Journal of Combinatorial Optimization
https://doi.org/10.1007/s10878-022-00858-x

Maximizing k-submodular functions under budget constraint: applications and streaming algorithms

Canh V. Pham[1] · Quang C. Vu[2] · Dung K. T. Ha[3] · Tai T. Nguyen[3] · Nguyen D. Le[4]

Accepted: 21 March 2022

Abstract
Motivated by the practical applications in solving plenty of important combinatorial optimization problems, this paper investigates the Budgeted k-Submodular Maximization problem defined as follows: Given a finite set V, a budget B and a k-submodular function $f : (k+1)^V \mapsto \mathbb{R}_+$, the problem asks to find a solution $\mathbf{s} = (S_1, S_2, \ldots, S_k) \in (k+1)^V$, in which an element $e \in V$ has a cost $c_i(e)$ when added into the i-th set S_i, with the total cost of $\mathbf{s}$ that does not exceed B so that $f(\mathbf{s})$ is maximized. To address this problem, we propose two single pass streaming algorithms with approximation guarantees: one for the case that an element e has only one cost value when added to all i-th sets and one for the general case with different values of $c_i(e)$. We further investigate the performance of our algorithms in two

A preliminary version appears in the proceedings of the 10th International Conference on Computational Data and Social Networks. This paper extends and revises the conference version by providing all the proofs more detail and experiment evaluation.

✉ Canh V. Pham
canh.phamvan@phenikaa-uni.edu.vn

Quang C. Vu
quangvc.hvan@gmail.com

Dung K. T. Ha
20028008@vnu.edu.vn

Tai T. Nguyen
20025032@vnu.edu.vn

Nguyen D. Le
nguyenld@dhhp.edu.vn

1 ORlab, Faculty of Computer Science, Phenikaa University, Hanoi 12116, Vietnam

2 Faculty of Information Security, People's Security Academy, Hanoi 10000, Vietnam

3 Faculty of Information Technology, University of Engineering and Technology, Vietnam National University, Hanoi 10000, Vietnam

4 Haiphong University, Haiphong 180000, Vietnam

Published online: 18 April 2022

applications of the problem, Influence Maximization with k topics and sensor placement of k types of measures. The experiment results indicate that our algorithms can return competitive results but require fewer the number of queries and running time than the state-of-the-art methods.

Keywords k-submodular · Budget constraint · Approximation algorithm · Streaming algorithm

1 Introduction

Maximizing k-submodular functions has attracted a lot of attentions because of its potential in solving various combinatorial optimization problems such as influence maximization (Ohsaka and Yoshida 2015; Rafiey and Yoshida 2020; Qian et al. 2018; Nguyen and Thai 2020), sensor placement (Ohsaka and Yoshida 2015; Rafiey and Yoshida 2020; Qian et al. 2018), feature selection (Singh et al. 2012) and information coverage maximization (Qian et al. 2018). Given a finite set V and an integer k, we define $[k] = \{1, 2, \ldots, k\}$ and $(k+1)^V = \{(X_1, X_2, \ldots, X_k) | X_i \subseteq V, \forall i \in [k], X_i \cap X_j = \emptyset, \forall i \neq j\}$ 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 $\mathbf{x} = (X_1, X_2, \ldots, X_k)$ 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

$$\mathbf{x} \sqcup \mathbf{y} = \left(X_1 \cup Y_1 \setminus (\bigcup_{i \neq 1} X_i \cup Y_i), \ldots, X_k \cup Y_k \setminus (\bigcup_{i \neq k} X_i \cup Y_i) \right)$$

In addition to unconstrained case (Ward and Zivný 2014; IWata et al. 2016; Soma 2019; Oshima 2017), researchers also solve the problem under size constraint (Rafiey and Yoshida 2020; Ohsaka and Yoshida 2015; Qian et al. 2018; Nguyen and Thai 2020), matroid constraint (Sakaue 2017; Rafiey and Yoshida 2020) and knapsack constraint (Tang et al. 2022; Wang and Zhou 2021). However, these problems does not cover several real applications which customizes each element in terms of requiring its cost as well as limits the budget. We are going to discuss the following two applications:

Influence Maximization with k topics under a budget constraint Given a social network under an information diffusion model and k topics. Each user has a cost to start the influence under a topic which manifests how hard it is to initially influence the respective person for that topic. Given the budget B, we consider the problem of finding a set of users (seed set), each initially adopts a topic, with the total cost that is at most B to maximize the expected numbers of users activated by at least one topic.

Sensor placement with k types of measures under a budget constraint. Given k types of sensors for different measures and n locations, each of which can be instrumented

with one sensor exactly. We assume that allocating a sensor to each location has a different cost depending on its position and the type of sensor. Given the budget B, we consider the problem of allocating those sensors to maximize the information gained with the total cost at most B.

In two above applications, the objective functions are k-submodular (Ohsaka and Yoshida 2015; Rafiey and Yoshida 2020; Nguyen and Thai 2020). Although there have been many attempts to find a solution that maximizes the k-submodular function, they did not cover the case that each element would have different costs when added into different sets of the solution with a limited budget as shown in two above examples. Motivated by that observation, in this work, we study a novel problem named *Budgeted* k-submodular maximization (BkSM), defined as follows.

Definition 1 (BkSM *problem*) Given a finite set V, a budget B and a k-submodular function $f : (k+1)^V \mapsto \mathbb{R}_+$. The problem asks to find a solution $\mathbf{s} = (S_1, S_2, \ldots, S_k)$ in which an element $e \in V$ has a cost $c_i(e) > 0$ when added into S_i, with total cost $c(\mathbf{s}) = \sum_{i\in[k]} \sum_{e\in S_i} c_i(e) \leq B$ so that $f(\mathbf{s})$ is maximized.

In addition, the constant increase of input data makes it impossible to be stored in computer memory. Therefore it is critical to devise streaming algorithms for BkSM, in which a streaming algorithm receives each element in the ground set sequentially, and keeps only a small number of the element in memory at any point. After scanning one or a few passes over the ground set, the algorithm can return a solution with performance guarantees (Badanidiyuru et al. 2014; Yang et al. 2019; Rafiey and Yoshida 2020).

1.1 Our contributions

To address above challenges, in this paper we propose two single-pass streaming algorithms which provide theoretical bounds of BkSM. Overall, our contributions are as follows:

- For the special case: an element has just one cost value when added into any i-th set, we first propose a deterministic streaming algorithm (Algorithm 2) which runs in a single pass, has $O(\frac{kn}{\epsilon}\log n)$ query complexity, $O\left(\frac{n}{\epsilon}\log n\right)$ space complexity and returns an approximation ratio of $\frac{1}{4} - \epsilon$ when f is *monotone* and $\frac{1}{5} - \epsilon$ when f is *non-monotone* for any input parameter $\epsilon \in (0, \frac{1}{5})$.
- For the general case, we propose a random streaming algorithm (Algorithm 4) which runs in a single pass, has $O(\frac{kn}{\epsilon}\log n)$ query complexity, $O\left(\frac{n}{\epsilon}\log n\right)$ space complexity and returns an approximation ratio of $\min\{\frac{\alpha}{2}, \frac{(1-\alpha)k}{(1+\beta)k-\beta}\} - \epsilon$ when f is *monotone* and $\min\{\frac{\alpha}{2}, \frac{(1-\alpha)k}{(1+2\beta)k-2\beta}\} - \epsilon$ when f is *non-monotone* in expectation where $\beta = \max_{e\in V, i, j\in[k], i\neq j} \frac{c_i(e)}{c_j(e)}$ and $\alpha \in (0, 1], \epsilon \in (0, 1)$ are input parameters.
- We conduct comprehensive experiments to investigate the performance of our algorithms in two applications of BkSM, Influence Maximization and Sensor Placement. The results have shown that our algorithms not only reduce the number of queries but also return comparable solutions in term of quality than the state-of-the-art non-streaming algorithms.

1.2 Related work

Although submodular maximization problems is NP-hard in general (Schrijver 2003), they have been extensively studied because of their important roles in combinatorial optimization and machine learning. Nemhauser et al. (1978) first studied the problem of maximizing a monotone submodular function under a cardinality constraint and showed that the traditional greedy algorithm could provide an approximation ratio of $(1 - 1/e)$. Since then, there have been many studies on this problem under various constraints such as cardinality constraint (Badanidiyuru and Vondrák 2014; Mirzasoleiman et al. 2015, 2016; Buchbinder et al. 2015; Krause et al. 2008), knapsack constraint (Wolsey 1982; Sviridenko 2004; Huang et al. 2020) matroid constraint (C**u**alinescu et al. 2011), p-set constraint (Haba et al. 2020), d-knapsack constraint (Yuet al. 2016). However, submodular maximization algorithms may not be applicable to k-submodular maximizaiton problems due to intrinsic differences between submodularity and k-submodularity.

Studying on k-submodular functions was initiated by Singh et al. (2012) but the authors only focused on the case $k = 2$, i.e., bisubmodular. Since then, more works have focused on the case of general k. Since submodular maximization problem, a special case of k-submodular maximization problem, is NP-hard (Schrijver 2003), the k-submodular maximization problem is also NP-hard. Though people proposed a polynomial-time algorithm in the case of minimizing a k-submodular (Thapper and Zivný S 2012), it's still a challenge of devising a similar algorithm to solve the problem of k-submodular maximization.

Ward and Zivný (2014) first studied an unconstrained maximization of k-submodular function, a special case of BkSM with cost values of all elements equal to 1 and $B = n$, and devised a deterministic greedy algorithm which provided an approximation ratio of $1/3$. Later on, the authors in IWata et al. (2016) introduced a random greedy approach which improved the approximation ratio to $\frac{k}{2k-1}$ by introducing a probability distribution to select a larger marginal element with higher probability. Work in Oshima (2017) eliminated the random told in IWata et al. (2016) but the number of queries increased to $O(n^2k^2)$. The unconstrained k-submodular maximization was further studied in Soma (2019) in online settings. In fact, the algorithms in Ward and Zivný (2014) and IWata et al. (2016) work as single pass streaming algorithms but they cannot be directly applied to our problem. Our streaming algorithm for BkSM also uses the idea of random selection in IWata et al. (2016) but it introduces a new distribution that can help to select an element with various costs and then establishes the relationship between the current solution and the optimal solution.

Maximizing k-submodular functions have been further studied with several types of constraints. Ohsaka and Yoshida (2015) first studied monotone k-submodular maximization problem with the size constraints. By using the greedy approaches, they proposed $1/2$-approximation algorithm for the total size constraint and $1/3$-approximation algorithm for the individual size constraint. Similarly, authors in Sakaue (2017) showed a greedy selection that could give an approximation ratio of $1/2$ under the matroid constraint. However, these works did not provide any approximation guarantee when f was non monotone. The authors in Qian et al. (2018) then

further proposed a multi-objective evolutionary algorithm for monotone k-submodular maximization problem under the total size constraint. Their algorithm provided $1/2$-approximation solution and took $O(kn \log^2 B)$ queries in expectation. Recently, Rafiey and Yoshida (2020) have considered the k-submodular maximization problem subject to the total size constraint under noises and proposed two streaming algorithms which provided the approximation ratio of $O(\epsilon(1-\epsilon)^{-2}B)$ when f was monotone and $O(\epsilon(1-\epsilon)^{-3}B)$ when f was non-monotone. Zheng et al. (2021) investigated the problem of maximizing approximately k-submodular functions subject to the size constraints by introducing an approximate function of the objective function and proposed several greedy algorithms with provable guarantees. However, these algorithms can not adapt to our studied problem because of the variety of an element's costs that makes devising an approximation algorithm more challenging.

