{f(\mathbf{s}^t)}{3} \tag{15}$$

The proof is completed. □

Lemma 3. *$f(\mathbf{o}_2^t) \le 2f(\mathbf{s})$.*

Proof. By the definition of $\mathbf{u}^t$, the elements in $supp(\mathbf{u}^t)$ do not pass the condition in Line 7 of Algorithm 1 and $c(\mathbf{u}^t) \le c(\mathbf{o}_2) \le B$. Denote by $\mathbf{s}^{<u_j}$ as $\mathbf{s}$ right before u_j arrives ($j \le r$), we have:

$$f(\mathbf{o}_2^t) - f(\mathbf{s}^t) = f(\mathbf{u}^t \sqcup \mathbf{s}^t) - f(\mathbf{s}^t) = \sum_{j=1}^{r}(f(\mathbf{u}_j^t) - f(\mathbf{u}_{j-1}^t))$$

$$\le \sum_{j=1}^{r}[f(\mathbf{s}^{<u_j} \sqcup (u_j, i_j)) - f(\mathbf{s}^{<u_j})] \tag{16}$$

$$\le \sum_{j=1}^{r} \Delta_{(u_j,i_j)} f(\mathbf{s}^{<u_j}) \tag{17}$$

$$< \sum_{j=1}^{r} c(u_j)\frac{f(\mathbf{s}^{<u_j})}{B} \tag{18}$$

$$\le \sum_{j=1}^{r} c(u_j)\frac{f(\mathbf{s}^{t})}{B} \tag{19}$$

$$\le c(\mathbf{u}^t)\frac{f(\mathbf{s}^t)}{B} \le f(\mathbf{s}^t) \tag{20}$$

where the inequality (16) is due to the k-submodularity of f, the inequality (17) is due to the definition of $\mathbf{s}^{<u_j}$, the inequality (18) is due to the selection of the algorithm and the inequality (19) is due to the monotonicity of f. Thus, we have: $f(\mathbf{o}_2^t) \le 2f(\mathbf{s}^t)$. □

Lemma 4. *$f(\mathbf{s}') \ge f(\mathbf{o}_2)/9$.*

Proof. Applying the Lemmas 1, 3, with $j = t$, we have

$$f(\mathbf{o}_2) - f(\mathbf{s}) = f(\mathbf{o}_2) - f(\mathbf{o}_2^t) + f(\mathbf{o}_2^t) - f(\mathbf{s})$$
$$\le f(\mathbf{s}) + f(\mathbf{o}_2^t) - f(\mathbf{s}) = f(\mathbf{o}_2^t)$$
$$\le 2f(\mathbf{s})$$

Thus $f(\mathbf{s}) \ge f(\mathbf{o}_2)/3$. Combine with Lemma 2, we have $f(\mathbf{s}') \ge f(\mathbf{o}_2)/9$. □

Theorem 1. *Algorithm 1 is a single-pass algorithm that returns an approximation ratio of $1/10$ and takes $O(nk)$ queries.*

Proof. The algorithm scans only once over the ground set, and each element e has k queries to find the position i_e. Therefore, the number of queries is nk. We now prove the approximation ratio of the algorithm. By the selection of (e_m, i_m) and the $\mathbf{o}_1$ contains at most one element so $f(\mathbf{o}_1) \le f((e_m, i_m))$. By the definition of $\mathbf{o}'_1, \mathbf{o}'_2$ and the k-submodularity of f, we obtain:

$$f(\mathbf{o}) \le f(\mathbf{o}'_1) + f(\mathbf{o}'_2) \tag{21}$$

$$\le f(\mathbf{o}_1) + f(\mathbf{o}_2) \tag{22}$$

$$\le f((e_m, i_m)) + 9f(\mathbf{s}') \le 10f(\mathbf{s}_{final}) \tag{23}$$

The proof was completed. □

4.2. Improved Fast Approximation algorithm

We next introduce the IFA algorithm, which improves the approximation ratio to $1/4 - \epsilon$ and takes $O(kn/\epsilon)$ query complexity. The key idea of IFA is to use the FA's solution to give an interval bound of opt and adapt a greedy threshold to improve the approximation ratio by conducting $O(1/\epsilon)$ times scanning over the ground set. The details of the algorithm are fully presented in Algorithm 2.

Algorithm 2: Improved Fast Approximation (IFA) Algorithm

Input: V, f, k, $B > 0$, $\epsilon > 0$.
Output: A solution $\mathbf{s}$

$\mathbf{s}_{max} \leftarrow$ Result of Algorithm 1, $\Gamma \leftarrow f(\mathbf{s}_{max})$
$S \leftarrow \{(1+\epsilon)^i : i \in \mathbb{N}, \Gamma \le (1+\epsilon)^i \le 10\Gamma\}$, $\mathbf{s}_v \leftarrow \mathbf{0}\ \forall v \in S$
for $e \in V$ **do**
  foreach $v \in S$ **do**
    $i_v \leftarrow \arg\max_{i\in[k]} \Delta_{(e,i)} f(\mathbf{s}_v)$
    $\theta_v = v/(2B)$
    if $c(\mathbf{s}_v) + c(e) \le B$ ***and*** $\Delta_{(e,i_v)} f(\mathbf{s}_v)/c(e) \ge \theta_v$ **then**
      $\mathbf{s}_v \leftarrow \mathbf{s}_v \sqcup (e, i_v)$
    end
  end
end
$\mathbf{s}_{final} \leftarrow \arg\max_{\mathbf{s}'\in\{\mathbf{s}_{max}, \mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_{|S|}\}} f(\mathbf{s}')$
return $\mathbf{s}_{final}$

Specifically, IFA takes an instance (V, f, k, B) of kSMK and an accuracy parameter $\epsilon > 0$ as inputs. IFA first calls FA as a subroutine and uses FA's solution, $\mathbf{s}_{max}$, to obtain a bound range of the optimal solution (line 1). From Theorem 1, we have $\Gamma \le \text{opt} \le 10\Gamma$.

The algorithm consists of two loops: the outer to scan each element e in the ground set V and the inner to consider each candidate solution $\mathbf{s}_v$ for each v filtered out from the set S. On the basis of Theorem 1, we construct the set S to bound the number of candidate solutions $\mathbf{s}_v$. We define (e, i_v) as the tuple that gives the largest marginal gain when added into $\mathbf{s}_v$. When an element e arrives, the algorithm handles these works: (1) chooses the position i_v with maximal marginal gain with respect to $\mathbf{s}_v$ and e (line (5)); (2) uses threshold $\theta_v = v/(2B)$ to add the element e into $\mathbf{s}_v$ if it has the high ratio of marginal gain over its cost (i.e. "*density gain*") without violating the budget constraint (line (7)). We define the notations regarding to Algorithm 2 as follows:

- $\mathbf{s}_v = \{(e_1, i_1), (e_2, i_2), \dots, (e_q, i_q)\}$ is the candidate solution with respect to some elements $v \in S$ after ending the outer loop.
- $\mathbf{s}_v^j = \{(e_1, i_1), (e_2, i_2), \dots, (e_j, i_j)\}, 1 \le j \le q$ and $\mathbf{s}_v^0 = \mathbf{0}$.
- $\mathbf{s}_v^{<e}$ as $\mathbf{s}_v$ immediately before e is processed.
- $\mathbf{u} = \{(u_1, i_1), (u_2, i_2), \dots, (u_r, i_r)\}$ as a set of elements that are in $\mathbf{o}$ but not in $\mathbf{s}_v$, $r = |supp(\mathbf{u})|$.
- $\mathbf{u}_l = \mathbf{s}_v \sqcup \{(u_1, i_1), (u_2, i_2), \dots, (u_l, i_l)\}, \forall 1 \le l \le r$ and $\mathbf{u}_0 = \mathbf{s}_v$.
- $\mathbf{o}^j = (\mathbf{o} \sqcup \mathbf{s}^j) \sqcup \mathbf{s}^j$.
- $\mathbf{o}^{j-1/2} = (\mathbf{o} \sqcup \mathbf{s}^j) \sqcup \mathbf{s}^{j-1}$.
- $\mathbf{s}^{j-1/2}$: If $e_j \in supp(\mathbf{o})$, then $\mathbf{s}^{j-1/2} = \mathbf{s}^{j-1} \sqcup (e_j, \mathbf{o}(e_j))$. If $e_j \notin supp(\mathbf{o})$, $\mathbf{s}^{j-1/2} = \mathbf{s}^{j-1}$.

