





Article

Efficient Streaming Algorithms for Maximizing Monotone DR-Submodular Function on the Integer Lattice

Bich-Ngan T. Nguyen ^{1,2} , Phuong N. H. Pham ^{1,2,*} , Van-Vang Le ³  and Václav Snášel ² ¹ Faculty of Information Technology, Ho Chi Minh City University of Food Industry, 140 Le Trong Tan Street, Ho Chi Minh City 700000, Vietnam² Department of Computer Science, Faculty of Electrical Engineering and Computer Science, VŠB-Technical University of Ostrava, 17.listopadu 15/2172, 708 33 Ostrava, Czech Republic³ Faculty of Information Technology, Ton Duc Thang University, Ho Chi Minh City 700000, Vietnam

* Correspondence: phuongpnh@hufi.edu.vn

Abstract: In recent years, the issue of maximizing submodular functions has attracted much interest from research communities. However, most submodular functions are specified in a set function. Meanwhile, recent advancements have been studied for maximizing a diminishing return submodular (DR-submodular) function on the integer lattice. Because plenty of publications show that the DR-submodular function has wide applications in optimization problems such as sensor placement, optimal budget allocation, social network, and especially machine learning. In this research, we propose two main streaming algorithms for the problem of maximizing a monotone DR-submodular function under cardinality constraints. Our two algorithms, which are called StrDRS1 and StrDRS2, have $(1/2 - \epsilon)$, $(1 - 1/e - \epsilon)$ of approximation ratios and $O(\frac{n}{\epsilon} \log(\frac{\log B}{\epsilon}) \log k)$, $O(\frac{n}{\epsilon} \log B)$, respectively. We conducted several experiments to investigate the performance of our algorithms based on the budget allocation problem over the bipartite influence model, an instance of the monotone submodular function maximization problem over the integer lattice. The experimental results indicate that our proposed algorithms not only provide solutions with a high value of the objective function, but also outperform the state-of-the-art algorithms in terms of both the number of queries and the running time.

Keywords: DR-submodular function; integer lattice; adaptive complexity; approximation algorithm**MSC:** 03G10; 06C05; 06D99; 30E10; 65K10

Citation: Nguyen, B.-N.T.; Pham, P.N.H.; Le, V.-V.; Snášel, V. Efficient Streaming Algorithms for Maximizing Monotone DR-Submodular Function on the Integer Lattice. *Mathematics* **2022**, *10*, 3772. <https://doi.org/10.3390/math10203772>

Academic Editors: Gaogao Dong and Jianguo Liu

Received: 1 September 2022

Accepted: 9 October 2022

Published: 13 October 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Submodular function maximization problems have recently received great interest in the research community. A satisfactory explanation for this attraction is the prevalence of optimization problems related to submodular functions in many real-world applications [1]. Prominent examples include sensor placement problems [2,3] and facility location [4] in operational improvement, the influence maximization problem in viral marketing [5,6], document summarization [7], experiment design [8], dictionary learning [9] in machine learning, etc. These problems can be presented with the concept of submodularity, and effective algorithms have been developed taking advantage of the submodular function [10]. Given a ground set E , a function $f: 2^E \rightarrow \mathbb{R}_{\geq 0}$ is called *submodular* if for all $A, B \subseteq E$,

$$f(A) + f(B) \geq f(A \cup B) + f(A \cap B) \quad (1)$$

The *submodularity* of a submodular function f is equivalent to the property *diminishing return*, i.e., for any $A \subseteq B \subseteq E$ and $\forall e \in E \setminus B$, it holds.

$$f(A \cup e) - f(A) \geq f(B \cup e) - f(B) \quad (2)$$

and the set function f is called *monotone* if

$$f(A) \leq f(B) \text{ for any } A \subseteq B \subseteq E \quad (3)$$

The submodular function maximization problem aims to select a subset A of the ground set E to maximize $f(A)$.

Most existing studies of the submodular function maximization problem consider submodular functions identified over a set function. It means that the problem has the input as a subset of the ground set and returns an actual value. However, there are many real-world situations in which it is crucial to know whether an element $e \in E$ is selected and how many copies of that element should be chosen. In other words, the problem considers submodular functions over a multiset, under the name *submodular function on the integer lattice* [11]. The submodularity defined on the integer lattice differs from set functions because it does not equate to the diminishing return property. Some notable examples include the optimal budget allocation problem [12], document summarization and sensor placement [13], the submodular welfare problem [14], and maximization of influence spread with partial incentives [15]. The definitions of a submodular function and diminishing return submodular function on the integer lattice are as follows:

A function $f : \mathbb{Z}_+^E \rightarrow \mathbb{R}$ is a *submodular function on the integer lattice* if for all $\mathbf{x}, \mathbf{y} \in \mathbb{Z}_+^E$

$$f(\mathbf{x}) + f(\mathbf{y}) \geq f(\mathbf{x} \vee \mathbf{y}) + f(\mathbf{x} \wedge \mathbf{y}) \quad (4)$$

where $\mathbf{x} \wedge \mathbf{y}$ and $\mathbf{x} \vee \mathbf{y}$ denote the coordinate-wise min and max operations, respectively.

A function $f : \mathbb{Z}_+^E \rightarrow \mathbb{R}$ is called *diminishing return submodular (DR-submodular)*, if for all $\mathbf{x}, \mathbf{y} \in \mathbb{Z}_+^E$ with $\mathbf{x} \leq \mathbf{y}$

$$f(\mathbf{x} + \chi_e) - f(\mathbf{x}) \geq f(\mathbf{y} + \chi_e) - f(\mathbf{y}) \quad (5)$$

where $e \in E$ and χ_e denote the unit vector with coordinate e being 1 and the other elements are 0.

The submodularity defined on the integer lattice differs from the set functions because it does not equate to the diminishing return property. In other words, lattice submodularity is weaker than DR-submodularity, i.e., a lattice submodular function may not be a DR-submodular function, but any DR-submodular function is a lattice submodular one [11]. Due to this cause, developing approximation algorithms is challenging; even for a single cardinality constraint, we need a more complicated method, such as partial enumeration [12,13]. Nevertheless, the diminishing return property of the DR-submodular function maximization problem often plays a fundamental role in some practical problems, such as optimizing budget allocation among channels and influencers [12], optimal budget allocation [13], and online submodular welfare maximization [14].

There have been many approaches to solving the problem of maximizing the submodular function under different constraints and contexts in the last decade. Two notable approaches to this problem are greedy algorithms [16–19] and streaming algorithms [20–22]. Plenty of studies show that the greedy method is often used for this optimization problem because it outputs a better result than other methods due to its “greedy” operation [16,23,24]. Understandably, the greedy method always scans data many times to find the best. However, this causes its algorithms to have a long runtime; it cannot even be applied to big data. Contrary to the greedy method, the streaming method scans the data once. As each element in the dataset arrives in order, the streaming algorithm must decide whether that element is selected before the next element arrives. Thus, the result of this method may not be as good as the result of greedy, because the elements it selects are not the best, but meet the selection condition. However, the outstanding advantage of the streaming method is that it runs much faster than the greedy method [25]. There are many studies that have used the streaming method to resolve the issue of submodular function maximization. Those studies have shown the advantages of the streaming method compared to the greedy

method. Some prominent studies include using the streaming algorithm for maximizing k -submodular functions under budget constraints [26], optimizing a submodular function under noise by streaming algorithms [27], maximizing a monotone submodular function by multi-pass streaming algorithms [20], and using fast streaming for the problem of submodular maximization [22].

Attracted by the usefulness of the maximizing DR-submodular function on the integer lattice issue in many practical problems, numerous studies on this problem have recently been published. These publications consider the problem under many different constraints and use greedy or streaming methods as the standard approach. Some prominent examples include the use of a fast double greedy algorithm to maximize the non-monotone DR submodular function [24], using a threshold greedy algorithm to maximize the monotone DR submodular constraint knapsack over an integer lattice [28], combining the threshold greedy algorithm with a partial element enumeration technique to maximize the monotone DR submodular knapsack constraint over an integer lattice [11], using a streaming method to maximize DR-submodular functions with d -knapsack constraints [29], using a one-pass streaming algorithm for DR-submodular maximization with a knapsack constraint over the integer lattice [30], and using streaming algorithms for maximizing a monotone DR-submodular function with a cardinality constraint on the integer lattice [31].

Our contribution. In this paper, we focus on *the maximization of monotone DR-submodular function under cardinality constraint on the integer lattice* (the MDRSCa problem in Definition 2). In surveying the literature, there are two novel methods for this problem. First, Soma et al. [11] proposed the Cardinality constraint/DR-submodular algorithm (called CaDRS), which interpolates between the classical greedy algorithm and a truly continuous algorithm. This algorithm achieves an approximation ratio of $(1 - 1/e - \epsilon)$ in $O(\frac{n}{\epsilon} \log B \log \frac{k}{\epsilon})$ complexity. Second, Zhang et al. [31] first devised a streaming algorithm based on Sieve streaming [32]. Zhang's method achieves an approximation ratio of $(1/2 - \epsilon)$ in $O(\frac{k}{\epsilon})$ memory and $O(\frac{k}{\epsilon} \log^2 k)$ query complexity. Inspired by Zhang's method [31], our study based on the streaming method devises two *improved streaming algorithms* for the problem and obtains some positive results compared to state-of-the-art algorithms. Specifically, our main contributions are as follows.

