Streaming Algorithms for Budgeted *k*-Submodular Maximization Problem

Canh V. Pham[1(✉)], Quang C. Vu[2], Dung K. T. Ha[2,3], and Tai T. Nguyen[3]

[1] ORlab, Faculty of Computer Science, Phenikaa University, Hanoi 12116, Vietnam
canh.phamvan@phenikaa-uni.edu.vn

[2] Faculty of Information Security, People's Security Academy, Hanoi, Vietnam
quangvc.hvan@gmail.com

[3] Faculty of Information Technology, University of Engineering and Technology, Vietnam National University, Hanoi, Vietnam
{20028008,20025032}@vnu.edu.vn

Abstract. Stimulated by practical applications arising from viral marketing. This paper investigates a novel 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 $\mathbf{s} = (S_1, S_2, \ldots, S_k)$, each element $e \in V$ has a cost $c_i(e)$ to be put into i-th set S_i, with the total cost of s does not exceed B so that $f(\mathbf{s})$ is maximized. To address this problem, we propose two streaming algorithms that provide approximation guarantees for the problem. In particular, in the case of each element e has the same cost for all i-th sets, we propose a deterministic streaming algorithm which provides an approximation ratio of $\frac{1}{4} - \epsilon$ when f is monotone and $\frac{1}{5} - \epsilon$ when f is non-monotone. For the general case, we propose a random streaming algorithm that provides 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 ϵ, α are fixed inputs.

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

1 Introduction

Maximizing k-submodular functions has attracted a lot of attention because of its potential in solving various combinatorial optimization problems such as influence maximization [8,9,11,12], sensor placement [9,11,12], feature selection [14] and information coverage maximization [11]. Given a finite set V and an integer k, we define $[k] = \{1, 2, \ldots, k\}$ and $(k+1)^V = \{(X_1, X_2, \ldots, X_k) | X_i \subseteq V, \forall i \in [k], X_i \cap X_j = \emptyset, \forall i \neq j\}$ be a family of k disjoint sets. 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:

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

D. Mohaisen and R. Jin (Eds.): CSoNet 2021, LNCS 13116, pp. 27–38, 2021.
https://doi.org/10.1007/978-3-030-91434-9_3

where

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

and

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

Although there exists a polynomial time to maximize a k-submodular function [16], maximizing a k-submodular function is still NP-hard. Studying on maximizing a k-submodular function was initiated by Singh *et al.* [14] with $k = 2$. Ward *et al.* [17] first studied to maximize an unconstrained k-submodular function for general k and devised a deterministic greedy algorithm which provided an approximation ratio of 1/3. Later on, [6] introduced a random greedy approach which improved the approximation ratio to $\frac{k}{2k-1}$ by applying a probability distribution to select any larger marginal element that has a higher probability. The authors in [10] eliminated the random told above but the number of queries increased to $O(n^2k^2)$. The unconstrained maximizing k-submodular function was further studied in [15] in online settings.

Under the size constraint, Oshaka et al. [9] first proposed 1/2-approximation algorithm by using a greedy approach for maximizing monotone k-submodular maximization functions. [13] showed a greedy selection that could give an approximation ratio of 1/2 under the matroid constraint. The authors in [11] then further proposed multi-objective evolutionary algorithms that provided 1/2-approximation ratio under the size constraint but took $O(kn \log^2 B)$ queries in expectation. Recently, Nguyen [12] *et al.* considered the k-submodular maximization problem subjected to the total size constraint under noises and devised 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.

Although there have been many attempts to solve the problem of maximizing a k-submodular function under several kinds of constraints, they did not cover several cases that could happens frequently in reality in which each element could be customized in terms of its private cost or a problem was provided with just limited budgets. Let's consider the following application:

Influence Maximization with k Topics. 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 to a respective person. Given a budget B, we consider the problem of finding a set of users (seed set), each initially adopts a topic, with the total cost is at most B to maximize the expected number of users who are eventually activated by at least one topic. In this application, the expected number of influenced users (objective) function is k-submodular where each user corresponds to each element in the set V [8,9,12].