Lemma 5. *For any $v \in S$, if there is no element $o \in supp(\mathbf{o}) \setminus supp(\mathbf{s}_v)$ so that $\Delta_{(o,\mathbf{o}(o))} f(\mathbf{s}_v^{<o})/c(o) \ge \theta_v$ and $c(\mathbf{s}_v^{<o}) + c(o) > B$, we have: $f(\mathbf{o}) \le 2f(\mathbf{s}_v) + c(\mathbf{o})\theta_v$.*

Proof. Due to the same selection rule between (e, i_v) of Algorithm 2 and (e, i'_e) of Algorithm 1, we have the same result with Lemma 1, i.e., $f(\mathbf{o}) - f(\mathbf{o}^q) \le f(\mathbf{s}_v)$ and thus:

$$f(\mathbf{o}) - f(\mathbf{s}_v) = f(\mathbf{o}) - f(\mathbf{o}^q) + f(\mathbf{o}^q) - f(\mathbf{s}_v) \tag{24}$$
$$\le f(\mathbf{s}_v) + \sum_{j=1}^{r} (f(\mathbf{u}_j) - f(\mathbf{u}_{j-1})) \tag{25}$$
$$\le f(\mathbf{s}_v) + \sum_{j=1}^{r} \Delta_{(u_j, i_j)} f(\mathbf{s}_v^{<u_j}) \tag{26}$$
$$\le f(\mathbf{s}_v) + \sum_{j=1}^{r} c(u_j)\theta_v \tag{27}$$
$$\le f(\mathbf{s}_v) + c(\mathbf{o})\theta_v \tag{28}$$

where the inequality (26) is due to the k-submodularity, the inequality (27) is due to the definition of $\mathbf{s}_v^{<o}$, and the inequality (28) is due to the definition of $\mathbf{u}$ and $\mathbf{o}$. Thus, the proof is completed. □

Theorem 2. *For $\epsilon \in (0, 1/4)$, Algorithm 2 returns an approximation ratio of $1/4 - \epsilon$, within $O(nk/\epsilon)$ queries.*

Proof. The algorithm needs nk queries to call FA and uses only 1-pass over the ground set for completing the outer loop (Line 3–11). For each incoming element, it takes at most $k \cdot \lceil \log_{(1+\epsilon)}(10) \rceil$ queries for updating $\mathbf{s}_v, v \in S$. Combine all tasks, the required number of queries is at most:

$$nk + nk\lceil \log_{(1+\epsilon)}(10) \rceil \le nk + nk(1 + \log_{(1+\epsilon)}(10)) = 2nk + nk\frac{\ln(10)}{\ln(1+\epsilon)}$$
$$\le 2nk + nk\frac{\ln(10)}{\ln\frac{1}{1-\frac{\epsilon}{2}}} = 2nk - nk\frac{\ln(10)}{\ln(1-\frac{\epsilon}{2})}$$
$$\le 2nk + \frac{2}{\epsilon}nk\ln(10) = O(\frac{nk}{\epsilon})$$

where the second inequality is due to $1 + \epsilon \ge \frac{1}{1-\frac{\epsilon}{2}}$, for $\epsilon \in (0, 1)$ and the third inequality is due to $\ln(x+1) \ge x$, for all $x \in (-1, 0)$. We now show the approximation ratio of the algorithm. By Theorem 1, we have $\Gamma \le \text{opt} \le 10\Gamma$. Therefore, there exists an integer number $v \in S$ so that $\text{opt}/(1+\epsilon) \le v < \text{opt}$. We have:

$$f(\mathbf{s}_v^j) = \sum_{i=1}^{j} (f(\mathbf{s}_v^i) - f(\mathbf{s}_v^{i-1})) \ge \sum_{i=1}^{j} c(e_i)\theta_v = c(\mathbf{s}_v^j)\theta_v. \tag{29}$$

We consider the following cases:

Case 1. There exists an element $o \in supp(\mathbf{o}) \setminus supp(\mathbf{s}_v)$ so that $\Delta_{(o,\mathbf{o}(o))} f(\mathbf{s}_v^{<o}) \ge c(o) \cdot \theta_v$ and $c(\mathbf{s}_v^{<o}) + c(o) > B$. Recall $(e_m, i_m) = \arg\max_{e \in V, i \in [k]} f((e, i))$, we have:

$$f(\mathbf{s}_{final}) \ge \max\{f(\mathbf{s}_v), f((e_m, i_m))\} \tag{30}$$
$$\ge \max\{f(\mathbf{s}_v^{<o}), f((o, \mathbf{o}(o)))\} \tag{31}$$
$$\ge \frac{f(\mathbf{s}_v^{<o}) + f((o, \mathbf{o}(o)))}{2} \tag{32}$$
$$\ge \frac{f(\mathbf{s}_v^{<o}) \sqcup (o, \mathbf{o}(o))}{2} \tag{33}$$
$$= \frac{\Delta_{(o,\mathbf{o}(o))} f(\mathbf{s}_v^{<o}) + f(\mathbf{s}_v^{<o})}{2} \tag{34}$$
$$\ge \frac{\theta_v c(o) + \theta_v c(\mathbf{s}_v^{<o})}{2} \ge \frac{B\theta_v}{2} \ge \frac{\text{opt}}{4(1+\epsilon)} \tag{35}$$
$$\ge (\frac{1}{4} - \epsilon)\text{opt}. \tag{36}$$

Case 2. There is no such an element o like Case 1. By Lemma 5, we have

$$f(\mathbf{o}) \le 2f(\mathbf{s}_v) + B\theta_v \le 2f(\mathbf{s}_v) + \frac{\text{opt}}{2}. \tag{37}$$

Hence $f(\mathbf{s}_{final}) \ge f(\mathbf{s}_v) \ge \text{opt}/4$. By combining the two above cases, we obtain the proof. □

4.3. Improved Fast Approximation Plus algorithm

To increase the approximation ratio of IFA, we expand Algorithm 2 to Algorithm 3, called Improved Fast Approximation Plus (IFA+). IFA+ provides a deterministic $1/3 - \epsilon$ approximation ratio.

The IFA+ algorithm consists of two phases. The first phase (lines 1–11) re-uses the algorithmic framework of IFA with some modifications. In this phase, the algorithm calls FA as a subroutine to get an approximate range of the optimal value $[\Gamma, 10\Gamma]$ (line 1). It then adapts the greedy threshold to add elements with high density gain (i.e., the value of marginal gain over its cost of an element) into the candidate solution $\mathbf{s}$. Specifically, this phase consists of multiple iterations; each scans one time over the ground set (lines 3–9). For an element e, the algorithm finds the best position $i_e = \arg\max_{i \in [k]} \Delta_{(e,i)} f(\mathbf{s})$ and e is added to the set $\mathbf{s}$ if its density gain $\Delta_{(e,i)} f(\mathbf{s})/c(e)$ is at least θ without violating the budget constraint. The threshold θ initiates to $10\Gamma/(3\epsilon B)$ and decreases by a factor of $(1-\epsilon)$ after each iteration until less than to $\Gamma(1-\epsilon)/(3B)$.

The second phase (lines 13–21) is to improve the approximation ratio of $\mathbf{s}$. The core idea of this phase is based on the exploiting that the quality of $\mathbf{s}$ can be raised by improving $\mathbf{s}^T$, where $\mathbf{s}^j = \{(e_1, i_1), \dots, (e_j, i_j)\}$ is the k-set $\mathbf{s}$ after adding j elements $1 \le j \le t$, $\mathbf{s}^0 = \mathbf{0}$ and $T = \max\{j \in \mathbb{N} : \mathbf{s}^j + c(o_1) \le B\}$.

To find $\mathbf{s}^T$, we find multiple guests $\mathbf{s}^q$ with the total budget at most l (line (14)) which gradually increases from ϵB to $(1+\epsilon)B$. It then re-scans the ground set V to add a new best element e_l from V to $\mathbf{s}'_l$ without violating the budget constraint (line (18)).

Finally, the final solution will choose the best among $\mathbf{s}_{max}, \mathbf{s}$, and the list of $\mathbf{s}_{(j)}$ with $j \in [l]$ (line 22). The details of the algorithm are fully presented in Algorithm 3.

We define the notations regarding to Algorithm 3 as follows:

- $\mathbf{s} = \{(e_1, i_1), \dots, (e_t, i_t)\}$ the k-set $\mathbf{s}$ after ending the first loop, $t = |supp(\mathbf{s})|$.

Algorithm 3: Improved Fast Approximation Plus (IFA+) Algorithm

Input: V, f, k, $B > 0$, $\epsilon > 0$.
Output: A solution $\mathbf{s}$
// Phase 1: Finding candidate solutions
1: $\mathbf{s}_{max} \leftarrow$ Result of Algorithm 1. , $\mathbf{s} \leftarrow \mathbf{0}$
2: $\Gamma \leftarrow f(\mathbf{s}_{max})$, $\theta \leftarrow 10\Gamma/(3\epsilon B)$
3: **while** $\theta \geq (1-\epsilon)\Gamma/(3B)$ **do**
4: **for** $e \in V \setminus supp(\mathbf{s})$ **do**
5: $i_e \leftarrow \arg\max_{i\in[k]} \Delta_{(e,i)} f(\mathbf{s})$
6: **if** $c(\mathbf{s}) + c(e) \leq B$ ***and*** $\Delta_{(e,i_e)} f(\mathbf{s})/c(e) \geq \theta$ **then**
7: $\mathbf{s} \leftarrow \mathbf{s} \sqcup (e, i_e)$
8: **end**
9: **end**
10: $\theta \leftarrow (1-\epsilon)\theta$
11: **end**
// Phase 2: Boosting quality of solutions
12: $l \leftarrow \epsilon B$
13: **while** $l \leq B$ **do**
14: Find $q \leftarrow \max\{i : i \leq |supp(\mathbf{s})|, c(\mathbf{s}^i) \leq l\}$
15: $\mathbf{s}'_l \leftarrow \mathbf{s}^q$
16: **if** $\mathbf{s}'_l \neq \mathbf{s}_{(l/(1+\epsilon))}$ **then**
17: $(e_l, i_l) \leftarrow \arg\max_{i\in[k], e\in V\setminus supp(\mathbf{s}'_l): c(\mathbf{s}'_l)+c(e)\leq B} \Delta_{(e,i)} f(\mathbf{s}'_l)$
18: $\mathbf{s}_{(l)} \leftarrow \mathbf{s}'_l \sqcup (e_l, i_l)$
19: **end**
20: $l \leftarrow (1+\epsilon)l$
21: **end**
22: $\mathbf{s} \leftarrow \arg\max_{\mathbf{s}''\in\{\mathbf{s}_{max}, \mathbf{s}, \mathbf{s}_{(\epsilon B)}, \mathbf{s}_{(\epsilon B(1+\epsilon))}, \ldots, \mathbf{s}_{(l)}\}} f(\mathbf{s}'')$
23: **return** $\mathbf{s}$