- To resolve the MDRSCa problem, we first devise an algorithm (called StrOpt) to handle each element by scanning the data with the assumption of a known optimal value (OPT). We prove that StrOpt guarantees the theoretical result with an approximation ratio of $(1/2)$. Next, we devise a $(1/4)$ -approximation streaming algorithm (called Stepping-Stone algorithm), which has the procedure role to calculate the threshold for the main algorithm. Later, based on StrOpt and Stepping-Stone algorithms, we provide two main streaming algorithms to solve this problem. They are named StrDRS1 and StrDRS2. Because OPT cannot be determined in actual situations, we estimate OPT based on a conventional method by observing $\text{OPT} \in [m, 2km]$ where $m = \max_{e \in E} \{f(\chi_e)\}$. Based on estimated OPT, StrDRS1, a one-pass streaming algorithm, has an approximation ratio of $(1/2 - \epsilon)$ and takes $O(\frac{n}{\epsilon} \log(\frac{\log B}{\epsilon}) \log k)$ queries. For StrDRS2, we first find a temporary result that satisfies the cardinality constraint by the Stepping-Stone algorithm. Subsequently, we increase the approximation solution ratio in StrDRS2 by finding elements that hold the threshold restriction of the above temporary result. The StrDRS2 is a multi-pass streaming algorithm that scans $O(\frac{1}{\epsilon})$ passes, takes $O(\frac{n}{\epsilon} \log B)$ queries, and returns an approximation ratio of $(1 - 1/e - \epsilon)$.
- We further investigate the performance of our algorithms by performing some experiments on some datasets of practical applications. We run four algorithms, StrDRS1, StrDRS2, CaDRS [11], and SieveStr++ [31], to compare their performance. The results indicate that our algorithms provide solutions with a theoretically guaranteed value of the objective function and outperform the state-of-the-art algorithm in both the number of queries and the runtime.

Table 1 shows how our algorithms compare theoretical properties with current state-of-the-art algorithms for the problem of maximizing monotone DR submodular functions with a cardinality constraint in the integer lattice.

Table 1. State-of-the-art algorithms for the problem of monotone DR-submodular function maximization with a cardinality constraint on the integer lattice in terms of time complexity.

Reference	Pass	Ratio	Query Complexity
CaDRS	$O(\frac{1}{\epsilon} \log \frac{1}{\epsilon})$	$1 - 1/e - \epsilon$	$O(\frac{n}{\epsilon} \log B \log \frac{k}{\epsilon})$
SieveStr + +	1	$1/2 - \epsilon$	$O(\frac{k}{\epsilon} \log^2 k)$
StrDRS1	1	$1/2 - \epsilon$	$O(\frac{n}{\epsilon} \log(\frac{\log B}{\epsilon}) \log k)$
StrDRS2	$O(\frac{1}{\epsilon})$	$1 - 1/e - \epsilon$	$O(\frac{n}{\epsilon} \log B)$

Organization. The structure of our paper is as follows: Section 1 introduces the development situation of the submodular function maximization on a-set and multi-set. Primarily, we focus on maximizing the monotone DR-submodular function on the integer lattice under cardinality constraint and present the main contributions of our study. Section 2 reviews the related work. The definition of the problem and some notation are introduced in Section 3. Section 4 contains our proposed algorithms and theoretical analysis. Section 5 shows the experimental results and evaluation. Finally, Section 6 concludes the paper and future work.

2. Related Work

A considerable amount of literature has been published on the maximization of monotone submodular functions under many different constraints over many decades. Nemhauser et al. [33] are pioneers in studying the approximations for maximizing submodular set functions in combinatorial optimization and machine learning. They proved that the standard greedy algorithm gives a $(1/2)$ -approximation under a matroid constraint and a $(1 - 1/e)$ -approximation under a cardinality constraint. Their method served as a model for further development. Later, Sviridenko [34] developed an improved greedy algorithm for maximizing a submodular set function subject to a knapsack constraint. This algorithm achieves a $(1 - 1/e)$ -approximation with $O(n^5)$ time complexity for a knapsack constraint. Subsequently, Calinescu et al. [35] first devise a $(1 - 1/e)$ -approximation algorithm for maximizing a monotone submodular function subject to a matroid constraint. This method combines a continuous greedy algorithm and pipage rounding. The pipage rounding rounds the approximate fractional solution of the continuous greedy approach to obtain an integral feasible solution. Recently, Badanidiyuru et al. [36] design a $(1 - 1/e - \epsilon)$ -approximation algorithm with any fixed constraint $\epsilon > 0$ for maximizing submodular functions. This algorithm takes $O(\frac{n}{\epsilon} \log \frac{n}{\epsilon})$ time complexity for the cardinality constraint.

Several studies have recently begun investigating the maximization of DR-submodular functions on the integer lattice under various constraints. Soma et al. (2014) [13] studied the monotone DR-submodular function maximization over integer lattices under a knapsack constraint. They proposed a simple greedy algorithm, which has an approximation ratio of $(1 - 1/e)$ and a pseudo-polynomial time complexity. Next, Soma et al. (2018) [11] continued to develop polynomial-time approximation algorithms for the problem of DR-submodular function maximization under a cardinality constraint, a knapsack constraint, and a polymatroid constraint on the integer lattice, respectively. For the cardinality constraint, they devised an algorithm based on the decreasing threshold greedy framework. For the polymatroid constraint, they developed an algorithm based on an extension of continuous greedy algorithms. For the knapsack constraint, they used the decreasing threshold greedy framework as the algorithm of the cardinality constraint. However this algorithm takes its initial solution as an input, whereas the algorithm for cardinality constraints always uses the zero vector as the initial solution. All three algorithms have polynomial time and

achieve a $(1 - 1/e - \epsilon)$ -approximation ratio. Besides, Some et al. (2017) [37] also studied the problem of non-monotone DR-submodular function maximization. They proposed a double greedy algorithm, which has $\frac{1}{2+\epsilon}$ -approximation and $O(\frac{n}{\epsilon} \log^2 B)$. Subsequently, Gu et al. (2020) [24] study the problem of maximizing the non-monotone DR-submodular function on the bounded integer lattice. They propose a fast double greedy algorithm that improves runtime. Their result achieves a $1/2$ -approximation algorithm with a $O(n \log B)$ time complexity. Liu et al. (2021) [29] develop two streaming algorithms for maximizing DR-submodular functions under the d -knapsack constraints. The first is a one-pass streaming algorithm that achieves a $(\frac{1-\theta}{1+d})$ -approximation with $O(\frac{\log(d\beta^{-1})}{\beta\epsilon})$ memory complexity and $O(\frac{\log(d\beta^{-1})}{\epsilon} \log B)$ update time per element, where $\theta = \min(\alpha + \epsilon, 0.5 + \epsilon)$ and α, β are the upper and lower bounds for the cost of each item in the stream. The second is an improved streaming algorithm to reduce the memory complexity to $O(\frac{d}{\beta\epsilon})$ with an unchanged approximation ratio and query complexity. Zhang et al. (2021) [31] based on the Sieve streaming method to develop a streaming algorithm for the problem of monotone DR-submodular function under cardinality constraint on the integer lattice. This algorithm achieves an approximation ratio of $(1/2 - \epsilon)$ and takes $O(\frac{k}{\epsilon} \log^2 k)$ complexity. This is the problem that we study in this paper. Most recently, Tan et al. (2022) [30] design an one-pass streaming algorithm for the problem of DR-submodular maximization with a knapsack constraint over the integer lattice, called DynamicMRT, which achieves a $(1/3 - \epsilon)$ -approximation ratio, a memory complexity $O(K \log K/\epsilon)$, and query complexity $O(\log^2 K/\epsilon)$ per element for the knapsack constraint K . Meanwhile, Gong et al. (2022) [28] consider the problem of non-negative monotone DR-submodular function maximization over a bounded integer lattice. They present a deterministic algorithm and theoretically reduce its runtime to a new record, $O((\frac{1}{\epsilon}^{O(1/\epsilon^5)} \cdot n \log \frac{1}{c_{\min}} \log B))$, (where $c_{\min} = \min_{e \in E} c(e)$ and $c(\cdot)$ is a cost function defined in E) with the approximate ratio of $(1 - 1/e - O(\epsilon))$.

All the studies mentioned above consider the problem of maximizing the submodular function on a set function or maximizing the DR-submodular function on the integer lattice under different constraints. Only the studies of Soma et al. in [11] and Zhang et al. in [31] consider the problem of MDRSCa, as mentioned in the contribution section. Motivated by these studies, we proposed two improved streaming algorithms for the MDRSCa problem. Our algorithms achieve better than state-of-the-art methods through theoretical analysis and experimental results.

3. Preliminaries

This section introduces the definitions of the monotone DR-submodular, MDRSCa problem and its associated notations. Table 2 summarizes the usually used notations in this paper.

Table 2. Table of the usually used notations in this paper.