Motivated by that observation, in this work, we study a novel problem named **Budgeted k-submodular maximization** (BkSM), defined as follows:

Definition 1. *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 $\mathbf{s} = (S_1, S_2, \dots, S_k)$,* *each element* $e \in V$ *has a cost* $c_i(e) > 0$ *to be put in to* S_i*, with total cost* $c(\boldsymbol{s}) = \sum_{i\in[k]} \sum_{e\in S_i} c_i(e) \le B$ *so that* $f(\boldsymbol{s})$ *is maximized.*

In addition, input data increasing constantly makes it impossible to be stored in computer memory. Therefore it is critical to devise streaming algorithms which not only reduce the requirement of stored memory but also be able to produce guaranteed solutions in a single pass or some passes. Although streaming algorithm is one of efficient methods for solving submodular maximization problems under various kinds of constraints such as cardinality constraint [1,3,7,18], knapsack constraint [5], k-set constraint [4] and matroid constraint [2], it is not potential to directly be applied to our BkSM problem due to intrinsic differences between submodularity and k-submodularity.

Our Contributions. In this paper we propose several algorithms which provide theoretical bounds of BkSM. Overall, our contributions are as follows:

- For a special case when every element has the same cost to be added to any i-th set, we first propose a deterministic streaming algorithm (Algorithm 2) which runs in a single pass, has $O(\frac{kn}{\epsilon}\log B)$ query complexity, $O(\frac{B}{\epsilon}\log B)$ 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 value of $\epsilon > 0$.
- For the general case, we propose a random streaming algorithm (Algorithm 4) which runs in a single pass, has $O(\frac{kn}{\epsilon}\log B)$ query complexity, $O(\frac{B}{\epsilon}\log B)$ 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 $\beta = \max_{e\in V, i,j\in[k], i\ne j} \frac{c_i(e)}{c_j(e)}$ and $\alpha \in (0,1], \epsilon \in (0,1)$ are inputs.

Our algorithms is an inspired suggestion from [1,5] 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. In addition, we introduce a new probability distribution to subsequently select a new element to candidate solutions.

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

2 Preliminaries

Given a finite set V and an integer k, denote $[k] = \{1, 2, \ldots, k\}$, let $(k+1)^V = \{(X_1, X_2, \ldots, X_k) | X_i \subseteq V, \forall i \in [k], X_i \cap X_j = \emptyset, \forall i \ne j\}$ be a family of k disjoint sets, called a k**-set**. We define $supp_i(\mathbf{x}) = X_i$, $supp(\mathbf{x}) = \cup_{i\in[k]} X_i$, X_i is called i**-th set of x** and an empty k-set is defined as $\mathbf{0} = (\emptyset, \ldots, \emptyset)$.

For $\mathbf{x} = (X_1, X_2, \ldots X_k)$ and $\mathbf{y} = (Y_1, Y_2, \ldots, Y_k) \in (k+1)^V$, if $e \in X_i$, we write $\mathbf{x}(e) = i$ else 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 $\mathbf{x}$ as (e, i). We denote $\mathbf{x} \sqsubseteq \mathbf{y}$ 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 $\mathbf{x} = (X_1, X_2, \dots, X_k)$ and $\mathbf{y} = (Y_1, Y_2, \dots, Y_k) \in (k+1)^V$, we have:

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

where

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

and

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

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

$$\Delta_{e,i} f(\mathbf{x}) = f(X_1, \dots, X_{i-1}, X_i \cup \{e\}, X_{i+1}, \dots, X_k) - f(X_1, \dots, X_k) \geq 0 \tag{3}$$

From [17], the k-submodularity of f implies the *orthant submodularity*, i.e.,

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

and the *pairwise monotonicity*, i.e.,

$$\Delta_{e,i} f(\mathbf{x}) + \Delta_{e,j} f(\mathbf{x}) \geq 0 \tag{5}$$

for any $\mathbf{x}, \mathbf{y} \in (k+1)^V$ with $\mathbf{x} \sqsubseteq \mathbf{y}$, $e \notin supp(\mathbf{y})$ 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 cost $c_i(e)$ to be added into 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 $c_i(e) \leq B$, otherwise we can simply remove it. We only consider $k \geq 2$ because if $k = 1$, the k-submodular becomes to submodular function.

3 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 i-th sets $c_i(e) = c_j(e), \forall e \in V, i \neq j$. 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 with a threshold which is set in advance, and (2) we use the maximum singleton value (e_{max}, i_{max}) defined as

$$(e_{max}, i_{max}) = \arg \max_{e \in V, i \in [k]} f((e, i)) \tag{6}$$

to obtain the final solution. We first assume that the optimal solution is known and then remove this assumption by using the method in [1] to approximate the optimal solution.