- $\mathbf{s}^j = \{(e_1, i_1), \ldots, (e_j, i_j)\}$: the k-set $\mathbf{s}$ after adding j elements $1 \leq j \leq t$, $\mathbf{s}^0 = \mathbf{0}$, $\mathbf{s}^t = \mathbf{s}$.
- $\mathbf{s}_j$: the k-set $\mathbf{s}$ after ending the iteration j of the first loop.
- $\mathbf{s}_j^i$: the k-set contains first i elements in $\mathbf{s}_j$.
- $T = \max\{j \in \mathbb{N} : \mathbf{s}^j + c(o_1) \leq B\}$, where $o_1 = \max_{o\in supp(\mathbf{o})} c(o)$.
- $\mathbf{o}^j = (\mathbf{o} \sqcup \mathbf{s}^j) \sqcup \mathbf{s}^j$.
- $\mathbf{o}' = \{(o_2, i_2^*), \ldots, (o_m, i_m^*)\}$, i.e., the optimal solution after picking (o_1, i_1^*).
- θ_j: the threshold θ at the jth-iteration of the first loop.
- $\theta_{(j)}$: the threshold θ when the tuple (e_j, i_j) is added into $\mathbf{s}$ in the first loop.

We first introduce the following simple but vital Lemma that plays an important role in analyzing the approximation ratio.

Lemma 6. *For any k-set $\mathbf{x}$ we have:*

$$f(\mathbf{x}) \leq 2f(\mathbf{s}_j) + \sum_{e\in supp(\mathbf{x})\setminus supp(\mathbf{s}_j)} \Delta_{(e,\mathbf{x}(e))} f(\mathbf{s}_j).$$

If $\max_{e\in supp(\mathbf{x})\setminus supp(\mathbf{s}_j)} c(e) + c(\mathbf{s}_j) \leq B$, we further have:

$$f(\mathbf{x}) < 2f(\mathbf{s}_j) + c(\mathbf{x})\theta_j.$$

Proof. By the similar reasoning with Lemma 5, we also get:

$$f(\mathbf{x}) - f(\mathbf{s}_j) = f(\mathbf{x}) - f(\mathbf{x}^i) + f(\mathbf{x}^i) - f(\mathbf{s}_j) \tag{38}$$
$$\leq f(\mathbf{s}_j) + \sum_{e\in supp(\mathbf{x})\setminus supp(\mathbf{s}_j)} \Delta_{(e,\mathbf{x}(e))} f(\mathbf{s}_j) \tag{39}$$

where $\mathbf{x}^i = (\mathbf{x} \sqcup \mathbf{s}_j^i) \sqcup \mathbf{s}_j^i$. For any element $e \notin supp(\mathbf{s}_j)$ without violating the total budget constraint, i.e., $c(e) + c(\mathbf{s}_j) \leq B$, it did not pass the condition in line 6 of Algorithm 3 at the iteration j of the first loop, where $\mathbf{s}^{<e}$ as $\mathbf{s}$ immediately before e is processed. This implies

$$\frac{\Delta_{(e,\mathbf{x}(e))} f(\mathbf{s}_j)}{c(e)} \leq \frac{\Delta_{(e,\mathbf{x}(e))} f(\mathbf{s}^{<e})}{c(e)} < \theta_j.$$

Therefore if $\max_{e\in supp(\mathbf{x})\setminus supp(\mathbf{s}_j)} c(e) + c(\mathbf{s}_j) \leq B$, then $\Delta_{(e,\mathbf{x}(e))} f(\mathbf{s}_j) < c(e)\theta_j, \forall e \in supp(\mathbf{x}) \setminus supp(\mathbf{s}_j)$. Thus

$$f(\mathbf{x}) - f(\mathbf{s}_j) < f(\mathbf{s}_j) + \sum_{e\in supp(\mathbf{x})\setminus supp(\mathbf{s}_j)} c(e)\theta_j \leq f(\mathbf{s}_j) + c(\mathbf{x})\theta_j \tag{40}$$

which implies the proof. □

Lemma 7. *If $c(o_1) > (1-\epsilon)B$, $f(\mathbf{s}) \geq (\frac{1}{3} - \epsilon)\mathsf{opt}$.*

Proof. By the definition of $\mathbf{o}'$, we have $c(\mathbf{o}') \leq B - c(o_1) \leq \epsilon B$. The threshold θ is decreasing from $\frac{10\Gamma}{3\epsilon B}$ to $\frac{(1-\epsilon)\Gamma}{3B}$ by a factor of $1-\epsilon$ after each iteration of the first loop. Therefore, the number of iterations of the first loop is at most

$$\lceil \log_{(1-\epsilon)}(\frac{\epsilon(1-\epsilon)}{10}) \rceil \leq 2 + \frac{\ln(\epsilon/10)}{\ln(1-\epsilon)} = 2 - \frac{\ln(10/\epsilon)}{\ln(1-\epsilon)} \leq 2 + \frac{\ln(10/\epsilon)}{\epsilon} \tag{41}$$

where the first inequality is due to $x \geq \ln(1+x)$, for all $x \in (-1, 0)$. Consider the iteration $j = \lceil \log_{(1-\epsilon)}(\frac{\epsilon\mathsf{opt}}{10\Gamma}) \rceil$ we have:

$$\frac{(1-\epsilon)\mathsf{opt}}{3B} < \theta_j = \frac{10(1-\epsilon)^j\Gamma}{3\epsilon B} \leq \frac{\mathsf{opt}}{3B}$$

We consider the moment after ending the iteration j and divide the proof into the following cases:

Case 1: If $supp(\mathbf{s}^T) \subseteq supp(\mathbf{s}_j)$, i.e., the algorithm obtains $\mathbf{s}^T$ before ending iteration j.

- If $c(\mathbf{s}_j) < (1-\epsilon)B$. By the k-submodularity of f, $f(\mathbf{o}') \geq f(\mathbf{o}) - f((o_1, i_1^*))$. For any element $e \in supp(\mathbf{o}')\setminus supp(\mathbf{s}_j)$, we have $c(\mathbf{s}_j) + \max_{e\in supp(\mathbf{o}')} c(e) \leq c(\mathbf{s}_j) + c(\mathbf{o}') < B$. By applying Lemma 6, we have $f(\mathbf{o}') \leq 2f(\mathbf{s}_j) + c(\mathbf{o}')\theta_j = 2f(\mathbf{s}_j) + \epsilon\mathsf{opt}/3$. Therefore,

$$f(\mathbf{o}) - f(\mathbf{s}) \leq f(\mathbf{o}) - f((o_1, i_1^*)) \tag{42}$$
$$\leq f(\mathbf{o}') \tag{43}$$
$$\leq 2f(\mathbf{s}_j) + \frac{\epsilon}{3}\mathsf{opt} \tag{44}$$
$$\leq 2f(\mathbf{s}) + \frac{\epsilon}{3}\mathsf{opt} \tag{45}$$

where the inequalities (42), (45) are due to the selection rule of the final solution, and the inequality (43) is due to the k-submodularity of f. Hence $f(\mathbf{s}) \geq (\frac{1}{3} - \frac{\epsilon}{9})\mathsf{opt}$.

If $c(\mathbf{s}_j) \geq (1-\epsilon)B$, we have:

$$f(\mathbf{s}_j) \geq c(\mathbf{s}_j)\theta_j = \frac{(1-\epsilon)^2\mathsf{opt}}{3} > (\frac{1}{3} - \epsilon)\mathsf{opt}$$

Case 2: If $supp(\mathbf{s}_j) \subset supp(\mathbf{s}^T)$. From the definition of T, we apply Lemma 6 with notice that $\theta_{(T)} \leq \theta_j$, we get:

$$f(\mathbf{o}) - f(\mathbf{s}^T) \leq f(\mathbf{s}^T) + c(\mathbf{o})\theta_{(T)} \tag{46}$$
$$\leq f(\mathbf{s}^T) + c(\mathbf{o})\theta_j \tag{47}$$
$$\leq f(\mathbf{s}^T) + \frac{\mathsf{opt}}{3} \tag{48}$$

which implies that $f(\mathbf{s}) \geq f(\mathbf{s}^T) \geq \mathsf{opt}/3$. This completes the proof. □

Lemma 8. *If $c(o_1) \leq (1-\epsilon)B$, $f(\mathbf{s}) \geq (\frac{1}{3} - \epsilon)\mathsf{opt}$.*

Proof. If $\mathbf{s}^T = \mathbf{s}$ after ending the main loop, by Lemma 6 we easily prove that

$$f(\mathbf{o}) - f(\mathbf{s}^T) \leq f(\mathbf{s}^T) + c(\mathbf{o})\frac{\Gamma(1-\epsilon)}{3B} \tag{49}$$
$$\leq f(\mathbf{s}^T) + \frac{(1-\epsilon)}{3}\mathsf{opt}. \tag{50}$$

Thus $f(\mathbf{s}^T) > (\frac{1}{3} + \frac{\epsilon}{6})\mathsf{opt}$. We consider the remaining case when $\mathbf{s}^T \neq \mathbf{s}$ after ending the main loop. In this case $\mathbf{s}$ contains at least $T+1$ elements and $c(\mathbf{s}^{T+1}) + c(o_1) \geq B$, implying $c(\mathbf{s}^{T+1}) \geq \epsilon B$. Consider the iteration $j = \lceil \log_{1-\epsilon}(\frac{\mathsf{opt}}{10I}) \rceil$ of the first loop, we have:

$$\frac{(1-\epsilon)\mathsf{opt}}{3\epsilon B} < \theta_j = \frac{10(1-\epsilon)^j \Gamma}{3\epsilon B} \leq \frac{\mathsf{opt}}{3\epsilon B}.$$

We now consider the second loop of the algorithm. Since l increases from ϵB to B by a factor of $(1+\epsilon)$ after each iteration, at the iteration $h \geq 1$, we have $l = \epsilon B(1+\epsilon)^{h-1}$. Since $B - c(o_1) \geq \epsilon B$, there exists an iteration h that

$$l = \epsilon B(1+\epsilon)^{h-1} \leq B - c(o_1) < \epsilon B(1+\epsilon)^h = l(1+\epsilon) \tag{51}$$

After that iteration, by the selection rule of $\mathbf{s}^q$ of the algorithm we have $c(\mathbf{s}^q) \leq l < c(\mathbf{s}^{q+1})$.