To our best knowledge, Zhang et al. (2019) first studied the problem of maximizing the k-submodular function with each i-th set in the solution having a budget constraint. In the seminal paper, they devised a $\frac{1}{5}(1-\frac{1}{e})$ approximation algorithm with $O(kn^2)$ query complexity but it did not keep this ratio when f was non-monotone. Furthermore, this problem is completely different from our studied problem. Firstly, we consider a general case where each element has a variety of costs when added into various i-th set of the solution. Secondly, instead of the individual budget constraint, we consider the total budget constraint. Besides, our algorithm also provides the approximation guarantee when f is non-monotone. More recently, a $(1-1/e)/2$-approximation algorithm within $O(n^4k^3)$ queries for the monotone k-submodular maximization under a knapsack constraint has been proposed (Tang et al. 2022). This approximation ratio has been improved to $1/2-\epsilon$ by Wang and Zhou (2021). However, two above algorithms only hold for a special case of BkSM when f is monotone and an element has just one cost value when added into any i-th set. On the other hand, our Algorithm 2 can give the approximate ratios for both monotone and non-monotone cases.

The streaming algorithm is one of efficient methods for solving submodular maximization problems under various kinds of constraints such as cardinality constraint (Gomes and Krause 2010; Badanidiyuru et al. 2014; Kumar et al. 2013; Yang et al. 2019), knapsack constraint (Huang et al. 2020), k-set constraint (Haba et al. 2020) and matroid constraint (Chakrabarti and Kale 2015) but it is not potential to be directly applied to our BkSM problem due to intrinsic differences between submodularity and k-submodularity. Instead, constructing our algorithms is an inspired suggestion from Huang et al. (2020) and Badanidiyuru et al. (2014) in which we also sequentially make decision based on the value of incremental objective function per cost of each element and guess the optimal solution through the maximum singleton value.

1.3 Organization

The rest of the paper is organized as follows: The notations and properties of k-submodular functions are presented in Sect. 2. Sections 3 and 4 present our algorithms and theoretical analysis. The extensive experiments are shown in Sect. 5. Finally, we conclude this work in Sect. 6.

2 Preliminaries

This section presents notations used throughout the paper and properties of the k-submodular function.

Given a finite set V and an integer k, for $\mathbf{x} = (X_1, X_2, \ldots, X_k), \mathbf{y} = (Y_1, Y_2, \ldots, Y_k) \in (k+1)^V$, we define $supp_i(\mathbf{x}) = X_i$, $supp(\mathbf{x}) = \cup_{i\in[k]} X_i$, X_i is called the *i-th set of* $\mathbf{x}$ and an empty k-set is defined as $\mathbf{0} = (\emptyset, \ldots, \emptyset)$. If $e \in X_i$, we write $\mathbf{x}(e) = i$; if $e \notin \cup_{i\in[k]} X_i$, we write $\mathbf{x}(e) = 0$ and i is called the *position* of e; adding $e \notin supp(\mathbf{x})$ into X_i can be represented by $\mathbf{x} \sqcup (e, i)$. In the case of $X_i = \{e\}$, and $X_j = \emptyset, \forall j \neq i$, we denote $\mathbf{x}$ by (e, i). We denote by $\mathbf{x} \sqsubseteq \mathbf{y}$ iff $X_i \subseteq Y_i$ for all $i \in [k]$.

A function $f : (k+1)^V \mapsto \mathbb{R}_+$ is *k-submodular* iff 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:

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

where

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

and

$$\mathbf{x} \sqcup \mathbf{y} = \left(X_1 \cup Y_1 \setminus (\bigcup_{i\neq 1} X_i \cup Y_i), \ldots, X_k \cup Y_k \setminus (\bigcup_{i\neq k} X_i \cup Y_i) \right)$$

A function $f : (k+1)^V \mapsto \mathbb{R}_+$ is monotone iff for any $\mathbf{x} \in (k+1)^V, e \notin supp(\mathbf{x})$ and $i \in [k]$, we have:

$$\Delta_{e,i} f(\mathbf{x}) = f(X_1, \ldots, X_{i-1}, X_i \cup \{e\}, X_{i+1}, \ldots, X_k) - f(X_1, \ldots, X_k) \geq 0 \quad (2)$$

Given a k-submodular function $f : (k+1)^V \mapsto \mathbb{R}_+$, from Ward and Zivný (2014) the k-submodularity of f implies the *orthant submodularity*, i.e,

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

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

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

for any $\mathbf{x} \in (k+1)^V$ with $e \notin supp(\mathbf{x})$ and $i, j \in [k]$ with $i \neq j$.

In this paper, we assume that f is normalized, i.e, $f(\mathbf{0}) = 0$ and each element e has a positive cost $c_i(e)$ when added into the i-th set of a solution and the total cost of k-set $\mathbf{x}$ is:

$$c(\mathbf{x}) = \sum_{i\in[k], e\in supp_i(\mathbf{x})} c_i(e)$$

We define β as the *largest ratio of different costs of an element*, i.e,

$$\beta = \max_{e \in V, i \neq j} \frac{c_i(e)}{c_j(e)}$$

Without loss of generality, throughout this paper, we assume that every element e satisfies $c_i(e) \geq 1, \forall i \in [k]$ and $c_i(e) \leq B$ as otherwise we can simply remove it. We only consider $k \geq 2$ because if $k = 1$, the k-submodular function becomes the submodular function.

3 A deterministic streaming algorithm when $\beta = 1$

In this section, we introduce a deterministic streaming algorithm for the special case when $\beta = 1$, i.e, each element has the same cost for all subsets $c_i(e) = c_j(e), \forall e \in V, i \neq j$. For simplicity, we denote $c(e) = c_i(e) = c_j(e)$.

The main idea of our algorithms is that (1) we select each observed element e based on comparing between the ratio of f per total cost at the current solution and a threshold which is set in advance, and (2) we use the maximum singleton value (e_{max}, i_{max}) defined as

$$(e_{max}, i_{max}) = \arg \max_{e \in V, i \in [k]} f((e, i)) \tag{5}$$

to obtain the final solution. We first assume that the optimal solution is known and then remove this assumption by using the method in Badanidiyuru et al. (2014).

3.1 A deterministic streaming algorithm with known optimal value

We first present a simplified version of our deterministic streaming algorithm when the optimal value is known. Denote by $\mathbf{o}$ an optimal solution and $\mathsf{opt} = f(\mathbf{o})$, the algorithm receives v such that $v \leq \mathsf{opt}$ and a parameter $\alpha \in (0, 1]$ as inputs. The role of these parameters are going to be clarified in main version. The details of the algorithm are fully presented in Algorithm 1. We define the following notations:

- (e^j, i^j) as the j-th element added in the main loop of the algorithm.
- $\mathbf{s}^j = \{(e^1, i^1), \ldots, (e^j, i^j)\}$ as the solution when adding j elements in the main loop of the algorithm.
- $\mathbf{o}^j = (\mathbf{o} \sqcup \mathbf{s}^j) \sqcup \mathbf{s}^j$
- $\mathbf{o}^{j-1/2} = (\mathbf{o} \sqcup \mathbf{s}^j) \sqcup \mathbf{s}^{j-1}$
- $\mathbf{s}^{j-1/2}$: 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 $e^j \notin supp(\mathbf{o})$, $\mathbf{s}^{j-1/2} = \mathbf{s}^{j-1}$
- $\mathbf{u}^t = \{(u_1, j_1), (u_2, j_2), \ldots, (u_r, j_r)\}$: a set of elements that are in $\mathbf{o}^t$ but not in $\mathbf{s}^t$, $r = |supp(\mathbf{u}^t)|$
- $\mathbf{u}_i^t = \mathbf{s}^t \sqcup \{(u_1, j_1), (u_2, j_2), \ldots, (u_i, j_i)\}, \forall 1 \leq i \leq r$ and $\mathbf{u}_0^t = \mathbf{s}^t$.

The algorithm initiates a candidate solution $\mathbf{s}^0$ as an empty k-set. For each new incoming element e, the algorithm updates a tuple (e_{max}, i_{max}) to find the maximal

singleton then checks that the total cost $c(\mathbf{s}^t) + c(e)$ exceeds B or not. If not, it finds a position $i' \in [k]$ that $f(\mathbf{s}^t \sqcup (e, i'))$ is maximal and adds (e, i) into $\mathbf{s}^t$ if $\frac{f(\mathbf{s}^t \sqcup (e,i'))}{c(\mathbf{s}^t)+c(e)} \geq \frac{\alpha v}{B}$. Otherwise, it ignores e and receives the next element. This step helps the algorithm select any element having high value of marginal value per its cost as well as eliminate bad ones.

After finishing the main loop, the algorithm returns the best solution between $\{\mathbf{s}^t\}$ and $\{(e_{max}, i_{max})\}$ when f is monotone or returns the best solution between $\{\mathbf{s}^j : j \leq t\}$ and $\{(e_{max}, i_{max})\}$ when f is non-monotone.

Algorithm 1: Deterministic streaming algorithm with known opt

Input: a k-submodular function f, $B > 0$, $\alpha \in (0, 1]$, v that $v \leq \mathsf{opt}$
Output: a solution $\mathbf{s}$

$\mathbf{s}^0 \leftarrow \mathbf{0}$, $t \leftarrow 0$
$(e_{max}, i_{max}) \leftarrow (\emptyset, 1)$
foreach $e \in V$ **do**
  $i_e \leftarrow \arg\max_{i \in [k]} f((e, i))$
  $(e_{max}, i_{max}) \leftarrow \arg\max_{(e_1,i_1) \in \{(e_{max},i_{max}),(e,i_e)\}} f((e_1, i_1))$
  if $c(\mathbf{s}^t) + c(e) \leq B$ **then**
    $i' \leftarrow \arg\max_{i \in [k]} f(\mathbf{s}^t \sqcup (e, i))$
    if $\frac{f(\mathbf{s}^t \sqcup (e,i'))}{c(\mathbf{s}^t)+c(e)} \geq \frac{\alpha v}{B}$ **then**
      $\mathbf{s}^{t+1} \leftarrow \mathbf{s}^t \sqcup (e, i')$
      $t \leftarrow t + 1$
    end
  end
end
return $\arg\max_{\mathbf{s} \in \{\mathbf{s}^t, (e_{max}, i_{max})\}} f(\mathbf{s})$ if f is monotone, $\arg\max_{\mathbf{s} \in \{\{\mathbf{s}^j : j \leq t\}, (e_{max}, i_{max})\}} f(\mathbf{s})$ if f is non-monotone.

We now analyze the approximation guarantee of Algorithm 1. By exploiting the relation among $\mathbf{o}$, $\mathbf{o}^t$ and $\mathbf{s}^t$, we obtain the following Lemma.