3.1 Deterministic Streaming Algorithm with Known Optimal Solution

We first present a simplified version of our deterministic streaming algorithm when the optimal solution is known. Denote $\mathbf{o}$ as 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 the main version. The details of the algorithm are fully presented in Algorithm 1. We define the notations as follows:

- (e^j, i^j) as the j-th element and its position added in the main loop of the algorithm;
- $\mathbf{s}^j$ - 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 $\mathbf{s}^t$, $r = |supp(\mathbf{u}^t)|$
- $\mathbf{u}_i^t = \mathbf{s}^t \sqcup \{(u_1, j_1), (u_2, j_2), \ldots, (u_i, j_i)\}$

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 $c(\mathbf{s}^t) + c(e)$ exceed 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 which has 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 in $\{\mathbf{s}^t\} \cup \{(e_{max}, i_{max})\}$ when f is monotone or returns the best solution in $\{\mathbf{s}^j : j \leq t\}$

Algorithm 1. Deterministic streaming algorithm with known opt

Input: a function $f : (k+1)^V \mapsto \mathbb{R}_+$, $B > 0$, $\alpha \in (0, 1]$, v that $v \leq \mathsf{opt}$
Output: a solution $\mathbf{s}$

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

$\cup \{(e_{max}, i_{max})\}$ when f is non-monotone. We now analysis the approximation guarantee of Algorithm 1. Denote e^t is the last addition of the main loop of the Algorithm 1. By exploiting the relation among $\mathbf{o}$, $\mathbf{o}^j$ and $\mathbf{s}^j, j \leq t$, we obtain the following Lemma.

Lemma 1. *If f is monotone then $v - f(\boldsymbol{o}^t) \leq f(\boldsymbol{s}^t)$ and if f is non-monotone then $v - f(\boldsymbol{o}^t) \leq 2f(\boldsymbol{s}^t)$.*

Due to the space constraint, we omit some proofs and presented them in a full version of this paper. 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 $\boldsymbol{s}$ satisfying:*

- *If f is monotone, $f(\boldsymbol{s}) \geq \min\{\frac{\alpha}{2}, \frac{1-\alpha}{2}\}v$, $f(\boldsymbol{s})$ is maximized to $\frac{v}{4}$ when $\alpha = \frac{1}{2}$.*
- *If f is non-monotone, $f(\boldsymbol{s}) \geq \min\{\frac{\alpha}{2}, \frac{1-\alpha}{3}\}v$, $f(\boldsymbol{s})$ is maximized to $\frac{v}{5}$ when $\alpha = \frac{2}{5}$.*

3.2 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 $\mathbf{o}$ is known. We use the dynamic update method in [1] to obtain a good approximation of opt.

To specific, denote $m = \max_{e \in V, i \in [k]} f((e, i))$, we have $m \leq \mathsf{opt} \leq Bm$. Therefore we use the value $v = (1 + \epsilon')^j$ for $\{j | m \leq (1 + \epsilon')^j \leq Bm, j \in \mathbb{Z}_+\}$ 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, we have to require at least one pass over V. Therefore, we adapt the dynamic update method in [1] which updates $m = \max\{m, \max_{i \in [k]} f((e, i))\}$ with an already observed element e to determine the range of guessed optimal values. This method can help algorithm maintain a good estimation of the optimal solution if that range shifts forward when next elements are observed. We implement this method by using variables $\mathbf{s}^{t_j}$ and t_j to store a candidate solution and the number of its elements in which $v = (1+\epsilon')^j$ is an guessed value of opt.

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 higher than ϵ to reduce the complexity but still ensure approximation ratios. The detail of our algorithm is presented in Algorithm 2.

Algorithm 2. Deterministic streaming algorithm

Input: a function $f : (k+1)^V \mapsto \mathbb{R}_+$, $B > 0$, $\epsilon > 0$.
Output: a solution $\mathbf{s}$