We consider two following cases:

Case 1. If the algorithm obtains $\mathbf{s}^{q+1}$ after the first iteration of the first loop, we have:

$$f(\mathbf{s}) \geq f(\mathbf{s}^{q+1}) \geq c(\mathbf{s}^{q+1})\theta_1 > \frac{B - c(o_1)}{1+\epsilon}\frac{\mathsf{opt}}{3\epsilon B} \geq \frac{\mathsf{opt}}{3(1+\epsilon)} \geq \frac{1}{3}(1-\epsilon)\mathsf{opt}. \tag{52}$$

Case 2. If the algorithm obtains $\mathbf{s}^{q+1}$ after the iteration $j > 1$ of the first loop. We also define $\mathbf{s}^{<e}$ as $\mathbf{s}$ immediately before e is processed. The density of any element $e \in supp(\mathbf{o}) \setminus supp(\mathbf{s}^q)$ with respect to $\mathbf{s}^q$ is less than the threshold at the previous iteration when (e_{q+1}, i_{q+1}) is added to $\mathbf{s}^q$, i.e.,

$$\frac{\Delta_{(e,i_e)} f(\mathbf{s}^q)}{c(e)} \leq \frac{\theta_{(q+1)}}{1-\epsilon}. \tag{53}$$

- If $o_1 \notin supp(\mathbf{s}^q)$, by the selection rule of $\mathbf{s}_{(l)}$ we have $\mathbf{s}_{(l)} = \mathbf{s}'_l \sqcup (e_l, i_l) = \mathbf{s}^q \sqcup (e_l, i_l)$. Recap that $\mathbf{o}^q = (\mathbf{o} \sqcup \mathbf{s}^q) \sqcup \mathbf{s}^q$ with a note that $o_1 \in supp(\mathbf{o}^q)$. By similar the reasoning of Lemma 1, we also have $f(\mathbf{o}) - f(\mathbf{o}^q) \leq f(\mathbf{s}^q)$. Therefore:

$$\begin{aligned}
f(\mathbf{o}) - f(\mathbf{s}^q \sqcup (o_1, i_1^*)) &\leq f(\mathbf{o}) - f(\mathbf{o}^q) + f(\mathbf{o}^q) - f(\mathbf{s}^q \sqcup (o_1, i_1^*)) && (54)\\
&\leq f(\mathbf{s}^q) + f(\mathbf{o}^q) - f(\mathbf{s}^q \sqcup (o_1, i_1^*)) && (55)\\
&\leq f(\mathbf{s}^q) + \sum_{e \in supp(\mathbf{o}) \setminus supp(\mathbf{s}^q \sqcup (o_1, i_1^*))} \Delta_{(e,\mathbf{o}(e))} f(\mathbf{s}^q \sqcup (o_1, i_1^*)) && (56)\\
&\leq f(\mathbf{s}^q) + \sum_{e \in supp(\mathbf{o}) \setminus supp(\mathbf{s}^q \sqcup (o_1, i_1^*))} \Delta_{(e,\mathbf{o}(e))} f(\mathbf{s}^q) && (57)\\
&\leq f(\mathbf{s}^q) + \sum_{e \in supp(\mathbf{o}) \setminus supp(\mathbf{s}^q \sqcup (o_1, i_1^*))} c(e)\frac{\theta_{(q+1)}}{1-\epsilon} && (58)\\
&\leq f(\mathbf{s}^q) + \frac{(B - c(o_1))\theta_{(q+1)}}{1-\epsilon} && (59)
\end{aligned}$$

where inequality (57) is due to the k-submodularity of f and inequality (58) is due to applying (53). By applying the inequality (59) and the selection rule of the final solution ($f(\mathbf{s}) \geq f(\mathbf{s}_{(l)}) \geq f(\mathbf{s}^q)$), we have:

$$\begin{aligned}
f(\mathbf{o}) - f(\mathbf{s}_{(l)}) &= f(\mathbf{o}) - f(\mathbf{s}^q \sqcup (e_l, i_l)) && (60)\\
&\leq f(\mathbf{o}) - f(\mathbf{s}^q \sqcup (o_1, i_1^*)) \text{ (Due to the selection rule of } \mathbf{s}_{(l)}) && (61)\\
&\leq f(\mathbf{s}^q) + \frac{(B - c(o_1))\theta_{(q+1)}}{1-\epsilon} && (62)\\
&\leq f(\mathbf{s}) + \frac{(B - c(o_1))\theta_{(q+1)}}{1-\epsilon}. && (63)
\end{aligned}$$

By re-arrange inequality (63), we obtain:

$$\begin{aligned}
\theta_{(q+1)} &\geq (1-\epsilon)\frac{f(\mathbf{o}) - f(\mathbf{s}_{(l)}) - f(\mathbf{s})}{B - c(o_1)} && (64)\\
&\geq (1-\epsilon)\frac{f(\mathbf{o}) - 2f(\mathbf{s})}{B - c(o_1)}. && (65)
\end{aligned}$$

On the other hand, at the first loop, the density gain of each additional element is at least the threshold, i.e., $\Delta_{(e_j,i_j)} f(\mathbf{s}^{j-1})/c(e_j) \geq \theta_{(j)}$ for all $j = 1, \ldots, q+1$. Thus,

$$\begin{aligned}
f(\mathbf{s}) \geq f(\mathbf{s}^{q+1}) - f(\mathbf{s}^0) &= \sum_{j=1}^{q+1} \Delta_{(e_j,i_j)} f(\mathbf{s}^{j-1}) && (66)\\
&\geq \sum_{j=1}^{q+1} c(e_j)\theta_{(j)} \geq c(\mathbf{s}^{q+1})\theta_{(q+1)} && (67)\\
&\geq \frac{B - c(o_1)}{1+\epsilon}\theta_{(q+1)} \text{ (Due to (53))} && (68)\\
&\geq \frac{1-\epsilon}{1+\epsilon}(f(\mathbf{o}) - 2f(\mathbf{s})). && (69)
\end{aligned}$$

Hence $f(\mathbf{s}) \geq (\frac{1}{3} - \frac{2\epsilon}{3(3-\epsilon)})f(\mathbf{o}) > (\frac{1}{3} - \epsilon)\mathsf{opt}$.

- If $o_1 \in supp(\mathbf{s}^q)$, $c(\mathbf{o}) - c(\mathbf{s}^{q+1}) \leq B - c(o_1)$. We also obtain (63) by the following transforms:

$$\begin{aligned}
f(\mathbf{o}) - f(\mathbf{s}_{(l)}) &\leq f(\mathbf{o}) - f(\mathbf{s}^{q+1}) && (70)\\
&\leq f(\mathbf{s}^{q+1}) + \sum_{e \in supp(\mathbf{o}) \setminus supp(\mathbf{s}^{q+1})} \Delta_{(e,\mathbf{o}(e))}\\
&\quad \times f(\mathbf{s}^{<e}) \text{ (By applying Lemma 6)} && (71)\\
&\leq f(\mathbf{s}^{q+1}) + \sum_{e \in supp(\mathbf{o}) \setminus supp(\mathbf{s}^{q+1})} c(e)\frac{\theta_{(q+1)}}{1-\epsilon} && (72)\\
&\leq f(\mathbf{s}) + \frac{\theta_{(q+1)}(B - c(o_1))}{1-\epsilon}. && (73)
\end{aligned}$$

From now on, by the similar arguments of the case $o_1 \notin supp(\mathbf{s}^q)$, we also get $f(\mathbf{s}) > (1/3 - \epsilon)\mathsf{opt}$. Combining all the above cases, we get the proof. □

Theorem 3. *For any $\epsilon \in (0, 1/3)$, Algorithm 3 has $O(kn\log(1/\epsilon)/\epsilon)$ query complexity and returns an approximation ratio $1/3 - \epsilon$.*