Notation	Description
E	a ground set, $E = \{e_1, \dots, e_n\}$
n	the number of elements in the ground set E .
2^E	the subset family of E .
A, B	the arbitrary subsets of E
\mathbf{x}, \mathbf{y}	the arbitrary vectors of \mathbb{Z}_+^E
χ_e	the unit vector with coordinate e , $e \in E$
$\{\mathbf{x}\}$	the multiset contains elements in vector \mathbf{x} , where each element $e \in E$ can appear many times.
$\mathbf{x}(e), \mathbf{y}(e)$	the coordinate value of entry e in vector \mathbf{x}, \mathbf{y} , where $e \in E$
$\ \mathbf{x}\ _\infty$	the infinity norm of vector \mathbf{x} , $\ \mathbf{x}\ _\infty := \max_{e \in E} \mathbf{x}(e)$
$\ \mathbf{x}\ _1$	the taxicab norm of vector \mathbf{x} , $\ \mathbf{x}\ _1 := \sum_{e \in E} \mathbf{x}(e)$.
$\mathbf{0}$	the vector zero whose value $\mathbf{0}(e) = 0, \forall e \in E$
\mathbf{B}	the upper bound vector of \mathbf{x} , $\mathbf{0} \leq \mathbf{x} \leq \mathbf{B}$
B	$B := \ \mathbf{B}\ _\infty$
k	the upper bound of total elements in vector \mathbf{x} on the integer lattice \mathbb{Z}_+^E , $\mathbf{x}(E) \leq k$
k_e	the number of copies of e to be considered for addition to \mathbf{x}
k'	the number of copies of e add to \mathbf{x}
$[k]$	the set of $\{1, \dots, k\}$
v	an optimal value of the object function, $(1 - \epsilon)\text{OPT} \leq v \leq \text{OPT}$ with $(\epsilon \in (0, 1/2))$
$\mathbf{x} \vee \mathbf{y}$	the coordinate-wise maximum of \mathbf{x} and \mathbf{y}
$(\mathbf{x} \vee \mathbf{y})(e)$	$\mathbf{x} \vee \mathbf{y} := \max\{\mathbf{x}(e), \mathbf{y}(e)\}$
$\mathbf{x} \wedge \mathbf{y}$	the coordinate-wise minimum of \mathbf{x} and \mathbf{y}
$(\mathbf{x} \wedge \mathbf{y})(e)$	$\mathbf{x} \wedge \mathbf{y} := \min\{\mathbf{x}(e), \mathbf{y}(e)\}$
$\mathbf{x} + \mathbf{y}$	sum of 2 vectors \mathbf{x} and \mathbf{y} , with the multiset $\{\mathbf{x} + \mathbf{y}\}$ whose e appears $(\mathbf{x}(e) + \mathbf{y}(e))$ times.
$\mathbf{x} - \mathbf{y}$	$\mathbf{x} - \mathbf{y} = \mathbf{x} + (-\mathbf{y})$
$f(\mathbf{x})$	the object function value of \mathbf{x}
$f(\mathbf{x} \mathbf{y})$	$f(\mathbf{x} \mathbf{y}) = f(\mathbf{x} + \mathbf{y}) - f(\mathbf{y})$

3.1. Notation

For a positive integer $k \in \mathbb{N}$, $[k]$ denotes the set $\{1, \dots, k\}$. Given a ground set $E = \{e_1, \dots, e_n\}$, we denote the i -th entry of a vector $\mathbf{x} \in \mathbb{Z}_+^E$ by $\mathbf{x}(i)$, and for each $e \in E$, we define the e -th unit vector with $\chi_e(t) = 1$ if $t = e$ and $\chi_e(t) = 0$ if $t \neq e$.

For $\mathbf{x} \in \mathbb{Z}_+^E$, $\{\mathbf{x}\}$ denotes the multiset where the element e appears $\mathbf{x}(e)$ times and with a subset $A \subseteq E$, $\mathbf{x}(A) = \sum_{e \in A} \mathbf{x}(e)$ and $\text{supp}^+(\mathbf{x}) = \{e \in E | \mathbf{x}(e) > 0\}$. According to the definition of the vector norm, we have $\|\mathbf{x}\|_\infty := \max_{e \in E} \mathbf{x}(e)$ and $\|\mathbf{x}\|_1 := \sum_{e \in E} \mathbf{x}(e)$.

For two vectors $\mathbf{x}, \mathbf{y} \in \mathbb{Z}_+^E$, $\mathbf{x} \leq \mathbf{y}$ signifies $\forall e \in E$ then $\mathbf{x}(e) \leq \mathbf{y}(e)$. Furthermore, given $\mathbf{x}, \mathbf{y} \in \mathbb{Z}_+^E$, $\mathbf{x} \vee \mathbf{y}$ and $\mathbf{x} \wedge \mathbf{y}$ denote the coordinate-wise maximum and minimum, respectively. This means that $(\mathbf{x} \vee \mathbf{y})(e) := \max\{\mathbf{x}(e), \mathbf{y}(e)\}$ and $(\mathbf{x} \wedge \mathbf{y})(e) := \min\{\mathbf{x}(e), \mathbf{y}(e)\}$. In addition, $\mathbf{x} + \mathbf{y}$ denotes the multiset $\{\mathbf{x} + \mathbf{y}\}$ where the element e appears $(\mathbf{x}(e) + \mathbf{y}(e))$ times. Thus, we can infer $\mathbf{x} - \mathbf{y} = \mathbf{x} + (-\mathbf{y})$.

3.2. Definition

For function $f : \mathbb{Z}_+^E \rightarrow \mathbb{R}_+$, we define $f(\mathbf{x}|\mathbf{y}) = f(\mathbf{x} + \mathbf{y}) - f(\mathbf{y})$.

Definition 1 (Monotone DR-submodular function). A function $f : \mathbb{Z}_+^E \rightarrow \mathbb{R}_+$ is monotone if $f(\mathbf{x}) \leq f(\mathbf{y})$ for all $\mathbf{x}, \mathbf{y} \in \mathbb{Z}_+^E$ with $\mathbf{x} \leq \mathbf{y}$ and f is said to be diminishing return submodular (DR-submodular), if

$$f(\mathbf{x} + \chi_e) - f(\mathbf{x}) \geq f(\mathbf{y} + \chi_e) - f(\mathbf{y}) \quad (6)$$

Definition 2 (Maximization of monotone DR-submodular function under cardinality constraint on the integer lattice—MDRSCa problem). Let $\mathbf{B} \in \mathbb{Z}_+^E$, $B = \|\mathbf{B}\|_\infty$ and an integer $k > 0$, we consider the DR-submodular function under cardinality constraint as follows

$$\text{maximize: } f(\mathbf{x}) \text{ subject to: } \mathbf{0} \leq \mathbf{x} \leq \mathbf{B}, x(E) \leq k \quad (7)$$

4. Proposed Algorithm

This section presents descriptions and theoretical analysis of the algorithms we have proposed for the MDRSCa problem, including a streaming algorithm with the assumption that the optimal value is known (StrOpt), the Stepping-Stone algorithm and two main streaming algorithms (StrDRS1, StrDRS2).

4.1. Streaming Algorithm with Approximation Ratio of $(1/2 - \epsilon)$

First, we propose StrOpt, a single-pass streaming algorithm for the MDRSCa problem under the assumption that the optimal value of the objective function is known. Afterwards, we use the traditional method to estimate the optimal value and devise the main one-pass streaming algorithm called StrDRS1.

4.1.1. Algorithm with Knowing Optimal Value—StrOpt

Algorithm description. The detail of StrOpt is fully presented in Algorithm 1.

Algorithm 1: StrOpt($f, \mathbf{B}, k, \epsilon, v$)

Input: $f : \mathbb{Z}_+^E \rightarrow \mathbb{R}_+$, \mathbf{B}, k, ϵ , a guess of optimal value v

Output: A vector \mathbf{x}

```

1:  $\mathbf{x} \leftarrow \mathbf{0}$ 
2: foreach  $e \in E$  do
3:    $I \leftarrow \{i_1, i_2, \dots, i_{|I|}\} : i_1 < i_2 < \dots < i_{|I|} \leftarrow \{\lceil \mathbf{B}(e)(1 - \epsilon)^i \rceil : i \in \mathbb{Z}, 1 \leq$ 
      $\mathbf{B}(e)(1 - \epsilon)^i \leq \mathbf{B}(e)\}$ 
4:   Find  $k_e \leftarrow \arg \min\{i_j - 1 : i_j \in I, f(i_j \chi_e | \mathbf{x}) / i_j < \frac{v}{2k}\}$  by a binary search
5:    $k' \leftarrow \min\{k_e, k - \|\mathbf{x}\|_1\}$ 
6:   if  $k' \neq 0$  then
7:      $\mathbf{x} \leftarrow \mathbf{x} + k' \cdot \chi_e$ 
8:   else
9:     break
10: return  $\mathbf{x}$ 

```

We assume that the optimal value OPT of the objective function of MDRSCa is already known. StrOpt is created to find vector \mathbf{x} using this OPT. Given a known optimal value v that satisfies $(1 - \epsilon)\text{OPT} \leq v \leq \text{OPT}$ for any $\epsilon \in (0, \frac{1}{2})$. When each element e arrives, we find a set I , which is the set of positive integers predicted to be the number of copies of e . Then, we use the binary search with threshold $\frac{v}{2k}$ to find the minimum k_e that holds $f(k_e \chi_e | \mathbf{x}) / k_e < \frac{v}{2k}$. We denote by k' the number of copies of e that adds the result vector \mathbf{x} . The value k' is the minimum of two values k_e and the rest of elements \mathbf{x}' in the cardinality k . If k' is equal to 0, then e is not selected in \mathbf{x} . Otherwise, e is selected in \mathbf{x} with k' copies.

Theoretical analysis. Lemma 1, Theorem 1, and their proofs demonstrate the theoretical solution guarantee of StrOpt. On the basis of that, we devise the first main streaming algorithm for the MDRSCa problem.

Lemma 1. We have $f(k_e \chi_e | \mathbf{x}) \geq (1 - \epsilon)k_e \frac{v}{2k}$.