Lemma 1 *Denote by e^t the last addition of the main loop of the Algorithm 1. If f is monotone then $v - f(\boldsymbol{o}^t) \leq f(\boldsymbol{s}^t)$ and if f is non-monotone then $v - f(\boldsymbol{o}^t) \leq 2f(\boldsymbol{s}^t)$.*

Proof The proof follows the analysis of the relationship among $\mathbf{s}^j$, $\mathbf{o}^j$, $\mathbf{o}$ in Nguyen and Thai (2020). We consider two following cases:

Case 1 If f is monotone. By the k-submodular property of f and note that $f(\mathbf{o}) = f(\mathbf{o}^0)$ we obtain:

$$
\begin{aligned}
v - f(\mathbf{o}^t) \leq f(\mathbf{o}) - f(\mathbf{o}^t) &= \sum_{j=1}^{t} (f(\mathbf{o}^{j-1}) - f(\mathbf{o}^j)) \\
&\leq \sum_{j=1}^{t} (f(\mathbf{o}^{j-1}) - f(\mathbf{o}^{j-1/2})) \quad \text{(due to the monotoncity of } f) \\
&\leq \sum_{j=1}^{t} (f(\mathbf{s}^{j-1/2}) - f(\mathbf{s}^{j-1})) \quad \text{(due to the} k - \text{submodularity)}
\end{aligned}
$$

$$\leq \sum_{j=1}^{t}(f(\mathbf{s}^j) - f(\mathbf{s}^{j-1})) \text{ (due to the selection of algorithm)}$$
$$\leq f(\mathbf{s}^t)$$

Case 2 If f is non-monotone, we further consider following sub-cases:

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

$$\begin{aligned} f(\mathbf{o}^{j-1}) - f(\mathbf{o}^j) &= f(\mathbf{o}_l^j) - f(\mathbf{o}^{j-1}) - (f(\mathbf{o}^j) + f(\mathbf{o}_l^j) - 2f(\mathbf{o}^{j-1})) \\ &\leq f(\mathbf{o}_l^j) - f(\mathbf{o}^{j-1}) \quad \text{(due to the pairwise-monotoncity)} \\ &\leq f(\mathbf{s}_l^j) - f(\mathbf{s}^{j-1}) \\ &\leq f(\mathbf{s}^j) - f(\mathbf{s}^{j-1}) \end{aligned}$$

- If $e^j \in supp(\mathbf{o})$. 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 obtain:

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

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

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

which completes the proof. □

Lemma 1 plays an important role for analyzing approximation ratio of the algorithm, which stated in the following Theorem.

Theorem 1 *Algorithm 1 is a single pass streaming algorithm and returns a solution* $\mathbf{s}$ *satisfying:*

- *If* f *is monotone,* $f(\mathbf{s}) \geq \min\{\frac{\alpha}{2}, \frac{1-\alpha}{2}\}v$. *The right hand side is maximized to* $\frac{v}{4}$ *when* $\alpha = \frac{1}{2}$.

- *If f is non-monotone, $f(\boldsymbol{s}) \geq \min\{\frac{\alpha}{2}, \frac{1-\alpha}{3}\}v$. The right hand side is maximized to $\frac{v}{5}$ when $\alpha = \frac{2}{5}$.*

Proof We observe that an element $e \in supp(\mathbf{o})$ does not belong to $supp(\mathbf{s}^t)$ if neither e does not pass the condition in line 8 nor its addition would cause the total cost of $\mathbf{s}^t$ to exceed B.

Denote by $e \in supp(\mathbf{o})$ as a *bad element* if it passes the condition in line 8 of Algorithm 1 but the total cost exceeds B, i.e., there exits an integer $i \in [k]$ satisfying:

$$\frac{f(\mathbf{s}^{t_e} \sqcup (e,i))}{c(\mathbf{s}^{t_e}) + c(e)} \geq \frac{\alpha v}{B} \text{ and } c(\mathbf{s}^{t_e}) + c(e) > B \tag{6}$$

where $\mathbf{s}^{t_e}$ is the candidate solution obtained right before e arrives.

Case 1 There is no bad element.

By applying Lemma 1, we obtain:

$$\begin{aligned}
v - f(\mathbf{s}^t) &= v - f(\mathbf{o}^t) + f(\mathbf{o}^t) - f(\mathbf{s}^t) \\
&\leq f(\mathbf{o}) - f(\mathbf{o}^t) + f(\mathbf{o}^t) - f(\mathbf{s}^t) \\
&\leq f(\mathbf{s}^t) + \sum_{i=1}^{r} \left(f(\mathbf{u}_i^t) - f(\mathbf{u}_{i-1}^t) \right) \text{ (Lemma 1)} \\
&\leq f(\mathbf{s}^t) + \sum_{i=1}^{r} (f(\mathbf{s}^{t_{u_i}} \sqcup (u_i, j_i)) - f(\mathbf{s}^{t_{u_i}}) \\
&\quad \text{(Due to the } k - \text{submodularity of } f) \\
&\leq f(\mathbf{s}^t) + \sum_{i=1}^{r} \left(\frac{\alpha v (c(\mathbf{s}^{t_{u_i}}) + c(u_i))}{B} - \frac{\alpha v c(\mathbf{s}^{t_{u_i}})}{B} \right) \\
&\leq f(\mathbf{s}^t) + \sum_{i=1}^{r} \frac{\alpha v c(u_i)}{B} \\
&\leq f(\mathbf{s}^t) + \alpha v
\end{aligned}$$

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

Case 2 If a bad element e exits, there is an integer $i \in [k]$ satisfying: $\frac{f(\mathbf{s}^{t_e} \sqcup (e,i))}{c(\mathbf{s}^{t_e}) + c(e)} \geq \frac{\alpha v}{B}$ and $c(\mathbf{s}^{t_e}) + c(e) > B$. Therefore:

$$f(\mathbf{s}^{t_e} \sqcup (e,i)) \geq \frac{(c(\mathbf{s}^{t_e}) + c(e))\alpha v}{B} > \alpha v$$

By the k-submodularity of f, we have:

$$f(\mathbf{s}^{t_e} \sqcup (e,i)) \leq f(\mathbf{s}^{t_e}) + f((e,i))$$

which implies that:

$$\begin{aligned} f(\mathbf{s}) &\geq \max\{f(\mathbf{s}^t),\, f((e_{max}, i_{max}))\} \\ &\geq \max\{f(\mathbf{s}^{t_e}),\, f((e,i))\} \\ &\geq \frac{f(\mathbf{s}^{t_e}) + f((e,i))}{2} \\ &> \frac{\alpha v}{2} \end{aligned}$$

Combine two above cases, we obtain $f(\mathbf{s}) = \min\{\frac{1-\alpha}{2}, \frac{\alpha}{2}\}v$ and $f(\mathbf{s})$ is maximized to $\frac{1}{4}v$ when $\alpha = \frac{1}{2}$.

If f is non-monotone, by arguments that are similar to the monotone case, we have $f(\mathbf{s}) = \min\{\frac{1-\alpha}{3}, \frac{\alpha}{2}\}v$. The proof is completed. □

3.2 A deterministic streaming algorithm

We present our deterministic streaming algorithm in the case of $\beta = 1$, which reuses the framework of Algorithm 1 but removes the assumption that opt is known.

Define $m = \max_{e\in V, i\in[k]} f((e,i))$, we have $m \leq \mathsf{opt} \leq n \cdot m$. Therefore we use the value $v = (1+\epsilon')^j$ with $m \leq (1+\epsilon')^j \leq n \cdot m$, $j \in \mathbb{Z}_+$ to guess the value of opt by showing that there exits v such that $(1-\epsilon')\mathsf{opt} \leq v \leq \mathsf{opt}$. However, in order to find m, it's necessary to require at least one pass over V. Therefore, we adapt the dynamic update method, which was first proposed by Badanidiyuru et al. (2014) and then widely used for streaming algorithms for both submodular and k-submodular optimizations (Yang et al. 2019; Huang et al. 2020; Nguyen and Thai 2020). It updates $m = \max\{m, \max_{i\in[k]} f((e,i))\}$ with already observed element e, to determine the range of the guessed optimal value.

This method can help algorithm maintain the good estimation of the optimal solution if that range shifts forward when next elements are observed. We implement this method by using variables $\mathbf{s}_j^{t_j}$ and t_j to store the candidate solution and the number of its elements with respect to j.

We set the value of α by using Theorem 1 which provides the best approximation guarantees. The value of ϵ' is set to several times ϵ to reduce the complexity but still ensure approximation ratios. The detail of our algorithm is presented in Algorithm 2.

Lemma 2 *After ending of the main loop of the Algorithm 2, there exists a number $j \in \mathbb{Z}_+$ that $v = (1+\epsilon')^j \in O$ satisfying $(1-\epsilon')\mathsf{opt} \leq v \leq \mathsf{opt}$.*

Proof Define $m = f((e_{max}, i_{max}))$. Due to k-submodularity of f, we have:

$$m \leq \mathsf{opt} = f(\mathbf{o}) \leq \sum_{e\in supp(\mathbf{o})} f(e, \mathbf{o}(e)) \leq m \cdot n$$

Algorithm 2: Deterministic streaming algorithm

Input: a k-submodular function f, $B > 0$, $\epsilon \in (0, 1/5)$.
Output: a solution $\mathbf{s}$

if f *is monotone* **then**
  $\alpha \leftarrow \frac{1}{2}, \epsilon' \leftarrow 4\epsilon$
else
  $\alpha \leftarrow \frac{2}{5}, \epsilon' \leftarrow 5\epsilon$
end
$(e_{max}, i_{max}) \leftarrow (\emptyset, 1), t_j \leftarrow 0 \; \forall j \in \mathbb{Z}^+$
foreach $e \in V$ **do**
  $i_e \leftarrow \arg\max_{i \in [k]} f((e, i))$
  $(e_{max}, i_{max}) \leftarrow \arg\max_{(e_1, i_1) \in \{(e_{max}, i_{max}), (e, i_e)\}} f((e_1, i_1))$
  $O \leftarrow \{j | f((e_{max}, i_{max})) \leq (1+\epsilon')^j \leq Bf((e_{max}, i_{max})), j \in \mathbb{Z}_+\}$
  for $j \in O$ **do**
    if $c(\mathbf{s}_j^{t_j}) + c(e) \leq B$ **then**
      $i' \leftarrow \arg\max_{i \in [k]} f(\mathbf{s}_j^{t_j} \sqcup (e, i))$
      if $\frac{f(\mathbf{s}_j^{t_j} \sqcup (e, i'))}{c(\mathbf{s}_j^{t_j}) + c(e)} \geq \frac{\alpha(1+\epsilon')^j}{B}$ **then**
        $\mathbf{s}_j^{t_j+1} \leftarrow \mathbf{s}^{t_j} \sqcup (e, i')$
        $t_j \leftarrow t_j + 1$
      end
    end
  end
end
return $\arg\max_{\mathbf{s} \in \{\{\mathbf{s}_j^{t_j}; j \in O\}, (e_{max}, i_{max})\}} f(\mathbf{s})$ if f is monotone,
$\arg\max_{\mathbf{s} \in \{\{\mathbf{s}_j^{i}: i \leq t_j, j \in O\}, (e_{max}, i_{max})\}} f(\mathbf{s})$ if f is non-monotone

Let $j = \lfloor \log_{1+\epsilon'} \mathsf{opt} \rfloor$, we have $v = (1+\epsilon')^j \leq \mathsf{opt} \leq n.m$, and

$$v \geq (1+\epsilon')^{\log_{1+\epsilon'}(\mathsf{opt})-1} = \frac{\mathsf{opt}}{1+\epsilon'} \geq \mathsf{opt}(1-\epsilon') \qquad \square$$

The performance of Algorithm 2 is claimed in the following Theorem.

Theorem 2 *Algorithm 2 is a single-pass streaming algorithm that has* $O\left(\frac{kn}{\epsilon}\log n\right)$ *query complexity,* $O\left(\frac{n}{\epsilon}\log n\right)$ *space complexity and provides an approximation ratio of* $\frac{1}{4} - \epsilon$ *when* f *is monotone and* $\frac{1}{5} - \epsilon$ *when* f *is non-monotone.*

Proof The size of O is at most $\frac{1}{\epsilon'}\log n$, finding each $\mathbf{s}_j^{t_j}$ takes at most $O(kn)$ queries and $\mathbf{s}_j^{t_j}$ includes at most n elements. Therefore, the query complexity is $O\left(\frac{kn}{\epsilon}\log n\right)$ and total space complexity is $O\left(\frac{n}{\epsilon}\log n\right)$.