Proof. The Algorithm 3 requires $O(nk)$ queries to run Algorithm 1, FA. Then, it consists of two loops. From (41), the first loop consists at most: $O(\frac{1}{\epsilon}\log(\frac{1}{\epsilon}))$ iterations. The number of iterations of the second loop is at most:

$$\begin{aligned}
\lceil \log_{(1+\epsilon)}(\frac{1}{\epsilon}) \rceil &\leq 1 + \log_{(1+\epsilon)}(\frac{1}{\epsilon}) = 1 + \frac{\ln(\frac{1}{\epsilon})}{\ln(1+\epsilon)} \leq 1 + \frac{\ln(\frac{1}{\epsilon})}{\ln\frac{1}{1-\frac{\epsilon}{2}}}\\
&= 1 - \frac{\ln(\frac{1}{\epsilon})}{\ln(1-\frac{\epsilon}{2})} \leq 1 + \frac{2}{\epsilon}\ln(\frac{1}{\epsilon}) = O(\frac{1}{\epsilon}\log(\frac{1}{\epsilon}))
\end{aligned}$$

where the first inequality is due to $1 + \epsilon \geq \frac{1}{1-\frac{\epsilon}{2}}$, for $\epsilon \in (0,1)$ and the second inequality is due to applying $\ln(1+x) \leq x$ for all $x \in (-1, 0)$. Since each iteration of the above loops scans one time over the data and takes kn queries, we obtain the number of queries at most:

$$O(nk) + 2 \cdot O(\frac{1}{\epsilon}\log(\frac{1}{\epsilon})) \cdot kn = O(\frac{nk}{\epsilon}\log(\frac{1}{\epsilon})). \tag{74}$$

The approximation ratio of the algorithm comes from the Lemmas 7, 8 by combining two cases: $c(o_1) \leq (1-\epsilon)B$ and $c(o_1) > (1-\epsilon)B$. We complete the proof. □

5. Experiments

In this section, we compare the performance between our algorithms and state-of-the-art algorithms for the kSMK problem listed below:

- **Greedy**: the $(1/2 - 1/(2e))$-approximation algorithm within $O(n^4k^3)$ in Tang et al. (2022).

Table 2
The dataset

Database	#Nodes	#Edges	Types
Facebook (Leskovec and Krevl, 2014)	4039	88 234	Directed
Hept (Chen et al., 2009)	15 233	58 894	Directed
Enron (Klimt and Yang, 2004)	36 692	367 662	Directed
Intel Lab sensors (Bodik et al., 2004)	56	–	–
DF-AMS WSN (Huy and Viet, 2015)	100	–	–

- **Deterministic Streaming (DS)**[1]: A streaming algorithm in Pham et al. (2022b) which returns an approximation ratio of $1/4 - \epsilon$, requires one pass and $O(kn\log(n)/\epsilon)$ queries.
- **Random Streaming (RS)**: Another streaming algorithm in Pham et al. (2022b) which returns an approximation ratio of $k/(4k-1)-\epsilon$ in expectation, requires one pass and $O(kn\log(n)/\epsilon)$ queries.

We conduct experiments on specific applications, which are *k*-**topic Influence Maximization under knapsack constraint** (**kIMK**), *k*-**topic information Coverage Maximization under Knapsack constraint** (**kCMK**), and *k*-**type Sensor Placement under Knapsack constraint** (**kSPK**) on three important measurements: the oracle value of the objective function, the number of queries, and running time. We further show the trade-off between the solution quality and the number of queries of algorithms with various settings of budget B.

We also use the dataset as mentioned in Chen et al. (2009), Nguyen et al. (2017), Mitrovic et al. (2016), Pham et al. (2022b) and Huy and Viet (2015) to illustrate the performance of compared algorithms (Table 2). To demonstrate the performance of algorithms via the above three measurements, we show some figures numbered and captioned, in which the terms Fig, K, and M stand for Figure, thousands, and millions, respectively.

All the implementations are on a Linux machine with configurations of 2× Intel Xeon Silver 4216 Processor @2.10 GHz and 16 threads × 256 GB DIMM ECC DDR4 @2666 MHz.

5.1. *k-topic Influence Maximization under Knapsack constraint (*kIMK*)*

The information diffusion model, called k-Linear Threshold (k-LT) model (Kempe et al., 2003; Nguyen and Thai, 2020) was briefed, and the k-topic Influence Maximization under Knapsack constraint (kIMK) problem using this model was defined as follows:

k-LT model. A social network is modeled by a directed graph $G = (V, E)$, where V, E represent sets of users and links, respectively. Each edge $(u, v) \in E$ is assigned weights $\{w^i(u,v)\}_{i\in[k]}$, where each $w^i(u,v)$ represents how powerful u influences to v on the ith topic. Each node $u \in V$ has a *influence threshold* with topic i, denoted by $\theta^i(u)$, which is chosen uniformly at random in $[0, 1]$. Given a seed set $\mathbf{s} = (S_1, S_2, \ldots, S_k) \in (k+1)^V$, the information propagation for topic i happens in discrete steps $t = 0, 1, \ldots$ as follows. At step $t = 0$, all nodes in S_i become active by topic i. At step $t \geq 1$, a node u becomes active if $\sum_{\text{actived node } v} w^i(v,u) \geq \theta^i(u)$.

The information diffusion process on topic i ends at step t if there is no new active node and the diffusion process of a topic is independent of the others. Denote by $\sigma(\mathbf{s})$ the number of nodes that becomes active in at least one of k topics after the diffusion process of a seed k-set $\mathbf{s}$, i.e.,

$$\sigma(\mathbf{s}) = \mathbb{E}[|\cup_{i\in[k]} \sigma_i(S_i)|] \tag{75}$$

where $\sigma_i(S_i)$ is a random variable representing the set of active users for topic i with the seed S_i.

[1] The kSMK problem is a special case of the k-submodular maximization under the budget constraint in Pham et al. (2022b) with $\beta = 1$.

The kIMK *problem.* The problem is formally defined as follows:

Definition 1 (kIMK *Problem*)**.** Assuming that each user e has a cost $c(e) > 0$ for every ith topic, which manifests how hard it is to influence the respective person for that topic initially. Given a positive integer-valued budget B, the problem asks to find a seed set $\mathbf{s}$ with $c(\mathbf{s}) = \sum_{e\in S_i, i\in k} c(e) \leq B$ so that $\sigma(\mathbf{s})$ maximal.

5.2. *k-topic information Coverage Maximization under Knapsack constraint*

The problem of k-topic information Coverage Maximization was proposed by Wang et al. (2015) when considering that an inactive node can be informed of information by at least one of its neighbors in the same diffusion model of the information influence problem. *Information coverage maximization* is to maximize the expected number of active and informed nodes. Hence, an inactive node is still informed if it has at least one active neighbor.

Authors Qian et al. (2018b) showed that if $v \in \sigma_i(S_i)\ \forall i \in [k]$, $N(v)$ is denoted the set of inactive neighbors of v, the set of active nodes and informed nodes by propagating from S_i could be:

$$\gamma_i(S_i) = \sigma_i(S_i) \cup \left(\cup_{v\in\sigma_i(S_i)} N(v)\right)$$

when $\sigma_i(S_i)$ was defined in Eq. (75). Hence, information coverage with k types of topics, as shown in Eq. (76), is the expected total number of nodes that get activated or informed in at least one propagation process:

$$\gamma(\mathbf{s}) = \mathbb{E}[|\cup_{i\in[k]} \gamma_i(S_i)|] \tag{76}$$

At that time, the k-topic information Coverage Maximization under Knapsack constraint (kCMK) problem is defined as follows:

Definition 2 (kCMK *Problem*)**.** Assuming that each user e has a cost $c(e) > 0$ for every ith topic, which manifests how hard it is to influence the respective person for that topic initially. Given a positive integer-valued budget B, and edge probabilities $p^i_{u,v}, (u,v) \in E, i \in [k]$, the problem asks to find a seed set $\mathbf{s}$ with $c(\mathbf{s}) = \sum_{e\in S_i, i\in k} c(e) \leq B$ so that $\gamma(\mathbf{s})$ is maximal.

5.3. *k-type Sensor Placement under Knapsack constraint*

We further study the performance of algorithms for *k-type Sensor Placement under Knapsack constraint* (kSPK) problem which is formally defined as follows:

Definition 3 (kSPK *Problem*)**.** Given k types of sensors for different measures and a set V of n locations, each of which is assigned with only one sensor. Assuming that each sensor e has a cost $c(e) > 0$ for every ith type. Given a positive integer-valued budget of B, the problem aims to locate these sensors to maximize the information gained with the total cost at most B.

Denote by X^i_e a random variable representing the observation collected from a i-type sensor and the information gained of a k-set $\mathbf{s}$ is

$$f(\mathbf{s}) = H(\cup_{e\in supp(\mathbf{s})}\{X^i_e\}) \tag{77}$$

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

In three problems, the objective is either monotone and submodular (Wang et al., 2015; Qian et al., 2018b; Nguyen and Thai, 2020; Ohsaka and Yoshida, 2015).

5.4. Results and discussion

5.4.1. Experiment settings

For kIMK and kCMK. We use three databases: Facebook, Hept, and Enron, which are popular in information propagation problems (Pham et al., 2022b; Nguyen et al., 2017; Nguyen and Thai, 2020) and set up the model as the recent work (Nguyen and Thai, 2020).

Since the computation of $\sigma(\cdot)$ is #P-hard (Chen et al., 2010), we adapt the sampling method in Nguyen and Thai (2020) and Borgs et al. (2014) to give an estimation $\hat{\sigma}(\cdot)$ with a (λ, δ)-approximation that is:

$$\Pr[(1+\lambda)\sigma(\mathbf{s}) \geq \hat{\sigma}(\mathbf{s}) \geq (1-\lambda)\sigma(\mathbf{s})] \geq 1-\delta. \tag{78}$$

In the experiment, we set parameters $\lambda = 0.8, \delta = 0.2$, $k = 3$ and $\epsilon = 0.1$ as in Nguyen and Thai (2020) to show a trade-off between solution quality and quantities of queries. The estimation $\hat{\gamma}(\cdot)$ of $\gamma(\cdot)$ was calculated by the Eqs. (76) and (78).