Proof. Assume that $k_e = i_j = \lceil x \rceil$ where $x = \mathbf{B}(e)(1 - \epsilon)^i$ with some $i \in I$. We have $i_j \geq i_{j-1} + 1$ and

$$\begin{aligned} k_e - i_{j-1} &= i_j - (i_{j-1} + 1) = \lceil x \rceil - (\lceil (1 - \epsilon)x \rceil + 1) \\ &\leq \lceil x \rceil - (\lceil x \rceil + \lceil (-\epsilon x) \rceil) = -\lceil (-\epsilon x) \rceil \\ &= \lfloor \epsilon x \rfloor \leq \epsilon x \leq \epsilon k_e \end{aligned}$$

Therefore, $i_{j-1} \geq (1 - \epsilon)k_e$. From the combination of the selection k_e and the monotonicity of f , we have the following.

$$f(k_e \chi_e | \mathbf{x}) \geq f(i_{j-1} \chi_e | \mathbf{x}) \geq i_{j-1} \frac{v}{2k} \geq (1 - \epsilon)k_e \frac{v}{2k} \quad (8)$$

The proof is completed. \square

Theorem 1. For any $\epsilon \in (0, 1/2)$ and $(1 - \epsilon)\text{OPT} \leq v \leq \text{OPT}$, the Algorithm 1 takes $O(n \log(\frac{1}{\epsilon} \log B))$ queries and returns a solution \mathbf{x} satisfying $f(\mathbf{x}) \geq (1 - \epsilon)v/2$.

Proof. The Algorithm 1 scans only one time over E and each incoming element e , it takes $\log |I| = O(\log(\frac{1}{\epsilon} \log B))$ queries to find k_e . The total number of required queries of the algorithm is $O(n \log(\frac{1}{\epsilon} \log B))$.

Denote \mathbf{x}_i and $k_i \chi_{e_i}$ as the solution at the beginning of iteration i and the additional vector in the current solution at iteration i , respectively. We consider two following cases:

Case 1. If $\|\mathbf{x}\|_1 = k$, we have $k_1 + k_2 + \dots + k_n = k$ thus:

$$f(\mathbf{x}) = \sum_{i=1}^n f(k_i \chi_{e_i} | \mathbf{x}_i) \geq \sum_{i=1}^n (1 - \epsilon)k_i \frac{v}{2k} = \frac{(1 - \epsilon)v}{2} \quad (9)$$

Case 2. If $\|\mathbf{x}\|_1 < k$, after ending the main loop, we have $f(e | \mathbf{x}) \leq \frac{v}{2k}$ for all $e \in \{\mathbf{B} - \mathbf{x}\}$. Therefore:

$$\begin{aligned} f(\mathbf{o}) - f(\mathbf{x}) &= f(\mathbf{o} \vee \mathbf{x}) - f(\mathbf{x}) \\ &= \sum_{e \in \{\mathbf{o} \vee \mathbf{x} - \mathbf{x}\}} f(\chi_e | \mathbf{x}) \\ &= \sum_{e \in \{\mathbf{o} - \mathbf{o} \wedge \mathbf{x}\}} f(\chi_e | \mathbf{x}) \\ &< \sum_{e \in \{\mathbf{o} - \mathbf{o} \wedge \mathbf{x}\}} \frac{v}{2k} \leq \frac{v}{2} \end{aligned}$$

where the second equality follows from the lattice identity $\mathbf{x} \vee \mathbf{y} - \mathbf{y} = \mathbf{x} - \mathbf{x} \wedge \mathbf{y}$ for $\mathbf{x}, \mathbf{y} \in \mathbb{Z}_+^E$. We have $f(\mathbf{x}) \geq \text{OPT} - v/2 \geq v/2$. The proof is completed. \square

4.1.2. $(1/2 - \epsilon)$ -Approximation Streaming Algorithm—StrDRS1 Algorithm

Algorithm description. The detail of this algorithm is fully presented in Algorithm 2.

Algorithm 2: Streaming-I algorithm (StrDRS1)

Input: $f : \mathbb{Z}_+^E \rightarrow \mathbb{R}_+, \mathbf{B}, k, \epsilon$
Output: A $(1/2 - \epsilon)$ -approximation solution \mathbf{x}

```

1:  $O = \{(1 + \epsilon)^i | i \in \mathbb{Z}_+\}$ 
2:  $\mathbf{x}^v = \mathbf{0}, \forall v \in O, m \leftarrow 0$ 
3: foreach  $e \in E$  do
4:    $m \leftarrow \max\{f(\chi_e), m\}$ 
5:    $O = \{(1 + \epsilon)^i | i \in \mathbb{Z}_+, m \leq (1 + \epsilon)^i \leq 2km\}$ 
6:    $I \leftarrow \{i_1, i_2, \dots, i_{|I|}\} : i_1 < i_2 < \dots < i_{|I|} \leftarrow \{\lceil \mathbf{B}(e)(1 - \epsilon)^i \rceil : i \in \mathbb{Z}, 1 \leq \mathbf{B}(e)(1 - \epsilon)^i \leq \mathbf{B}(e)\}$ 
7:   for  $v \in O$  do
8:     Find  $k_e \leftarrow \arg \min\{i \in I : f(i\chi_e | \mathbf{x}^v) < i \cdot \frac{v}{2k}\}$  by a binary search
9:      $k' \leftarrow \min\{k_e, k - \|\mathbf{x}^v\|_1\}$ 
10:    if  $k' \neq 0$  then
11:       $\mathbf{x}^v \leftarrow \mathbf{x}^v + k' \cdot \chi_e$ 
12:    else
13:      break
14: return  $\arg \max_{\mathbf{x}^v, v \in O} f(\mathbf{x}^v)$ 

```

Based on the analysis of StrOpt, and the working frame of the Sieve streaming algorithm [38], we design the StrDRS1 algorithm for the MDRSCa problem with the following main idea. We find a set of solutions \mathbf{x}^v of OPT, where $v \in O$ and O is the set of values that changes according to the maximum value of the unit standard vector on the arriving elements. Besides, we find a set I , which contains positive integers predicted to be the number of copies of each element e if e is selected in \mathbf{x}^v . For each solution \mathbf{x}^v , $v \in O$, the algorithm finds k_e , is the smallest value in I so that the current element e satisfies the condition in line 8 by binary search. Then, we choose k' , which is the minimum value between k_e and $k - \|\mathbf{x}^v\|_1$. If k' is not equal to 0, this means that e is selected in \mathbf{x}^v with k' copies. Otherwise, e is not selected in \mathbf{x}^v . In the end, the result \mathbf{x} is \mathbf{x}^v , which makes $f(\mathbf{x})$ maximal.

Theoretical analysis. We analyze the complexity of StrDRS1, stated in Theorem 2.

Theorem 2. StrDRS1 is a single-pass streaming algorithm, has an approximation ratio of $(\frac{1}{2} - \epsilon)$ and takes $O(\frac{n}{\epsilon} \log(\frac{\log B}{\epsilon}) \log k)$ queries.

Proof. By the definition of O , there exists an integer i such that

$$(1 - \epsilon)\text{OPT} \leq \frac{\text{OPT}}{1 + \epsilon} \leq v = (1 + \epsilon)^i \leq \text{OPT}$$

By applying the proof of Theorem 1, and the working frame of the Sieve streaming algorithm in [38], we obtain:

$$f(\mathbf{x}^v) \geq \frac{(1 - \epsilon)}{2} v \geq \frac{(1 - \epsilon)^2}{2} \text{OPT} \geq (\frac{1}{2} - \epsilon) \text{OPT} \quad (10)$$

The proof is completed. \square

4.2. Streaming Algorithm with Approximation Ratio of $(1 - 1/e - \epsilon)$

In this section, we introduce two more algorithms for the MDRSCa problem, including one with the role of a stepping stone (called Stepping-Stone algorithm) and the second main algorithm (called StrDRS2 algorithm) in our study.

4.2.1. $(1/4)$ -Approximation Streaming Algorithm—Stepping-Stone Algorithm

Algorithm description. The detail of this algorithm is fully presented in Algorithm 3.

Algorithm 3: $(1/4)$ -approximation algorithm (Stepping-Stone algorithm)

Input: $f : \mathbb{Z}_+^E \rightarrow \mathbb{R}_+, \mathbf{B}, k, \epsilon$

Output: A vector \mathbf{x}

1: **foreach** $e \in E$ **do**

2: $I \leftarrow \{i_1, i_2, \dots, i_{|I|}\} : i_1 < i_2 < \dots < i_{|I|} \leftarrow \{\lceil \mathbf{B}(e)(1 - \epsilon)^i \rceil : i \in \mathbb{Z}, 1 \leq \mathbf{B}(e)(1 - \epsilon)^i \leq \mathbf{B}(e)\}$

3: Find $k_e \leftarrow \arg \max \{i_t - 1 : i_t \in I, f(\chi_e | \mathbf{x} + (i_t - 1)\chi_e) < f(\mathbf{x} + i_{t-1}\chi_e)/k \text{ and } f(\chi_e | \mathbf{x} + (i_j - 1)\chi_e) \geq f(\mathbf{x} + i_{j-1}\chi_e)/k, \forall j \leq t - 1, j \in I\}$

4: $\mathbf{x} \leftarrow \mathbf{x} + k_e \cdot \chi_e$

5: $\mathbf{x} \leftarrow$ last elements in \mathbf{x} with $\|\mathbf{x}\|_1 = k$

6: **return** \mathbf{x}

We design the Stepping-Stone algorithm, which is a $(1/4)$ -approximation streaming algorithm. The Stepping-Stone algorithm differs from StrDRS1 and StrDRS2 in that it only selects elements for exactly one solution and has an approximately constant value. In contrast, the other two algorithms find multiple solution candidates and choose the best candidate.