By Lemma 2, there exists an integer number $j \in \mathbb{Z}_+$ that $v = (1+\epsilon')^j \in O$ satisfies $(1-\epsilon')\mathsf{opt} \leq v \leq \mathsf{opt}$. Apply Theorem 1, for the monotone case we have:

$$f(\mathbf{s}) \geq \frac{1}{4}v \geq \frac{1}{4}(1-\epsilon')\mathsf{opt} = \left(\frac{1}{4} - \epsilon\right)\mathsf{opt} \tag{7}$$

and for the non-monotone case:

$$f(\mathbf{s}) \geq \frac{1}{5}v \geq \frac{1}{5}(1-\epsilon')\mathsf{opt} = \left(\frac{1}{5}-\epsilon\right)\mathsf{opt} \tag{8}$$

The theorem is proved. □

4 A random streaming algorithm for general case

Since in this case each element having multiple different costs makes the problem more challenging and we cannot apply previous algorithms. We introduce a one pass streaming which provides approximation ratio in expectation for BkSM problem. At the core of our algorithm, we introduce a new probability distribution for choosing a position for each element to establish the relationship among $\mathbf{o}$, $\mathbf{o}^j$ and $\mathbf{s}^j$ (Lemma 3) and analyze the performance of our algorithm. Besides, we also use the predefined threshold to filter high-value elements into the candidate solutions and the maximum singleton value to give the final solution.

Similar to the previous section, we first introduce a simplified version of streaming algorithm when the optimal solution is known in advance.

4.1 A random algorithm with known the optimal value

This algorithm also receives inputs $\alpha \in (0, 1)$ and v that $v \leq \mathsf{opt}$. We use the same notations as in Sect. 3. This algorithm also requires one pass over V.

The algorithm initializes an empty k-set $\mathbf{s}^0$ and subsequently updates the solution after one-pass over V. Differing from Algorithm 1, for each $e \in V$ being observed, the algorithm finds a set collection J that contains some positions satisfying the total cost at most B and provides the ***ratio*** of increment of objective function per cost at least a given threshold, i.e,

$$J = \left\{ i \in [k] : c(\mathbf{s}^t) + c_i(e) \leq B \text{ and } \frac{f(\mathbf{s}^t \sqcup (e,i)) - f(\mathbf{s}^t)}{c_i(e)} \geq \frac{\alpha v}{B} \right\} \tag{9}$$

These constraints help the algorithm eliminate the positions having low such ratio. If $J \neq \emptyset$, the algorithm puts e into the set i of $\mathbf{s}^t$ with a probability:

$$\frac{p_i^{|J|-1}}{T} = \frac{\left(\frac{f(\mathbf{s}^t \sqcup (e,i)) - f(\mathbf{s}^t)}{c_i(e)}\right)^{|J|-1}}{\sum_{i \in J}\left(\frac{f(\mathbf{s}^t \sqcup (e,i)) - f(\mathbf{s}^t)}{c_i(e)}\right)^{|J|-1}} \tag{10}$$

Simultaneously, the algorithm finds the maximum singleton value (e_{max}, i_{max}) by updating the current maximal value from the set of observed elements. As Algorithm 3, the algorithm also uses (e_{max}, i_{max}) as one of candidate solutions and finds the best among them. The full detail of this algorithm is described in Algorithm 3.

Algorithm 3: RanStreamWithOpt(f, opt, α)

Input: a k-submodular function f, $B > 0$, $\epsilon \in (0, 1)$ $\alpha \in (0, 1]$, v that $v \leq \text{opt}$
Output: a solution $\mathbf{s}$

$\mathbf{s}_0 \leftarrow \mathbf{0}$, $t \leftarrow 0$
$(e_{max}, i_{max}) \leftarrow (\emptyset, 1)$, $t_j \leftarrow 0 \; \forall j \in \mathbb{Z}^+$
foreach $e \in V$ **do**
  $i_e \leftarrow \arg\max_{i \in [k]} f((e, i))$
  $(e_{max}, i_{max}) \leftarrow \arg\max_{(e_1, i_1) \in \{(e_{max}, i_{max}), (e, i_e)\}} f((e_1, i_1))$
  $J \leftarrow \emptyset$
  foreach $i \in [k]$ **do**
    if $c(\boldsymbol{s}^t) + c_i(e) \leq B$ *and* $\frac{f(\boldsymbol{s}^t \sqcup (e,i)) - f(\boldsymbol{s}^t)}{c_i(e)} \geq \frac{\alpha v}{B}$ **then**
      $p_i \leftarrow \frac{f(\mathbf{s}^t \sqcup (e,i)) - f(\mathbf{s}^t)}{c_i(e)}$
      $J \leftarrow J \cup \{i\}$
    end
  end
  if $J \neq \emptyset$ **then**
    $T \leftarrow \sum_{i \in J} p_i^{|J|-1}$
    Select a possition $i \in J$ with probability $\frac{p_i^{|J|-1}}{T}$
    $\mathbf{s}^{t+1} \leftarrow \mathbf{s}^t \sqcup (e, i)$
    $t \leftarrow t + 1$
  end
end
return $\arg\max_{\mathbf{s} \in \{\mathbf{s}^t, (e_{max}, i_{max})\}} f(\mathbf{s})$ if f is monotone, $\arg\max_{\mathbf{s} \in \{\{\mathbf{s}^j : j \leq t\}, (e_{max}, i_{max})\}} f(\mathbf{s})$ if f is non-monotone

Denote by $e \in supp(\mathbf{o})$ as a ***bad element*** if

$$\frac{f(\mathbf{s}^{t_e} \sqcup (e, \mathbf{o}(e)))}{c(\mathbf{s}^{t_e}) + c_{\mathbf{o}(e)}(e)} \geq \frac{\alpha v}{B} \text{ and } c(\mathbf{s}^{t_e}) + c_{\mathbf{o}(e)}(e) > B \tag{11}$$

where $\mathbf{s}^{t_e}$ is the candidate solution obtained right before e arrives.

Lemma 3 provides the relationship among $\mathbf{o}$, $\mathbf{o}^j$ and $\mathbf{s}^j$ that plays an important role for analyzing the performance of the algorithm.

Lemma 3 *Assume that there is no bad element. In the Algorithm 3, we have:*

– *If f is monotone, then:*

$$f(\boldsymbol{o}^{j-1}) - \mathbb{E}[f(\boldsymbol{o}^j)] \leq \beta \left(1 - \frac{1}{k}\right) (\mathbb{E}[f(\boldsymbol{s}^j)] - f(\boldsymbol{s}^{j-1}) + \frac{\alpha v c_{j^*}(e^j)}{kB} \tag{12}$$

– *If f is non-monotone, then:*

$$f(\boldsymbol{o}^{j-1}) - \mathbb{E}[f(\boldsymbol{o}^j)] \leq 2\beta \left(1 - \frac{1}{k}\right) (\mathbb{E}[f(\boldsymbol{s}^j)] - f(\boldsymbol{s}^{j-1})) + \frac{2\alpha v c_{j^*}(e^j)}{kB} \tag{13}$$

Proof We consider following cases:

Case 1 If f is monotone. If $J = \emptyset$ the algorithm returns current $\mathbf{s}^j$ so we consider the case $J \neq \emptyset$. We further consider two following sub-cases.

Case 1.1 $e^j \notin supp(\mathbf{o})$, due to the monotonicity of f we have:

$$f(\mathbf{o}^{j-1}) - f(\mathbf{o}^j) \leq 0 < \beta\left(1 - \frac{1}{k}\right)\left(f(\mathbf{s}^j) - f(\mathbf{s}^{j-1})\right)$$

Case 1.2 If $e^j \in supp(\mathbf{o})$. We reuse the notations $\mathbf{o}^j, \mathbf{o}^{j-1/2}, \mathbf{s}^j, \mathbf{s}^{j-1/2}$ as in Sect. 2 and define $\mathbf{o}_l^j, \mathbf{s}_l^j, j^*$ as follows: $\mathbf{o}_l^j(e) = \mathbf{o}^j(e), \forall e \in V \setminus \{e^j\}$ and $\mathbf{o}_l^j(e^j) = l$; $\mathbf{s}_l^j = \mathbf{s}^{j-1} \sqcup (e^j, l)$; $j^* = \mathbf{o}(e^j)$. Since there is no bad element, we consider two following sub-cases.

- If $j^* \in J$, i.e, $\frac{f(\mathbf{s}^{j-1} \sqcup (e^j, j^*))}{c(\mathbf{s}^{j-1}) + c_{j^*}(e^j)} \geq \frac{\alpha v}{B}$ and $c(\mathbf{s}^j) + c_{j^*}(e^j) \leq B$. If $|J| = 1$, we have $f(\mathbf{o}^{j-1}) - f(\mathbf{o}^j) = 0 < \beta\left(1 - \frac{1}{k}\right)\left(f(\mathbf{s}^j) - f(\mathbf{s}^{j-1})\right)$ so we consider $|J| > 1$. In this case we have:

$$f(\mathbf{o}^{j-1}) - \mathbb{E}[f(\mathbf{o}^j)] = \sum_{l \in J}(f(\mathbf{o}^{j-1}) - f(\mathbf{o}_l^j))\frac{p_l^{|J|-1}}{T} \tag{14}$$

$$= \sum_{l \in J \setminus \{j^*\}}(f(\mathbf{o}^{j-1}) - f(\mathbf{o}_l^j))\frac{p_l^{|J|-1}}{T} \tag{15}$$

$$\leq \sum_{l \in J \setminus \{j^*\}}(f(\mathbf{o}^{j-1}) - f(\mathbf{o}^{j-1/2}))\frac{p_j^{|J|-1}}{T} \tag{16}$$

$$\leq \sum_{l \in J \setminus \{j^*\}}(f(\mathbf{s}_{j^*}^j) - f(\mathbf{s}^{j-1}))\frac{p_l^{|J|-1}}{T} \tag{17}$$

$$= \frac{1}{T}\sum_{l \in J \setminus \{j^*\}} c_{j^*}(e^j) \cdot p_{j^*} \cdot p_l^{|J|-1} = \frac{c_{j^*}(e^j)}{T}\sum_{j \in J \setminus \{j^*\}} p_{j^*} \cdot \underbrace{p_l \cdot p_l}_{|J|-1} \tag{18}$$

$$\leq \frac{c_{j^*}(e^j)}{T}\sum_{l \in J \setminus \{j^*\}}\frac{1}{|J|}(p_{j^*}^{|J|} + \underbrace{p_l^{|J|} + \ldots + p_l^{|J|}}_{|J|-1}) \text{ (By applying AG-GM inequality)} \tag{19}$$

$$= c_{j^*}(e^j)\left(1 - \frac{1}{|J|}\right)\sum_{l \in J}\frac{p_j^{|J|}}{T} \tag{20}$$

$$\leq c_{j^*}(e^j)\left(1 - \frac{1}{k}\right)\sum_{l \in J}\frac{f(\mathbf{s}_l^j) - f(\mathbf{s}^{j-1})}{c_l(e^j)}\frac{p_l^{|J|-1}}{T} \tag{21}$$

$$\leq \beta\left(1 - \frac{1}{k}\right)(\mathbb{E}[f(\mathbf{s}^j)] - f(\mathbf{s}^{j-1})) \tag{22}$$