We also budget for kIMK and kCMK with several B in $\{0.5K, 1K, 1.5K, 2K\}$ to illustrate the expense to influence k topics via large networks such as social networks is not a small number. We set the cost of each element according to the Normalized Linear model (Pham et al., 2022b). Whereby, we set the cost from 1 to 10 with Facebook and increase the cost range from 1 to 50 with Hept and Enron.

As mentioned in Wang et al. (2015) and Qian et al. (2018b), the problem of Information Coverage Maximization is a variant of the Influence Maximization problem. The edge probability of each edge $(u, v) \in E$ has a probability vector $(1/k, \ldots, 1/k)$ as setup in Qian et al. (2018b). Besides, we keep the setting for both kIMK for kCMK by running two problems in the same experiment. In the experiment, we collect results of kIMK and kCMK about objective values, and they have the same running time and queries.

For kSPK. We use 2 datasets including Intel Lab (Bodik et al., 2004) and DF-AMS WSN (Huy and Viet, 2015) to illustrate the kSPK problem. Intel Lab sensor data (Bodik et al., 2004) was collected from 2.3 million readings of a Mica2dot weather board sensor system, including temperature, humidity, light, and voltage, while DF-AMS WSN (Huy and Viet, 2015) was collected from the output of the NS2 simulation software that includes temperature, humidity, wind speed, and energy consumption. The data were all preprocessed to remove missing data fields and be consistent and compatible with the data reading function in the experiment. Moreover, we set $k = 3$, $\epsilon = 0.1$ as in the experiment of kIMK, and the cost range from 1 to 10 for the Intel Lab dataset and from 1 to 50 for the DF-AMS WSN dataset, whereas the values of B are fixed at several points from 10 to 50. This setting depends on the number of sensors and the similarity among algorithms.

5.4.2. Experiment results

To provide a comprehensive experiment, we ran the above algorithms several times and collected results about objective values, the number of queries, and the running time according to the B milestones. For each milestone, the average values were calculated. Figs. 1–5 illustrate the result on the above databases for three instances kIMK, kCMK, and kSPK.

Regarding kIMK and kCMK. The graph lines of the Greedy Algorithm in Tang et al. (2022) were not the final results because it took too long to complete running this algorithm. To the experiment fit within our system configuration, we had to limit the time to process the Greedy according to the number of nodes in each database, in which $4K$-node Facebook ran within 12 h, $15K$-node Hept ran in 2 days, and $36K$-node Enron worked for 4 days.

First, Figs. 1 and 2 represent the quality of algorithms via values of the objective functions $\sigma(\cdot)$ and $\gamma(\cdot)$. The Greedy lines just show the estimations of $\sigma(\cdot)$ and $\gamma(\cdot)$ in a limited time because it wasted extremely long to complete. In the remaining, IFA+ always provides the best results, followed by IFA, DS, and RS. Values of FA fluctuate because FA's lines mark the lowest points in some cases and better in others. In Fig. 1, the gaps between FA, IFA, IFA+, DS, and RS seem bigger when $B \geq 1.5K$. And when B and n (the size of V) increase, the performance of IFA+ and IFA seem better. For instance, at $B = 2K$, with Hept ($n \approx 15K$ nodes) IFA+ and IFA is about 1.5 times higher than DS and RS, respectively, while with Enron ($n \approx 36K$ nodes), IFA+ and IFA are about 1.25 times higher than both DS and RS. Especially, IFA+ and IFA show outstanding results compared to the others in the Enron network. Regarding kCMK, the results in Fig. 2 look similar to those in Fig. 1. We can see that the performance in terms of solution quality of IFA+ is always higher than the others. FA, IFA, and DS are indifferent, while RS has a bit of fluctuation.

In general, the result clusters algorithms into three groups from top to bottom: IFA+-DS, IFA-FA-RS, and Greedy. We can see the gap between algorithms in each group when B grows from $0.7K$ to $1.5K$, which seems relatively small, and separate when B increases more than $1.5K$. These lines also fluctuate according to each B milestone, yet we can realize the general trend of IFA+, IFA, DS, and RS going up while FA values seem unpredictable. These results reflect the theoretical guarantees when the approximation ratio of IFA+ is better than the others, FA is the lowest, and the role of FA is just bounding the rage of the optimal.

Second, Figs. 3 and 4 display the amounts of queries called and the time needed to run these algorithms. As mentioned, we ran kIMK and kCMK in the same experiment. Hence, the results of queries and the running time belong to both kIMK and kCMK. The Greedy takes the most quantities of queries in this experiment. Although Greedy is processed in a limited time, it still costs almost $1M-5M$ queries, which is typically higher than the others. We now focus on the results of the remaining. FA shows an advantage over others in terms of query complexity. It is sharply from several to dozens of times lower than the remaining. Specifically, RS is about 15–20 times and 4–5 times higher, while DS is about 7–10 times and 1.5 times higher than FA and IFA, respectively. Besides, the number of queries of IFA+ is always higher than DS and lower than RS, respectively. Significantly, IFA and FA explicitly determine and go straight over B milestones. Overall, the number of queries of RS is the highest, followed by IFA+, and DS is a little higher than the others. Finally, the quantities of queries of our algorithms give better results than the others.

As the query complexity directly influences running time, the representation of the time graph in Fig. 4 looks quite similar to the representation on the query graph in Fig. 3 in which FA line was drawn typically lowest. It shows the running time of FA is several to dozens of times faster than the others. IFA runs considerably faster than DS and RS, while Greedy runs the slowest. IFA+ is faster than RS but slower than DS.

The above figures show the trade-off between our proposed algorithms' solution qualities and the query complexities. FA tries to target the near-optimal value by dividing the ground into two subsets according to the cost values of elements and reduces query complexity by the filtering condition of the algorithm 1. Hence, the query complexity decreases significantly. Nevertheless, the performance of FA regarding solution quality is not high. IFA and IFA+ enhance FA by using FA as an input and the decreasing constant threshold. As a result, the objective of IFA is better than FA while the number of queries is a little higher but still deterministic. IFA+ reconstructs the solution in the second phase to upgrade the performance. It leads to the objective value increasing, yet the number of queries also increases. Moreover, when the ground set and B value grow, the solution quality improves while running time and query complexity are linear. This is extremely important when working with big data.

Besides, the difference in quality between our algorithms and those in Pham et al. (2022b) is due to the construction of algorithms. DS and RS focus on finding more candidate solutions than FA and IFA;

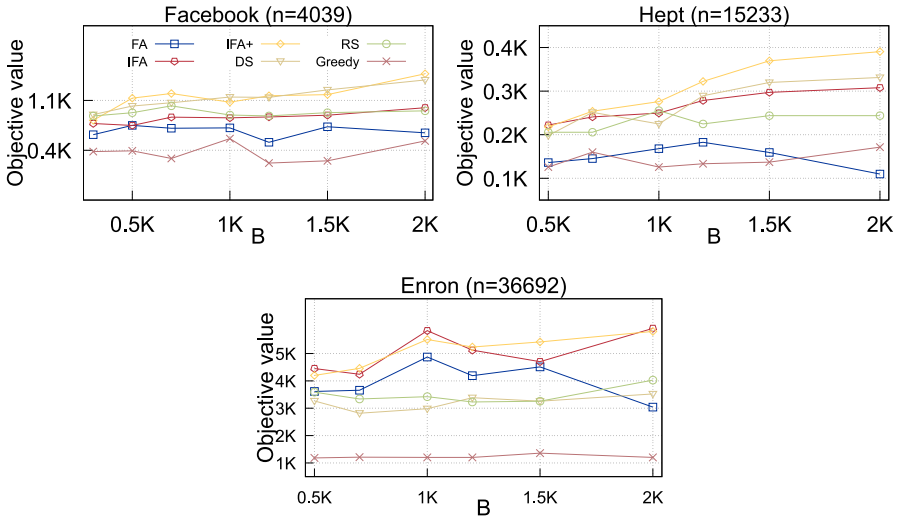

Fig. 1. Performance of algorithms for kIMK.

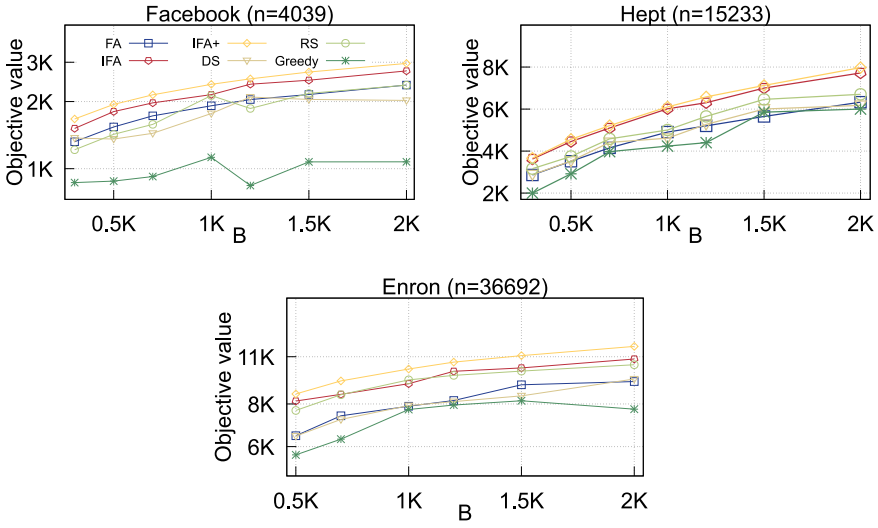

Fig. 2. Solution quality of algorithms for kCMK.