In more detail, the main idea of this algorithm differs from StrDRS1, that is, the Stepping-Stone algorithm is a single-pass streaming algorithm and finds k_e without relying on a given v . In this way, after finding the set I as StrDRS1, for each element $e, e \in E$, k_e is the largest $i_t - 1, i_t \in I$ so that it meets the conditions in line 3. Finally, the output contains the last elements of \mathbf{x} with $\|\mathbf{x}\|_1 = k$.

Theoretical analysis. Lemmas 2–4, and Theorem 3 clarify the theoretical analysis of the Stepping-Stone algorithm.

Lemma 2. After each iteration of the Stepping-Stone algorithm, we have $f(k_e \chi_e | \mathbf{x}) \geq (1 - \epsilon)k_e f(\mathbf{x})/k$.

Proof. Due to the definition of I , after each iteration of the main loop, we have $i_t - 1 \geq i_{t-1}$. Similarly to the proof of Lemma 2, we have $k_e - i_{t-1} = i_t - 1 - i_{t-1} \leq \epsilon k_e$. By selection of the algorithm, for $1 \leq j < t$ we have

$$f(i_j | \mathbf{x} + i_{j-1}\chi_e) = \sum_{l=i_{j-1}+1}^{i_j} f(\chi_e | \mathbf{x} + i_{j-1}\chi_e) \geq \sum_{l=i_{j-1}+1}^{i_j} f(\mathbf{x} + i_{j-1}\chi_e)/k \quad (11)$$

$$\geq (i_j - i_{j-1})f(\mathbf{x} + i_{j-1}\chi_e)/k \quad (12)$$

Therefore:

$$f(k_e \chi_e | \mathbf{x}) \geq f(i_{t-1} \cdot \chi_e | \mathbf{x}) \quad (13)$$

$$\geq \sum_{j=1}^{t-1} (i_j - i_{j-1}) f(\chi_e | \mathbf{x} + i_{j-1} \cdot \chi_e) \quad (14)$$

$$= i_{t-1} f(\mathbf{x}) / k \geq (1 - \epsilon) k_e f(\mathbf{x}) / k \quad (15)$$

The proof is completed. \square

Lemma 3. After the main loop of the Stepping-Stone algorithm, we have $2f(\mathbf{x}) \geq \text{OPT}$.

Proof. Denote $\mathbf{x}_{(e)}$ is \mathbf{x} right before the element e starts to proceed. We have the following.

$$f(\mathbf{o}) - f(\mathbf{x}) = f(\mathbf{o} \vee \mathbf{x}) - f(\mathbf{x}) \quad (16)$$

$$= \sum_{e \in \{\mathbf{o} \vee \mathbf{x} - \mathbf{x}\}} f(\chi_e | \mathbf{x}) \quad (17)$$

$$= \sum_{e \in \{\mathbf{o} - \mathbf{o} \wedge \mathbf{x}\}} f(\chi_e | \mathbf{x}) \quad (18)$$

$$\leq \sum_{e \in \{\mathbf{o} - \mathbf{o} \wedge \mathbf{x}\}} f(\chi_e | \mathbf{x}_{(e)}) \quad (19)$$

$$< \sum_{e \in \{\mathbf{o} - \mathbf{o} \wedge \mathbf{x}\}} \frac{f(\mathbf{x}_{(e)})}{k} \leq f(\mathbf{x}) \quad (20)$$

which implies the proof. \square

Lemma 4. At the end of the Stepping-Stone algorithm, we have $f(\mathbf{x}') \geq \frac{1-3\epsilon}{2-3\epsilon} f(\mathbf{x})$.

Proof. If $\|\mathbf{x}'\|_1 < k$, then $\mathbf{x}' = \mathbf{x}$ and Lemma 4 holds. We consider the case $\|\mathbf{x}'\|_1 = k$. Assume that $\text{supp}(\mathbf{x}) = \{e_1, e_2, \dots, e_l\}$, $\mathbf{x}_i = \sum_{j=1}^i \mathbf{x}(e_j) \cdot \chi_{e_j}$, $\text{supp}(\mathbf{x}') = \{e_p, e_{p+1}, \dots, e_l\}$ and $\mathbf{x}^1 = \mathbf{x} - \mathbf{x}'$ where e_j is added to \mathbf{x} immediately after e_{j-1} and $1 \leq p < l$.

We further consider two cases.

Case 1. If $\{\mathbf{x}^1\} \cap \{\mathbf{x}'\} = \emptyset$, we have $k = \sum_{i=p}^l k_{e_i}$ and

$$f(\mathbf{x}) - f(\mathbf{x}^1) = \sum_{i=p}^l f(k_{e_i} \chi_{e_i} | \mathbf{x}_i) \geq \sum_{i=p}^l (1 - \epsilon) k_{e_i} \frac{f(\mathbf{x}_i)}{k} \quad (\text{Lemma 2}) \quad (21)$$

$$\geq (1 - \epsilon) \sum_{i=p}^l k_{e_i} \frac{f(\mathbf{x}^1)}{k} = (1 - \epsilon) f(\mathbf{x}^1) \quad (22)$$

Case 2. If $\{\mathbf{x}^1\} \cap \{\mathbf{x}'\} = \{e_p\}$. Denote $q = k_{e_p} - \mathbf{x}_1(e_p)$ and $c = \min\{i_j \in I : i_j \geq \mathbf{x}_1(e_p)\}$. We have $k = q + \sum_{i=p+1}^l k_{e_i}$ and $i_{j-1} < \mathbf{x}_1(e_p) \leq c = i_j$. Similarly to the proof of Lemma 2, we have $c - \mathbf{x}_1(e_p) \leq \epsilon i_j \leq \epsilon k_{e_p}$ and thus $c \leq \epsilon i_j + \mathbf{x}_1(e_p) \leq \epsilon k_{e_p} + \mathbf{x}_1(e_p)$. Let $\mathbf{x}_l^1 = \mathbf{x}^1 + l \chi_{e_p}$, then

$$f(q \chi_e | \mathbf{x}^1) \geq \sum_{l=c+1}^{i_{t-1}} \frac{f(\chi_{e_p} | \mathbf{x}^1 + (l-1)k_{e_p})}{k} \quad (23)$$

$$\geq (k_{e_p}(1 - \epsilon) - c) \frac{f(\mathbf{x}_c^1)}{k} = (k_{e_p}(1 - \epsilon) - (\mathbf{x}_1(e_p) + \epsilon k_{e_p})) \frac{f(\mathbf{x}^1)}{k} \quad (24)$$

$$\geq (q - 2\epsilon k_{e_p}) \frac{f(\mathbf{x}^1)}{k} \quad (25)$$

implying that $f(q\chi_e|\mathbf{x}^1) \geq (q - 2\epsilon k_{e_p}) \frac{f(\mathbf{x}^1)}{k}$. Therefore:

$$f(\mathbf{x}) - f(\mathbf{x}^1) = f(q\chi_e|\mathbf{x}^1) + \sum_{i=p+1}^l f(k_{e_i}\chi_{e_i}|\mathbf{x}_i) \quad (26)$$

$$\geq (q - 2\epsilon k_{e_p}) \frac{f(\mathbf{x}^1)}{k} + \sum_{i=p+1}^l (1 - \epsilon) k_{e_i} \frac{f(\mathbf{x}_i)}{k} \quad (\text{Lemma 2}) \quad (27)$$

$$\geq (q + \sum_{i=p+1}^l k_{e_i} - 2\epsilon k_{e_p} - \epsilon \sum_{i=p+1}^l k_{e_i}) \frac{f(\mathbf{x}^1)}{k} \quad (28)$$

$$\geq (k - 3\epsilon k) \frac{f(\mathbf{x}^1)}{k} = (1 - 3\epsilon) f(\mathbf{x}^1) \quad (29)$$

Hence, $f(\mathbf{x}) \geq (2 - 3\epsilon) f(\mathbf{x}^1)$. Combined with the fact that $f(\mathbf{x}) \leq f(\mathbf{x}') + f(\mathbf{x}^1)$, we have $f(\mathbf{x}') \geq \frac{1-3\epsilon}{2-3\epsilon} f(\mathbf{x})$, which completes the proof. \square

Theorem 3. The Stepping-Stone algorithm is a single-pass streaming algorithm that takes $O(\frac{n}{\epsilon} \log B)$ and provides an approximation ratio of $1/4 - 3\epsilon/4$.

Proof. The algorithm scans only one time over the ground set E and each element e , it calculates $f(\chi_e|\mathbf{x} + (i_j - 1)\chi_e)$ for all $i_j \in I$ to find k_e . This task takes at most $\frac{1}{\epsilon} \log(\mathbf{B}(e)) = O(\frac{1}{\epsilon} \log B)$ queries. Thus, the total number of required queries is $O(\frac{1}{\epsilon} \log B)$. For the proof of the approximation ratio, by using Lemmas 3 and 4, we have:

$$f(\mathbf{x}') \geq \frac{1-3\epsilon}{2-3\epsilon} f(\mathbf{x}) \geq \frac{1-3\epsilon}{2(2-3\epsilon)} \text{OPT} \geq (\frac{1}{4} - \frac{3}{4}\epsilon) \text{OPT} \quad (30)$$

The proof is completed. \square

4.2.2. $(1 - 1/e - \epsilon)$ -Approximation Streaming Algorithm—StrDRS2 Algorithm

Algorithm description. The detail of StrDRS2 is fully presented in Algorithm 4.