If f is monotone $\alpha \leftarrow \frac{1}{2}$, $\epsilon' \leftarrow 4\epsilon$ **else** $\alpha \leftarrow \frac{2}{5}$, $\epsilon' \leftarrow 5\epsilon$
foreach $e \in V$ **do**
    $i_e \leftarrow \arg\max_{i \in [k]} f((e,i))$
    $(e_{max}, i_{max}) \leftarrow \arg\max_{(e_1,i_1) \in \{(e_{max},i_{max}),(e,i_e)\}} f((e_1,i_1))$
    $O \leftarrow \{j | f((e_{max},i_{max})) \le (1+\epsilon')^j \le Bf((e_{max},i_{max})), j \in \mathbb{Z}_+\}$
    for $j \in O$ **do**
        if $c(\mathbf{s}^{t_j}) + c(e) \le B$ **then**
            $i' \leftarrow \arg\max_{i \in [k]} f(\mathbf{s}^{t_j} \sqcup (e,i))$ **if** $\frac{f(\mathbf{s}^{t_j} \sqcup (e,i'))}{c(\mathbf{s}^{t_j}) + c(e)} \ge \frac{\alpha(1+\epsilon')^j}{B}$ **then**
                $\mathbf{s}^{t_j+1} \leftarrow \mathbf{s}^{t_j} \sqcup (e,i')$, $t_j \leftarrow t_j + 1$

Lemma 2. *In Algorithm 2, there exists a number $j \in \mathbb{Z}_+$ so that $v = (1+\epsilon')^j \in O$ satisfies* $(1-\epsilon')\mathsf{opt} \le v \le \mathsf{opt}$

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

$$m \le \mathsf{opt} = f(\mathbf{o}) \le \sum_{e \in supp(\mathbf{o})} f(e, \mathbf{o}(e)) \le Bm$$

Let $j = \lfloor \log_{1+\epsilon'} \mathsf{opt} \rfloor$, we have $v = (1+\epsilon')^j \le \mathsf{opt} \le Bm$ and $v \ge (1+\epsilon')^{\log_{1+\epsilon'}(\mathsf{opt})-1} = \frac{\mathsf{opt}}{1+\epsilon'} \ge \mathsf{opt}(1-\epsilon')$.

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

Theorem 2. *Algorithm 2 is a single pass streaming algorithm that has $O(\frac{kn}{\epsilon}\log B)$ query complexity, $O(\frac{B}{\epsilon}\log B)$ 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 $\frac{1}{\epsilon'}\log B$, finding each $\mathbf{s}^{t_j}$ takes at most $O(kn)$ queries and $\mathbf{s}^{t_j}$ includes at most B elements. Therefore, the query complexity is $O(\frac{kn}{\epsilon}\log B)$ and total space complexity is $O(\frac{B}{\epsilon}\log B)$.

By Lemma 2, there exists an integer number $j \in \mathbb{Z}_+$ so that $v = (1+\epsilon')^j \in O$ satisfies $(1-\epsilon')\mathsf{opt} \le v \le \mathsf{opt}$. Apply Theorem 1, for the monotone case we have: $f(\mathbf{s}) \ge \frac{1}{4}v \ge \frac{1}{4}(1-\epsilon')\mathsf{opt} = (\frac{1}{4} - \epsilon)\mathsf{opt}$ and for the non-monotone case: $f(\mathbf{s}) \ge \frac{1}{5}v \ge \frac{1}{5}(1-\epsilon')\mathsf{opt} = (\frac{1}{5} - \epsilon)\mathsf{opt}$. Hence, the theorem is proved. □

4 Random Streaming Algorithm for General Case

In the case each element e has multiple different cost $c_i(e)$ for each i-th set, we can not apply previous algorithms. Therefore, in this section we introduce one pass streaming which provides approximation ratios in expectation for BkSM problem.

At the core of our algorithm, we introduce a new probability distribution to choose 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 a predefined threshold to filter high-value elements into candidate solutions and the maximum singleton value to give the final solution. Similar to the previous section, we first introduce a simplified version of the streaming algorithm when the optimal solution is known in advance.

4.1 Random Algorithm with Known Optimal Solution

This algorithm also receives the 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 imitates an empty k-set $\mathbf{s}^0$ and subsequently updates the solution after once passing over V. Be different from Algorithm 1, for each $e \in V$ being observed, the algorithm finds a set collection J that contains positions satisfying the total cost is at most B and the ratio of the increment of the objective function per cost is at least a given threshold, i.e.,

$$J = \left\{ i \in [k] : c(\mathbf{s}^t) + c_i(e) \leq B \text{ and } \frac{f(\mathbf{s}^t \sqcup (e,i)) - f(\mathbf{s}^t)}{c_i(e)} \geq \frac{\alpha v}{B} \right\} \tag{7}$$

These constraints help the algorithm eliminate which position having low increment of the objective function over its cost. If $J \neq \emptyset$, the algorithm puts e into set i of $\mathbf{s}^t$ with a probability:

$$\frac{p_i^{|J|-1}}{T} = \frac{\left(\frac{f(\mathbf{s}^t \sqcup (e,i)) - f(\mathbf{s}^t)}{c_i(e)}\right)^{|J|-1}}{\sum_{i \in J} \left(\frac{f(\mathbf{s}^t \sqcup (e,i)) - f(\mathbf{s}^t)}{c_i(e)}\right)^{|J|-1}} \tag{8}$$

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.