- If $j^* \notin J$, i.e, $\frac{f(\mathbf{s}^{j-1} \sqcup (e^j, j^*)) - f(\mathbf{s}^{j-1})}{c_{j^*}(e^j)} < \frac{\alpha v}{B}$, then $p_{j^*} \leq p_l, \forall l \in J$. Similar to the transform from (14) to (22), we have:

$$\begin{aligned} c_{j^*}(e^j) p_{j^*} &= c_{j^*}(e^j) \sum_{l \in J} p_{j^*} \frac{p_l^{|J|-1}}{T} \\ &\leq c_{j^*}(e^j) \sum_{l \in J} p_l \frac{p_l^{|J|-1}}{T} \\ &\leq \beta(\mathbb{E}[f(\mathbf{s}^j)] - f(\mathbf{s}^{j-1})) \end{aligned}$$

Therefore,

$$\begin{aligned} f(\mathbf{o}^{j-1}) - \mathbb{E}[f(\mathbf{o}^j)] &\leq f(\mathbf{s}^{j-1} \sqcup (e^j, j^*)) - f(\mathbf{s}^{j-1}) = c_{j^*}(e) p_{j^*} \\ &\leq \beta(1 - \frac{1}{k})(\mathbb{E}[f(\mathbf{s}^j)] - f(\mathbf{s}^{j-1})) + \frac{c_{j^*}(e^j) p_{j^*}}{k} \\ &\leq \beta \left(1 - \frac{1}{k}\right) (\mathbb{E}[f(\mathbf{s}^j)] - f(\mathbf{s}^{j-1})) + \frac{c_{j^*}(e^j) \alpha v}{kB} \end{aligned}$$

Case 2 If f is non-monotone, similar to the monotone case, we only consider $J \neq \emptyset$ and two following cases:

Case 2.1 If $e^j \notin supp(\mathbf{o})$, we consider two sub-cases:

- If there exist $l \in [k] \setminus J$ satisfying $\frac{f(\mathbf{s}^{j-1} \sqcup (e^j, l)) - f(\mathbf{s}^{j-1})}{c_l(e^j)} < \frac{\alpha v}{B}$ and $c(\mathbf{s}^{j-1}) + c_l(e) \leq B$, then $p_l < p_x \forall x \in J$. By the *pairwise monotonicity* and *k-submodularity* properties of f, we obtain:

$$\begin{aligned} f(\mathbf{o}^{j-1}) - \mathbb{E}[f(\mathbf{o}^j)] &= f(\mathbf{o}_l^j) - f(\mathbf{o}^{j-1}) - (\mathbb{E}[f(\mathbf{o}^j)] + f(\mathbf{o}_l^j) - 2f(\mathbf{o}^{j-1})) \\ &\leq f(\mathbf{o}_l^j) - f(\mathbf{o}^{j-1}) \leq f(\mathbf{s}_l^j) - f(\mathbf{s}^{j-1}) \\ &< \mathbb{E}[f(\mathbf{s}^j)] - f(\mathbf{s}^{j-1}) \end{aligned}$$

- If there does not exist such integer $l \in [k] \setminus J$, we define a permutation $\pi : J \mapsto J$ such that $\pi(i) \neq i, \forall i \in J$. We have:

$$\begin{aligned} f(\mathbf{o}^{j-1}) - \mathbb{E}[f(\mathbf{o}^j)] &= \sum_{i \in J} (f(\mathbf{o}^{j-1}) - f(\mathbf{o}_i^j)) \frac{p_i^{|J|-1}}{T} \\ &= \sum_{i \in J} \Big(f(\mathbf{o}_{\pi(i)}^j) - f(\mathbf{o}^{j-1}) - (f(\mathbf{o}_i^j) \\ &\quad + f(\mathbf{o}_{\pi(i)}^j) - 2f(\mathbf{o}^{j-1})) \Big) \frac{p_i^{|J|-1}}{T} \\ &\leq \sum_{i \in J} (f(\mathbf{o}_{\pi(i)}^j) - f(\mathbf{o}^{j-1})) \frac{p_i^{|J|-1}}{T} \end{aligned}$$

$$
\begin{aligned}
&\leq \sum_{i\in J}(f(\mathbf{s}^j_{\pi(i)}) - f(\mathbf{s}^{j-1}))\frac{p_i^{|J|-1}}{T}\\
&\leq \sum_{i\in J} c_{\pi(i)}(e^j)p_{\pi(i)}\frac{p_i^{|J|-1}}{T} \leq \frac{c_{max}}{T}\sum_{i\in J} p_{\pi(i)}p_i^{|J|-1}\\
&= \frac{c_{max}}{T}\sum_{i\in J} p_{\pi(i)}\cdot \underbrace{p_i\cdot p_i}_{|J|-1}\\
&\leq \frac{c_{max}}{T}\sum_{i\in J}\frac{1}{|J|}(p_{\pi(i)}^{|J|}\\
&\quad + \underbrace{p_i^{|J|}+\ldots+p_i^{|J|}}_{|J|-1}) \text{ (By applying AG-GM inequality)}\\
&\leq c_{max}\sum_{i\in J}\frac{p_i^{|J|}}{T}\\
&\leq \beta(\mathbb{E}[f(\mathbf{s}^j)] - f(\mathbf{s}^{j-1}))
\end{aligned}
$$

Case 2.2 If $e^j \in supp(\mathbf{o})$. Similar to the monotone case with notice that $f(\mathbf{o}^{j-1}) = f(\mathbf{o}^j_{j^*})$, we further consider two following sub-cases:

- <u>If</u> $j^* \in J$, i.e, $\frac{f(\mathbf{s}^{j-1}\sqcup(e^j,j^*))}{c(\mathbf{s}^{j-1})+c_{j^*}(e^j)} \geq \frac{\alpha v}{B}$ and $c(\mathbf{s}^j)+c_{j^*}(e^j)\leq B$.

$$
\begin{aligned}
&f(\mathbf{o}^{j-1}) - \mathbb{E}[f(\mathbf{o}^j)]\\
&= \sum_{l\in J\setminus\{j^*\}}(f(\mathbf{o}^{j-1}) - f(\mathbf{o}^j_l))\frac{p_l^{|J|-1}}{T}\\
&= \sum_{l\in J\setminus\{j^*\}}(2f(\mathbf{o}^j_{j^*}) - 2f(\mathbf{o}^{j-1/2}) - (f(\mathbf{o}^j_{j^*}) + f(\mathbf{o}^j_l) - 2f(\mathbf{o}^{j-1/2})))\frac{p_l^{|J|-1}}{T}\\
&\leq \sum_{l\in J\setminus\{j^*\}} 2(f(\mathbf{o}^j_{j^*}) - f(\mathbf{o}^{j-1/2}))\frac{p_l^{|J|-1}}{T}\\
&\leq \sum_{l\in J\setminus\{j^*\}} 2(f(\mathbf{s}^j_{j^*}) - f(\mathbf{s}^{j-1}))\frac{p_l^{|J|-1}}{T}\\
&\leq 2\beta\left(1-\frac{1}{k}\right)(\mathbb{E}[f(\mathbf{s}^j)] - f(\mathbf{s}^{j-1}))
\end{aligned}
$$

- <u>If</u> $j^* \notin J$, i.e, $\frac{f(\mathbf{s}^{j-1}\sqcup(e^j,j^*))-f(\mathbf{s}^{j-1})}{c_{j^*}(e^j)} < \frac{\alpha v}{B}$, then $p_{j^*} \leq p_l, \forall l \in J$. Apply the transform as in case 2.1, we have:

$$f(\mathbf{o}^{j-1}) - \mathbb{E}[f(\mathbf{o}^j)] \leq 2\beta(f(\mathbf{s}^{j-1} \sqcup (e^j, j^*)) - f(\mathbf{s}^{j-1})) = 2c_{j^*}(e^j)p_{j^*}$$
$$\leq 2\beta\left(1 - \frac{1}{k}\right)(\mathbb{E}[f(\mathbf{s}^j)] - f(\mathbf{s}^{j-1})) + \frac{2c_{j^*}(e^j)\alpha v}{kB}$$

Combine all cases, in general we obtain the proof. □

Theorem 3 *Algorithm 2 returns a solution* $\boldsymbol{s}$ *satisfying:*

- *If* f *is monotone,* $\mathbb{E}[f(\boldsymbol{s})] \geq \min\{\frac{\alpha}{2}, \frac{(1-\alpha)k}{(1+\beta)k-\beta}\}v$. *The right hand side is maximized to* $\frac{v}{3+\beta-\frac{\beta}{k}}$ *when* $\alpha = \frac{2}{3+\beta-\frac{\beta}{k}}$.
- *If* f *is non-monotone,* $\mathbb{E}[f(\boldsymbol{s})] \geq \min\{\frac{\alpha}{2}, \frac{(1-\alpha)k}{(1+2\beta)k-2\beta}\}v$. *The right hand side is maximized to* $\frac{v}{3+2\beta-\frac{2\beta}{k}}$ *when* $\alpha = \frac{2}{3+2\beta-\frac{2\beta}{k}}$.

Proof If f is monotone, denote by e^t the last addition of the main loop of the algorithm, we consider two sub-cases as follows:

Case 1 There is no bad element e. By applying Lemma 3, we obtain:

$$\begin{aligned}
&v - \mathbb{E}[f(\mathbf{s}^t)] \\
&\leq f(\mathbf{o}) - \mathbb{E}[f(\mathbf{s}^t)] = f(\mathbf{o}) - f(\mathbf{o}^t) + f(\mathbf{o}^t) - \mathbb{E}[f(\mathbf{s}^t)] \\
&= \sum_{j=1}^{t}(f(\mathbf{o}^{j-1}) - f(\mathbf{o}^j)) + \sum_{i=1}^{t}(f(\mathbf{u}_i^t) - f(\mathbf{u}_{i-1}^t)) \\
&\leq \beta\left(1 - \frac{1}{k}\right)\sum_{j=1}^{t}(f(\mathbf{s}^j) - f(\mathbf{s}^{j-1}) + \frac{\alpha v c_{j^*}(e^j)}{kB}) \\
&\quad + \sum_{i=1}^{r}(f(\mathbf{s}^{t_{u_i}} \sqcup (u_i, j_i)) - f(\mathbf{s}^{t_{u_i}}) \\
&\leq \beta\left(1 - \frac{1}{k}\right)\mathbb{E}[f(\mathbf{s}^t)] + \sum_{j=1}^{t}\frac{\alpha v c_{j^*}(e^j)}{kB} + \sum_{i=1}^{r}\frac{\alpha v c(u_i)}{B} \\
&\leq \beta\left(1 - \frac{1}{k}\right)\mathbb{E}[f(\mathbf{s}^t)] + \sum_{e \in supp(o) \cap supp(\mathbf{s}^t)}\frac{\alpha v c_{\mathbf{o}(e)}(e)}{B} \\
&\quad + \sum_{e \in supp(o) \setminus supp(\mathbf{s}^t)}\frac{\alpha v c_{\mathbf{o}(e)}(e)}{B} \leq \beta\left(1 - \frac{1}{k}\right)\mathbb{E}[f(\mathbf{s}^t)] + \alpha v
\end{aligned}$$

This implies that $\mathbb{E}[f(\mathbf{s}^t)] \geq \frac{(1-\alpha)v}{(1+\beta)k-\beta}$.