Algorithm 4: $(1 - 1/e - \epsilon)$ -approximation algorithm (StrDRS2)

Input: $f : \mathbb{Z}_+^E \rightarrow \mathbb{R}_+, \mathbf{B}, k, \epsilon$

Output: A vector \mathbf{x}

```

1:  $\mathbf{x}_0 \leftarrow$  Result of Algorithm 3,  $\Gamma \leftarrow f(\mathbf{x}_0)$ 
2:  $\theta = \frac{(4-3\epsilon)\Gamma}{(1-3\epsilon)k}$ ,  $\mathbf{x} \leftarrow \mathbf{0}$ 
3: while  $\theta \geq (1 - \epsilon)\Gamma / (4k)$  do
4:   foreach  $e \in E$  do
5:     Find  $k_e \leftarrow \arg \min\{i - 1 : i \in \{1, 2, \dots, \mathbf{B}(e)\}, f(i\chi_e|\mathbf{x})/i < \theta\}$  by a binary
       search
6:      $k' \leftarrow \min\{k_e, k - \|\mathbf{x}\|_1\}$ 
7:     if  $k' \neq 0$  then
8:        $\mathbf{x} \leftarrow \mathbf{x} + k' \cdot \chi_e$ 
9:     else
10:      break
11:    $\theta = (1 - \epsilon)\theta$ 
12: return  $\mathbf{x}$ 

```

We propose a $(1 - 1/e - \epsilon)$ -approximation algorithm, called StrDRS2. It is a multi-pass streaming algorithm and is based on the output of the Stepping-Stone algorithm (Algorithm 3) to compute the threshold θ of $f(k_e\chi_e|\mathbf{x})$ of each element e . The k_e of each e is

the minimal value $i, i \in \{1, 2, \dots, \mathbf{B}(e)\}$, so that $f(i\chi_e|\mathbf{x})/i < \theta$. The threshold θ decreases $(1 - \epsilon)$ times after each iteration.

Theoretical analysis. Lemma 5 and Theorem 4 clearly demonstrate the theoretical solution-ability guarantee of the StrDRS2 algorithm.

Lemma 5. *In the StrDRS2 algorithm, at any iteration of the outer loop, we have:*

$$f(k'_e\chi_e|\mathbf{x}) \geq \frac{(1-\epsilon)k_e}{k}(\text{OPT} - f(\mathbf{x})) \quad (31)$$

Proof. For the first iteration of the outer loop, we have $\mathbf{x} = \mathbf{0}$, and thus

$$f(\mathbf{0}) - f(\mathbf{x}) = k \frac{\text{OPT}}{k} \leq k \frac{(4-3\epsilon)\Gamma}{(1-3\epsilon)k} < \frac{k\theta}{1-\epsilon} \leq \frac{f(k'_e\chi_e|\mathbf{x}_i)}{(1-\epsilon)k'_e} \quad (32)$$

Thus, $f(k'_e\chi_e|\mathbf{x}) \geq \frac{(1-\epsilon)k'_e}{k}(\text{OPT} - f(\mathbf{x}))$, Lemma 5 is valid. For the latter iterations, the marginal gain of any element e with current vector \mathbf{x} is less than the threshold of previous iterations of the outer loop, i.e., $f(\chi_e|\mathbf{x}) \leq \frac{\theta}{1-\epsilon}$ for $e \in \{\mathbf{B} - \mathbf{x}\}$. Then,

$$f(\mathbf{0}) - f(\mathbf{x}_i) \leq f(\mathbf{0} \vee \mathbf{x}) - f(\mathbf{x}) \quad (33)$$

$$= \sum_{e \in \{\mathbf{0} \vee \mathbf{x} - \mathbf{x}\}} f(\chi_e|\mathbf{x}) \quad (34)$$

$$= \sum_{e \in \{\mathbf{0} - \mathbf{0} \wedge \mathbf{x}\}} f(\chi_e|\mathbf{x}) \quad (35)$$

$$\leq k \frac{\theta}{1-\epsilon} \leq \frac{f(k'_e\chi_e|\mathbf{x}_i)}{(1-\epsilon)k'_e} \quad (36)$$

The proof is completed. \square

Theorem 4. *The StrDRS2 algorithm is a multi-pass streaming algorithm that scans $O(\frac{1}{\epsilon})$ passes over the ground set, takes $O(\frac{n}{\epsilon} \log B)$ queries, and returns an approximation ratio of $(1 - 1/e - \epsilon)$.*

Proof. We consider following cases:

Case 1. If $\|\mathbf{x}\|_1 < k$, after the last iteration of the outer loop we have:

$$f(\mathbf{0}) - f(\mathbf{x}) \leq f(\mathbf{0} \vee \mathbf{x}) - f(\mathbf{x}) \quad (37)$$

$$= \sum_{e \in \{\mathbf{0} \vee \mathbf{x} - \mathbf{x}\}} f(\chi_e|\mathbf{x}) \quad (38)$$

$$= \sum_{e \in \{\mathbf{0} - \mathbf{0} \wedge \mathbf{x}\}} f(\chi_e|\mathbf{x}) \quad (39)$$

$$\leq k\theta_{\min} \leq k(1-\epsilon) \frac{\Gamma}{4k} \leq (1-\epsilon) \frac{\text{OPT}}{4} \quad (40)$$

Hence, $f(\mathbf{x}) \geq \frac{3+\epsilon}{4} \text{OPT}$.

Case 2. If $\|\mathbf{x}\|_1 = k$. Denote \mathbf{x}_i as \mathbf{x} after i -th update, $k'_{e_i}\chi_{e_i}$ is the vector added to \mathbf{x} at the i -th update, and the final solution $\mathbf{x} = \mathbf{x}_l$, Lemma 5 gives

$$f(\mathbf{x}_{i+1}) - f(\mathbf{x}_i) = f(k'_{e_{i+1}}\chi_{e_{i+1}}|\mathbf{x}_i) \geq \frac{(1-\epsilon)k'_{e_{i+1}}}{k}(\text{OPT} - f(\mathbf{x}_i)) \quad (41)$$

Rearrange the above inequality for $i + 1 = l$, and we have:

$$\text{OPT} - f(\mathbf{x}_l) \leq \left(1 - \frac{(1-\epsilon)k'_{e_l}}{k}\right)(\text{OPT} - f(\mathbf{x}_{l-1})) \quad (42)$$

$$\leq e^{-\frac{(1-\epsilon)k'_{e_l}}{k}}(\text{OPT} - f(\mathbf{x}_{l-1})) \quad (43)$$

$$\dots \leq e^{-\sum_{j=1}^l \frac{(1-\epsilon)k'_{e_j}}{k}} \text{OPT} \quad (44)$$

$$e^{-(1-\epsilon)} \text{OPT} \leq \left(\frac{1}{e} + \epsilon\right) \text{OPT} \quad (45)$$

Therefore, $f(\mathbf{x}) \geq (1 - 1/e - \epsilon) \text{OPT}$, the proof is completed. \square

5. Experiment

We conducted experiments based on the *budget allocation problem over the bipartite influence model* [39]. This problem is an instance of the *monotone submodular function maximization problem over the integer lattice under a constraint* [3]. As mentioned above, we consider the problem under a cardinality constraint.

Suppose that we consider the context of the algorithmic marketing approach. The budget allocation problem can be explained as follows. In a marketing strategy, one of the crucial choices is deciding how much of a given budget to spend on different media, including television, websites, newspapers, and social media, to reach as many potential customers as possible. In other words, given a bipartite graph $G(V; E)$, where V is a bipartition $(V_1; V_2)$ of the vertex set, V_1 denotes the set of source nodes (such as ad sources), V_2 denotes the set of target nodes (such as people/customers), and $E \subseteq V_1 \times V_2$ is the edge set. Each source node v_1 has a capacity $B_{v_1} \in \mathbb{Z}_+$, which represents the number of available budgets of the ad source corresponding to v_1 . Each edge $v_1 v_2 \in E$ is associated with a probability $p(v_1 v_2) \in [0; 1]$, which means that putting an advertisement to a slot of v_1 activated customer v_2 with probability $p(v_1 v_2)$. Each source node v_1 will be allocated a budget $\mathbf{x}(v_1) \in \{0, 1, \dots, B_{v_1}\}$ such that $\sum_{v_1 \in V_1} \mathbf{x}(v_1) \leq k$ where $k \in \mathbb{Z}_+$ denotes a *total budget capacity*. The object value function f , which means the expected number of target vertices activated by \mathbf{x} , is defined as follows [3].

$$f : \mathbb{Z}_+^V \rightarrow \mathbb{R}_+ \text{ as } f(\mathbf{x}) = \sum_{v_2 \in V_2} \left(1 - \prod_{v_1 v_2 \in E} (1 - p(v_1 v_2))^{\mathbf{x}(v_1)}\right) \quad (46)$$

All experiments are carried out to compare the performance of StrDRS1, StrDRS2, CaDRS and SieveStr + +. We evaluated the performance of each algorithm based on the number of oracle queries, runtime, and influence $f(\mathbf{x})$.

5.1. Experimental Setting

Datasets. For the exhaustive experiment, we choose two datasets of different sizes regarding the number of nodes and edges. They are two real networks that are *bipartite, undirected type and weighted* of the KONECT (<http://konect.cc>) (accessed on 1 September 2022) project [40]: the network of the *FilmTrust ratings* project, and the *NIPS* is a doc-word dataset of NIPS full papers. The weighted of the rating datasets is the rating value, and one of the doc-word datasets is the number of occurrences of the word in the document. The description of the datasets is presented in Table 3.

Table 3. Statistics of datasets. All datasets have the type of bipartite and undirected.