Lemma 3 provides the relationship among $\mathbf{o}, \mathbf{o}^j$ and $\mathbf{s}^j, j \leq t$ that play an importance role in analyzing algorithm's performance.

Lemma 3. *In Algorithm 3, if there is no pair* $(e,i) \in \boldsymbol{o}$ *satisfying* $\exists j \in [t] : e \notin$ *supp*$(\boldsymbol{s}^j)$ *so that* $\frac{f(\boldsymbol{s}^j \sqcup (e,i))}{c(\boldsymbol{s}^j) + c_i(e)} \geq \frac{\alpha v}{B}$ *and* $c(\boldsymbol{s}^j) + c_i(e) > B$, *we have:*

- *If f is monotone, then*

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

Algorithm 3. Random streaming algorithm with known opt-RanStreamWithOpt(f, opt, α)

Input: a function $f : (k+1)^V \mapsto \mathbb{R}_+$, $B > 0$, $\alpha \in (0,1]$, $v, v \leq \mathsf{opt}$
Output: a solution $\mathbf{s}$

$\mathbf{s}_0 \leftarrow \mathbf{0}$, $t \leftarrow 0$
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**
            $p_i \leftarrow \frac{f(\mathbf{s}^t \sqcup (e,i)) - f(\mathbf{s}^t)}{c_i(e)}$; $J \leftarrow J \cup \{i\}$
    if $J \neq \emptyset$ **then**
        $T \leftarrow \sum_{i \in J} p_i^{|J|-1}$
        Select a position $i \in J$ with probability $\frac{p_i^{|J|-1}}{T}$
        $\mathbf{s}^{t+1} \leftarrow \mathbf{s}^t \sqcup (e,i)$; $t \leftarrow t+1$
return $\arg\max_{\mathbf{s} \in \{\mathbf{s}^t\} \cup \{(e_{max},i_{max})\}} f(\mathbf{s})$ if f is monotone, $\arg\max_{\mathbf{s} \in \{\mathbf{s}^j : j \leq t\} \cup \{(e_{max},i_{max})\}} f(\mathbf{s})$ if f is non-monotone

– *If f is non-monotone, then*

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

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

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

4.2 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 [1] to estimate opt. We assume that we know β 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 $\mathbf{s}^{t_j}$ and t_j to save candidate solutions, which is updated by using the probability distribution as in Algorithm 3 with $(1+\epsilon)^j$ is an estimation of optimal solution. The algorithm finally compares all candidate solutions to select the best one. The details of algorithm is presented in Algorithm 4.

Algorithm 4. Random streaming algorithm

Input: a k-submodular function $f : (k+1)^V \mapsto \mathbb{R}_+$, $B > 0$, $\epsilon > 0$, $\alpha \in (0, 1]$
Output: a solution $\mathbf{s}$

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

return $\arg\max_{\mathbf{s} \in \{\mathbf{s}^{t_j} : j \in O\} \cup \{(e_{max}, i_{max})\}} f(\mathbf{s})$ if f is monotone,
$\arg\max_{\mathbf{s}\{\mathbf{s}_i^{t_j} : j \in O, i \leq j\} \cup \{(e_{max}, i_{max})\}} f(\mathbf{s})$ if f is non-monotone

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

Proof. By Lemma 2, there exists $j \in \mathbb{Z}_+$ that $v = (1+\epsilon)^j \in O$ satisfies $(1-\epsilon)\mathsf{opt} \leq v \leq \mathsf{opt}$. Using similar arguments of the proof of Theorem 3, for the monotone case

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

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

5 Conclusions

This paper studies the BkSM, a generalized version of maximizing k-submodular functions problem. In order to find the solution, we propose several 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 applying a new probability distribution to select elements with high contributions to a current solution. In the future we are going to conduct experiments on so some instance of BkSM to show the performance of our algorithms in practice.