Case 2 There exists a bad element e. Let $j^* = \mathbf{o}(e)$ we have:

$$f(\mathbf{s}^{t_e} \sqcup (e, j^*)) \geq \frac{c_{j^*}(e)\alpha v}{B} + f(\mathbf{s}^{t_e}) \geq \frac{c_{j^*}(e)\alpha v}{B} + \sum_{j=1}^{t_e}(f(\mathbf{s}^j) - f(\mathbf{s}^{j-1}))$$
$$\geq \frac{(c(\mathbf{s}^{t_e}) + c_{j^*}(e))\alpha v}{B} \geq \alpha v$$

Therefore,

$$f(\mathbf{s}) \geq \max\{f(\mathbf{s}^{t_e}),\, f((e_{max}, i_{max}))\} \tag{23}$$

$$\geq \max\{f(\mathbf{s}^{t_e}),\, f((e, j^*))\} \geq \frac{f(\mathbf{s}^{t_e}) + f((e, j^*))}{2} \tag{24}$$

$$\geq \frac{f(\mathbf{s}^{t_e} \sqcup (e, j^*))}{2} \text{ (Due to the } k\text{-submodularity)} \tag{25}$$

$$\geq \frac{\alpha v}{2} \tag{26}$$

Combine two above cases, we obtain $\mathbb{E}[f(\mathbf{s})] \geq \min\{\frac{\alpha}{2}, \frac{(1-\alpha)k}{(1+\beta)k-\beta}\}v$, $f(\mathbf{s})$ is maximized to $\frac{v}{3+\beta-\frac{\beta}{k}}$ when $\alpha = \frac{2}{3+\beta-\frac{\beta}{k}}$.

If f is non-monotone, similar to the monotone case and combine with Lemma 3, we obtain the proof. □

4.2 A random streaming algorithm

In this section we remove the assumption that the optimal solution is known and present the random streaming algorithm which reuses the framework of Algorithm 3.

Similar to the Algorithm 2, we use the method in Badanidiyuru et al. (2014) to estimate opt. We assume that β is known in advance. This is feasible because we can calculate the value of β in $O(kn)$. We set α according to the properties of f to provide the best performance of the algorithm. The algorithm continuously updates $O \leftarrow \{j | f((e_{max}, i_{max})) \leq (1+\epsilon)^j \leq Bf((e_{max}, i_{max})), j \in \mathbb{Z}_+\}$ in order to estimate the value of maximal singleton and uses $\mathbf{s}_j^{t_j}$ and t_j to save candidate solutions, which is updated by using the probability distribution as in Algorithm 3 with $(1+\epsilon)^j$ as an estimation of the optimal solution. The algorithm finally compares all candidate solutions to select the best one. The details of algorithm is presented in Algorithm 4.

Theorem 4 *Algorithm 4 is one pass streaming algorithm that has* $O\left(\frac{kn}{\epsilon}\log n\right)$ *query complexity,* $O\left(\frac{n}{\epsilon}\log n\right)$ *space complexity and*

- *If* f *is monotone,* $\mathbb{E}[f(\boldsymbol{s})] \geq \left(\min\{\frac{\alpha}{2}, \frac{(1-\alpha)k}{(1+\beta)k-\beta}\} - \epsilon\right)$ opt. *The right hand side is maximized to* $\left(\frac{1}{3+\beta-\frac{\beta}{k}} - \epsilon\right)$ opt *when* $\alpha = \frac{2}{3+\beta-\frac{\beta}{k}}$.
- *If* f *is non-monotone,* $\mathbb{E}[f(\boldsymbol{s})] \geq \left(\min\{\frac{\alpha}{2}, \frac{(1-\alpha)k}{(1+2\beta)k-2\beta}\} - \epsilon\right)$ opt. *The right hand side is maximized to* $\left(\frac{1}{3+2\beta-\frac{2\beta}{k}} - \epsilon\right)$ opt *when* $\alpha = \frac{2}{3+2\beta-\frac{2\beta}{k}}$.

Proof By Lemma 2, there exists $j \in \mathbb{Z}_+$ that $v = (1+\epsilon)^j \in O$ satisfies $(1-\epsilon)\mathsf{opt} \leq v \leq \mathsf{opt}$. Similar to the proof of Theorem 2, we easily show the query and space complexities of Algorithm 4. Using similar arguments of the proof of Theorem 3, for the monotone case:

Algorithm 4: Random streaming algorithm

Input: a k-submodular function f, $B > 0$, $\epsilon \in (0, 1)$, $\alpha \in (0, 1]$
Output: a solution $\mathbf{s}$

$\mathbf{s}^0 \leftarrow \mathbf{0}$
$(e_{max}, i_{max}) \leftarrow (\emptyset, 1), t_j \leftarrow 0 \; \forall j \in \mathbb{Z}^+$
foreach $e \in V$ **do**
  $i_e \leftarrow \arg\max_{i \in [k]} f((e, i))$
  $(e_{max}, i_{max}) \leftarrow \arg\max_{(e_1, i_1) \in \{(e_{max}, i_{max}), (e, i_e)\}} f((e_1, i_1))$
  $O \leftarrow \{j | f((e_{max}, i_{max})) \leq (1+\epsilon)^j \leq B f((e_{max}, i_{max})), j \in \mathbb{Z}_+\}$
  foreach $j \in O$ **do**
    $J \leftarrow \emptyset$
    foreach $i \in [k]$ **do**
      if $c(\mathbf{s}_j^{t_j}) + c_i(e) \leq B$ *and* $\frac{f(\mathbf{s}_j^{t_j} \sqcup (e,i)) - f(\mathbf{s}_j^{t_j})}{c_i(e)} \geq \frac{\alpha(1+\epsilon)^j}{B}$ **then**
        $p_i \leftarrow \frac{f(\mathbf{s}_j^{t_j} \sqcup (e,i)) - f(\mathbf{s}_j^{t_j})}{c_i(e)}$
        $J \leftarrow J \cup \{i\}$
      end
    end
    $T \leftarrow \sum_{i \in J} p_i^{|J|-1}$
    Select a possition $i \in J$ with probability $\frac{p_i^{|J|-1}}{T}$
    $\mathbf{s}_j^{t_j+1} \leftarrow \mathbf{s}_j^{t_j} \sqcup (e, i)$
    $t_j \leftarrow t_j + 1$
  end
end
return $\arg\max_{\mathbf{s} \in \{\{\mathbf{s}_j^{t_j} : j \in O\}, (e_{max}, i_{max})\}} f(\mathbf{s})$ if f is monotone,
$\arg\max_{\mathbf{s} \in \{\{\mathbf{s}_j^{i} : j \in O, i \leq t_j\}, (e_{max}, i_{max})\}} f(\mathbf{s})$ if f is non-monotone

$$\mathbb{E}[f(\mathbf{s})] \geq \min\{\frac{\alpha}{2}, \frac{(1-\alpha)k}{(1+\beta)k-\beta}\}v \geq \left(\min\{\frac{\alpha}{2}, \frac{(1-\alpha)k}{(1+\beta)k-\beta}\} - \epsilon\right)\mathsf{opt}$$

and if $\alpha = \frac{2}{3+\beta-\frac{\beta}{k}}$, we have:

$$\mathbb{E}[f(\mathbf{s})] \geq \frac{v}{3+\beta-\frac{\beta}{k}} \geq \left(\frac{1}{3+\beta-\frac{\beta}{k}} - \epsilon\right)\mathsf{opt}$$

For the non-monotone case we also obtain the proof by applying the same arguments. □

Remark 1 In the case of $\beta = 1$, the algorithm returns an approximation ratio of $\frac{1}{4-1/k} - \epsilon$ when f is monotone and $\frac{1}{5-2/k} - \epsilon$ when f is non-monotone in expectation. Thus, these approximation ratios are better than that of Algorithm 2 in expectation.

5 Experiments

In this section, we compare the performance of our algorithms with a greedy algorithm in two applications of BkSM: Influence Maximization (IM) with k topics and Sensor Placement on *three major metrics*: value of objective function, the number of queries and running time. Besides, we further show the trade-off between the solution quality and the number of queries of algorithms with various settings of ϵ.

5.1 The Greedy algorithm

Since existing algorithms for k-submodular maximization problems cannot be applied directly to BkSM, we adapt the recent Greedy algorithm (Ohsaka and Yoshida 2015) which is the best current non-streaming algorithm with some modifications. The Greedy algorithm iteratively adds a pair (e, i) into the current solution $\mathbf{s}$, which maximizes the marginal gain per its cost $\frac{f(\mathbf{s}\sqcup(e,i))-f(\mathbf{s})}{c_i(e)}$ until there is no remaining cost to add any element. The algorithm has $O(k^2n^2)$ query complexity. The pseudo-code is presented in Algorithm 5.

Algorithm 5: Greedy algorithm

Input: V, a k-submodular function f, $B > 0$
Output: a solution $\mathbf{s}$

$\mathbf{s} \leftarrow \mathbf{0}$, $U \leftarrow \{(e, i) | e \in V, i \in [k]\}$
while $U \neq \emptyset$ **do**
  $(e_m, i_m) \leftarrow \arg\max_{(e,i)\in U} \frac{f(\mathbf{s}\sqcup(e,i))-f(\mathbf{s})}{c_i(e)}$
  if $c(s) + c_{i_m}(e) > B$ **then**
    $U \leftarrow U \setminus (e_m, i_m)$
  else
    $\mathbf{s} \leftarrow \mathbf{s} \sqcup (e_m, i_m)$
    $U \leftarrow U \setminus (e_m, i_m)$
  end
end
return s

5.2 Influence Maximization with k topics subject to a budget constraint

We first recap the information diffusion model, called Linear Threshold (LT) model (Kempe et al. 2003; Nguyen and Thai 2020) and then define the Influence Maximization with k topics subject to the budget constraint problem (IMkB) under this model.

LT model. In this model, a social network is modeled by a directed graph $G = (V, E)$, where V, E represent a set of users and a set of 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 the strength of influence from u to v on the i-th 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 $\sum_{\text{active node v}} w^i(v, u) \geq \theta^i(u)$.

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

$$\sigma(\mathbf{s}) = \mathbb{E}[|\cup_{i\in[k]} \sigma_i(S_i)|] \tag{27}$$

where $\sigma_i(S_i)$ is a random variable representing the set of active users for topic i with seed S_i. The IMkB problem is formally defined as follows:

Definition 2 (IMkB *problem*) Assume that each user u has a cost $c_i(u)$ for i-th topic which manifests how hard it is to initially influence the respective person for that topic. Given the budget B, the problem asks to find a seed set $\mathbf{s}$ with $c(\mathbf{s}) \leq B$ so that $\sigma(\mathbf{s})$ is maximal.

Experiment settings We use the Facebook social network dataset from SNAP (Leskovec and Krevl 2014). The network contains 4,039 nodes and 88,234 edges. Weights $\{w^i(u, v)\}_{i\in[k]}$ of the edge (u, v) are randomly selected from a set $\{\frac{1}{kN(v)}, \frac{2}{kN(v)}, \ldots, \frac{k}{kN(v)}\}$ according to the recent work (Nguyen and Thai 2020), where d_v is in-degree of v.