Dataset	#Nodes	#Edges	Node Meaning ($n_1; n_2$)	Edge Meaning
FilmTrust	3579	35,494	(user, film) (1508;2071)	rating
NIPS	13,875	1,932,365	(doc, word) (1500;12,375)	occurrence

Environment. We conducted our experiments on a Linux machine with Intel Xeon Gold 6154 (720) @ 3.700 GHz CPUs and 3TB RAM. Our implementation is written in Python.

Parameter Setting. We set the parameters as follows: $\epsilon = 0.1$, $B = 5$ for all experiments. Because FilmTrust has a small set of nodes, $k \in \{60, 70, 80, 90, 100\}$. Meanwhile, NIPS has a large set of nodes and edges, so $k \in \{120, 140, 160, 180, 200\}$. Besides, we do a simple preprocessing for the edge-weighted of the datasets, which refers to the probability $p(v_1v_2)$. For FilmTrust, edge weighted is the ratio of the rated value and the maximum rated value ($\frac{\text{rated value}}{\text{maximum rated value}}$). While, it is the ratio of the number of the word's occurrences in the document and the number of words in the document ($\frac{\text{number of the word's occurrences in the document}}{\text{number of words in the document}}$) for NIPS.

5.2. Experimental Results

This section discusses the experimental results to clarify the benefits and drawbacks of the algorithms through three metrics: *number of oracle queries*, *runtime*, and *influence*. Two outstanding advantages of our algorithms over CaDRS and SieveStr++ are (1) *our algorithms' runtime and the number of oracle queries are faster many times than those of CaDRS and SieveStr++*; (2) *the influence of our algorithms is often smaller than that of SieveStr++ and CaDRS. However, for some datasets, the influence of StrDRS1 and StrDRS2 can be equal to or greater than that of SieveStr++ and CaDRS if we suitably set parameters B and k for the dataset.* Figure 1 clearly shows the results achieved.

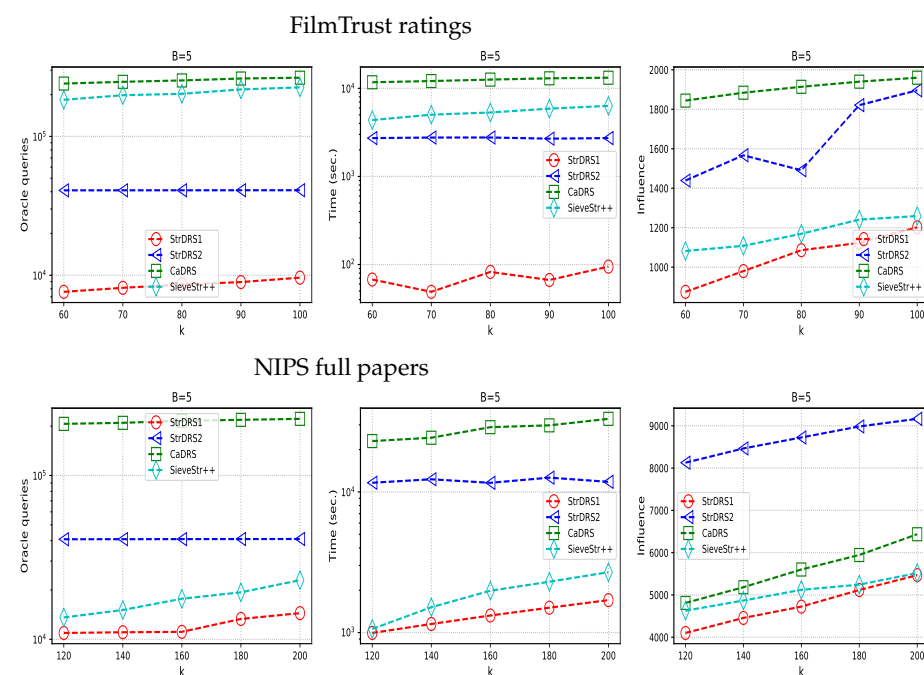


Figure 1. The results of the experimental comparison of algorithms on the datasets.

Oracle queries and Runtime. Because most of the execution time of the algorithms is consumed by the number of queries to compute the function f , the runtime is directly proportional to the number of oracle queries. In detail, for comparing StrDRS1 to SieveStr++, the number of oracle queries of StrDRS1 is 1.2 to 24.4 times smaller than SieveStr++; and the runtime of StrDRS1 is 1.1 to 102.5 times faster than SieveStr++. For comparing StrDRS2 to CaDRS, the number of oracle queries of StrDRS2 is 5.1 to 6.5 times smaller than CaDRS; and the runtime of StrDRS2 is 2.0 to 4.8 times faster than CaDRS. Especially, even if k increases many times, the number of queries and runtime of StrDRS2 only increase very small compared to the other algorithms. This cause makes it possible for us to mistake them for constants when looking at the charts. Table 4 clearly shows the variation in the

number of queries.

Table 4. Statistics of the number of queries.

k	StrDRS1	StrDRS2	CaDRS	SieveStr ++
FilmTrust rating				
60	7601	40,895	240,202	183,453
70	8126	40,917	247,335	198,063
80	8582	40,939	252,978	202,716
90	8933	40,957	261,258	217,947
100	9613	40,979	264,730	225,569
NIPS full papers				
120	10,934	40,807	206,599	13,595
140	11,040	40,847	209,606	15,106
160	11,108	40,887	215,624	17,690
180	13,313	40,927	218,639	19,362
200	14,446	40,963	221,672	22,958

Influence. Through the analysis of experimental results, the difference in the influence value of the algorithms is as follows. For the comparison of SieveStr ++ and StrDRS1, the influence of StrDRS1 is 1.1 to 1.2 times smaller than SieveStr ++. For the comparison of CaDRS and StrDRS2, the influence of StrDRS2 is 1.0 to 1.3 times smaller than CaDRS for FilmTrust dataset. However, for NIPS dataset, the influence of StrDRS2 is 1.4 to 1.7 times greater than CaDRS in this parameters set. Generally, because CaDRS uses a greedy technique, the influence of this algorithm is always at its best. As k increases, this value of CaDRS can reach the best values. On the contrary, the remaining three algorithms use streaming techniques, so it is difficult to achieve the same influence as CaDRS's. However, the difference in the influence of streaming and greedy algorithms is not too large. Especially, this gap will decrease as k increases. Thus, the time benefit of our algorithms is a significant strength against this disparity in influence.

For the convenience of the readers, we summarize the experimental results in Table 5.

Table 5. Statistical comparison of experimental results.

StrDRS1 vs. SieveStr ++	Oracle queries	StrDRS1 is 1.2 to 24.4 times smaller than SieveStr ++
	Time	StrDRS1 is 1.1 to 102.5 times faster than SieveStr ++
	Influence	StrDRS1 is 1.1 to 1.2 times smaller than SieveStr ++
StrDRS2 vs. CaDRS	Oracle queries	StrDRS2 is 5.1 to 6.5 times smaller than CaDRS
	Time	StrDRS2 is 2.0 to 4.8 times faster than CaDRS
	Influence	For FilmTrust, StrDRS2 is 1.0 to 1.3 times smaller than CaDRS but StrDRS2 is 1.4 to 1.7 times greater than CaDRS for NIPS.

6. Conclusions and Future Work

This paper studies the maximization of monotone DR-submodular functions with a cardinality constraint on the integer lattice. We propose two streaming algorithms that have determined approximation ratios and significantly reduce query and time complexity compared to state-of-the-art algorithms. We conducted some experiments to evaluate the efficiency of our algorithms and novel algorithms for this problem. The results indicate that our algorithms are highly scalable and outperform the compared algorithms in terms of both runtime and number of queries, and the influence is slightly smaller.

For our future work, one direction is to study the monotone DR-submodular function maximization problem under a polymatroid constraint and knapsack constraint. In another

direction, we consider the maximization of the non-monotone DR-submodular function under a cardinality constraint.

Author Contributions: Conceptualization, B.-N.T.N. and V.S.; formal analysis, B.-N.T.N.; investigation, B.-N.T.N. and P.N.H.P.; methodology, B.-N.T.N., P.N.H.P. and V.-V.L.; project administration, B.-N.T.N.; resources, B.-N.T.N.; software, B.-N.T.N. and P.N.H.P.; supervision, V.S.; validation, V.S.; writing—original draft, B.-N.T.N.; Writing—review and editing, B.-N.T.N., P.N.H.P., V.-V.L. and V.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by Ho Chi Minh City University of Food Industry (HUFI), Ton Duc Thang University (TDTU), and VŠB-Technical University of Ostrava (VŠB-TUO).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: All real-world network datasets used in the experiment can be downloaded at <http://konect.cc/> (accessed on 1 September 2022).

Acknowledgments: The authors would like to give thanks for the support of Ho Chi Minh City University of Food Industry (HUFI), Ton Duc Thang University (TDTU), and VŠB-Technical University of Ostrava (VŠB-TUO).

