VSB TECHNICAL UNIVERSITY OF OSTRAVA | FACULTY OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

Approximation Algorithms For Submodular Optimization And Applications

Aproximační Algoritmy Pro Submodulární Optimalizaci A Aplikace

Thi Bich Ngan Nguyen

PhD Thesis

Supervisor: prof. RNDr. Václav Snášel, CSc.

Ostrava, 2022

Thesis Assignment

This is a sample assignment for a bachelor's thesis, master's thesis or dissertation. You can download the genuine assignment from the Edison system.

Here is the end of a very long assignment over two pages.

Abstrakt a přínos práce

No Czech or Slovak abstract is given

Klíčová slova

No Czech or Slovak keywords are given

Abstract and Contributions

This study proposes approximation algorithms by using several strategies such as streaming, improved-greedy, stop-and-stare, and reverse influence sampling (RIS) to solve three variants of the submodular optimization problem, and perform experiments of these algorithms on the well-known application problems of submodular optimization such as Influence Threshold (IT) and Influence Maximization (IM). Specifically, in the first problem, we propose the two single-pass streaming algorithms (Str-SCN-A and Str-SCN-M) for minimizing the cost of the submodular cover problem under the multiplicative and additive noise models. Str-SCN-A and Str-SCN-M provide bicriteria approximation solutions. These algorithms effectively increase performance computing the objective function, reduce complexity, and apply to big data. For the second problem, we focus on maximizing a submodular function on fairness constraints. This problem is also known as the problem of fairness budget distribution for influence maximization. We design three algorithms (FBIM1, FBIM2, and FBIM3) by combining several strategies such as the threshold greedy algorithm, dynamic stop-and-stare technique, generating samplings by reverse influence sampling framework, and seeds selection to ensure max coverage. FBIM1, FBIM2, and FBIM3 perform effectively on big data, provide $(1/2 - \epsilon)$-approximation to the optimum solutions, and require complexities of the comparison algorithms. Finally, we devise two effective streaming algorithm (StrDRS1 and StrDRS2) to maximize the Diminishing Returns submodular (DR-submodular) function with a cardinality constraint on the integer lattice for the third problem. StrDRS1 and StrDRS2 provide $(1/2 - \epsilon)$-approximation ratio and $(1 - 1/e - \epsilon)$-approximation ratio, respectively. Simultaneously, compared with the state-of-the-art, these two algorithms have reduced complexity, superior runtime performance, and negligible difference in objective function values. In each problem, we further investigate the performance of our proposed algorithms by conducting many experiments. The experimental results have indicated that our approximation algorithms provide high-efficiency solutions, outperform the-state-of-art algorithms in complexity, runtime, and satisfy the specified constraints. Some of the results have been confirmed through five publications at the Scopus international conferences (RIVF 2021, ICABDE 2021) and the SCIE journals (Computer Standards & Interfaces (Elsevier) and Mathematics (MDPI)).

Keywords

Approximation Algorithm; DR-Submodular function; Fairness Constraint; Integer Lattice; Submodular Optimization; Submodular Cover; Noises; Submodular Function; Submodular Maximization; Streaming Algorithm; Greedy Algorithm; Threshold Greedy Algorithm.

Acknowledgement

First and foremost, I would like to express my thanks to my supervisor, prof. RNDr. Václav Snášel, CSc. for his invaluable advice and continuous support; his immense knowledge and plentiful experience have encouraged me in my academic research life, especially the learning process to complete this Ph.D. program.

Secondly, I would like to express my thanks to doc. Ing. Pavel Krömer, Ph.D.; prof. Ing. Jan Platoš, Ph.D; prof. Ing. Ivan Zelinka, Ph.D.; prof. Ing. Michal Krátký, Ph.D.; doc. Mgr. Jiří Dvorský, Ph.D.; and doc. Mgr. Miloš Kudělka, Ph.D. of FEI VŠB-TUO, and the professors are the opponents and committees for reviewing my study. The professors have been kind and willing to comment, suggest and discuss research ideas. Those comments encouraged me to finish some of the most rewarding results of this dissertation.

Thirdly, I would also like to express my thanks to Canh Van Pham, Ph.D. (ORLab, Faculty of Computer Science, Phenikaa University, Hanoi, Vietnam) for enthusiastic support to me during my studies and for offering valuable remarks, feedback, and insights regarding my research.

Personally, I thank God for giving me this opportunity to study what I love and have wonderful experiences. I would like to express my gratitude to my husband (the most wonderful husband), my parent, my parent-in-law, friends, colleagues, and relatives for their support and encouragement because this work would not have happened without them.

Finally, I want to thank VŠB - Technical University of Ostrava for giving me a wonderful place to study and improve my skills during my work. I also want to thank Ho Chi Minh City University of Food Industry (HUFI) and European Cooperation Center, Ton Duc Thang University for their support and help throughout my studies.

Contents

List of symbols and abbreviations

CaDRS	–	Cadinality constraint/Diminishing Returns Submodular
CELF/CELF++	–	Cost-Effective Lazy Forward selection
DR-submodular	–	Diminishing Returns submodular
D-SSA	–	Dynamic Stop-and-Stare Algorithm
FBIM	–	Fairness Budget Influence Maximization
FBIM1	–	Fairness Budget Influence Maximization 1
FBIM2	–	Fairness Budget Influence Maximization 2
FBIM3	–	Fairness Budget Influence Maximization 3
FMC	–	Fairness-Max-Coverage
FIM	–	Fair Infuence Maximization
FSM	–	Fair Submodular Maximization
KONECT	–	Koblenz Network Collection
IC	–	Independent Cascade model
IMM	–	Influence Maximization via Martingales
IM	–	Influence Maximization
IT	–	Influence Threshold model
LT	–	Linear Threshold model
MDRSCa	–	Maximization of monotone Diminishing Returns Submodular function under Cardinality constraint on the integer lattice
NIPS	–	Neural Information Processing Systems
NP-hard	–	Non-deterministic polynomial-time hard
OPT	–	Optimal (solution)
OPIM	–	Online Processing Influence Maximization
OPIMC	–	OPIM-based method for conventional influence maximization
OSNs	–	Online social networks
P-hard	–	Polynomial-time hard
RIS	–	Reverse Influence Sampling
RR	–	Reachable Reverse

SC	–	Submodular Cover
SCN	–	Submodular Cover under Noise
SCSC	–	Submodular Cost Submodular Cover
SieveStr++	–	Sieve Streaming ++
SMM	–	Submodular Maximization under a Matroid constraint
SNAP	–	Stanford Network Analysis Project
SSA	–	Stop-and-Stare Algorithm
Str-SCN-A	–	Streaming algorithm for SCN under additive noise
Str-SCN-M	–	Streaming algorithm for SCN under multiplicative noise
StrDRS1	–	Streaming Diminishing Returns Submodular 1
StrDRS2	–	Streaming Diminishing Returns Submodular 2
StrOpt	–	Streaming Optimal
TIM/TIM+	–	Two-phase Influence Maximization
Str-SCN-A	–	Streaming algorithm for SCN under additive noise
Str-SCN-A	–	Streaming algorithm for SCN under additive noise
UBLF	–	Upper Bound based Lazy Forward

List of Figures

List of Tables

Chapter 1

Introduction

Studying combinatorial optimization problems with submodular objective functions has recently received much attention from research communities. It is motivated by the principles of economies of scale, prevalent in real-world applications. At the same time, the submodular function is also widely utilized in optimization problems of machine learning and algorithmic game theory. Submodularity is a fundamental phenomenon of combinatorial optimization. Submodularity is not an artificial mathematical concept, but it naturally arises in many applications, either as a structural characteristic of combinatorial problems or as a presumption on particular valuation functions [1]. Submodular functions can be utilized in a variety of combinatorial scenarios, including: economic and social problems such as viral marketing [2, 3], welfare maximization [4, 5], leader-selection problem in multi-agent systems [6], monitor placement [7], maximum cut [8]; and problems in computer science such as machine learning [9, 10, 11], data mining [11, 12], data summarization [13, 14], social network analysis [15, 16, 17, 18], game theory [19, 20], and many more.

In mathematics, a submodular function (also known as a submodular set function) is a set function whose value has the following feature: *as the size of the input set rises, the difference in the incremental value of the function that a single element makes when added to the set decreases* [21]. From a theoretical viewpoint, submodular function and submodular optimization are crucial in graph theory, combinatorics, and combinatorial optimization. Inspired by the usefulness and importance of submodular functions, this dissertation studies some approximation algorithms for submodular optimization and their applications.

1.1 Motivation

As mentioned above, the submodular function is crucial in optimization problems. In a submodular optimization problem, *the objective is to select a set that either maximizes or minimizes a submodular function dependent on specific constraints of the problem on the* *allowed sets.* The submodular optimization problems have many variants when they are placed in different contexts. According to Iyer *et al.* [11], most of these problems fall into the following two main categories: *minimizing cost submodular cover* and *maximizing submodular function under a budget constraint.*

Although there exists a vast literature for submodular optimization problems, in different contexts and with the expansion of data, these problems still face some challenges as follows.

1. **How to compute near-exactly the submodular objective function for extensive data under noises?**

2. **How to find good (near-optimal) solutions for these problems under certain constraints (such as cardinality, time, fairness, matroid, and more) when placing them in different contexts?**

3. **How to design viable and efficient algorithms with reasonable computational costs (linear time) for these problems whose data is becoming more extensive or diverse properties (such as set, multiset)?**

Motivated by the above challenges, we study and propose efficient approximation algorithms for some variants of the submodular optimization problem. The main goal of this thesis briefly introduces our proposed algorithms, and their detailed description is presented in the rest of the thesis.

1.2 Main Goals and Contributions

The main research goals of this doctoral thesis are to **study and propose efficient approximation algorithms for the following three submodular optimization problems** and **conduct experiments to investigate the performance and efficient of the proposed algorithms**. The results obtained contribute to solving the challenges mentioned above. The main contributions of the thesis can be summarized as follows.

- **Problem 1. Minimizing cost submodular cover under noises**.
  - Study the problem of minimizing a submodular function subject to a submodular lower bound constraint (also known as the submodular cover [11]) under the multiplicative and additive noise models.
  - Propose two single-pass streaming algorithms (Streaming algorithm under multiplicative noise (Str-SCN-M) and Streaming algorithm under additive noise (Str-SCN-A)) to solve this problem.

  - Conduct experiments on these algorithms to solve the Influence Threshold (IT) problem and evaluate their effectiveness. IT is an instance of the submodular cover problem [22]. The experimental results indicate that Str-SCN-M and Str-SCN-A not only provide the solution with a high value of the objective function, but also outperform the state-of-the-art algorithm in terms of both the number of queries and the running time.
  - Publish two papers, one in the RIVF2021 conference (Scopus) [Publication 1] and one in the Computer Standards & Interfaces journal (Elsevier, SCIE) [Publication 8].

- **Problem 2. Fairness budget distribution for Influence maximization.**
  - Study the Influence Maximization (IM) under budget threshold constraints, which set an upper and lower bounded budgets to choose seeds in each input community to guarantee the fairness constraint (FBIM problem in short). IM is an instance of the submodular function maximization problem [23].
  - Propose three algorithms (called FBIM1, FBIM2, and FBIM3), which use a combination of many methods, including selecting the seed set so that ensuring the fairness constraint, a threshold greedy algorithm, generating sampling by the Reverse Influence Sampling (RIS) framework [24] with the dynamic stop-and-stare algorithm [2], and the online processing influence maximization method [25].
  - Conduct experiments for these algorithms on diverse datasets. Compared to state-of-the-art methods, our approximation algorithms achieve $(1/2 - \epsilon)$ and $(1/2 - 2\epsilon)$ optimal solutions, guarantee a high ratio for the fairness constraint, and work efficiently in big data.
  - Publish two papers, one at the ICABDE2021 conference (Scopus) [Publication 2] and one at the Mathematics journal (MDPI, SCIE) [Publication 9].

- **Problem 3. Maximizing DR-submodular function on the integer lattice**
  - Study the problem of maximizing a monotone diminishing return submodular (DR-submodular) function under cardinality constraint on the integer lattice. It is called MDRSCa problem shortly.
  - Propose two streaming algorithms for solving the MDRSCa problem, including an one-pass streaming algorithm and a multi-pass streaming algorithm (called StrDRS1 and StrDRS2, respectively).
  - Conduct experiments on these algorithms. The results indicate that our approximation algorithms provide solutions with a theoretically guaranteed value of the objective function (achieving $(1/2 - \epsilon)$ and $(1 - 1/e - \epsilon)$ optimal solutions) and

outperform the state-of-the-art algorithms in both the number of queries and the runtime.

– Publish one paper in the Mathematics journal (MDPI, SCIE) [Publication 10].

1.3 Outline of the Thesis

The rest of the thesis is organized into the following sections. Chapter 2 introduces an overview of the submodular function, two categories of optimization problems of submodular functions (minimizing submodular cover and maximizing a submodular function under a certain constraint), and some definitions related to the problems studied in this thesis. Chapters 3, 4, and 5 present Problems 1,2, and 3, respectively. Each chapter includes the definitions related to its problem, the related work, the proposed algorithms, the experiment, and the evaluation of the result. Chapter 7 summarizes the thesis and suggests future research directions. Finally, it is the list of our publication activities.

Chapter 2

Background

This section introduces the principles of submodular function, two main problem types of the submodular function optimization, and the instances of the influence propagation problem, which is an application of submodular function optimization problem. Table 2.1 summarizes the usually used notations in Chapter 2.

2.1 Submodular function optimization

2.1.1 Submodularity

Submodularity has long been known in economic problems but under other names, such as *decreasing marginal values*, *diminishing returns*, etc. Over the years, submodularity has re-emerged strongly in the domain of combinatorial optimization [1]. Submodularity is a property of set functions, i.e., a submodular function $f : 2^V \to \mathbb{R}$ determined on a finite set of items V. V is commonly called the *ground set.* The natural explanation of $f(S)$ is the value of a subset of items S. As we previously said, the economic interpretation of the submodularity feature states that as the size of the set we own increases, the value of additional items decreases. Formally, the definition of a submodular function is as follows [21].

Definition 1 (Submodular function) *A set function $f : 2^V \to \mathbb{R}$, where V is a finite ground set, is said to be submodular if and only if for all subsets $A, B \subseteq V$, it holds that*

$$f(A \cup B) + f(A \cap B) \leq f(A) + f(B) \tag{2.1}$$

By Definition 1, "uncrossing" two sets reduces their total function value. Besides, if we founded on the diminishing return property of a submodular function, the submodular function has an alternative definition as Definition 2 [21].

Table 2.1: Table of the usually used notations in Chapter 2.

Notation	**Description**
G	a graph.
n	the number of nodes in the graph G.
V	the set of nodes in the graph G. $\|V\| = n$.
e	an arbitrary node/element in V
$e_1, \ldots, e_n$	all nodes in V
2^V	the subset family of V.
E	the set of edges in the graph G.
m	the number of edges in the graph. $\|E\| = m$.
A, B	the subsets of V
S	a seed set S
f, h	the submodular functions
$f(S)$	the number of influenced nodes/users by the seed set S after the information diffusion process is done.
T	the threshold of $f(S)$
S_i	S at the i-th step where $i = 0, 1, 2, ...$
v	an arbitrary node in V.
u	a neighbor node of v in V.
w	the set of weighted values of all edges in the graph G.
$p(u, v)$	the weighted value of the edge (u, v)
k	an upper bounded of $\|S\|$.
g	a sample graph in G.
r	the number of sample graph generated from the original graph $G = (V, E)$.
$G_i = (V, E_i)$	the i-th sample graph generated from the original graph $G = (V, E)$, where $i = 1, 2, ..., r$
a, b, c, d, e	nodes in the graph G.
Λ_v	the threshold Λ_v of node v, which is the weighted proportion of v's neighbors that must become active to influence v becoming active.
α	a cover parameter
β	a budget

Definition 2 (Submodular function) *A set function $f : 2^V \to \mathbb{R}$, where V is a finite ground set, is said to be submodular if and only if for all subsets $A \subseteq B \subseteq V$, and $e \in V \setminus B$,*

$$\underbrace{f(A \cup \{e\}) - f(A)}_{\text{Gain of adding an element } e \text{ to a small solution}} \geq \underbrace{f(B \cup \{e\}) - f(B)}_{\text{Gain of adding an element } e \text{ to a large solution}} \tag{2.2}$$

By Definition 2, the marginal value of an additional element exhibits *diminishing marginal returns*. This means that the incremental value, cost, or gain of e decreases (diminishes) as the context in which e is considered grows from A to B.