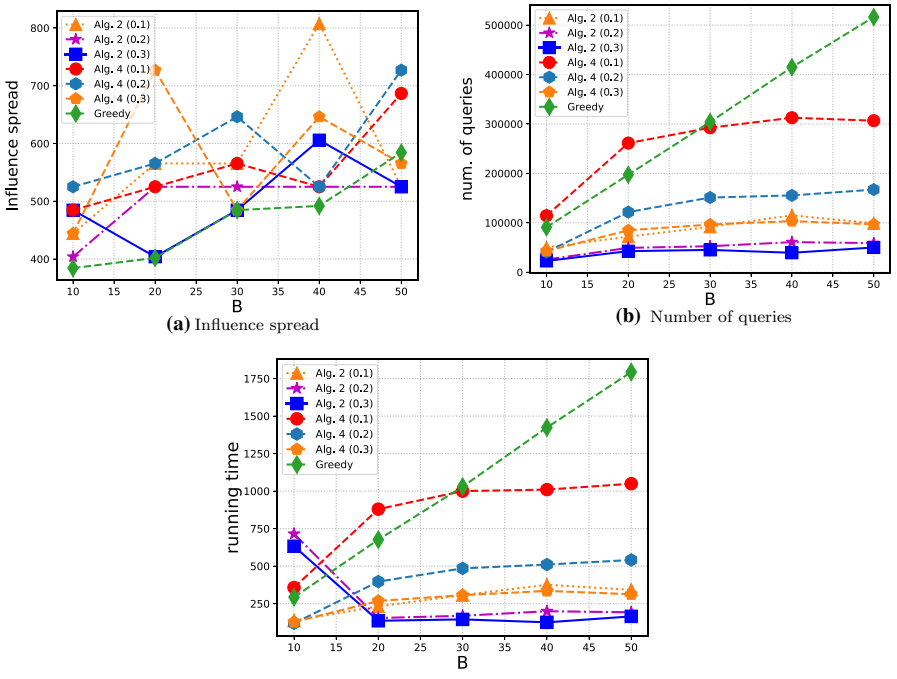
Since the computation of $\sigma(\cdot)$ is #P-hard (Chen et al. 2010), we adapt the sampling method in Nguyen and Thai (2020); Borgs et al. (2014) to give an estimation $\hat{\sigma}(\cdot)$ with a (λ, δ)-approximation that is:

$$\Pr[(1+\lambda)\sigma(\mathbf{s}) \geq \hat{\sigma}(\mathbf{s}) \geq (1-\lambda)\sigma(\mathbf{s})] \geq 1 - \delta \tag{28}$$

In algorithms, we set parameters $\lambda = 0.5, \delta = 0.2$ and $k = 3$ as in Nguyen and Thai (2020). In our algorithms, we set ϵ on varying in $\{0.1, 0.2, 0.3\}$ to show a trade-off between solution quality and number of queries. The costs of each element e are established in the following two cases:

- *Case 1* $\beta = 1$, we set $c_i(e) = c_j(e) = c(e), i, j \in [k]$ with $c(e)$ is calculated under the Normalized Linear model with the support $[1, 2]$ according to recent works Nguyen and Zheng (2013) and Li et al. (2019) and the budget B varies in $\{10, 20, 30, 40, 50\}$.
- *Case 2: General* β $c_i(u)$ is also calculated under the Normalized Linear model with the support $[1, 2]$ and the budget B varies in $\{10, 20, 30, 40, 50\}$. For Algorithm 4, $\alpha = \frac{2}{3+\beta-\frac{\beta}{k}}$ if f is monotone and $\alpha = \frac{2}{3+2\beta-\frac{2\beta}{k}}$ if f is non-monotone with $\beta = 2$.

Experiment results. For the purpose of providing a comprehensive experiment, we divide the experiment into two cases: the special case when $\beta = 1$ and the general case. Figure 1 shows the performance of algorithms for $\beta = 1$. We denote Algorithm



(a) Influence spread

(b) Number of queries

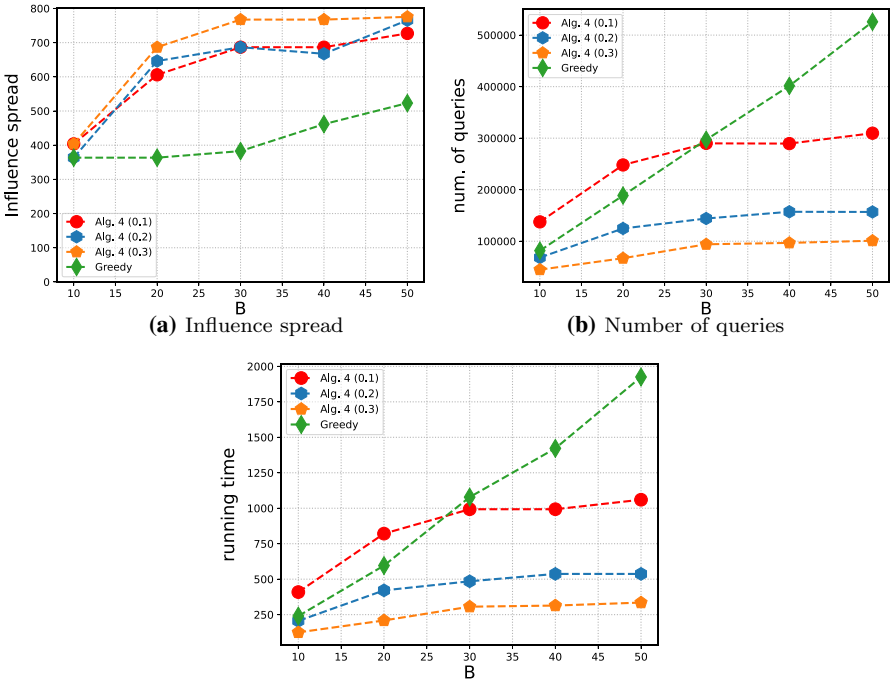
(c) Running time

Fig. 1 Performance of algorithms for IMkB when $\beta = 1$: **a** Influence spread, **b** Number of queries, **c** Running time

x with $\epsilon = y$ by Alg.$x(y)$. With solution quality (influence spread), our algorithms outperform Greedy algorithm in most cases. For the query complexity, our algorithms totally outperform Greedy algorithm by a large gap except Alg. 4(0.1) with $B = 10, 20$. They require up to 10 times fewer queries than the Greedy algorithm.

Compare Algorithm 2 with Algorithm 4, we can find that Algorithm 4 provides the better solution than Algorithm 2 with the same value of ϵ for most cases. This is consistent with the theoretical analysis and the discussion in Remark 1. However, Algorithm 4 requires more queries and running time than Algorithm 2. It might be because of Algorithm 2 that reaches to the budget B faster than Algorithm 4.

Figure 2 shows the performance of algorithms for the general case. Algorithm 2 can not adapt for IMkB in this case, therefore there is no plot of this algorithm in Fig. 2. Again, our random streaming algorithm gives better quality solutions and takes fewer queries and running time than Greedy algorithm. This consists of results of the case $\beta = 1$. We now investigate the affect of ϵ on the performance of these algorithms and the trade-off between solution quality, number of queries and running time of our streaming ones. As ϵ increases, our streaming algorithms tend to take fewer queries and running time but give lower quality of solutions. This is more clearly reflected for the general case in Fig. 2.



(a) Influence spread **(b)** Number of queries

(c) Running time

Fig. 2 Performance of algorithms for IMkB in general case: **a** Influence spread, **b** Number of queries, **c** Running time

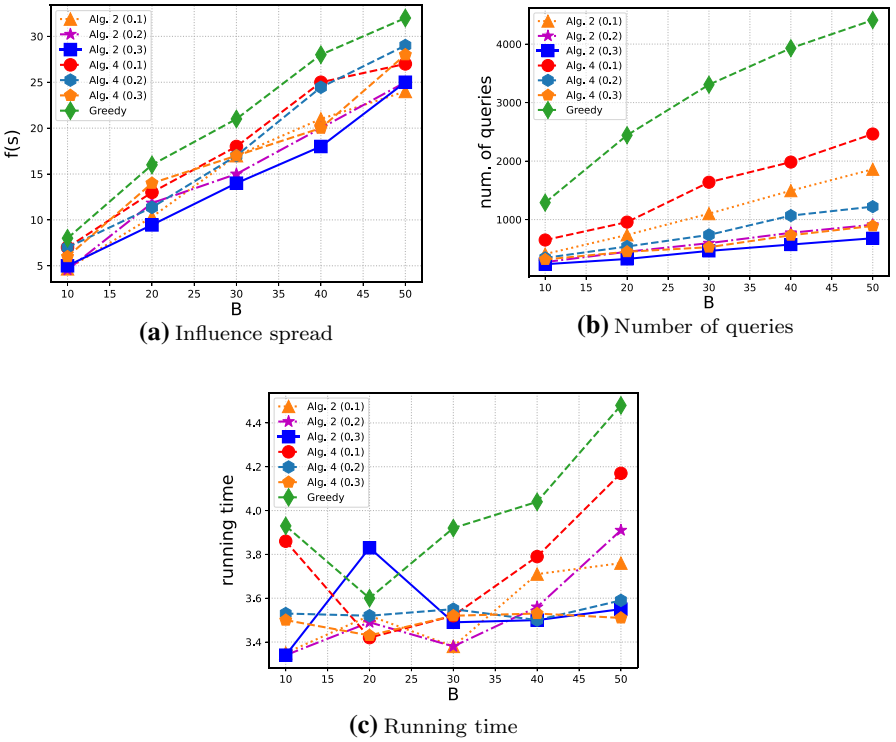
5.3 Sensor placement with *k* types of measures subject to a budget constraint

We further study the performance of algorithms for *Sensor placement with k types of measures subject to a budget constraint* (SPkB) problem. In this problem, we have k types of sensors for different measures and a set V of n locations, each of which can be instrumented with only one sensor. Denote by X_e^i a random variable representing the observation collected from a sensor of kind i and the information gained of a k-set $\mathbf{s}$ is

$$f(\mathbf{s}) = H(\cup_{e \in supp(\mathbf{s})}\{X_e^i\}) \tag{29}$$

where H is *entropy function*. The function f is monotone and k-submodular Ohsaka and Yoshida (2015). We assume that allocating of a sensor to each location has a different cost depending on its position and the kind of sensor. Given the budget B, the SPkB problem aims at allocating sensors to maximize the information gained with the total cost is at most B.

Experiment settings As previous works (Ohsaka and Yoshida 2015; Nguyen and Thai 2020), we use Intel Lab dataset (Bodik et al. 2004). This contains a log approximately 2.3 million readings collected from 54 sensors deployed in the Intel Berkeley research



(a) Influence spread

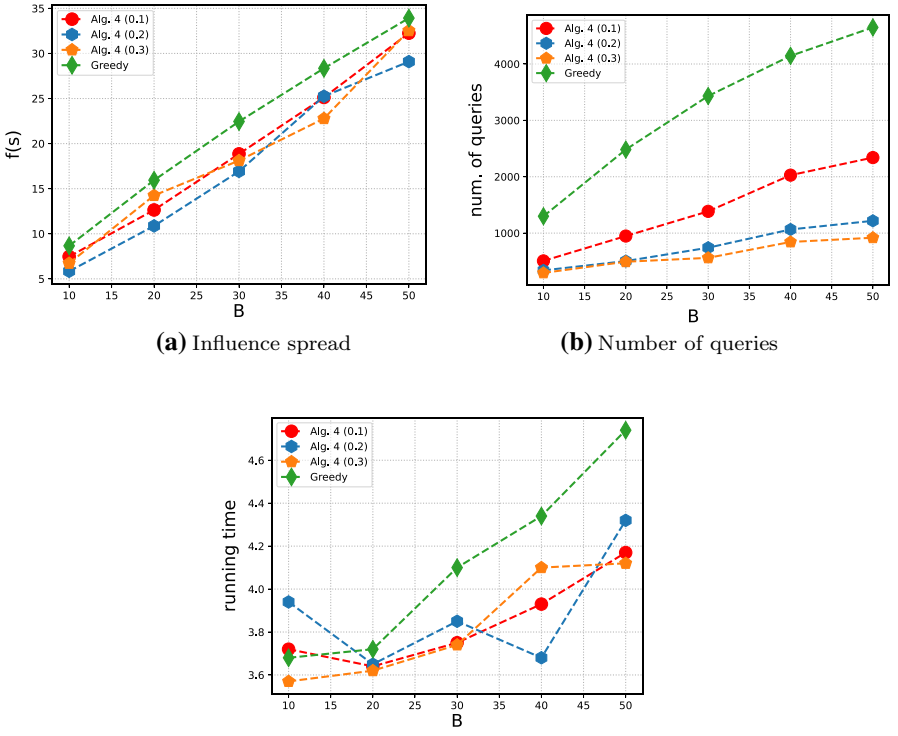
(b) Number of queries

(c) Running time

Fig. 3 Performance of algorithms for SPkB when $\beta = 1$: **a** Influence spread, **b** Number of queries, **c** Running time

lab between February 28th and April 5th, 2004. Temperature, humidity, and light values are extracted and discretized into several bins of 2 degrees Celsius each, 5 points each, and 100 luxes each, respectively. Finally, we set $k = 3$, ϵ, costs and the budget B as in the experiments of IMkB.