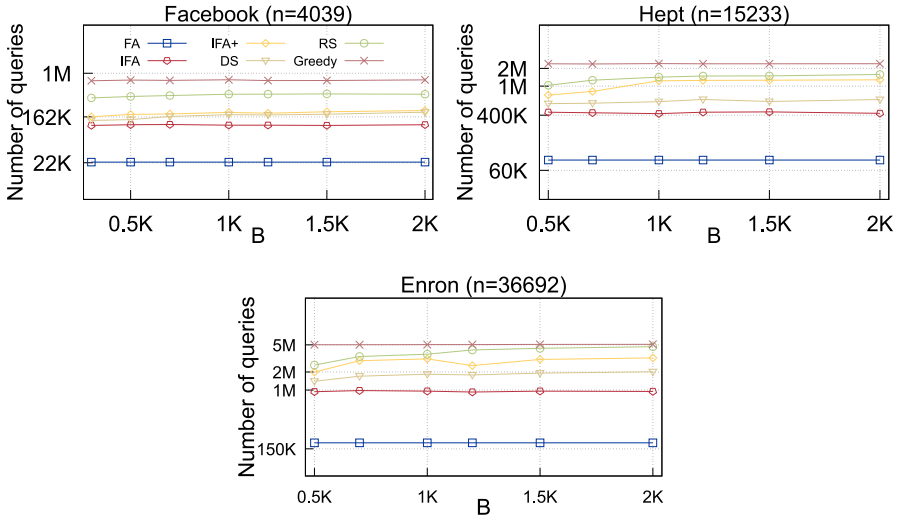

Fig. 3. Number of queries of algorithms for kIMK and kCMK.

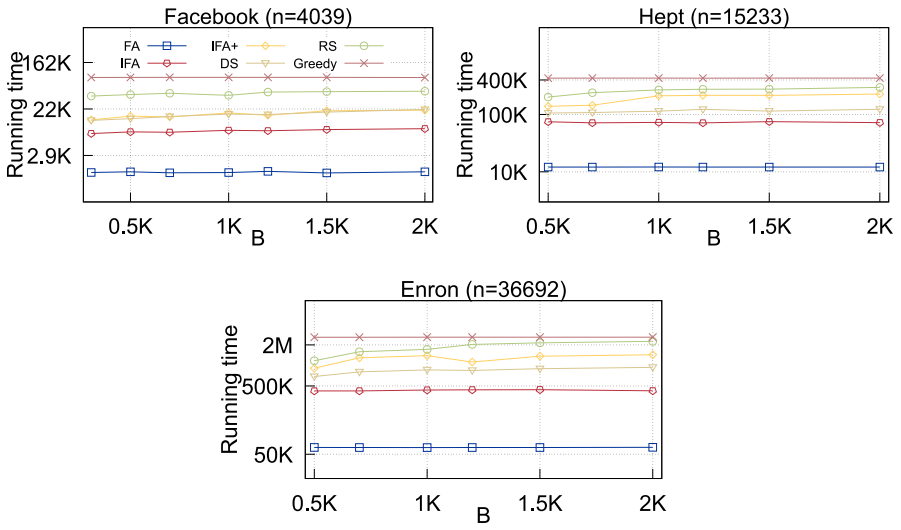

Fig. 4. Running time of algorithms for kIMK and kCMK.

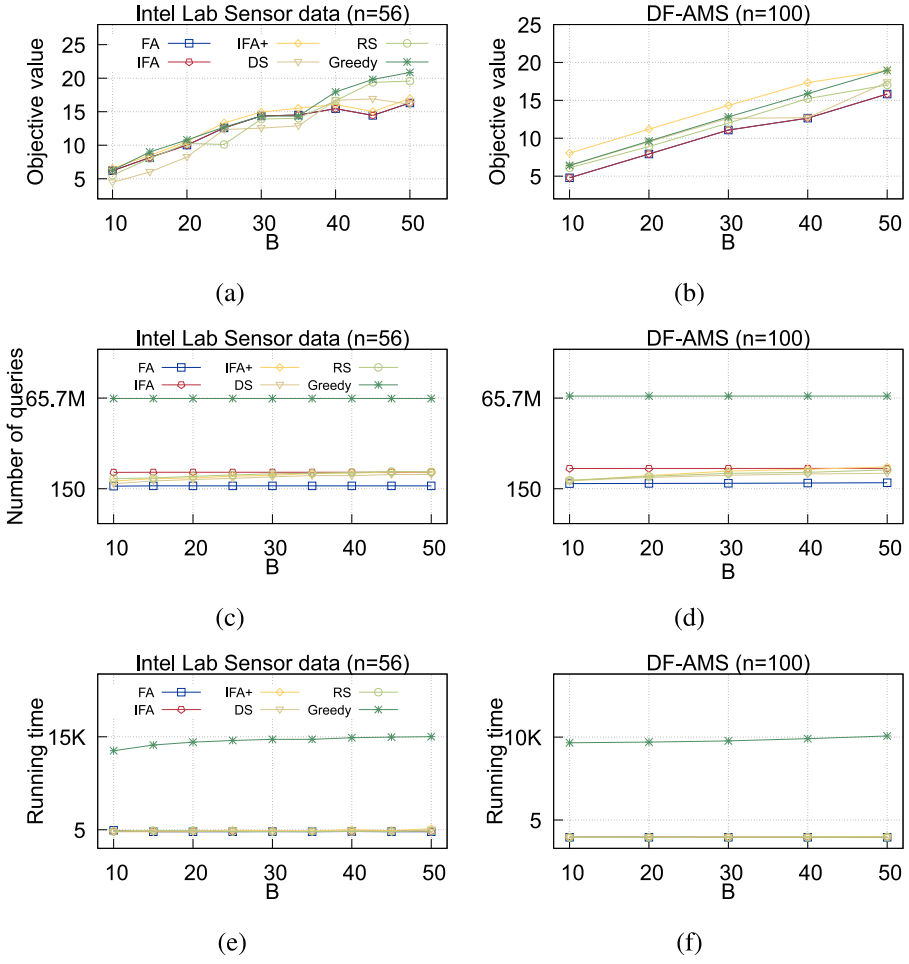

Fig. 5. Performance of algorithms for kSPK: (a), (b) Information gain; (c), (d) number of queries; (e), (f) running time.

therefore, they have more opportunities to find a better solution. In IFA+, re-selecting a small portion of the elements in the solution and replacing it with better elements in the set V increases performance with a bit more query overhead. Hence, when input data grows, our algorithms overcome others, both solution quality and the number of queries.

In addition, from the above figures, the experiment shows our algorithms dozens of times outperform Greedy regarding the number of queries and running time. This correctly reflects theory guarantees that our query complexity is $O(kn)$ while Greedy complexity is $O(n^4k^3)$.

Regarding kSPK. The result was shown in Fig. 5. First, the objective function of kSPK is an oracle of the Entropy function. Besides, in this

case, the number of sensor nodes is small. Hence, the discrimination between objective values of experimented algorithms is not large. Nevertheless, the result of IFA+, RS, and Greedy is generally better than the others. Especially, IFA+'s line always lies on the others when $B \leq 40$. FA and IFA overlap, while DS results fluctuate slightly. Overall, the general trend is increasing.

Second, the gap between the Greedy line and the others in Fig. 5(c)–(f) is significantly large. While Greedy needs millions of queries to output the final solution, the remaining algorithms just use approximately 150 queries. Similarly, the running time of Greedy is also thousands of times higher than the others. Moreover, query lines and timelines of the above algorithms are almost horizontal over B's milestones. On the other hand, the number of queries and time of FA is the smallest, and it is a little different among IFA, IFA+, DS, and RS. This result illustrates the query complexity of our algorithms is more optimized than the others.

Overall, from three actual uses of kIMK, kCMK, and kSPK, our proposed algorithms are described to outperform or be comparable to the state-of-the-art.

6. Conclusion

This paper studies the problem of maximizing a k-submodular function under the knapsack constraint. We propose three deterministic algorithms that take just $O(kn)$ query complexity. The key of our algorithms lines in a novel framework that contains two components. Firstly, it first divides the ground set into an appropriate subset to find a near-optimal solution with $O(nk)$ queries. Secondly, it boosts the solution quality without increasing query complexity by adapting greedy threshold strategies.

To investigate the performance of our algorithms in practice, we conduct some experiments on three applications: Influence Maximization, Information Coverage Maximization, and Sensor Placement. Experimental results have shown that our algorithms not only return reasonable solutions regarding quality requirements but also take a sharply smaller number of queries than state-of-the-art algorithms. In the future, we will further work with the problem whose objective function is non-monotone.

CRediT authorship contribution statement

Dung T.K. Ha: Conceptualization, Methodology, Software, Writing – original draft. **Canh V. Pham:** Data curation, Methodology, Writing – original draft. **Tan D. Tran:** Visualization, Software, Data curation.

Data availability

Data will be made available on request.

Acknowledgments

We would like to thank the Editors and reviewers whose valuable comments helped us improve this paper. The work has been carried out partly at the Vietnam Institute for Advanced Study in Mathematics (VIASM). The second author (Canh V. Pham) would like to thank VIASM for its hospitality and financial support.

References

Amir, R., 2005. Supermodularity and complementarity in economics: An elementary survey. South. Econ. J. 71 (3), 636–660.