Moreover, a submodular function is a set function, so it also has other possibly useful properties that a set function may have. Given a submodular set function $f : 2^V \to \mathbb{R}$, f has the following properties [21].

- *Normalized*: $f(\emptyset) = 0$.
- *Non-negative*: $f(S) \geq 0$ for all $S \subseteq V$.
- *Monotone*: if $A \subseteq B \subseteq V$, then $f(A) \leq f(B)$.
- *Symmetric*: $f(S) = f(V \setminus S)$ for all $S \subseteq V$.

2.1.2 Submodular optimization

Over the years, submodular optimization has attracted extraordinary interest in both theory and practice. Because there are many problems in machine learning, auction theory, and combinatorial optimization that have submodular structures. In practice, these problems often relate to large amounts of data and must be solved in a distributed method [26]. A large body of literature is a testament to the important role of submodular optimization in many fields such as economics [27, 28], social welfare [29], and combinatorial optimization [21, 30]. In computer science, it has recently been identified and utilized in domains that include machine learning [9, 31], viral marketing [23], image segmentation [32], feature selection [33], document summarization [34] and speeding up satisfiability solvers [35].

As mentioned in the Introduction, the submodular optimization problem has many variations under different constraints when placed in separate contexts. These variations usually fall into two main types: *minimizing cost submodular cover* and *maximizing submodular function under a budget constraint* [11].

Definition 3 (Submodular function optimization) *Given f and h are monotone non-decreasing submodular functions and where α and β refer to cover parameters and budget, respectively. The problem asks to find a subset S, $S \subseteq V$ so that*

- *Minimizing Cost Submodular Cover:* $\arg\min\{h(S)|f(S) \geq \alpha\}$
- *Maximizing submodular function under budget constraint:* $\arg\max\{f(S)|h(S) \leq \beta\}$

The submodular optimization problems that we study in this thesis also belong to these two problems.

2.2 Influence propagation - an application of submodular optimization

The influence propagation problem is one of the important applications of submodular optimization, and information diffusion in online social networks is an instance of influence propagation. Many studies have been published that focus on influence propagation as well as information diffusion, such as blocking misinformation on online social networks [36, 37, 38], viral marketing [2, 25, 39], influence maximization [40, 41, 42, 43, 44], influence threshold in information diffusion [22, 45].

In the experiment progress of this thesis, we have applied our proposed methods to the celebrated problems of information diffusion, submodular cover minimization, and influence maximization on the online social networks. Specifically, we applied our algorithms to two famous problems, *Influence Threshold* and *Influence Maximization*. The Influence Threshold problem is an instance of the submodular cover problem [22], and the Influence Maximization problem is an instance of maximizing the submodular function [23].

Definition 4 (Influence Threshold problem - IT) *Let $G = (V, E)$ be a social network where the vertices V represent users and the edges E represent social connections. Assume that $G_1 = (V, E_1), G_2 = (V, E_2), ..., G_r = (V, E_r)$, where $E_i \subseteq E$, represent r instances of communities of social connections. In each instance, influenced users on the social network start from an initial seed set and then propagate across edges. Denote $f : 2^V \rightarrow \mathbb{R}^+$ as the influence spread function of a seed set S, i.e., the number of users influenced after the information diffusion process. For $S \subseteq V, f(S)$ is the average number of reachable vertices from S over r instances, and f is a monotone and submodular function, given threshold $T \leq f(V)$, the* IT *problem is to find the seed set S with minimal size so that $f(S) \geq T$, i.e., find*

$$S \leftarrow \arg \min_{S \subseteq V, f(S) \geq T} |S| \tag{2.3}$$

The *influence maximization* (IM) problem has received great attention from researchers in the field of network diffusion. Although Kemp *et al.* first introduced this problem under the name influence maximization in 2003 [23], Richardson *et al.* was the first to study the problem in 2002 as the problem of maximizing the profit of an advertiser on a social network. The IM is crucial in a wide variety of applications, including viral marketing [2, 46], social network analysis [23, 47, 48], social problems such as financial inclusion [49], HIV prevention for homeless youth [50], propagation of information for disease spread [51] and more. In the IM problem, nodes and edges of a social network, respectively, represent individuals and connections. The classical definition of IM is the diffusion to reach the maximum number of nodes, while only disseminating the information to a few initial individuals, also called seeds [23]. Formally, IM can be described as follows:

Definition 5 (Influence Maximization problem - IM) *Given a ground set V, which is the set of users in a social network. $S \subseteq V$ and $|S| \leq k$, is the set of key users that need to be identified, and $f(S)$ is the influence function that measures the expected number of users in V, which can be influenced by members in S according to an information diffusion model. The problem asks to find*

$$S \leftarrow \arg \max_{S \subseteq V, |S| \leq k} f(S) \tag{2.4}$$

The IM problem operates on a certain information diffusion. There are two well-known information diffusion that are commonly used in studies of spreading information and influence. These are the *independent cascade* (IC) model and the *linear threshold* (LT) model. These models were formalized by Kemp *et al.* in 2003 [23]. In their study, they also formulated the IM problem as a combinatorial optimization problem. Besides, they proved that this problem is NP-hard by reducing from the NP-complete Set Cover problem and $f(S)$ is a monotone submodular function under both the IC model and LT model. Furthermore, the calculation of influence spread $f(S)$ given a seed set S has been proved to be P-hard [46].

The following are definitions describing operating principles and illustrations of the models IC and LT in Definition 6, Figure 2.1 and Definition 7, Figure 2.2, respectively.

Definition 6 (Independent Cascade model - IC) *Given a directed graph $G = (V, E, w)$, V is the set of nodes, and $|V| = n$, E is the set of edges and $|E| = m$, and w is the set of weighted values of all edges in E. At first, the nodes of the seed set S are active, while all the remaining nodes are inactive. The process spreads according to the following rule. At step t, if node v first becomes active, it has only a chance to activate each neighbor u. The probability of success is $p(v, u)$, $p(v, u) \in w$. If u has multiple active neighbors, their diffusion is sequenced in random order. As soon as v succeeds, u will become active in step $t + 1$. But whether v succeeds or not, it cannot make any further attempts to activate u in subsequent rounds. In this way, the process works until no more activation is feasible.*

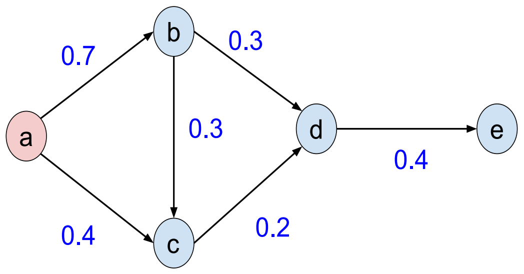


Figure 2.1: An example of Independent Cascade model

Figure 2.1 shows an example of the information propagation process on the IC model. The seed set is $S = a$, and at each edge there is a corresponding influence probability. The process of information transmission takes place as follows:

- At step $t = 0$, $S_0 = S = \{a\}$.
- At step $t = 1$, vertex a activates b and c with the probability of success 0.7 and 0.4. Assume that the vertex b is successfully activated. We have $S_1 = \{a, b\}$.
- At step $t = 2$, similarly, vertex b activates vertices d and c with success probability 0.3 and 0.4. Vertex a is not activated c because it was done the last time. Assume that in this case, c is active. We have $S_2 = \{a, b, c\}$.
- At step $t = 3$, similarly, b and c activate d with probabilities 0.3 and 0.2. In this case, if d is not activated, then the propagation stops.

Definition 7 (Linear Threshold model - LT) *Given a directed graph $G = (V, E, w)$, V is the set of nodes, and $|V| = n$, E is the set of edges and $|E| = m$, and w is the set of weighted values of all edges in E. A node v in G is affected by each of its neighbors u with probability $p(u, v)$, $p(u, v) \in w$, and $\sum_{u \text{ neighbor of } v} p(u, v) \leq 1$. Each node v is assigned a threshold Λ_v from the interval $[0; 1]$ at random. This threshold is the weighted proportion of v's neighbors that must become active to influence v becoming active. The process spreads as follows. At the beginning, we initialize a random set of threshold values and a seed set of active nodes S, while other nodes are inactive. At step t, all nodes that were active in step $t-1$ remain active, and we activate any node v with a total weight of active neighbors greater than or equal to Λ_v.*

$$\sum_{u \text{ active neighbor of } v} p(u, v) \leq 1 \tag{2.5}$$

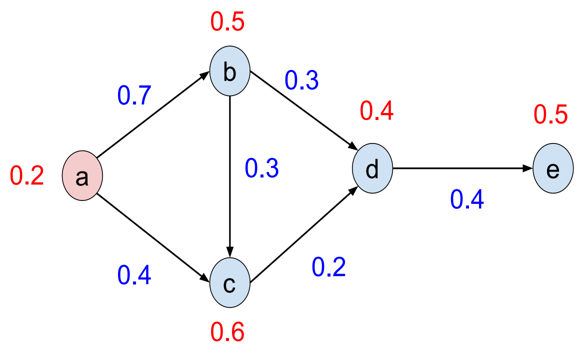


Figure 2.2: An example of Linear Threshold model

Figure 2.2 shows an example of the information propagation process on the LT model. Assume that in the seed set $S = a$, each vertex has activation thresholds and each edge has a corresponding weight. The process of information transmission takes place as follows.

- At step $t = 0$, $S_0 = S = \{a\}$.
- At step $t = 1$, the sum of the weights that affect the vertex b is 0.7, greater than the threshold $\Lambda_b = 0.5$, so b is activated. The vertex c has a total weight of 0.4, less than the threshold $\Lambda_c = 0.6$, so it is not activated. We have $S_1 = \{a, b\}$.
- At step $t = 2$, the vertex c is activated. We have $S_2 = \{a, b, c\}$.
- At step $t = 3$, the vertex d activated, so $S_3 = \{a, b, c, d\}$.
- At step $t = 4$, no vertices are activated, the propagation stops.

2.3 Concluding remarks

This chapter presented important fundamental components, which serve as the theoretical basis for the problems studied in the thesis. Especially they include the concept and properties of submodular functions, types of submodular optimization problem, application problems of submodular optimization in influence propagation, and their propagation models.

Chapter 3

Problem 1. Minimizing cost submodular cover under noises

This chapter describes the problem of minimizing cost submodular cover under multiplicative and additive noise models in detail, including the motivation to study the problem, problem definition, related work, proposed algorithms, experiment and result evaluation. Table 3.1 summarizes the usually used notations in this chapter.

Table 3.1: Table of the usually used notations in the Submodular cover under noise problem.

Notation	**Description**
G	a graph.
n	the number of nodes in the graph.
V	the set of nodes in the graph G. $\|V\| = n$.
2^V	the subset family of V.
E	the set of edges in the graph G.
F	the function calculates the influence of an element e or subset S in the graph.
T	the threshold to calculate F.
S_i	seed set for threshold T.
S	the returned size-k seed set of the Str-SCN-M and Str-SCN-A algorithms.
S^*	an optimal size-k seed set.
OPT	OPT $= \|S^*\|$
v	a random node in V.
u	a neighbor node of v in V.
k	an upper bounded of $\|S\|$.
g	a sample graph in G.
$f(S)$	an optimal influence spread of seed set S.
r	the number of sample graph generated from the original graph $G = (V, E)$.

3.1 Introduction

One of the important problems of the submodular optimization problem is to minimize a submodular function subject to a submodular lower bound constraint, also known as the minimizing cost submodular cover (Submodular Cover problem in short). Formally, the submodular cover problem can be defined as in Definition 8 [52].

Definition 8 (Submodular Cover problem - SC) *Given a ground set V, a monotone and submodular function $f : 2^V \to \mathbb{R}^+$ and the threshold $T \leq f(V)$, the problem asks to find the subset $S \subseteq V$ with minimal cardinality so that $f(S) \geq T$.*

This problem is found in many applications, such as data summarization [52, 53], monitor placement [16, 54], influence threshold in online social networks [55, 56], and active set selection [57]. According to recent studies, existing solutions to the SC problem often assume that the value oracle access to f, meaning that f can be queried at any subset $S \subseteq V$ [54, 58, 59]. Unfortunately, for many rising applications in submodular optimization, computing the function f takes exponential time [22, 55, 60]. Instead of directly accessing f, we only query a noise oracle F of f in polynomial time under noise models. There are two common noise models that are *multiplicative noise* [17, 22, 61], i.e., $(1-\epsilon)f(S) \leq F(S) \leq (1+\epsilon)f(S)$ and *additive noise* [22], i.e., $f(S)-\epsilon \leq F(S) \leq f(S)+\epsilon$. Unfortunately, in many cases, F does not inherit the good properties (monotone or submodular) of f, which can help to approximate SC with a theoretical guarantee [22]. Therefore, it becomes more challenging to approximate SC.

Motivated by this phenomenon, Crawford *et al.* [22] first investigated SC problem under an *additive noise* model and proposed a greedy algorithm with theoretical bounds. However, this algorithm takes $O(n^2)$ query complexity, and in some cases it has not been applicable when the data is large due to the number of n passes over the data.

In reality, some applications' data is often on a large scale, making it difficult to store data on the computer. Therefore, it is critical to devise fast and efficient algorithms that not only provide theoretical guarantees, but can also scan over the data in a single-pass or a few passes. To address this challenge, we propose two efficient streaming algorithms for SC under noise, which need one time to scan over the data and guarantee the theoretical bounds of the solution. Specifically, our contributions are as follows:

- We propose two single-pass streaming algorithms for the Submodular Cover under Noises (SCN) problem, named Str-SCN-M and Str-SCN-A, under multiplicative noise model and additive noise model, respectively. Under the multiplicative noise model, Str-SCN-M provides a $(\frac{1-\epsilon}{1+\epsilon}(1-\frac{1}{\alpha}), \alpha(1+m))$ bicriteria approximation solution. In the additive noise model, Str-SCN-A provides a solution S satisfying $f(S) \geq (1-\frac{1}{\alpha})T - 2\epsilon$ and

$|S| \leq \alpha(1+m)|S^*|$, where $\alpha, m > 0$ are the input parameters and S^* is the optimal solution.

- We further investigate the performance of our algorithms by performing some experiments on the IT problem (in Definition 7). The results indicate that our algorithms provide solutions with a high value of function F and outperform the state-of-the-art algorithm in both the number of queries and the running time.

3.2 Related work

Due to the wide application of the SC problem, many works have focused on approximating the problem, despite being an NP-hard problem [23, 62]. The greedy strategy was first proposed to find an approximate solution for the SC problem with value oracle access to f. Specifically, Wolsey *et al.*[58] proved that if f is an integral value, the approximation ratio of the greedy algorithm is $\ln(\alpha)$, where α is the largest singleton value of f. Otherwise, if f is a real value, the approximation ratio is $1 + \ln(\alpha/\beta)$, where β is the smallest nonzero marginal gain. Later, Wan *et al.* [59] investigated SC by considering the cost function of the set of elements and showed that the greedy algorithm has an approximation ratio of $\gamma \ln(\alpha)$, where γ is the curvature of the cost function. Soma and Yoshida [54] generalized the SC problem on the integer lattice domain. They proposed a decreasing threshold algorithm that has a bicriteria approximation ratio of $(1+3\gamma)\rho(1+\ln(\alpha/\beta))$, where $\gamma < 1$ is an input. In addition, Mirzasoleiman *et al.* [53] proposed a distributed algorithm that ran in $O(\log(|S^*| + \log(T)))$ rounds and provided a solution S with $f(S) \geq T\lambda$ and the size was at most $2(1+\log(T))|S^*|/\lambda$, where λ worked as an input.