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

References

- Tohidi, E.; Amiri, R.; Coutino, M.; Gesbert, D.; Leus, G.; Karbasi, A. Submodularity in action: From machine learning to signal processing applications. *IEEE Signal Process. Mag.* **2020**, *37*, 120–133. [\[CrossRef\]](#)
- Krause, A.; Guestrin, C.; Gupta, A.; Kleinberg, J. Near-optimal sensor placements: Maximizing information while minimizing communication cost. In Proceedings of the 5th International Conference on Information Processing in Sensor Networks, Nashville, TN, USA, 19–21 April 2006; pp. 2–10.
- Soma, T.; Yoshida, Y. A generalization of submodular cover via the diminishing return property on the integer lattice. In Proceedings of the Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems, Montreal, QC, Canada, 7–12 December 2015; pp. 847–855.
- Cornuejols, G.; Fisher, M.; Nemhauser, G.L. On the uncapacitated location problem. *Ann. Discret. Math.* **1977**, *1*, 163–177.
- Nguyen, B.-N.T.; Pham, P.N.; Tran, L.H.; Pham, C.V.; Snášel, V. Fairness budget distribution for influence maximization in online social networks. In Proceedings of the International Conference on Artificial Intelligence and Big Data in Digital Era, Ho Chi Minh City, Vietnam, 18–19 December 2021; pp. 225–237.
- Pham, C.V.; Thai, M.T.; Ha, D.; Ngo, D.Q.; Hoang, H.X. Time-critical viral marketing strategy with the competition on online social networks. In Proceedings of the International Conference on Computational Social Networks, Ho Chi Minh City, Vietnam, 2–4 August 2016; pp. 111–122.
- Lin, H.; Bilmes, J. A class of submodular functions for document summarization. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Portland, OR, USA, 19–24 June 2011; pp. 510–520.
- Agrawal, R.; Squires, C.; Yang, K.; Shanmugam, K.; Uhler, C. Abcd-strategy: Budgeted experimental design for targeted causal structure discovery. In Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics, PMLR, Naha, Japan, 16–18 April 2019; pp. 3400–3409.
- Das, A.; Kempe, D. Submodular meets spectral: Greedy algorithms for subset selection, sparse approximation and dictionary selection. In Proceedings of the 28th International Conference on Machine Learning, ICML, Bellevue, WA, USA, 28 June–2 July 2011; pp. 1057–1064.
- Liu, S. A review for submodular optimization on machine scheduling problems. *Complex. Approx.* **2020**, *12000*, 252–267.
- Soma, T.; Yoshida, Y. Maximizing monotone submodular functions over the integer lattice. *Math. Program.* **2018**, *172*, 539–563. [\[CrossRef\]](#)
- Alon, N.; Gamzu, I.; Tennenholtz, M. Optimizing budget allocation among channels and influencers. In Proceedings of the 21st International Conference on World Wide Web, Lyon, France, 16–20 April 2012; pp. 381–388.
- Soma, T.; Kakimura, N.; Inaba, K.; Kawarabayashi, K.-I. Optimal budget allocation: Theoretical guarantee and efficient algorithm. In Proceedings of the International Conference on Machine Learning, PMLR, Beijing, China, 21–26 June 2014; pp. 351–359.
- Kapralov, M.; Post, I.; Vondrák, J. Online submodular welfare maximization: Greedy is optimal. In Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SIAM, New Orleans, LA, USA, 6–8 January 2013; pp. 1216–1225.

15. Demaine, E.D.; Hajiaghayi, M.; Mahini, H.; Malec, D.L.; Raghavan, S.; Sawant, A.; Zadimoghaddam, M. How to influence people with partial incentives. In Proceedings of the 23rd International Conference on World Wide Web, Seoul, Korea, 7–11 April 2014; pp. 937–948.
16. Bian, A.; Buhmann, J.; Krause, A.; Tschischek, S. Guarantees for greedy maximization of non-submodular functions with applications. In Proceedings of the 34th International Conference on Machine Learning, PMLR, Sydney, Australia, 6–11 August 2017; pp. 498–507.
17. Feldman, M.; Naor, J.; Schwartz, R. A unified continuous greedy algorithm for submodular maximization. In Proceedings of the 2011 IEEE 52nd Annual Symposium on Foundations of Computer Science, Palm Springs, CA, USA, 22–25 October 2011; pp. 570–579.
18. Korula, N.; Mirrokni, V.; Zadimoghaddam, M. Online submodular welfare maximization: Greedy beats $1/2$ in random order. In Proceedings of the Forty-Seventh Annual ACM Symposium on Theory of Computing, Portland, OR, USA, 14–17 June 2015; pp. 889–898.
19. Ha, D.T.; Pham, C.V.; Hoang, H.X. Submodular Maximization Subject to a Knapsack Constraint Under Noise Models. *Asia-Pac. J. Oper. Res.* **2022**, 2250013. [\[CrossRef\]](#)
20. Huang, C.; Kakimura, N. Multi-pass streaming algorithms for monotone submodular function maximization. *Theory Comput. Syst.* **2022**, 66, 354–394. [\[CrossRef\]](#)
21. Chekuri, C.; Gupta, S.; Quanrud, K. Streaming algorithms for submodular function maximization. *Int. Colloq. Autom. Lang. Program.* **2015**, 9134, 318–330.
22. Buschjäger, S.; Honysz, P.; Pfahler, L.; Morik, K. Very fast streaming submodular function maximization. In Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Bilbao, Spain, 13–17 September 2021; pp. 151–166.
23. Pham, C.; Pham, D.; Bui, B.; Nguyen, A. Minimum budget for misinformation detection in online social networks with provable guarantees. *Optim. Lett.* **2022**, 16, 515–544. [\[CrossRef\]](#)
24. Gu, S.; Shi, G.; Wu, W.; Lu, C. A fast double greedy algorithm for non-monotone dr-submodular function maximization. *Discret. Math. Algorithms Appl.* **2020**, 12, 2050007. [\[CrossRef\]](#)
25. Mitrovic, S.; Bogunovic, I.; Norouzi-Fard, A.; Tarnawski, J.; Cevher, V. Streaming robust submodular maximization: A partitioned thresholding approach. In Proceedings of the Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; pp. 4557–4566.
26. Pham, C.; Vu, Q.; Ha, D.; Nguyen, T.; Le, N. Maximizing k-submodular functions under budget constraint: Applications and streaming algorithms. *J. Comb. Optim.* **2022**, 44, 723–751. [\[CrossRef\]](#)
27. Nguyen, B.; Pham, P.; Pham, C.; Su, A.; Snášel, V. Streaming Algorithm for Submodular Cover Problem Under Noise. In Proceedings of the 2021 RIVF International Conference on Computing and Communication Technologies (RIVF), Hanoi, Vietnam, 19–21 August 2021; pp. 1–6.
28. Gong, S.; Nong, Q.; Bao, S.; Fang, Q.; Du, D.-Z. A fast and deterministic algorithm for knapsack-constrained monotone dr-submodular maximization over an integer lattice. *J. Glob. Optim.* **2022**, 1–24. [\[CrossRef\]](#)
29. Liu, B.; Chen, Z.; Du, H.W. Streaming algorithms for maximizing dr-submodular functions with d-knapsack constraints. In Proceedings of the Algorithmic Aspects in Information and Management—15th International Conference, AAIM, Virtual Event, 20–22 December 2021; Volume 13153, pp. 159–169.
30. Tan, J.; Zhang, D.; Zhang, H.; Zhang, Z. One-pass streaming algorithm for dr-submodular maximization with a knapsack constraint over the integer lattice. *Comput. Electr. Eng.* **2022**, 99, 107766. [\[CrossRef\]](#)
31. Zhang, Z.; Guo, L.; Wang, Y.; Xu, D.; Zhang, D. Streaming algorithms for maximizing monotone dr-submodular functions with a cardinality constraint on the integer lattice. *Asia Pac. J. Oper. Res.* **2021**, 38, 2140004:1–2140004:14. [\[CrossRef\]](#)
32. Kazemi, E.; Mitrovic, M.; Zadimoghaddam, M.; Lattanzi, S.; Karbasi, A. Submodular streaming in all its glory: Tight approximation, minimum memory and low adaptive complexity. In Proceedings of the 36th International Conference on Machine Learning, PMLR, Long Beach, CA, USA, 9–15 June 2019; Volume 97, pp. 3311–3320.
33. Nemhauser, G.L.; Wolsey, L.A.; Fisher, M.L. An analysis of approximations for maximizing submodular set functions—I. *Math. Program.* **1978**, 14, 265–294. [\[CrossRef\]](#)
34. Sviridenko, M. A note on maximizing a submodular set function subject to a knapsack constraint. *Oper. Res. Lett.* **2004**, 32, 41–43. [\[CrossRef\]](#)
35. Călinescu, G.; Chekuri, C.; Păxăl, M.; Vondrák, J. Maximizing a monotone submodular function subject to a matroid constraint. *SIAM J. Comput.* **2011**, 40, 1740–1766. [\[CrossRef\]](#)
36. Badanidiyuru, A.; Vondrák, J. Fast algorithms for maximizing submodular functions. In Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA, SIAM, Portland, OR, USA, 5–7 January 2014; pp. 1497–1514.
37. Soma, T.; Yoshida, Y. Non-Monotone DR-Submodular Function Maximization. In Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, San Francisco, CA, USA, 4–9 February 2017; pp. 898–904.
38. Badanidiyuru, A.; Mirzasoleiman, B.; Karbasi, A.; Krause, A. Streaming submodular maximization: Massive data summarization on the fly. In Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD’14, Association for Computing Machinery, New York, NY, USA, 24–27 August 2014; pp. 671–680.

-
39. Hatano, D.; Fukunaga, T.; Maehara, T.; Kawarabayashi, K. Lagrangian decomposition algorithm for allocating marketing channels. In Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, Austin, TX, USA, 25–30 January 2015; pp. 1144–1150.
 40. Kunegis, J. KONECT: The koblenz network collection. In Proceedings of the 22nd International World Wide Web Conference, WWW'13, ACM, Rio de Janeiro, Brazil, 13–17 May 2013; pp. 1343–1350.