Balkanski, E., Qian, S., Singer, Y., 2021. Instance specific approximations for submodular maximization. In: Proceedings of the 38th International Conference on Machine Learning, Vol. 139. pp. 609–618.

Bilbao, J., 2000. Cooperative games on combinatorial structures.

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

Borgs, C., Brautbar, M., Chayes, J.T., Lucier, B., 2014. Maximizing social influence in nearly optimal time. In: Proceedings of the 2014 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA). pp. 946–957.

Buchbinder, N., Feldman, M., Naor, J., Schwartz, R., 2015. A tight linear time (1/2)-Approximation for unconstrained submodular maximization. SIAM J. Comput. 44 (5), 1384–1402.

Chen, W., Wang, Y., Yang, S., 2009. Efficient influence maximization in social networks. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining 2009. ACM, pp. 199–208.

Chen, W., Yuan, Y., Zhang, L., 2010. Scalable influence maximization in social networks under the linear threshold model. In: Proceedings of the IEEE International Conference on Data Mining (ICDM). pp. 88–97.

Dam, T.T., Ta, T.A., Mai, T., 2022. Submodularity and local search approaches for maximum capture problems under generalized extreme value models. European J. Oper. Res. 300 (3), 953–965.

Du, W., Xing, Z., Li, M., He, B., Chua, L.H.C., Miao, H., 2014. Optimal sensor placement and measurement of wind for water quality studies in urban reservoirs. In: IPSN'14, Proceedings of the 13th International Symposium on Information Processing in Sensor Networks (Part of CPS Week). pp. 167–178.

Ene, A., Nguyen, H.L., 2022. Streaming algorithm for monotone k-submodular maximization with cardinality constraints. In: Chaudhuri, K., Jegelka, S., Song, L., Szepesvári, C., Niu, G., Sabato, S. (Eds.), Proceedings of the 39th International Conference on Machine Learning, Vol. 162. pp. 5944–5967.

Guillory, A., Bilmes, J.A., 2011. Simultaneous learning and covering with adversarial noise. In: Getoor, L., Scheffer, T. (Eds.), Proceedings of the International Conference on Machine Learning, ICML. Omni Press, pp. 369–376.

Güney, E., Leitner, M., Ruthmair, M., Sinnl, M., 2021. Large-scale influence maximization via maximal covering location. European J. Oper. Res. 289 (1), 144–164.

Huy, D.V., Viet, N.D., 2015. DF-AMS: Proposed solutions for multi-sensor data fusion in wireless sensor networks. In: Proceedings of the InternationalKnowledge and Systems Engineering Conference KSE. pp. 1–6. http://dx.doi.org/10.1109/KSE.2015.28.

Iwata, S., Tanigawa, S., Yoshida, Y., 2016. Improved approximation algorithms for k-submodular function maximization. In: Krauthgamer, R. (Ed.), Proceedings of the 2016 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA). SIAM, pp. 404–413.

Kempe, D., Kleinberg, J.M., Tardos, É., 2003. Maximizing the spread of influence through a social network. In: Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 137–146.

Khuller, S., Moss, A., Naor, J., 1999. The budgeted maximum coverage problem. Inform. Process. Lett. 70 (1), 39–45.

Klimt, B., Yang, Y., 2004. Introducing the enron corpus. In: CEAS 2004 - Proceedings of the First Conference on Email and Anti-Spam.

Krause, A., Guestrin, C., 2007. Near-optimal observation selection using submodular functions. In: Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence.

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

Krause, A., Singh, A.P., Guestrin, C., 2008. Near-optimal sensor placements in Gaussian processes: Theory, efficient algorithms and empirical studies. J. Mach. Learn. Res. (JMLR) 9, 235–284.

Kuhnle, A., 2021. Quick streaming algorithms for maximization of monotone submodular functions in linear time. In: Proceedings of the 24th International Conference on Artificial Intelligence and Statistics (AISTATS). pp. 1360–1368.

Lee, J., Kim, G., Moon, I., 2021. A mobile multi-agent sensing problem with submodular functions under a partition matroid. Comput. Oper. Res. 132, 105265.

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

Lopes, M., Ramos, T.R., 2023. Efficient sensor placement and online scheduling of bin collection. Comput. Oper. Res. 151, 106113.

Lovász, L., 1982. Submodular functions and convexity. In: Proceedings of the Mathematical Programming the State of the Art, XIth International Symposium on Mathematical Programming. Springer, pp. 235–257.

Milgrom, P., Roberts, J., 1990. The economics of modern manufacturing: Technology, strategy, and organization. Am. Econ. Rev. 80 (3), 511–528.

Mirzasoleiman, B., Badanidiyuru, A., Karbasi, A., 2016a. Fast constrained submodular maximization: Personalized data summarization. In: Proceedings of the International Conference on Machine Learning, ICML. 48, pp. 1358–1367.

Mirzasoleiman, B., Karbasi, A., Badanidiyuru, A., Krause, A., 2015. Distributed submodular cover: Succinctly summarizing massive data. In: Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS). pp. 2881–2889.

Mirzasoleiman, B., Zadimoghaddam, M., Karbasi, A., 2016b. Fast distributed submodular cover: Public-private data summarization. In: Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS). pp. 3594–3602.

Mitrovic, M.Z.M., Kazemi, E., Karbasi, A., 2016. Data summarization at scale: A two-stage submodular approach. In: Proceedings of the International Conference on Machine Learning, ICML. pp. 3593–3602.

Nguyen, L., Thai, M., 2020. Streaming k-submodular maximization under noise subject to size constraint. In: Proceedings of the 37th International Conference on Machine Learning ICML 2020. pp. 7338–7347.

Nguyen, H.T., Thai, M.T., Dinh, T.N., 2017. A billion-scale approximation algorithm for maximizing benefit in viral marketing. IEEE/ACM Trans. Netw. 25 (4), 2419–2429.

Ohsaka, N., Yoshida, Y., 2015. Monotone k-submodular function maximization with size constraints. In: Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems. pp. 694–702.

Ortiz-Astorquiza, C., Contreras, I.A., Laporte, G., 2015. Multi-level facility location as the maximization of a submodular set function. European J. Oper. Res. 247 (3), 1013–1016.

Oshima, H., 2017. Derandomization for k-submodular maximization. In: Proceedings of the 28th International Workshop on Combinatorial Algorithms. pp. 88–99.

Pham, C.V., Ha, D.K.T., Hoang, H.X., Tran, T.D., 2022a. Fast streaming algorithms for k-submodular maximization under a knapsack constraint. In: Proceedings of International Conference on Data Science and Advanced Analytics (DSAA). pp. 1–10.

Pham, C.V., Vu, Q.C., Ha, D.K., Nguyen, T.T., Le, N.D., 2022b. Maximizing k-submodular functions under budget constraint: applications and streaming algorithms. J. Combin. Optim. 44 (1), 723–751.

Qian, C., Shi, J., Tang, K., Zhou, Z., 2018a. Constrained monotone k-submodular function maximization using multiobjective evolutionary algorithms with theoretical guarantee. IEEE Trans. Evol. Comput. 22 (4), 595–608.

Qian, C., Shi, J., Tang, K., Zhou, Z., 2018b. Constrained monotone k-submodular function maximization using multiobjective evolutionary algorithms with theoretical guarantee. IEEE Trans. Evol. Comput. 22 (4), 595–608.

Rafiey, A., Yoshida, Y., 2020a. Fast and private submodular and k-submodular functions maximization with matroid constraints. In: Proceedings of the 37th International Conference on Machine Learning ICML. pp. 7887–7897.

Rafiey, A., Yoshida, Y., 2020b. Fast and private submodular and k-submodular functions maximization with matroid constraints. In: Proceedings of 37th International Conference on Machine Learning. pp. 7887–7897.

Saeys, Y., Inza, I., Larrañaga, P., 2007. A review of feature selection techniques in bioinformatics. Bioinformatics 23 (19), 2507–2517.

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

Singh, A.P., Guillory, A., Bilmes, J.A., 2012. On bisubmodular maximization. In: Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics AISTATS. pp. 1055–1063.

Soma, T., 2019. No-regret algorithms for online k-submodular maximization. In: Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2019. pp. 1205–1214.

Sviridenko, M., 2004. A note on maximizing a submodular set function subject to a knapsack constraint. Oper. Res. Lett. 32 (1), 41–43.

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

2023. Using submodularity in solving the robust bandwidth packing problem with queuing delay guarantees. Comput. Oper. Res. 160, 106374.

Vondrák, J., 2008. Optimal approximation for the submodular welfare problem in the value oracle model. In: Dwork, C. (Ed.), Proceedings of the 40th Annual ACM Symposium on Theory of Computing. pp. 67–74.

Wang, Z., Chen, E., Liu, Q., Yang, Y., Ge, Y., Chang, B., 2015. Maximizing the coverage of information propagation in social networks. In: Yang, Q., Wooldridge, M.J. (Eds.), Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI). pp. 2104–2110.

Wang, B., Zhou, H., 2021. Multilinear extension of k-submodular functions. CoRR, abs/2107.07103.

Ward, J., Zivný, S., 2014. Maximizing bisubmodular and *k*-submodular functions. In: Proceedings of the 2014 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA). pp. 1468–1481.

Zheng, L., Chan, H., Loukides, G., Li, M., 2021. Maximizing approximately *k*-submodular functions. In: Proceedings of the 2021 SIAM International Conference on Data Mining, SDM 2021, Virtual Event, April 29 - May 1, 2021. SIAM, pp. 414–422.