Norouzi-Fard*et al.* [57] proposed the single-pass streaming algorithm providing a $(1 - \frac{1}{\ln(1/\epsilon)}, 2\ln(1/\epsilon))$-bicriteria approximation for the SC problem. However, this algorithm cannot be directly applied to the submodular cover under noise (SCN) problem to obtain theoretical bounds for the following reasons: the approximation of this algorithm is obtained by exploiting the submodular property of f but we can only apply this algorithm by calling F (multiplicative/additive noise oracle) instead of f but F does not inherit the properties of f, i.e, F is not submodular and may not be monotone. In addition, the noise can mislead a process of constructing a solution by magnifying the marginal gain of a selection whose contribution may be insignificant. As a result, the approximation guarantees of this algorithm do not hold. To handle the noises, we give appropriate estimates for $f(S \cup e) - f(S)$ by using noise oracle F, where S is the current solution and e is an incoming element. Besides, we provide the upper bound of the optimal solution by finding X such that $F(X)/(1+\epsilon) \geq T$ (or $F(X) \geq T + \epsilon$). This helps our algorithm reduce the number of queries as well as the memory to find and store unnecessary candidate solutions. In addition, our algorithm provides a

$(\frac{1-\epsilon}{1+\epsilon}(1-\frac{1}{\alpha}), \alpha(1+m))$ bi-criteria solution that depends on ϵ, α and m. It makes the efficiency of the algorithm easy to control when these parameters are independent of each other. In contrast, the Greedy algorithm depends only on ϵ, so we must make a trade-off between the solution cost and the objective function value.

Most previous studies assumed that the objective function is noise-free, that is, it is easy to calculate the objective function precisely. However, we must have a noisy evaluation in many practical applications. For example, in the influence threshold problem [55, 56], calculating the influence spread function has been shown to be P-hard [60]. Thus, it is often estimated by simulating the random diffusion process [23] or using the sketch-based method [22, 62]. Another example is the sparse regression problem, when only a set of limited data can be used for evaluation, making the evaluation noisy. In addition, some works have studied the maximizing of the submodular under a noise model and proposed approximation algorithms. Chao *et al.* [61] studied the problem of maximizing a set function characterized by a submodularity ratio γ with two types of noise. Under the multiplicative noise model, they gave a $\frac{1-\epsilon}{1+\epsilon}(1-e^{-\gamma})$ approximation algorithm, and under the additive noise model, they gave a $(1-e^{-\gamma})-2\epsilon$ approximation algorithm via a Pareto method.

Recently, Crawford *et al.* [22] studied Submodular Cost Submodular Cover (SCSC) problem, a generalization version of SC under the additive noise model. SCSC considers a cost function $c : 2^V \to \mathbb{R}_{\geq 0}$, and the problem finds a subset S with minimal cost such that $f(S) \geq T$. In the seminal paper, they proposed the Greedy algorithm that returns a solution S satisfying $f(S) \geq T - \epsilon$ and $c(S) \leq \frac{\rho}{1-\frac{4\epsilon c_{max}\rho}{c_{min}\mu}-\gamma}\left(2+\ln(\frac{n\alpha\rho}{\gamma\mu})\right)c(S^*)$ where S^* is an optimal solution, $\mu > 4\epsilon c_{max}\rho/c_{min}$, $\gamma \in (0, 1-4\epsilon c_{max}\rho/c_{min}\mu)$, with $c_{min} = \min_{e\in S} c(e)$ and $c_{max} = \max_{e\in S} c(e)$. However, the Greedy algorithm requires $|S|$ passes over the input data and has $O(|S|n)$ number of calls for F. Its query complexity tends to $O(n^2)$, so it may not be applicable to big data. Meanwhile, our proposed algorithms do not only provide guaranteed theoretical bounds, but also reduce query complexity to $O(n \log n)$.

3.3 Problem definition

Given the ground set $V = \{e_1, \ldots, e_n\}$ and a utility function $f : 2^V \to \mathbb{R}^+$ that measures the quality of a subset $S \subseteq V$ and f is the submodular function.

For simplicity, we denote $S + e$ as $S \cup \{e\}$ and assume that f is normalized, i.e., $f(\emptyset) = 0$. Assume that the values of the utility function $f(S)$ cannot be obtained while a noise oracle, $F(S)$, is easily computed. In our work, we consider the SC problem in multiplicative and additive noise models (in Definition 11) with ϵ-multiplicative noise oracle and ϵ-additive noise oracle, defined as follows.

Definition 9 ***(ϵ-multiplicative noise oracle)*** *A function $F : 2^V \rightarrow \mathbb{R}^+$ is an ϵ-multiplicative noise oracle of f if for all $S \subseteq V$, we have:*

$$(1-\epsilon)f(S) \leq F(S) \leq (1+\epsilon)f(S) \tag{3.1}$$

Definition 10 (ϵ-additive noise oracle) *A function $F : 2^V \rightarrow \mathbb{R}^+$ is an ϵ-additive noise oracle of f if for all $S \subseteq V$, we have:*

$$f(S) - \epsilon \leq F(S) \leq f(S) + \epsilon \tag{3.2}$$

In this thesis, we study the Submodular Cover under Noise problem, defined as follows.

Definition 11 (Submodular Cover under Noise - SCN) *Given a group set V, a threshold T, an ϵ-multiplicative noise oracle F (ϵ-additive noise oracle) of a monotone and submodular function $f : 2^V \rightarrow \mathbb{R}^+$ under the multiplicative noise (additive noise) model. The problem asks to find the subset $S \subseteq V$ with minimal size such that $f(S) \geq T$.*

Throughout this thesis, we denote S^* as an optimal solution and $\mathsf{OPT} = |S^*|$. We call an algorithm (α, β)-*bicriteria approximation* for the SCN problem if it returns a solution S satisfying $f(S) \geq \alpha \cdot T$ and $|S| \leq \beta \cdot \mathsf{OPT}$, for $\alpha, \beta > 0$.

3.4 Proposed algorithms

This section introduces two streaming algorithms for the SCN problem under multiplicative and additive noise models.

3.4.1 Streaming Algorithm under multiplicative noise

At a high level, the main idea of the algorithm is that (1) *we give the prediction to the value of* OPT*, based on the set of observed elements, and update the prediction to reduce storing memories of the candidate solutions*, and (2) *we compare the incremental value of F for each element with the given threshold to select high value elements.*

3.4.1.1 Algorithm description

The detail of the algorithm is fully presented in Algorithm 1.

The algorithm receives a threshold T, parameters m ($0 < m < 1$), α ($\alpha > 1$), which are related to the quality of the solution as well as the running time of our algorithm. They allow

Algorithm 1: Streaming algorithm under the multiplicative noise for SCN (Str-SCN-M)

```
Input: A noise oracle $F$, a threshold $T$, parameters $m$ $(0 < m < 1), \alpha$ $(\alpha > 1)$
Output: A solution $S$
1:  $l \leftarrow \lceil \log_{1+m}(\alpha n) \rceil$
2:  $S_i \leftarrow \emptyset, \forall i = 1, \ldots, l$
3:  $X \leftarrow \emptyset$
4:  foreach $e \in V$ do
5:      if $\frac{F(X \cup \{e\})}{1+\epsilon} \geq T$ then
6:          Update: $X \leftarrow X \cup \{e\}$, $l \leftarrow \lceil \log_{1+m}(\alpha |X|) \rceil$
7:          Delete $S_i$ for all $i > l$.
8:      else
9:          $X \leftarrow X \cup \{e\}$
10:     for $i = 1$ to $l$ do
11:         if $\frac{F(S_i \cup \{e\})}{1-\epsilon} - \frac{F(S_i)}{1+\epsilon} \geq \frac{T}{(1+m)^i}$ and $|S_i| + 1 \leq (1+m)^i$ then
12:             $S_i \leftarrow S_i + \{e\}$
13: Find $S \leftarrow \arg\min_{S_i \in \{S_1, S_2, \ldots, S_l\}} \{|S_i| : F(S_i) \geq (1-\epsilon) T (1 - \frac{1}{\alpha})\}$ by doing a binary
    search.
14: return $S$.
```

us to adjust the solution quality trade-off with runtime. Specifically, α is to guarantee that the value of the objective function f is near to T while α and m are to guarantee the bound of the size of the returned solution. Besides, α and m will affect the running time of the algorithm. If α is larger and m is smaller, the algorithm gives good solution quality, but the running time increases and vice versa.

The algorithm initiates $l = \lceil \log_{1+m}(\alpha n) \rceil$ candidate solutions $S_i, i = 1, \ldots, l$ and a prediction of the optimal solution X which is initiated as an empty set. For each observed element e, if e passes the condition that

$$f(X) \geq \frac{F(X \cup \{e\})}{1+\epsilon} \geq T$$

the algorithm updates X by adding e. Also, the value of l is updated to $\lceil \log_{1+m}(\alpha |X|) \rceil$ at this time to reduce the number of candidate solutions and the running time of the algorithm. Candidate solutions $S_j, j > l$ will be removed.

The algorithm next updates the candidate solution S_i as follows. If the incoming element e satisfies conditions $|S_i| + 1 \leq (1+m)^i$ and

$$\frac{F(S_i \cup \{e\})}{1-\epsilon} - \frac{F(S_i)}{1+\epsilon} \geq \frac{T}{(1+m)^i}$$

then the algorithm adds e to S_i. Otherwise, it ignores S_i and moves into the next candidate solution S_{i+1}. The value of $\frac{F(S_i \cup \{e\})}{1-\epsilon} - \frac{F(S_i)}{1+\epsilon}$ represents the marginal value of f at e when we add it to S_i.

After scanning one pass over the ground set, the algorithm finds the best solution among the candidates (line 17) by doing a binary search.

3.4.1.2 Theoretical analysis

We first analyze the complexity of the algorithm, stated in Lemma 1

Lemma 1 *Algorithm 1 is a single-pass streaming algorithm that has $O(n \log_{1+m}(n))$ query complexity and takes $O(n \log_{1+m}(n))$ memory complexity.*

Proof The algorithm scans only once over the ground set V. There are at most $\alpha \log_{1+m}(n)$ candidate solutions, so the algorithm takes $O(n \log_{1+m}(n))$ memory complexity. Furthermore, each candidate solution is updated at most n times, so the total number of queries is $O(n \log_{1+m}(n))$. ■

The approximation guaranteed by the algorithm is shown in the following theorem.

Theorem 1 *Algorithm 1 is a $\left(\frac{1-\epsilon}{1+\epsilon}(1 - \frac{1}{\alpha}), \alpha(1+m)\right)$-bicriteria approximation algorithm.*

Proof When the condition in line 5 is satisfied, we have $f(X) \geq \frac{F(X)}{1+\epsilon} \geq T$ which implies $|X| \geq \mathsf{OPT}$ and $\log_{1+m}|X| \geq \log_{1+m}\mathsf{OPT}$. Since $l = \lceil \log_{1+m}(\alpha|X|) \rceil$, therefore there exists a positive number $i \in [l]$ such that:

$$\frac{(1+m)^{i-1}}{\alpha} \leq \mathsf{OPT} \leq \frac{(1+m)^i}{\alpha} \tag{3.3}$$

The algorithm returns one candidate solution S_j for some $j \in [l]$, we consider all possible cases of the final solution as follows:

Case 1. The algorithm returns S_i. Due to the condition of selecting the additional element e in line 12, we always have:

$$S_i \leq (1+m)^i = (1+m)(1+m)^{i-1} \leq (1+m)\alpha\mathsf{OPT} \tag{3.4}$$

To give the bound of $f(S_i)$, we consider the following cases:

Case 1.1. If $|S_i| = (1+m)^i$, denote $S_i = \{s_1, s_2, \ldots, s_{|S_i|}\}$, where s^t is the t-th element added to S_i and $S_i^t = \{s_1, s_2, \dots, s_t\}, t \le |S_i|$. We have:

$$f(S_i) \ge \sum_{j=1}^{|S_i|}(f(S_i^j) - f(S_i^{j-1})) \tag{3.5}$$

$$\ge \sum_{j=1}^{|S_i|}\left(\frac{F(S_i^j)}{1-\epsilon} - \frac{F(S_i^{j-1})}{1+\epsilon}\right) \tag{3.6}$$

$$\ge \frac{T|S_i|}{(1+m)^i} \ge T \tag{3.7}$$

Case 1.2. If $|S_i| < (1+m)^i$. Assume that S^* is an optimal solution, $S' = S^* \setminus S_i = \{s'_1, s'_2, \dots, s'_d\}$, $S'_t = \{s'_1, s'_2, \dots, s'_t\}, t \le d$. Due to $\mathsf{OPT} \le \frac{(1+m)^i}{\alpha}$, we have:

$$f(S^* \cup S_i) - f(S_i) = \sum_{t=1}^{d}(f(S_i \cup S'_t) - f(S_i \cup S'_{t-1})) \tag{3.8}$$

$$\le \sum_{t=1}^{d}(f(S_i \cup s'_t) - f(S_i)) \tag{3.9}$$

$$\le \mathsf{OPT} \cdot \max_{s'_t \in S'}(f(S_i \cup s'_t) - f(S_i)) \tag{3.10}$$

$$\le \mathsf{OPT} \max_{s'_t \in S'}\left(\frac{F(S_i \cup s'_t)}{1-\epsilon} - \frac{F(S_i)}{1+\epsilon}\right) \tag{3.11}$$

$$< \mathsf{OPT}\frac{T}{(1+m)^i} \le \frac{T}{\alpha} \tag{3.12}$$

It implies that $f(S_i) \ge (1 - \frac{1}{\alpha})T$ and $F(S_i) \ge (1-\epsilon)(1-\frac{1}{\alpha})T$.
Case 2. If Algorithm 1 returns a solution $S_{i'}$ with $i' < i$, then we also have:

$$|S_{i'}| \le (1+m)^{i'} < (1+m)^i = (1+m)(1+m)^{i-1} \le (1+m)\alpha\mathsf{OPT} \tag{3.13}$$

Due to the selection of the final solution in line 17, we have: $f(S_{i'}) \ge \frac{F(S_{i'})}{1+\epsilon} \ge \frac{1-\epsilon}{1+\epsilon}\left(1 - \frac{1}{\alpha}\right)T$.
Case 3. If the Algorithm 1 returns a solution S_k with $k > i$, similar to the arguments in Cases 1.1 and 1.2, we also have: $f(S_k) \ge \frac{1-\epsilon}{1+\epsilon}(1-\frac{1}{\alpha})T$. In the other hand, due to the selection of the final solution in line 17, we also have: $|S_k| \le |S_i| \le \alpha(1+m)\mathsf{OPT}$. By combining all cases, we obtain the proof. ■

3.4.2 Streaming Algorithm under additive noise

3.4.2.1 Algorithm description

In this section, we introduce a streaming algorithm for SCN under the adaptive noise model, named Str-SCN-A (Algorithm 2). Str-SCN-A inherits the operating principle of Str-SCN-M with some modifications to be suitable for additive noise. Accordingly, we give a prediction of the optimal solution by adding elements until $F(X \cup \{e\}) - \epsilon \geq T$ (line 5). Str-SCN-A adds a new element e to a candidate solution S_i if $|S_i| + 1 \leq (1+m)^i$ and $F(S_i \cup \{e\}) - F(S_i) + 2\epsilon \geq \frac{T}{(1+m)^i}$. The full details of this algorithm are described in Algorithm 2.