Acknowledgements. This work is supported by Vietnam National Foundation for Science and Technology Development (NAFOSTED) under Grant No. 102.01-2020.21.

References

1. Badanidiyuru, A., Mirzasoleiman, B., Karbasi, A., Krause, A.: Streaming submodular maximization: massive data summarization on the fly. In: Macskassy, S.A., Perlich, C., Leskovec, J., Wang, W., Ghani, R. (eds.) The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2014, pp. 671–680. ACM (2014)
2. Chakrabarti, A., Kale, S.: Submodular maximization meets streaming: matchings, matroids, and more. Math. Program. 225–247 (2015). https://doi.org/10.1007/s10107-015-0900-7
3. Gomes, R., Krause, A.: Budgeted nonparametric learning from data streams. In: Fürnkranz, J., Joachims, T. (eds.) Proceedings of the 27th International Conference on Machine Learning (ICML 2010), Haifa, Israel, 21–24 June 2010, pp. 391–398. Omnipress (2010)
4. Haba, R., Kazemi, E., Feldman, M., Karbasi, A.: Streaming submodular maximization under a k-set system constraint. In: Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13–18 July 2020, Virtual Event. Proceedings of Machine Learning Research, vol. 119, pp. 3939–3949. PMLR (2020)
5. Huang, C., Kakimura, N., Yoshida, Y.: Streaming algorithms for maximizing monotone submodular functions under a Knapsack constraint. Algorithmica **82**(4), 1006–1032 (2020)
6. Iwata, S., Tanigawa, S., Yoshida, Y.: 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, 10–12 January 2016, pp. 404–413. SIAM (2016)
7. Kumar, R., Moseley, B., Vassilvitskii, S., Vattani, A.: Fast greedy algorithms in mapreduce and streaming. In: Blelloch, G.E., Vöcking, B. (eds.) 25th ACM Symposium on Parallelism in Algorithms and Architectures, SPAA 2013, pp. 1–10. ACM (2013)
8. Nguyen, L., Thai, M.: 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 (2020)

9. Ohsaka, N., Yoshida, Y.: Monotone k-submodular function maximization with size constraints. In: Cortes, C., Lawrence, N.D., Lee, D.D., Sugiyama, M., Garnett, R. (eds.) Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, Montreal, Quebec, Canada, 7–12 December 2015, pp. 694–702 (2015)
10. Oshima, H.: Derandomization for k-submodular maximization. In: Brankovic, L., Ryan, J., Smyth, W.F. (eds.) IWOCA 2017. LNCS, vol. 10765, pp. 88–99. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-78825-8_8
11. Qian, C., Shi, J., Tang, K., Zhou, Z.: Constrained monotone k-submodular function maximization using multiobjective evolutionary algorithms with theoretical guarantee. IEEE Trans. Evol. Comput. **22**(4), 595–608 (2018)
12. Rafiey, A., Yoshida, Y.: Fast and private submodular and k-submodular functions maximization with matroid constraints. In: Proceedings of the 37th International Conference on Machine Learning, ICML 2020. Proceedings of Machine Learning Research, vol. 119, pp. 7887–7897. PMLR (2020)
13. Sakaue, S.: On maximizing a monotone k-submodular function subject to a matroid constraint. Discret. Optim. **23**, 105–113 (2017)
14. Singh, A.P., Guillory, A., Bilmes, J.A.: On bisubmodular maximization. In: Lawrence, N.D., Girolami, M.A. (eds.) Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2012. JMLR Proceedings, vol. 22, pp. 1055–1063. JMLR.org (2012)
15. Soma, T.: 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, Naha, Okinawa, Japan, 16–18 April 2019. Proceedings of Machine Learning Research, vol. 89, pp. 1205–1214. PMLR (2019)
16. Thapper, J., Zivný, S.: The power of linear programming for valued CSPs. In: 53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012, New Brunswick, NJ, USA, 20–23 October 2012, pp. 669–678. IEEE Computer Society (2012)
17. Ward, J., Zivný, S.: 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, 5–7 January 2014, pp. 1468–1481. SIAM (2014)
18. Yang, R., Xu, D., Cheng, Y., Gao, C., Du, D.: Streaming submodular maximization under noises. In: 39th IEEE International Conference on Distributed Computing Systems, ICDCS 2019, Dallas, TX, USA, 7–10 July 2019, pp. 348–357. IEEE (2019)