Experiment results We also conduct this experiment for two cases as previous experiment. Figure 3 shows the performance of algorithms for the case $\beta = 1$. Differing from the results of IMkB, Greedy gives the best solution quality but the gap with our algorithms is not significant. Once again, Algorithm 4 is usually better than Algorithm 2. However, it needs more number of queries and running time than Algorithm 2. The performance of random streaming algorithm and Greedy for the general case is shown in Fig. 4. Our algorithm is able to perform approximately to Greedy but it runs faster and takes 6 times fewer queries than Greedy.



(a) Influence spread

(b) Number of queries

(c) Running time

Fig. 4 Performance of algorithms for SPkB in general case: **a** Influence spread, **b** Number of queries, **c** Running time

6 Conclusions

This paper studies the BkSM problem, which generalizes the problem of maximizing a k-submodular function under size constraint by considering the cost of each element and the a limited budget. We propose two single pass streaming algorithms with provable guarantees. The core of our algorithms is to exploit the relation between candidate solutions and the optimal solution by analyzing intermediate quantities and using a new probability distribution then comparing the contribution value (marginal objective per cost) to a given appropriate threshold.

In order to investigate the performance of our algorithms in practice, we conduct some experiments on two applications of Influence maximization and Sensor placement. Experimental results have shown that our algorithms not only return good solutions in term of quality requirement but also take a sharply smaller number of queries than that of the state-of-the-art Greedy algorithm. In the future, we further investigate the k-submodular maximization under an individual budget constraint in which each subset S_i of the solution has a budget constraint.

Acknowledgements This work was supported by Vietnam National Foundation for Science and Technology Development (NAFOSTED) under Grant No. 102.01-2020.21. The work has been carried out partly at the Vietnam Institute for Advanced Study in Mathematics (VIASM). The first author (Canh V. Pham) would like to thank VIASM for the hospitality and financial support.

References

Badanidiyuru A, Vondrák J (2014) Fast algorithms for maximizing submodular functions. In: Chekuri C (ed) Proceedings of the twenty-fifth annual ACM-SIAM symposium on discrete algorithms, SODA 2014, Portland, Oregon, USA, January 5–7, 2014, SIAM, pp 1497–1514. https://doi.org/10.1137/1.9781611973402.110

Badanidiyuru A, Mirzasoleiman B, Karbasi A, Krause A (2014) Streaming submodular maximization: massive data summarization on the fly. In: Macskassy SA, Perlich C, Leskovec J, Wang W, Ghani R (eds) The 20th ACM SIGKDD international conference on knowledge discovery and data mining, KDD '14, New York, NY, USA—August 24–27, 2014, ACM, pp 671–680

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

Borgs C, Brautbar M, Chayes JT, Lucier B (2014) Maximizing social influence in nearly optimal time. In: Proceedings of the twenty-fifth annual ACM-SIAM symposium on discrete algorithms, SODA 2014, Portland, Oregon, USA, January 5–7, 2014, 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

Cualinescu G, Chekuri C, Pál M, Vondrák J (2011) Maximizing a monotone submodular function subject to a matroid constraint. SIAM J Comput 40(6):1740–1766

Chakrabarti A, Kale S (2015) Submodular maximization meets streaming: matchings, matroids, and more. Math Program 154(1–2):225–247

Chen W, Yuan Y, Zhang L (2010) Scalable influence maximization in social networks under the linear threshold model. In: ICDM 2010, The 10th IEEE international conference on data mining, Sydney, Australia, 14–17 December 2010, pp 88–97

Gomes R, Krause A (2010) Budgeted nonparametric learning from data streams. In: Fürnkranz J, Joachims T (eds) Proceedings of the 27th international conference on machine learning (ICML-10), June 21–24, 2010, Haifa, Israel, Omnipress, pp 391–398

Haba R, Kazemi E, Feldman M, Karbasi A (2020) Streaming submodular maximization under a k-set system constraint. In: Proceedings of the 37th international conference on machine learning, ICML 2020, 2020, virtual event, PMLR, proceedings of machine learning research, vol 119, pp 3939–3949

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

Iwata S, Tanigawa S, Yoshida Y (2016) Improved approximation algorithms for k-submodular function maximization. In: Krauthgamer R (ed) Proceedings of the twenty-seventh annual ACM-SIAM symposium on discrete algorithms, SODA 2016, Arlington, VA, USA, 2016, SIAM, pp 404–413

Kempe D, Kleinberg JM, 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, Washington, DC, USA, August 24–27, 2003, pp 137–146. https://doi.org/10.1145/956750.956769

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

Kumar R, Moseley B, Vassilvitskii S, Vattani A (2013) Fast greedy algorithms in mapreduce and streaming. In: Blelloch GE, Vöcking B (eds) 25th ACM symposium on parallelism in algorithms and architectures, SPAA '13, Montreal, QC, Canada—23–25, 2013, ACM, pp 1–10

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

Li X, Smith JD, Dinh TN, Thai MT (2019) Tiptop: (almost) exact solutions for influence maximization in billion-scale networks. IEEE/ACM Trans Netw 27(2):649–661

Mirzasoleiman B, Badanidiyuru A, Karbasi A, Vondrák J, Krause A (2015) Lazier than lazy greedy. In: Bonet B, Koenig S (eds) Proceedings of the twenty-ninth AAAI conference on artificial intelligence, 2015, Austin, Texas, USA, AAAI Press, pp 1812–1818

Mirzasoleiman B, Badanidiyuru A, Karbasi A (2016) Fast constrained submodular maximization: personalized data summarization. In: Balcan M, Weinberger KQ (eds) Proceedings of the 33nd international conference on machine learning, ICML 2016, New York City, NY, USA, June 19–24, 2016, JMLR.org, JMLR workshop and conference proceedings, vol 48, pp 1358–1367

Nemhauser GL, Wolsey LA, Fisher ML (1978) An analysis of approximations for maximizing submodular set functions - I. Math Program 14(1):265–294. https://doi.org/10.1007/BF01588971

Nguyen H, Zheng R (2013) On budgeted influence maximization in social networks. IEEE J Sel Areas Commun 31(6):1084–1094. https://doi.org/10.1109/JSAC.2013.130610

Nguyen L, Thai M (2020) Streaming k-submodular maximization under noise subject to size constraint. In: Daumé H, Singh A (eds) Proceedings of the international conference on machine learning, (ICML-2020), thirty-seventh international conference on machine learning

Ohsaka N, Yoshida Y (2015) Monotone k-submodular function maximization with size constraints. In: Cortes C, Lawrence ND, Lee DD, Sugiyama M, Garnett R (eds) Advances in neural information processing systems 28: annual conference on neural information processing systems 2015, Montreal, Quebec, Canada, pp 694–702

Oshima H (2017) Derandomization for k-submodular maximization. In: Brankovic L, Ryan J, Smyth WF (eds) Combinatorial algorithms—28th international workshop, IWOCA 2017, Lecture notes in computer science, vol 10765, pp 88–99

Qian C, Shi J, Tang K, Zhou Z (2018) 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 (2020) Fast and private submodular and k-submodular functions maximization with matroid constraints. In: Proceedings of the 37th international conference on machine learning, ICML 2020, 13–18, 2020, virtual event, proceedings of machine learning research, vol 119, pp 7887–7897

Rafiey A, Yoshida Y (2020) Fast and private submodular and k-submodular functions maximization with matroid constraints. In: Proceedings of the 37th international conference on machine learning, ICML 2020, 13-18 July 2020, virtual event, PMLR, proceedings of machine learning research, vol 119, pp 7887–7897

Sakaue S (2017) On maximizing a monotone k-submodular function subject to a matroid constraint. Discret Optim 23:105–113

Schrijver A (2003) Combinatorial optimization: polyhedra and efficiency. Springer, Algorithms and Combinatorics

Singh AP, Guillory A, Bilmes JA (2012) On bisubmodular maximization. In: Lawrence ND, Girolami MA (eds) Proceedings of the fifteenth international conference on artificial intelligence and statistics, AISTATS 2012, La Palma, Canary Islands, Spain, 21–23, 2012, JMLR.org, JMLR proceedings, vol 22, pp 1055–1063

Soma T (2019) No-regret algorithms for online k-submodular maximization. In: Chaudhuri K, Sugiyama M (eds) The 22nd international conference on artificial intelligence and statistics, AISTATS 2019, 16–18 April 2019, Naha, Okinawa, Japan, proceedings of machine learning research, vol 89, 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

Thapper J, Zivný S (2012) The power of linear programming for valued csps. In: 53rd annual IEEE symposium on foundations of computer science, FOCS 2012, New Brunswick, NJ, USA, 2012, IEEE Computer Society, pp 669–678

Wang B, Zhou H (2021) Multilinear extension of k-submodular functions. CoRR abs/2107.07103, https://arxiv.org/abs/2107.07103, eprint2107.07103

Ward J, Zivný S (2014) Maximizing bisubmodular and k-submodular functions. In: Chekuri C (ed) Proceedings of the twenty-fifth annual ACM-SIAM symposium on discrete algorithms, SODA 2014, Portland, Oregon, USA, 2014, SIAM, pp 1468–1481

Wolsey LA (1982) Maximising real-valued submodular functions: primal and dual heuristics for location problems. Math Oper Res 7(3):410–425

Yang R, Xu D, Cheng Y, Gao C, Du D (2019) Streaming submodular maximization under noises. In: 39th IEEE international conference on distributed computing systems, ICDCS 2019, Dallas, TX, USA, July 7–10, 2019, IEEE, pp 348–357

Yu Q, Xu EL, Cui S (2016) Submodular maximization with multi-knapsack constraints and its applications in scientific literature recommendations. In: 2016 IEEE global conference on signal and information processing, GlobalSIP 2016, Washington, DC, USA, 2016, IEEE, pp 1295–1299

Zhang Y, Li M, Yang D, Xue G (2019) A budget feasible mechanism for k-topic influence maximization in social networks. In: 2019 IEEE global communications conference, GLOBECOM 2019, Waikoloa, HI, USA, December 9–13, 2019, IEEE, pp 1–6

Zheng L, Chan H, Loukides G, Li M (2021) Maximizing approximately k-submodular functions. In: Demeniconi C, Davidson I (eds) Proceedings of the 2021 SIAM international conference on data mining, SDM 2021, Virtual Event, 2021, SIAM, pp 414–422