Algorithm 2: Streaming algorithm for SCN under the additive noise (Str-SCN-A)

```
    Input: A oracle $F$, a threshold $T$, parameters $m$ $(0 < m < 1), \alpha$ $(\alpha > 1)$
    Output: Subset $S$
1:  $l \leftarrow \lceil \log_{1+m}(\alpha n) \rceil$
2:  $S_i \leftarrow \emptyset, \forall i = 1, \ldots, l$
3:  $X \leftarrow \emptyset$
4:  foreach $e \in V$ do
5:      if $F(X \cup \{e\}) - \epsilon \geq T$ then
6:          Update: $X \leftarrow X \cup \{e\}, l \leftarrow \lceil \log_{1+m}(\alpha |X|) \rceil$
7:          Delete $S_i$ for all $i > l$.
8:      else
9:          $X \leftarrow X \cup \{e\}$
10:     for $i = 1$ to $l$ do
11:         if $F(S_i \cup \{e\}) - F(S_i) + 2\epsilon \geq \frac{T}{(1+m)^i}$ and $|S_i| + 1 \leq (1+m)^i$ then
12:             $S_i \leftarrow S_i + \{e\}$
13: Find $S \leftarrow \arg\min_{S_i \in \{S_1, S_2, \ldots, S_l\}} \{|S_i| : F(S_i) \geq T(1 - \frac{1}{\alpha}) - \epsilon\}$ by using binary search.
14: return $S$
```

3.4.2.2 Theoretical analysis

We now analyze the performance of the algorithm in Theorem 2.

Theorem 2 *Algorithm 2 is a single-pass streaming algorithm, has $O(n \log_{1+m}(n))$ queries, and returns a solution S satisfying $f(S) \geq (1 - \frac{1}{\alpha})T - 2\epsilon$ and $|S| \leq \alpha(1+m)\mathsf{OPT}$.*

Proof We prove this theorem by using the same argument as the proofs of Lemma 1 and Theorem 1. Since the operations of Algorithm 1 and 2 is similar, the query complexity of Algorithm 2 is $O(n \log_{1+m}(n))$. When the condition in line 5 is satisfied, we also have $f(X) \geq F(X) - \epsilon \geq T$ and $l = \lceil \log_{1+m}(\alpha |X|) \rceil$. Therefore, there exists a positive number $i \in [l]$ such that:

$$\frac{(1+m)^{i-1}}{\alpha} \leq \mathsf{OPT} \leq \frac{(1+m)^i}{\alpha}$$

If the algorithm returns S_i as the final solution, we consider the two following cases:
Case 1. If $|S_i| = (1+m)^i$, denote $S_i = \{s_1, s_2, \ldots, s_{|S_i|}\}$, where s^t is the t-th element added

to S_i and $S_i^t = \{s_1, s_2, \ldots, s_t\}, t \leq |S_i|$. We have the following:

$$\begin{aligned} f(S_i) &\geq \sum_{j=1}^{|S_i|}(f(S_i^j) - f(S_i^{j-1})) && (3.14)\\ &\geq \sum_{j=1}^{|S_i|}((F(S_i^j) - \epsilon) - (F(S_i^{j-1}) + \epsilon)) && (3.15)\\ &= \sum_{j=1}^{|S_i|}(F(S_i^j) - F(S_i^{j-1}) - 2\epsilon) && (3.16)\\ &\geq \frac{T|S_i|}{(1+m)^i} - 2\epsilon \geq T - 2\epsilon && (3.17) \end{aligned}$$

Case 2. If $|S_i| < (1+m)^i$. Assume that S^* is an optimal solution, $S' = S^* \setminus S_i^j = \{s'_1, s'_2, \ldots, s'_d\}$, $S'_t = \{s'_1, s'_2, \ldots, s'_t\}, t \leq d$. Since $\mathsf{OPT} \leq \frac{(1+m)^i}{\alpha}$, we have:

$$\begin{aligned} f(S^* \cup S_i) - f(S_i) &= \sum_{t=1}^{d}((f(S_i \cup S'_t) - f(S_i \cup S'_{t-1})) && (3.18)\\ &\leq \sum_{t=1}^{d}(f(S_i \cup s'_t) - f(S_i)) && (3.19)\\ &\leq \mathsf{OPT} \cdot \max_{s'_t \in S'}(f(S_i \cup s'_t) - f(S_i)) && (3.20)\\ &\leq \mathsf{OPT} \cdot \max_{s'_t \in S'}((F(S_i \cup s'_t) + \epsilon) - (F(S_i) - \epsilon)) && (3.21)\\ &= \mathsf{OPT} \cdot \max_{s'_t \in S'}(F(S_i \cup s'_t) - F(S_i) + 2\epsilon) && (3.22)\\ &\leq \mathsf{OPT}\frac{T}{(1+m)^i} \leq \frac{T}{\alpha} && (3.23) \end{aligned}$$

It implies that $f(S_i^j) \geq (1 - \frac{1}{\alpha})T$ and $F(S_i) \geq (1 - \frac{1}{\alpha})T - \epsilon$. Combining the two above cases, we have $F(S_i) \geq (1 - \frac{1}{\alpha})T - \epsilon$.

In the case when Algorithm 2 returns a solution $S_{i'}$ with $i' < i$, we have

$$|S_{i'}| \leq (1+m)^{i'} < (1+m)^i \leq \alpha(1+m)\mathsf{OPT}$$

Due to the condition in line 13, we have: $f(S_{i'}) \geq F(S_{i'}) - \epsilon \geq \left(1 - \frac{1}{\alpha}\right)T - 2\epsilon$
If Algorithm 2 returns a solution S_k with $k > i$, we also have $f(S_k) \geq (1 - \frac{1}{\alpha})T - 2\epsilon$ and $|S_k| \leq |S_i| \leq \alpha(1+m)\,\mathsf{OPT}$. By combining all the cases, we obtain the proof. ■

3.5 Experiment and Result evaluation

As mentioned above, we conduct some experiments of Problem 1 on the Influence Threshold problem (Definition 4), which is an instance of SC problem.

Since calculating f of the IT problem is P-hard [63], i.e, it cannot be calculated exactly in polynomial time. The value of function f is calculated based on the sample graphs (deterministic graph) $G_1, G_2, \ldots, G_r$ which generated from the original graph $G = (V, E)$ and the number of all sample graphs is exponential. Therefore, we only compute the function F, an ϵ-multiplicative or ϵ-additive noise oracle of f by using the approximate reachability method proposed in [62], which is based only on a given finite number of sample graphs (typically $poly(n)$). Now we present the details of this method. It was also used in the recent work of Crawford *et al.* [22].

Approximate reachability method. In the IT problem, the value of the function f is calculated based on the sample graphs (deterministic graphs) $G_1, G_2, \ldots, G_r$ which generated from the original graph $G = (V, E)$. Cohen *et al.*[62] proposed to compute the approximate average reachability oracle of f by basing on bottom-k min-hash sketches. Given $k \in \mathbb{Z}_{>0}$, F is computed as follows: For every vertex v, instance pair $(v, i) \in V \times \{1, ..., r\}$ and a random rank value $rank_v^i$ is drawn from the uniform distribution on $[0, 1]$. For every vertex $u \in V$, the combined reachability sketch S_u of u is the smallest k value from the set $\{rank_v^i : v$ is reachable from u on instance $G_i\}$. S_u is stored for all $u \in V$.

Let $S \subseteq T$. If $|\cup_{u \in S} S_u| < k$, $F(S) = |\cup_{u \in S} S_u| / r$. Otherwise, let t be the k-th smallest value in $\cup_{u \in S} S_u$, $F(S) = (k-1)/(rt)$. Hence, F can be an ϵ-approximation to f by choosing a sufficiently large k: for $c > 2$, if $k = c\epsilon^{-2} \log(n)$ the relative error of all queries over the duration of three algorithms is within ϵ with probability at least $1 - 1/n^{c-2}$.

In this section, all experiments are conducted to compare the performance of the Str-SCN-M, Str-SCN-A, and the Greedy algorithm [22], the state-of-the-art algorithm for the SCN problem. We evaluated the performance of each algorithm based on five important metrics, such as running time, value of $\frac{F}{T(1+\epsilon)}$ for Str-SCN-M, value of $\frac{F}{T+\epsilon}$ for Str-SCN-A, number of queries F, size of seed set S, and memory usage. Note that $\frac{F}{T(1+\epsilon)}$ and $\frac{F}{T+\epsilon}$ are lower bounds of $\frac{f}{T}$ which are to quantify the closeness between the value of f for the returned solutions and the threshold T.

3.5.1 Experimental Settings

3.5.1.1 Datasets

For the comprehensive experiment purpose, we choose three datasets with different sizes. They are three real social networks from SNAP [64]: the Facebook ego network (ego-Facebook), the ArXiV General Relativity collaboration network (ca-GrQc), and the Gnutella peer-to-peer file sharing network (p2p-Gnutella05). They are applied commonly for submodular optimization problems [22, 61, 65]. The description of the datasets used is presented in Table 3.2.

- **ego-Facebook**: This dataset consists of 'circles' (or 'friend lists') from Facebook. This dataset includes node features (profiles), circles, and ego networks.
- **ca-GrQc**: ArXiV GR-QC (General Relativity and Quantum Cosmology) collaboration network is from the e-print arXiv and covers scientific collaborations between the papers of authors, which were submitted to the General Relativity and Quantum Cosmology category. If an author i co-authored a paper with the author j, the graph contains an undirected edge from i to j. If the paper is co-authored by k authors, this generates a completely connected (sub)graph on k nodes. The data cover papers in the period from January 1993 to April 2003 (124 months).
- **p2p-Gnutella05**: A sequence of snapshots of the Gnutella peer-to-peer file-sharing network from August 2002. There are a total of 9 snapshots of the Gnutella network collected in August 2002. The nodes represent hosts in the Gnutella network topology, and the edges represent connections between Gnutella hosts.

Table 3.2: Statistic of Datasets and r is the number of sample graphs, which are generated from the original graph $G = (V, E)$ to compute the function F

Dataset	**Nodes**	**Edges**	**Type**	r
ego-Facebook	4039	88234	Undirected, Unweighted	256
ca-GrQc	5242	14496	Undirected, Unweighted	256
p2p-Gnutella05	8846	31839	Directed, Unweighted	256

3.5.1.2 Environment

All our experiments are carried out using a Linux machine with a 2 x Intel(R) Xeon(R) CPU E5-2630 v4 @ 2.20GHz, 64GB RAM DDR4 @ 2400MHz. Our implementation is written in C/C++ language and compiled with g++ 11. We use the THREAD and OMP libraries for parallel programming.

3.5.1.3 Parameter Setting

Based on the Greedy [22], we tailor the test-specific parameters to the following: datasets implemented in the experiment are a directed graph and directly weighted. We prepossessed datasets from undirected graphs to directed graphs and assigned edges weight by using the *Weighted Cascade edge-weight* assignment method [23, 62]. The weights are assigned as $\left|\frac{1}{|In(v)|}\right|$, where $|In(v)|$ denotes in-degree of node v in the graph V. The calculation of the F function is based on the *Reachability* method of [62]. Default parameters

$\epsilon = 0.1$ and $\epsilon = 0.2$; $\alpha = \lceil \log_2 n \rceil / 2$; $m \in \{0.5, 0.6, 0.7, 0.8\}$. We choose a threshold T according to the number of vertices in the dataset, as follows: for p2p-Gnutella05, $T \in \{1000, 2000, 3000, 4000, 5000, 6000, 7000, 8000\}$, for ca-GrQc, $T \in \{1000, 2000, 3000, 4000, 5000\}$, and for ego-Facebook, $T \in \{1000, 2000, 3000, 4000\}$. For the number of sample graphs r, we set $r = 256$ according to the recent work in [22].

3.5.2 Experiment Results and Evaluation

In the experiment, we compare similar approximation guaranteed multiple thresholds regarding running time, number of queries F, ratio of $\frac{F}{T(1+\epsilon)}$ for Str-SCN-M and $\frac{F}{T+\epsilon}$ for Str-SCN-A, seed set size S, and memory usage. Two outstanding advantages of our algorithms over Greedy in the experiment are (1) *the f function value is close to threshold T and the S seed set size is close to size of Greedy's solution;* (2) *the running time and the number queries of our algorithms are 15 to 237 times faster and 12 to 208 times smaller than that of Greedy algorithm, respectively.* Our algorithms have good results as above because the complexity of Greedy is $O(n^2)$ [22], while the complexity of ours is only $O(n \log_{1+m}(n))$. The larger n is, the larger the deviation of the two complexities. We discuss thoroughly the experimental results of each metric in the following, and the results are shown in the figures of Figure 3.1, 3.2, and 3.3.

3.5.2.1 Runtime

The Greedy algorithm must always spend a lot of running time because it sequentially selects elements until $f(S) \geq T - \epsilon$ (for additive noise) and $f(S) \geq T(1-\epsilon)$ (for multiplicative noise). Therefore, it has $|S|$ passes over the data and takes $O(|S|n)$ query complexity. If $|S| = \Omega(n)$, its query complexity becomes $O(n^2)$. Thus, the runtime of Greedy depends on the size of the data n. The larger n is, the longer Greedy's runtime is. On the contrary, since the Str-SCN-A and Str-SCN-M algorithms are based on the streaming method, they only do one-pass over data to find elements that have F influence meeting the condition of threshold T. The results indicate that our algorithms are *16 to 196 times faster* for Str-SCN-M and *15 to 237 times faster* for Str-SCN-A than Greedy. Also, because the running time of our algorithms depends only on T and n, and does not depend on ϵ, the running time is not much different in two experiments $\epsilon = 0.1$ and $\epsilon = 0.2$. The results are shown in Figure 3.1.

3.5.2.2 Number of queries to function F

The running time of three algorithms is long due to the number of queries to calculate the function F. These two values are proportional to each other. Hence, the number of queries F of Greedy is also much larger than our algorithms, and increases exponentially as T and n get larger. In our experiments, the number of queries F of the Greedy is more *12 to 197 times*

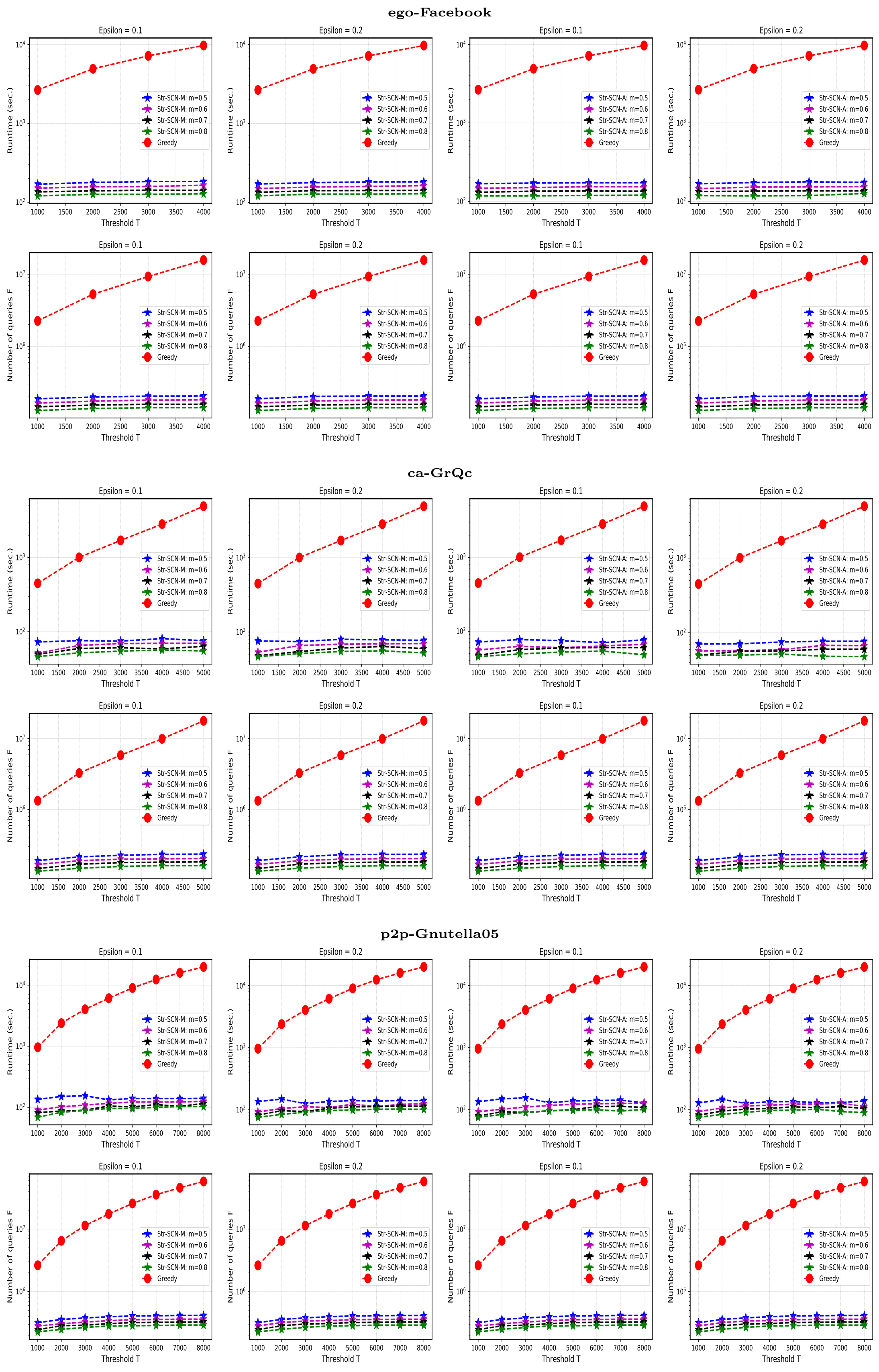
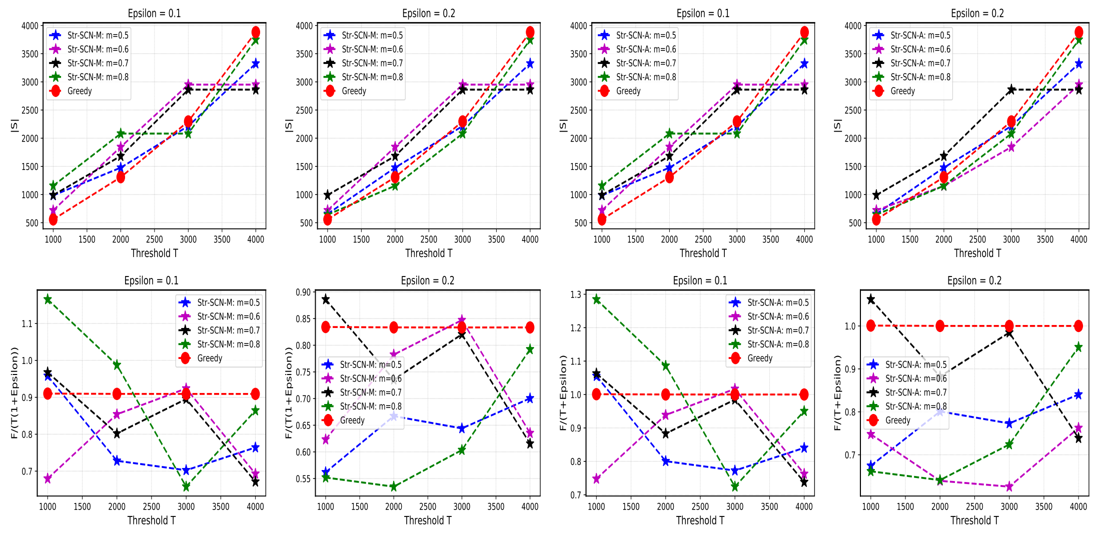
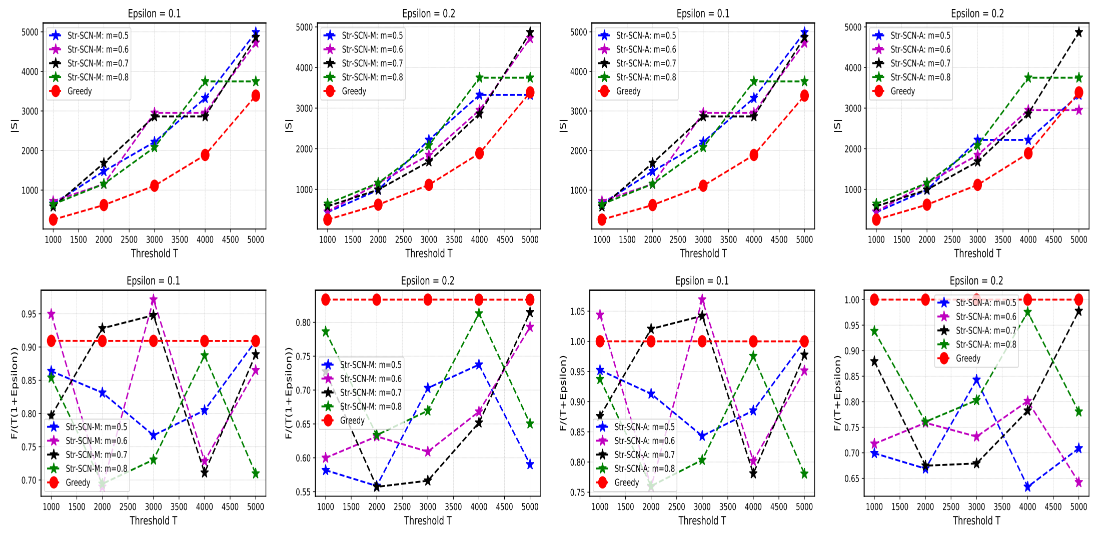


Figure 3.1: The runtime and number of queries to F of Str-SCN-M, Str-SCN-A and Greedy on three datasets

ego-Facebook



ca-GrQc



p2p-Gnutella05

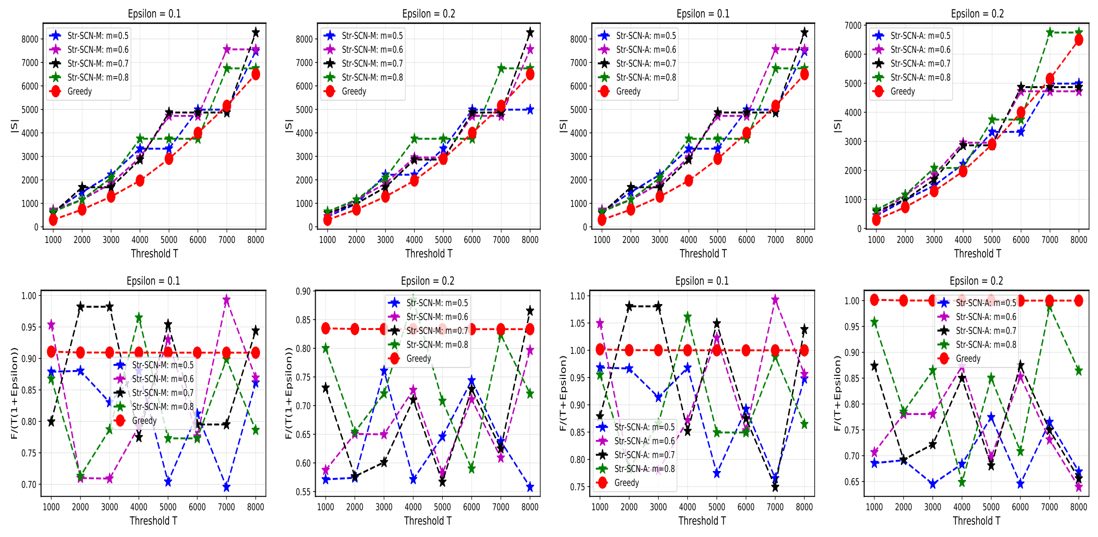


Figure 3.2: The size of seed set S and the $\frac{F}{T(1+\epsilon)}$ (multiplicative noise), $\frac{F}{T+\epsilon}$ (additive noise) ratios of three algorithms on datasets.

greater than Str-SCN-M *and 12 to 208 times greater than* Str-SCN-A. The result is shown in Figure 3.1.

3.5.2.3 Ratios of $\frac{f}{T(1+\epsilon)}$ (for multiplicative noise) and $\frac{f}{T+\epsilon}$ (for additive noise)

We compare the ratios $\frac{f}{T(1+\epsilon)}$ and $\frac{f}{T+\epsilon}$, lower bounds of $\frac{f}{T}$ under two considered noise models of our algorithms with Greedy algorithm (Figure 3.2). Because the Greedy algorithm calculates F for all elements of V, it always finds a seed set that is almost equal to the threshold values T. Hence, the ratio values of Greedy have almost no change for each experimental case. In contrast, our algorithms only need to find a S that guarantees $f(S) \geq (1-\frac{1}{\alpha})\frac{1-\epsilon}{1+\epsilon}T$ (for Str-SCN-M), and $f(S) \geq (1-\frac{1}{\alpha})T - 2\epsilon$ (for Str-SCN-A). As a result, the ratio values of our algorithms have fluctuations but still ensure the set approximation of the problem. Specifically, the experimental results obtained are as follows. For the multiplicative noise, with $\epsilon = 0.1$, the $\frac{F}{T(1+\epsilon)}$ of Str-SCN-M from 66% *to* 117%, meanwhile $\epsilon = 0.2$, the value of ratios from 53% *to* 89%. Concurrently, the ratio values of Greedy are equal to almost 91% when $\epsilon = 0.1$ and 83% when $\epsilon = 0.2$. For the additive noise, $\epsilon = 0.1$, the $\frac{F}{T+\epsilon}$ of Str-SCN-A from 72% *to* 128%, and with $\epsilon = 0.2$, the value of ratios from 63% *to* 106%. Conversely, Greedy's values are nearly equal to 100% for two values of ϵ in the additive noise case. Concisely, for both multiplication and addition noise algorithms, the results (values of ratio) of our algorithm are at least equal to 63%($\epsilon = 0.1$) and 72%($\epsilon = 0.2$) compared to Greedy's results.

3.5.2.4 Size of seed set S

The size of seed sets of our algorithms is *0.7 to 2.8 times larger* than the Greedy algorithm. The algorithms Str-SCN-M and Str-SCN-A do not depend on ϵ. Their $|S|$ are not much different in two cases $\epsilon = 0.1$ and $\epsilon = 0.2$. In summary, the difference in $|S|$ between our algorithms and Greedy is too small compared to the huge run-time benefit of ours. The result is shown in Figure 3.2.

3.5.2.5 Memory usage

The results on the memory usage of Greedy do not depend on the threshold T and m. They depend on the number of nodes and edges in the dataset, especially the more edges, the more the list of neighbors will be saved. Therefore, the experimental results of the Greedy test cases T and m have the same memory usage value. Thus, in the three datasets, although the ego-Facebook data has the smallest number of nodes, it has 2.8 times more edges than ca-GrQc and 6.1 times more than p2p-Gnutella05, so its memory usage in Greedy is 5 times higher than the others. Additionally, when comparing the results of memory usage between Greedy and our algorithms, the experimental results show that our algorithms must store data like Greedy and save a list of sets of S_i, which is used to choose the smallest S_i. Therefore, the memory usage of our algorithms is *1.2 to 5.2 times larger* than Greedy. Furthermore, the memory usage of our algorithms does not differ much between the test cases T and m. When T and m are large, the memory usage reaches the maximum fixed result for each data set. The results are shown in Figure 3.3.

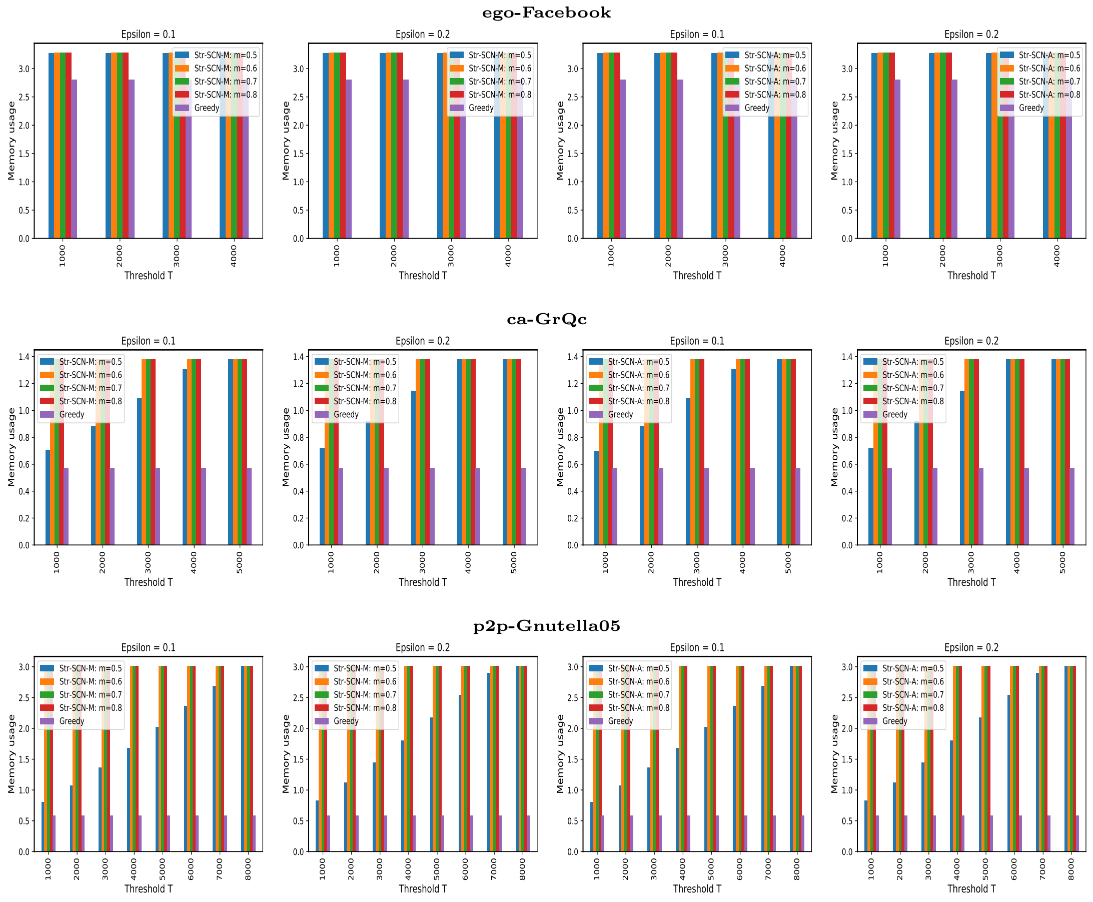


Figure 3.3: Memory usage (GB) of Str-SCN-M, Str-SCN-A and Greedy on three datasets.

3.6 Concluding remarks

We studied SCN and proposed two efficient streaming algorithms with theoretical bounds under two common noise models that are multiplicative and additive noise models. We compare our algorithms with the-state-of-art algorithm by conducting some experiments. The results reveal that our algorithms are highly scalable and outperform the adaptations in terms of both the running time and the number of queries. In future work, we will further improve the running time by developing more accelerated techniques.

Chapter 4

Problem 2. Fairness budget distribution for Influence Maximization

This chapter describes maximizing the submodular function in the IM problem (Definition 5) under budget threshold constraints, which set an pair of upper and lower bounded budgets to choose seeds in each input community to guarantee the fairness constraint (FBIM problem in short). IM is an instance of the submodular function maximization problem [23]. The content of this chapter includes the motivation to study the problem, problem definition, related work, proposed algorithms, experiment and result evaluation. Table 4.1 summarizes the notations commonly used in this chapter.

4.1 Introduction

In the digital information age, using online social networks (OSNs) has become indispensable and widespread for people. Currently, many OSNs have millions or billions of users, such as Facebook, Twitter, Instagram, LinkedIn, Youtube, and others. As a result, OSNs can rapidly influence by sharing behavior, spreading content, or messages from one person to another [66]. This propagation is similar to the way viruses spread exponentially. Based on this powerful feature of OSNs, brands or organizations use an online marketing tactic through OSNs, also known as viral marketing. Because this tactic can rapidly spread information, effectively promote products, usefully support candidates in elections, etc., on a large scale, it often helps to achieve high results with modest investment costs [67]. However, an effective viral marketing campaign must seed content with groups of influential people on OSNs. The process of identifying a group of such individuals is referred to as the IM problem.

There are many efficient approaches for the IM problem that assess the spread of influence, such as the formulation of the discrete optimization problem, which is NP-hard [23]; using continuous-time models [68]; ranking and score-based heuristics [69], or an excited approach,

Table 4.1: Table of the usually used notations in Chapter 4

Notation	**Description**
G	a graph
n	the number of nodes in the graph G.
V	the set of nodes in the graph G, $\|V\| = n$.
2^V	the subset family of V.
m	the number of edges in the graph G.
E	the set of edges in the graph G, $\|E\| = m$.
w	the set of edge weights in the graph.
v	a random node in the graph.
u	a neighbor node of v in the graph.
k	a total budget, which is upper bound of $\|S\|$.
K	the number of target communities is selected for the FBIM's input.
$\mathcal{C}^{'}$	the set of K communities in network G.
$\mathbf{C}$	a set of disjoint communities of the graph, $\|\mathbf{C}\| = N$
N	the size of a set $\mathbf{C}$
C_i, C_j	the i-th community and the j-th community.
V_i	the set of nodes of the community C_i
k_i^l	the lower bounded budget of the community C_i.
k_i^u	the upper bounded budget of the community C_i.
S	the returned size-k seed set of algorithms.
S^*	an optimal size-k seed set.
R_i, R_j	a random RR set.
T	the number of nodes in $\mathcal{C}^{'}$, $T = \left\| \bigcup_{C_i \in \mathcal{C}'} (C_i) \right\|$ and $T \leq n$.
$\mathcal{R}, \mathcal{R}^{'}, \mathcal{R}_1, \mathcal{R}_2$	the set of random RR sets.
$\mathsf{Cov}_{\mathcal{R}}(S)$	number of RR sets in R incident at some node in S.
$f(S)$	influence spread of a seed set S
$f_l(S), f_u(S^*)$	the lower bound of $f(S)$, the upper bound of $f(S^*)$.
$\hat{f}(S)$	an estimation of $f(S)$ on a collection of RR sets $\mathcal{R}$
$\mathbb{E}$	the expected value
$\mathcal{M}$	a matroid

using sketches and the *Reverse Influence Sampling* (RIS) framework, which was suggested by Borgs *et al.*[24]. Numerous publications have used RIS to solve the problem IM with positive results [2, 15, 70, 71].

However, most of these existing solutions to the problem IM focus on the most influential nodes to maximize the total number of affected nodes. That means these methods only aim to find the most influential users to maximize the number of affected people. Meanwhile, they lost interest in whether the influenced people are fairly distributed over the network. Thus, there is a high probability that users in groups that do not contain seeded users will not be influenced or will not receive the spread information. On the contrary, it is these users who must need to be affected.

For example, viral marketing is a significant and standard application of the IM problem. The objective is to maximize the profits of advertisers by promoting products to users on OSNs [72]. OSNs are the fertile ground of the advertising field, which has millions or billions of users, and the number of users continuously increases every day. However, it is troubling that users of OSNs often encounter advertising content that has been viewed too many times or purchased, so they do not care about them. In contrast, users who are the right customers for these ads do not receive them. That is boring and frustrating for users. Therefore, the challenge is how to spread the advertising content to the right customers and other potential customer groups in the OSNs communities.

In recent years, to conquer the above weakness of IM, a new variant of it has been developed that has attracted the attention of researchers. That is, the problem *fair influence maximization* (FIM) which aims to ensure a fair distribution for the groups in the final set of selected nodes [43, 73]. In other words, it guarantees coverage propagation in the target communities. However, each of the existing methods offers a unique perspective on fairness. For all we know, there are no publications that consider both minimum and maximum budget constraints for each group to guarantee equitable distribution.

Fueled by this challenge, we study FIM under budget threshold constraint, setting an upper and lower bounded budget to choose seeds in each community to guarantee fairness. This problem is called FBIM problem shortly. Specifically, our contributions are as follows.

- We propose three algorithms to solve the problem FBIM. These algorithms provide a $(1/2 - \epsilon)$-approximation to the optimum solution work efficiently in big data.

  (1) The first is the FBIM1 algorithm that uses a combination of some methods: generating sampling by the RIS framework with the *dynamic stop-and-stare algorithm*, also known as. DSSA in [2], and adding the fairness constraint in seed set selection. Our algorithm has $O\left(kT\log\left((8+2\epsilon)n\frac{\ln\frac{2}{\delta}+\ln\binom{n}{k}}{\epsilon^2}\right)\right)$ complexity. The results show that our algorithm has a runtime, and the objective function value can be equal to or better than

DSSA, which depends on the adjustment of the dependent parameters. Particularly, our method resolves the fairness constraint, while DSSA lacks that [39].

(2) The second is the FBIM2 algorithm that combines seed selection to ensure maximum coverage and fairness constraint with the *online processing influence maximization* algorithm, also known as OPIMC in [15]. FBIM2 has $O\left(kT \log \frac{n}{\epsilon^2 k}\right)$ complexity. Furthermore, like DSSA, OPIMC does not solve the fairness constraint.

(3) The last is the FBIM3 algorithm that improves FBIM2 using the greedy technique with a threshold criterion for selecting a seed set. FBIM3 has $O\left(\frac{T}{\epsilon} \log \frac{k}{\epsilon} \log \frac{n}{\epsilon^2 k}\right)$ complexity. Significantly, the seed set's distribution guarantees a high coverage ratio, which is an expression of ensuring the fairness constraint.

- We further investigate the performance of our algorithms by conducting some experiments for the FBIM under both well-known diffusion models, *Linear Threshold* and *Independent Cascade* [23], on real social networks. The results indicate the seed sets of our algorithms, whose coverage ratio over communities is greater than the result of OPIMC. It is 2x to 10x larger and there are even some cases in which FBIM reaches 100% coverage on target communities. This process depends on the appropriate parameter selection for each dataset. An extensive coverage ratio signifies the number of chosen seeds covering the target communities, ensuring that the impacted individuals are the ones we want to influence. Besides, the efficiency of FBIM's algorithms must be influenced by objective factors due to the implementation method and the fairness constraints. This leads to more cost but lower objective function value than OPIMC. Nevertheless, the results still guarantee the optimal theoretical approximation, especially the fairness constraint.

4.2 Related work

According to much earlier research [23, 47, 60, 74], the problem IM is often addressed using a greedy technique with an approximation of $(1 - 1/e)$. Although the greedy strategy is quite successful for IM, computing the influence function $f(S)$ is still challenging because it is P-hard. Existing approaches for IM may be classified into three primary classes based on how the influence function is calculated [75].

(1) The simulation-based approaches, such as Greedy [23], CELF [76], CELF++[41, 42], UBLF[77], calculate the influence function using Monte Carlo sampling. To achieve highly scalable algorithms for IM, they merged heuristic techniques based on the greedy algorithm. These algorithms aim to produce an $(1 - 1/e)$ approximation answer. These methods have an advantage that is suitable for diffusion models. Nevertheless, the disadvantage is that, if

we want to calculate the objective function with minor errors, we must generate many sample cases. Hence, it significantly increases computational costs.

(2) *The proxy-based approaches*, such as SimPath [78], Degree [41], PageRank [79, 80, 81], EasyIM [82], approximate the influence function $f(S)$ to conquer the P-hard by devising proxy models. The approximate solution obtained is $(1 - 1/e - \epsilon)$ for any $\epsilon > 0$. Many algorithms have demonstrated that the proxy-based strategy is efficient for empiricism but does not provide theoretical guarantees.

(3) *The sketch-based approaches*, such as TIM, TIM+ [83], IMM [84], use a novel RIS sampling method. The goal of these techniques is to produce an $(1 - 1/e - \epsilon)$-approximate solution with minimal numbers of RIS samples [24]. The drawback of these approaches is that their lower bounded budget is unknown, and the number of samplings generated can be rather large. Furthermore, these algorithms guarantee theoretically efficient, having rigorously bounded solutions and minimal time complexities. However, because they must ensure the approximation ratio for the worst-case scenario, the sketch-based strategy's practical efficiency may be lower than that of the proxy-based approaches.

Subsequently, Nguyen *et al.* [2] proposed two new sampling techniques, SSA and DSSA, which attempt to obtain a small number of RIS samples while ensuring $(1-1/e-\epsilon)$-approximations. Despite this, Huang *et al.* [85] discovered that SSA/DSSA has certain flaws. They also provided SSA-Fix, an updated version of SSA. Afterward, Nguyen *et al.* presented D-SSA-Fix [86], a significantly updated version of DSSA that produces $(1 - 1/e - \epsilon)$-approximations.

In addition, there are many studies that resolve other variants of IM [70, 83, 84, 86]. However, almost all of these methods focus on *offline processing*. This means that the user does not receive any output until the final result is obtained. Thus, the user cannot terminate the algorithm early to trade the solution quality for efficiency. Motivated by this phenomenon, Tang *et al.* proposed the OPIMC algorithm for *online processing* of influence maximization in [15]. This method can allow the user to terminate any timestamp, then get a seed set S and report an approximate guarantee $(1 - 1/e - \epsilon)$, such that S is the IM problem's approximate solution with at least $(1 - \alpha)$ probability, with $\alpha \in (0, 1)$ is a user-specified parameter.

Unsuccessfully, virtually all of the above methods have focused on identifying the most influential nodes to increase the number of nodes affected. They fail to ensure that chosen nodes are evenly distributed across the partitions of the dataset. This shortcoming as a driving force has attracted the attention of researchers. Recent publications have proposed multiple definitions of fairness and explicitly integrated fairness into the IM problem. One of these fairness concerns, known as group fairness, is to ensure that each group receives a fair share of resources and that the distribution of those resources respects the various makeup of the groups. These studies for the group fairness constraint in the IM problem have obtained positive results.

Tsang *et al.* (2019)[87] proposed the issue of optimizing the dissemination of a strategy while keeping a group-fairness restriction in mind. The authors developed two fairness measures (*maximin fairness* and *group rationality*) to assess group-fairness in IM. Maximin fairness measures the smallest number of nodes within each group that are influenced must be maximized. Meanwhile, the main principle of group rationality is that no group increases its influence when it withdraws from IM with its proportional allocation of resources and distributes those resources internally. Both measures aim to ensure that each group receives an equitable share of resources.

Stoica *et al.* (2020) [88] studied the problem of fair resource allocation in influence maximization. The authors provided an algorithmic framework to locate solutions that satisfy fairness constraints for multi-objective submodular maximization problems. This method increases the diversity of nodes in the seed set and may have an impact on the effectiveness and fairness of the information diffusion process. The authors demonstrated that, in some circumstances, seeding methods that consider the diversity of nodes in the seed set are more effective and fair.

Halabi *et al.* (2020) [89] worked on the problem of maximizing fair submodular functions (including monotone and non-monotone submodular functions). They proposed streaming algorithms for this problem. For monotone case, the authors achieved two results, the (1/2)-approximation algorithm and the (1/4)-approximation algorithm use $O(\log k)$ time. For non-monotone case, they achieved a $(q + \epsilon)$-approximation with q is an input excess ratio and requires $O(k)$ time. These approaches apply to creating fair summaries for a massive dataset.

Next, Khajehnejad *et al.* (2020) [43] studied the fair influence maximization in an effort to more fairly reach minorities. The authors used machine learning approaches to pick a seed set using an adversarial graph embedding technique, which allows for strong impact propagation as well as fairness amongst communities.

Rahmattalabi *et al.* (2021) [44] resolved the problem that Tsang *et al.* studied in [87]. This is the group-fairness in the influence maximization problem. However, Rahmattalabi *et al.* took a different approach, which is to offer a principled characterization of the properties that a fair influence maximization algorithm must meet. The authors designed a framework founded on the social welfare theory that aggregates the cardinal utilities each community derives using isoelastic social welfare functions. In this framework, the trade-off between fairness and efficiency can be handled by a single inequality aversion design parameter.

More recently, Becker *et al.* (2022) [73] also considered the problem the same as Tsang *et al.*[87] and also proposed a new approach. The authors modeled the problem on the basis of the probabilistic techniques used to choose seed sets rather than purely deterministic ones. They provided two variations of this probabilistic problem: the node-based problem, which uses probabilistic strategies over nodes, and the set-based problem, which uses probabilistic strategies over sets of nodes. After examining the relationship between the two probabilistic problems, the authors demonstrated that both probabilistic variants give approximation algorithms that achieve a constant multiplicative factor of $(1 - 1/e)$ minus an arbitrarily small additive error caused by the simulation of the information spread.

Ali *et al.* (2022) [90] solved the fairness of the spread process in various groups. They focused on the time-critical aspect of IM, examining the number of affected people and the time step at which they are influenced. After doing these things, maximizing the expected number of individuals reached by selecting a fixed cardinality seed set and minimizing the amount of seeds needed to affect a given portion of the network can lead in unfair solutions. The authors proposed an objective function that balances two goals, such as maximizing the expected number of nodes reached and minimizing the maximum disparity in influence between any two communities.

Razaghi *et al.* (2022) [91] also worked on the group-fairness-aware influence maximization in social networks as Tsang *et al.* [87]. However, they fixed an important omission that Tsang *et al.* overlooked, that is, did not assess the time for receiving resources by nodes of groups. The authors expand the concept of group-fairness in the IM problem by examining the speed of nodes' activation in different groups of social networks. They proposed a multi-objective meta-heuristic (SetMOGWO), founded on the multi-objective gray wolf optimizer, to increase the fair propagation of information in IM problem relating to various fairness measures.

Through the literature available, these publications show that each study is concerned with a different aspect or constraint on the problem of group fairness in IM. As far as we know, there has not been a single study that mentions the fairness constraints on the lower and upper bounded budgets of the target communities in the dataset. That is the reason and impetus for us to research and find a solution to this shortcoming.

4.3 Problem definition

In this section, we first introduce the definition of the Fair Submodular Maximization (FSM) problem [89], the generation of our studied problem, and recap the Fair Greedy algorithm that returns an approximation of 1/2 within $O(nk)$ queries. We then introduce our studied fairness influence maximization problem and recap some properties of the problem.

4.3.1 Fair Submodular Maximization problem

Given a ground set V of n elements and a submodular function $f : 2^V \to \mathbb{R}_+$, with f is a monotone submodular function. The problem FSM is defined as follows.

Definition 12 (FSM problem) *Assume that V is divided into $\mathcal{C}'$ disjoint subsets $V_1, V_2, \dots, V_K$, and $C_i \cap C_j = \emptyset$. Each subset V_i has a lower and an upper bound on the number of elements*

k_i^l and k_i^u representing the fairness constraint. Let a positive number k be a global cardinality constraint. The FSM *asks to find:*

$$\textit{max: } f(S) \tag{4.1}$$

$$\textit{subject to: } |S| \le k \tag{4.2}$$

$$k_i^l \le |S \cap V_i| \le k_i^u, \quad i = 1 \ldots K \tag{4.3}$$

We defined an instance of FSM problem as a tuple $(f, V, V_1, \ldots, V_K, k)$. The problem FSM can be reduced to submodular maximization under a matroid constraint (SMM) problem [89] defined as follows.

Definition 13 (SMM problem) *The problem ask to select a set $S \subseteq V$ with $S \in \mathcal{M}$ such that it maximize $f(S)$, where $\mathcal{M}$ is a matroid. If a family of sets $\mathcal{M}$ subset 2^V satisfies the following conditions, it is referred to as a matroid.*

1. *$\mathcal{M} \neq \emptyset$;*
2. *downward-closedness: if $A \subseteq B$ and $B \in \mathcal{M}$, then $A \in \mathcal{M}$;*
3. *augmentation: if $A, B \in \mathcal{M}$ with $|A| \le |B|$, then there exists $s \in B$ such that $A + s \in \mathcal{M}$.*

Fair Greedy algorithm. We now recap the Fair Greedy algorithm (Algorithm 3) [89] The operation of this algorithm based on the notation of an extendable set.

```
Algorithm 3: Fair Greedy algorithm
   Input: An instan of FSM problem $(f, V, V_1, \ldots, V_K, k)$
   Output: A 1/2-approximation solution $S$
1. $S \leftarrow \emptyset$
2. for $i = 1$ to $k$ do
3.     $U \leftarrow \{s \in V | S \cup s$ is extendable $\}$
4.     $v \leftarrow \arg\max_{s \in U} f(s|S)$
5.     $S \leftarrow S \cup \{v\}$
6. return $S$
```

Definition 14 (An extendable set S [89]) *A set $S \subseteq V$ is extendable if and only if*

$$|S \cap C_i| \le k_i^u, \quad i = 1, \ldots, K \textit{ and } \sum_{i=1}^{K} \max(|S \cap C_i|, k_i^l) \le k \tag{4.4}$$

4.3.2 Fairness Budget Influence Maximization problem

In this section, we introduce the problem FBIM, which is the focus of this study, and its related definitions. FBIM inherits two issues: *fair submodular maximization* (FSM) [89] and *fair influence maximization* (FIM) [44].

Definition 15 (FIM problem) *Given a graph $G = (V, E)$, V is the set of vertices, $|V| = n$, E is the set of edges, $|E| = m$ and a global cardinality constraint $k \in \mathbb{Z}^+$. Let $\mathbf{C}$ be a set of disjoint communities (empty intersection) of the graph. Each vertex v of V belongs to one of the communities $C_i \in \mathbf{C}$, $i \in 1, ..., N$ such that $V_1 \cup ... \cup V_N = V$ where V_i denotes the set of vertices of the community C_i. Furthermore, communities may be disconnected, that is, $\forall C_i, C_j \in \mathbf{C}$ and $\forall v \in V_i, u \in V_j$, there is no edge between v and u. $\mathcal{A}$ denotes the initial set of vertices (referred to as influencer vertices), and we define $\mathcal{A}^* := \{\mathcal{A} \subset V||\mathcal{A}| \leq k\}$ as the set of feasible budget influencers. Lastly, for any choice of $\mathcal{A} \in \mathcal{A}^*$, we let $h_{C_i}(\mathcal{A})$ denote the expected fraction of the influenced vertices of the community C_i, where the expectation is taken under a diffusion model in the spread of influence. The fair influence maximization problem solves the optimization problem*

$$\underset{\mathcal{A} \in \mathcal{A}^*}{maximize} \sum_{C_i \in \mathbf{C}} |V_i|.h_{C_i}(\mathcal{A}) \tag{4.5}$$

Definition 16 (FBIM problem) *Given a social network $G = (V, E, w)$, $|V| = n$ and a set $\mathcal{C}' = \{C_1, C_2, \ldots, C_K\}$ where $\forall C_i \subset G$ and $C_i \bigcap_{i \neq j} C_j = \emptyset$. Each group C_i has a pair of lower k_i^l and upper k_i^u bounded budgets, $k_i^l \leq k_i^u$. For a given total budget k, the problem asks to find a seed set $S, |S| \leq k$, which satisfies $k_i^l \leq |S \cap C_i| \leq k_i^u$ so that $f(S)$ is maximal, where $f(S)$ is the influence function. The $f(S)$ measures the expected number of users in V that can be influenced by the elements in S under a diffusion model.*

In this study, we solve the problem FBIM under both information diffusion models, IC (in Definition 6) and LT (in Definition 7).

For problem FBIM, according to the proposition of Halabi *et al.* in [89], the greedy method selects the element with the highest *marginal gain* (the marginal gain of an element s is the value of $(f(S \cup s) - f(S))$) while meeting the constraints. We note that the greedy approach might not produce a feasible solution if this element was only required to meet the upper bound and cardinality constraints. It may satisfy the global cardinality restriction without meeting the lower bounds. As a result, more careful element selection is required. To this end, the seed set S must be an extendable set.

Besides, there are some definitions that are usually used in this study. They are *Reverse Influence Sampling* (RIS), *Reachable Reverse set* (RR set) and the *Coverage of seed set S on the set of Reachable Reverse sets* ($\mathsf{Cov}_{\mathcal{R}}(S)$).

Definition 17 (RIS **[24]**) *Given a graph $G = (V, E, w)$,* RIS *apprehend the influence scene of G by generating a set $\mathcal{R}$ of random* RR *sets. A* RR *set contains nodes that can reach v in g, where v is a random node in V and g is a sample graph from G.*

Definition 18 (**RR set** **[24]**) *Given a graph $G = (V, E, w)$ under the* IC *model. A random* RR *set R_i is generated from G according to the following steps:*
- *Step 1. Choose a source node $v \in V$.*
- *Step 2. Generate a sample graph g from G.*
- *Step 3. Return a R_i that contains nodes can be reached from v in g.*

We denote $\mathcal{R}$ as the set of random RR sets. As reported by [2], finding a seed set S and influence spread $f(S)$ is based on computing the coverage of S on most RR sets. Due to the generation of a set $\mathcal{R}$ of multiple random RR sets, influential nodes may appear frequently in the RR sets. Therefore, we find the nodes that appear in the most RR sets to add the seed set S. Besides, the influence spread $f(S)$ on a random RR set R_i is proportional to the probability that S intersects with R_i. Thus, a seed set S covers a set RR R_i if $S \cap R_i \neq \emptyset$. For simplicity, we denote the coverage of S on $\mathcal{R}$ as $\mathsf{Cov}_{\mathcal{R}}(S)$, and it is calculated as follows.

$$\mathsf{Cov}_{\mathcal{R}}(S) = \sum_{R_i \in \mathcal{R}} \min\{1, |S \cap R_i|\} \tag{4.6}$$

Furthermore, the influence spread $f(S)$ on a random RR set R_i is proportional to the probability that S intersects with R_i. According to recent work [24], we can calculate the influence spread function $f(\cdot)$ as follows:

$$f(S) = n \cdot \mathbb{E}[\min\{1, |S \cap R_i|\}] \text{ (with } \mathbb{E} \text{ is the expected value)} \tag{4.7}$$

and an estimation of $f(S)$ over a collection of RR sets $\mathcal{R}$ is

$$\hat{f}(S) = n \cdot \mathsf{Cov}_{\mathcal{R}}(S)/|\mathcal{R}| \tag{4.8}$$

4.4 Proposed algorithms

In this section, we design a Threshold Greedy algorithm for the FSM problem. Later, based on this Greedy algorithm, we improved to develop one of the algorithms for FBIM because FSM is the generation of the FBIM.

4.4.1 Threshold Greedy algorithm for FSM

The Threshold Greedy algorithm applies the decreasing threshold greedy strategy for the FSM problem. This algorithm is improved from Algorithm 3 by using a decreasing threshold to reduce the number of data traversals while still ensuring the approximate solution.

a. Algorithm description

The Threshold Greedy algorithm operates as follows. At the beginning, it initiates the initial threshold t with M, where $M = max_{u \in V} f(u)$. One loop of the inner loop 'for' is named one *iteration.* It scans all elements in S. At each iteration, if the current element s satisfies two conditions, $S \cup s$ is extendable and $f(s|S) \geq t$, s will be selected into S. After each loop of the "while" loop, t will be decremented by $(1-\epsilon)$ times until $t < \epsilon M/k$ then the algorithm terminates and returns S. A detailed description of this algorithm is given in Algorithm 4.

Algorithm 4: Threshold Greedy algorithm

```
Input: An instance of FSM problem $(f, V, V_1, \dots, V_K, k)$, accuracy parameter $\epsilon$.
Output: A $(1/2 - \epsilon)$-approximation solution $S$
1. $S \leftarrow \emptyset$
2. $M \leftarrow \max_{u \in V} f(u)$
3. $t \leftarrow M$
4. while $t \geq \epsilon M/k$ do
5.     for $s \in V$ do
6.         if $S \cup \{s\}$ is extendable and $f(s|S) \geq t$ then
7.             $S \leftarrow S \cup \{s\}$
8.     $t \leftarrow (1-\epsilon)t$
9. return $S$
```

b. Theoretical analysis

The following theoretical analysis and proofs in Lemma 2 and Theorem 3 will demonstrate the feasibility and efficiency of this algorithm to guarantee the approximation solution.

Lemma 2 *Denote t_i be threshold t at the i-th iteration and S_i be S at the beginning of the i-th iteration. We first show that:*

$$t_i \geq (1-\epsilon) \max_{s \in V: S_i \cup \{s\} \text{ is extendable}} f(s|S_i) \tag{4.9}$$

Proof We prove Lemma 2 using the induction. If $i = 1$, $S_1 = \emptyset$ and $t_1 = M \geq (1-$

$\epsilon)\max_{s\in V:S_1\cup\{s\}\text{ is extendable}} f(s)$. Assume that the lemma holds for $i \geq 1$. We have:

$$t_{i+1} = (1-\epsilon)t_i \geq (1-\epsilon) \max_{s\in V:S_{i+1}\cup\{s\}\text{ is extendable}} f(s|S_{i+1}) \tag{4.10}$$

The inequality is due to the fact that the element

$$s_{max} = \arg \max_{s\in V:S_{i+1}\cup\{s\}\text{ is extendable}} f(s|S_{i+1})$$

was not added to S in iteration i. The lemma is proved. ■

Theorem 3 *Algorithm 4 requires $O(\frac{n}{\epsilon}\log\frac{k}{\epsilon})$ runtime and returns a $(1/2-\epsilon)$-approximation solution for* FSM *problem.*

Proof The computational complexity can easily be proved. Assume that there are in total x number of iterations in the "while" loop of Algorithm 4. Therefore, we have $(1-\epsilon)^x = \frac{\epsilon}{k}$. Solving this equation yields

$$x = \frac{\log\frac{k}{\epsilon}}{\log\frac{1}{1-\epsilon}} \leq \frac{1}{\epsilon}\log\frac{k}{\epsilon} \tag{4.11}$$

The "for" loop does n iterations. Thus, the time complexity of this algorithm is $O(\frac{n}{\epsilon}\log\frac{k}{\epsilon})$. Besides, assume that $s_{max} = \max_{s\in V:S_i\cup\{s\}\text{ is extendable}} f(s|S_i)$, $S_i = S_{i-1}\cup\{s_1, s_2, \ldots, s_l\}$, and S^s be S just before going s is processed and $S^s_i \cup \{s\}$ is extendable. We have the following:

$$t_i \geq f(s_{max}|S^s) \geq f(s_{max}|S_i) \tag{4.12}$$

Now, assume that $S = \{s_1, s_2, \ldots, s_u\}, u \leq k$ is S after ending the main loop of the algorithm. We consider the following two cases:

Case 1. We have $|S| = u, u = k$, assume that $O = \{o_1, o_2, \ldots, o_k\}$ is the optimal solution such that $\{s_1, s_2, \ldots, s_{i-1}, o_i\}$ can be extended, which exists because the extendability is matroids [89] and satisfies the augmentation property. Denote S^i be S right before s_i is added to S and $t(s_i)$ be t at the iteration s_i be added to S. We have the following:

$$\begin{aligned}
f(S) &= \sum_{i=1}^{u} f(s_i|\{s_1, s_2, \dots, s_{i-1}\}) \\
&\geq \sum_{i=1}^{u} t(s_i) \\
&\geq \sum_{i=1}^{u} (1-\epsilon) f(s_i|S^i) (\text{By Lemma 2}) \\
&\geq \sum_{i=1}^{u} (1-\epsilon) f(o_i|S^i) \\
&\geq \sum_{i=1}^{u} (1-\epsilon) f(o_i|\{s_1, s_2, \dots, s_{i-1}\}) \\
&\geq \sum_{i=1}^{u} (1-\epsilon) f(o_i|\{s_1, s_2, \dots, s_u\} \cup \{o_1, \dots, o_{i-1}\}) (\text{Due to the submodularity of } f) \\
&= \sum_{i=1}^{u} (1-\epsilon) f(o_i|S \cup \{o_1, \dots, o_{i-1}\}) \\
&= (1-\epsilon)(f(O \cup S) - f(S)) (\text{by equality (2)})
\end{aligned}$$

and so

$$\begin{aligned}
f(S) &\geq \frac{(1-\epsilon)}{(2-\epsilon)} f(O) \\
&\geq \frac{(1-2\epsilon)}{2} f(O) \\
&= (\frac{1}{2} - \epsilon) f(O)
\end{aligned}$$

Case 2. We have $|S| = u, u < k$ and $S = \{s_1, s_2, \dots, s_u\}$, assume that O is the optimal solution, $S' = \{s_{u+1}, \dots, s_k\}$ and $S_k = \{s_1, s_2, \dots, s_u\} \cup \{s_{u+1}, \dots, s_k\}$, so $S_k = S \cup S'$. Denote t_{last} be t after ending the main loop of the algorithm. Thus, we have the following:

$$\begin{aligned}
\sum_{i=u+1}^{k} f(s_i|\{s_1, s_2, \dots, s_{i-1}\}) &\leq kt_{last} \text{ (Due to } f(s_i|S)_{i=u+1,\dots,k} \leq t_{last}) \\
&\leq k\frac{\epsilon M}{k} \leq \epsilon M \leq \epsilon f(O)
\end{aligned}$$

and so

$$f(S_k) - f(S) \leq \epsilon f(O)$$

We have the equivalent inequality as follows:

$$\begin{aligned} f(S) &\geq f(S_k) - \epsilon f(O) \\ &\geq (\frac{1}{2} - \epsilon) f(O) - \epsilon f(O) \text{ (By the proof of Case 1)} \\ &\geq (\frac{1}{2} - 2\epsilon) f(O) \\ &\geq (\frac{1}{2} - \epsilon') f(O) \text{ (with } \epsilon' = 2\epsilon) \end{aligned}$$

The proof is completed. ■

As above mentioned, FSM is the generation of the FBIM problem. Therefore, we based on the Threshold Greedy algorithm as an important key to designing an efficient algorithm for the FBIM problem in the next section. It is the Threshold Greedy algorithm for Fairness Max Cover (Algorithm 8), which finds a seed set S so that it satisfies to maximize the coverage and fairness constraint.

4.4.2 Proposed Algorithms for FBIM

This section introduces three algorithms for the FBIM problem, including FBIM1, FBIM2, and FBIM3. The details of these algorithms are fully presented in Algorithms 6, 7, and 9, respectively. Because our proposed algorithms are based on the DSSA [2] and OPIMC [15] methods, we do not perform the proofs given in their originals.

4.4.2.1 FBIM1 - an algorithm based on the stop-and-stare method

This method combines an improved greedy strategy for selecting seeds satisfied the fairness constraint with generating sampling of the DSSA. The FBIM1 algorithm's fundamental principle is: *(1) choose k nodes that occur on the majority of communities and add them to S to maximize the coverage of S on the set of* RR *sets*; *(2) calculate $f(S)$ by the stop-and-stare technique. If the result does not satisfy the threshold condition of the algorithm, the process repeats the search for a new S on the new set of RR sets with double the number of elements.* The details of the algorithm are fully presented in Algorithms 5 and 6.

a. Algorithm description

The FBIM1 algorithm (Algorithm 6) inputs a K-size communities set $\mathcal{C}'$, a budget k, parameters ϵ and δ, $(0 \leq \epsilon, \delta \leq 1)$, which are related to the solution's quality and the algorithm's runtime. They allow us to adjust the solution quality trade-off with runtime. Especially, ϵ and δ guarantee the size bound of S. In contrast, k and the set $\mathcal{C}'$ guarantee the value of the objective function $f(S)$ and the coverage ratio of S in communities in $\mathcal{C}'$, which is considered to satisfy the fairness constraint.

Algorithm 5: Fairness-Max-Coverage (FMC) procedure

```
Input: Graph $G$, $\mathcal{C}'$, $k$, RR sets ($\mathcal{R}$).
Output: A seed set $S$ that satisfies fairness constraint, $|S| \leq k$ and its estimated
        influence $f(S)$.
1. $S \leftarrow \emptyset$
2. for $i = 1$ to $k$ do
3.     $\mathbb{U} \leftarrow \{s \in (C_{j_{C_j \in \mathcal{C}'}} \setminus S) | S \cup s$ is extendable $\}$
4.     $v \leftarrow \arg\max_{\{s \in \mathbb{U}\}} (\mathsf{Cov}_{\mathcal{R}}(S \cup s) - \mathsf{Cov}_{\mathcal{R}}(S))$
5.     $S \leftarrow v$
6. return $S$.
```

Algorithm 6: FBIM1 algorithm

```
Input: Graph $G$, $\mathcal{C}'$, $k$, and $0 \leq \epsilon, \delta \leq 1$
Output: A seed set $S$, $|S| \leq k$.
1. $\Gamma \leftarrow \sqrt{8}(1+\epsilon)^2 ln(\frac{2}{\delta})\frac{1}{\epsilon^2}$
2. $\mathcal{R} \leftarrow$ Generate $\Gamma$ random RR sets by RIS
3. $S \leftarrow \mathsf{FMC}(G, \mathcal{C}', k, \mathcal{R})$
4. $f(S) \leftarrow n.\mathsf{Cov}_{\mathcal{R}}(S)/|\mathcal{R}|$
5. repeat
6.     $\mathcal{R}' \leftarrow$ Generate $|\mathcal{R}|$ random RR sets by RIS
7.     $f'(S) \leftarrow n.\mathsf{Cov}_{\mathcal{R}'}(S)/|\mathcal{R}'|$
8.     $\epsilon_1 \leftarrow (f(S)/f'(S)) - 1$
9.     if $\epsilon_1 \leq \epsilon$ then
10.        $\epsilon_2 \leftarrow (\epsilon - \epsilon_1)/(2(1+\epsilon_1))$
11.        $\epsilon_3 \leftarrow (\epsilon - \epsilon_1)/(2(1-\frac{1}{e}))$
12.        $\delta_1 \leftarrow e^{-\frac{\mathsf{Cov}_{\mathcal{R}}(S).\epsilon_3^2}{\sqrt{8}(1+\epsilon_1)(1+\epsilon_2)}}$
13.        $\delta_2 \leftarrow e^{-\frac{(\mathsf{Cov}_{\mathcal{R}'}(S)-1).\epsilon_2^2}{\sqrt{8}(1+\epsilon_2)}}$
14.        if $\delta_1 + \delta_2 \leq \delta$ then
15.            return $S$.
16.    $\mathcal{R} \leftarrow \mathcal{R} \cup \mathcal{R}'$
17.    $S \leftarrow \mathsf{FMC}(G, \mathcal{C}', k, \mathcal{R})$
18.    $f(S) \leftarrow n.\mathsf{Cov}_{\mathcal{R}}(S)/|\mathcal{R}|$
19. until $|\mathcal{R}| \geq (8+2\epsilon)n.\frac{ln\frac{2}{\delta} + ln\binom{n}{k}}{\epsilon^2}$;
20. return $S$.
```

In the initial step, the algorithm generates a set $\mathcal{R}$ that contains Γ the random RR sets by RIS, with the value of Γ initialized as line 1. The formula of Γ was proved in [2]. Subsequently, based on this $\mathcal{R}$, the FMC procedure (Algorithm 5) returns a seed set S.

Algorithm 5 receives a K-size community $\mathcal{C}^{'}$ set, a budget k and a set $\mathcal{R}$. This algorithm produces a k-size seed set S that satisfies fairness constraint. Initiate an empty set S. It performs a loop of k times, each iteration finds an element s in all C_j of $\mathcal{C}^{'}$, except in S, so that $(S+s)$ is extendable and the coverage of s on the RR sets of $\mathcal{R}$ is maximal. This element s is added to S. After the loop ends, the algorithm returns the seed set S.

We now turn our attention to Algorithm 6. The algorithm executes an indefinite loop. In each iteration, it evaluates the efficiency of S by computing $f^{'}(S)$ on the set $\mathcal{R}^{'}$ and checks whether $(f(S)/f^{'}(S))-1$ meets the conditions in lines 9 and 14. If it is true, the algorithm stops. Otherwise, $\mathcal{R}$ doubles by adding $\mathcal{R}^{'}$ to find a new S and goes to the next iteration. The loop stops when $|\mathcal{R}|$ is at least $(8+2\epsilon)n.\dfrac{ln\frac{2}{\delta}+ln\binom{n}{k}}{\epsilon^2}$.

b. Theoretical analysis

Theorem 4 *Algorithm 5 is an improved greedy algorithm, has $O(k.T)$ complexity with $T=\left|\bigcup_{C_i\in\mathcal{C}'}(C_i)\right|$ and $T\leq n$.*

Proof Algorithm 5 iterates k times to select seeds for the seed set S of size k. Each iteration scans the majority of the elements of C_i ($\forall C_i\in\mathcal{C}^{'}$) to select an element that has the greatest coverage on $\mathcal{R}$. As a result, the complexity of the algorithm is $O(kT)$. ∎

Theorem 5 *Algorithm 6 has $O\left(k.T.log\left((8+2\epsilon)n.\frac{ln\frac{2}{\delta}+ln\binom{n}{k}}{\epsilon^2}\right)\right)$ complexity.*

Proof Algorithm 6 iterates the generation of the set $\mathcal{R}$ of random RR sets with randomly chosen source elements from C_i and finding $\langle S,f(S)\rangle$ founded on $\mathcal{R}$ through Algorithm 5. At lines 14 and 19, Algorithm 6 has two requirements to break the loop. Because, in each iteration, $|\mathcal{R}|$ is doubled in size and calls Algorithm 5 to find a new S, in the worst case, this algorithm stops when it meets condition line 19, which indicates that the maximum number of iterations is $\log\left((8+2\epsilon)n\dfrac{\ln\frac{2}{\delta}+\ln\binom{n}{k}}{\epsilon^2}\right)$. Consequently, the complexity of this method is $O\left(kT\log\left((8+2\epsilon)n\dfrac{\ln\frac{2}{\delta}+\ln\binom{n}{k}}{\epsilon^2}\right)\right)$. ∎

4.4.2.2 FBIM2 & FBIM3 - algorithms based on the online processing of influence maximization method

a. Algorithm description

Algorithm 7: FBIM2 algorithm

```
Input: Graph $G$, $\mathcal{C}^{'}$, $k$, $\epsilon$, $\delta$
Output: An $(\frac{1}{2}-\epsilon)$-optimal solution $S$, $|S| \leq k$ with probility at least $1-\delta$
1.  $\Gamma_{max} \leftarrow \frac{2n\left(\frac{1}{2}\sqrt{ln\frac{6}{\delta}}+\sqrt{\frac{1}{2}(ln\binom{n}{k}+ln\frac{6}{\delta})}\right)^2}{\epsilon^2 k}$
2.  $\Gamma_0 \leftarrow \Gamma_{max}.\epsilon^2 k/n$
3.  $\mathcal{R}_1 \leftarrow$ Generate $\Gamma_0$ random RR sets by RIS
4.  $\mathcal{R}_2 \leftarrow$ Generate $\Gamma_0$ random RR sets by RIS
5.  $i_{max} \leftarrow \lceil \log_2(\Gamma_{max}/\Gamma_0) \rceil$
6.  for $i \leftarrow 1$ to $i_{max}$ do
7.      $S \leftarrow \mathsf{FMC}(G, \mathcal{C}^{'}, k, \mathcal{R}_1)$
8.      $\delta_1 = \delta_2 = \delta/3i_{max}$
9.      compute $f_l(S)$ and $f_u(S^*)$ by (4.13) and (4.14)
10.     $\epsilon^{'} \leftarrow f_l(S)/f_u(S^*)$
11.     if $\epsilon^{'} \geq (\frac{1}{2}-\epsilon)$ or $i == i_{max}$ then
12.         return $S$.
13.     $\Gamma_0 \leftarrow 2\Gamma_0$ and generate $\Gamma_0$ new random RR sets for $\mathcal{R}_1, \mathcal{R}_2$.
```

Exactly, FBIM2 (Algorithm 7) and FBIM3 (Algorithm 9) algorithms are similar in the main idea and only different in the finding seed set S step. The main idea of these algorithms is to perform a finite loop that does the following processing: *(1) generate two set of* RR *sets, $\mathcal{R}_1$ and $\mathcal{R}_2$; (2) find a seed set S based on $\mathcal{R}_1$ so that S satisfies the fairness constraint; (3) compute $f_l(S)$ based on $\mathcal{R}_2$ and $f_u(S^*)$ based on $\mathcal{R}_1$. If the ratio $f_l(S)/f_u(S^*)$ satisfies the optimal solution approximation (for* FBIM2*, it is at least* $1/2-\epsilon$*, and for* FBIM3*, it is at least* $1/2-2\epsilon$*), the algorithm stops before the limit. Otherwise, the algorithm repeats the finding S with $\mathcal{R}_1$ and $\mathcal{R}_2$ doubled in size.*

As mentioned above, FBIM2 and FBIM3 are inherited from the OPIMC method of Tang *et al.* [15], so we have the expressions $f_l(S)$ and $f_u(S^{'})$ as follows.

$$f_l(S) = \left(\left(\sqrt{\mathsf{Cov}_{\mathcal{R}_2}(S) + \frac{2\ln(1/\delta_2)}{9}} - \sqrt{\frac{\ln(1/\delta_2)}{2}}\right)^2 - \frac{\ln(1/\delta_2)}{18}\right) \cdot \frac{n}{\Gamma_0} \tag{4.13}$$

$$f_u(S^*) = \left(\sqrt{\frac{\mathsf{Cov}_{\mathcal{R}_1}(S)}{1/2} + \frac{\ln(1/\delta_1)}{2}} + \sqrt{\frac{\ln(1/\delta_1)}{2}}\right)^2 \cdot \frac{n}{\Gamma_0} \tag{4.14}$$

For FBIM2, finding the seed set S is similar to FBIM1 (Algorithm 5), that is, select k elements in all C_i communities of $\mathcal{C}^{'}$ so that each element appears the most in communities and guarantees S extendable after adding to S, which is known as the mandatory condition of the fairness constraint.The detail of this algorithm is fully presented in Algorithm 7.

For FBIM3, finding the seed set S is improved by decreasing the search elements times for S. Instead of having to do k iterations as FBIM2 does, FBIM3 only requires at most $\frac{1}{\epsilon}\log(\frac{k}{\epsilon})$ iterations. The detail of this process is fully presented in the Threshold Greedy algorithm for Fairness Max Cover procedure (Algorithm 8). The main idea of this algorithm is to find elements s in C_j, $(\forall C_j \in \mathcal{C}')$ that make $S+s$ is extendable and the gain of coverage ratio of s when adding to S on $\mathcal{R}$ is maximum (line 4 and 5). This idea was based on the principle of a simple near-linear time algorithm for the problem of maximization of monotone submodular functions, which was studied by Badanidiyuru and Vondrák [92].

b. Theoretical analysis

Theorem 6 *Algorithm 7 has* $O\left(kT\log\frac{n}{\epsilon^2 k}\right)$ *complexity.*

Proof Algorithm 7 iterates generating two random RR sets, $\mathcal{R}_1$ and $\mathcal{R}_2$, the size of each set is Γ_0. Later, the algorithm finds a seed set S based on $\mathcal{R}_1$ through Algorithm 5. Next, it calculates the approximation solution ϵ' of S, with $\epsilon' = \dfrac{f_l(S)}{f_u(S^*)}$. This algorithm has two conditions to stop and return the result on line 11. Because, in each iteration, Γ_0 doubles in size and calls Algorithm 5 to find a new S, in the worst case, this algorithm stops when it has not met a S satisfying $\epsilon' \geq (1/2-\epsilon)$ and the number of iterations has reached i_{max}, with $i_{max} = \lceil\log(\Gamma_{max}/\Gamma_0)\rceil$ in line 5. We have:
$\Gamma_{max} = \dfrac{2n\left(\frac{1}{2}\sqrt{\ln\frac{6}{\delta}} + \sqrt{\frac{1}{2}(\ln\binom{n}{k} + \ln\frac{6}{\delta})}\right)^2}{\epsilon^2 k}$ (at line 1).
$\Gamma_0 = \Gamma_{max}.\epsilon^2 k/n$ (at line 2).
After calculating to reduce for i_{max}, we have $i_{max} = \lceil\log\frac{n}{\epsilon^2 k}\rceil$. As a result, this algorithm takes $O\left(kT\log\frac{n}{\epsilon^2 k}\right)$ complexity. ■

Theorem 7 *Algorithm 8 has* $O\left(\frac{T}{\epsilon}log\frac{k}{\epsilon}\right)$ *complexity.*

Proof Similar to Algorithm 4, in Algorithm 8, the outer "for" loop requires at most $\frac{1}{\epsilon}\log\frac{k}{\epsilon}$ iterations, and the inner "for" loop requires maximum T iterations. As a result, this algorithm takes $O\left(\frac{T}{\epsilon}\log\frac{k}{\epsilon}\right)$ complexity. ■

Theorem 8 *Algorithm 9 has* $O\left(\frac{T}{\epsilon}\log\frac{k}{\epsilon}\log\frac{n}{\epsilon^2 k}\right)$ *complexity.*

Proof Algorithm 9 operates similarly to Algorithm 7, only differing in the step of finding the set S (in line 7), that is, Algorithm 9 uses Algorithm 8. Therefore, we can prove likewise for Algorithm 7, Algorithm 9 takes $O\left(\frac{T}{\epsilon}\log\frac{k}{\epsilon}\log\frac{n}{\epsilon^2 k}\right)$. ■

Algorithm 8: Threshold Greedy algorithm for Fairness Max Cover (ThresholdGreedy($\mathcal{R}$, ϵ)) procedure

```
   Input: Graph $G$, $\mathcal{C}^{'}$, $k$, RR sets ($\mathcal{R}$), $\epsilon > 0$
   Output: An $(\frac{1}{2} - \epsilon)$-optimal solution, $S$, $|S| \leq k$
1. $S \leftarrow \emptyset$
2. $M \leftarrow \max_{u \in V} \mathsf{Cov}_{\mathcal{R}}(u)$
3. for $(t = M; t \geq \epsilon \frac{M}{k}; t \leftarrow (1-\epsilon)t)$ do
4.     for $s \in ({C_j}_{C_j \in \mathcal{C}'} \setminus S) | S + s$ is extendable do
5.         if $(\mathsf{Cov}_{\mathcal{R}}(S \cup s) - \mathsf{Cov}_{\mathcal{R}}(S)) \geq t$ then
6.             $S \leftarrow S \cup \{s\}$
7. return $S$.
```

Algorithm 9: FBIM3 algorithm

```
    Input: Graph $G$, $\mathcal{C}^{'}$, $k$, $\epsilon$, $\delta$
    Output: An $(\frac{1}{2} - 2\epsilon)$-optimal solution $S$, $|S| \leq k$ with probility at least $1 - \delta$
1.  $\Gamma_{max} \leftarrow \frac{2n\left(\frac{1}{2}\sqrt{ln\frac{6}{\delta}} + \sqrt{\frac{1}{2}\left(ln\binom{n}{k} + ln\frac{6}{\delta}\right)}\right)^2}{\epsilon^2 k}$
2.  $\Gamma_0 \leftarrow \Gamma_{max}\frac{\epsilon^2 k}{n}$
3.  $\mathcal{R}_1 \leftarrow$ Generate $\Gamma_0$ random RR sets by RIS
4.  $\mathcal{R}_2 \leftarrow$ Generate $\Gamma_0$ random RR sets by RIS
5.  $i_{max} \leftarrow \lceil \log_2(\Gamma_{max}/\Gamma_0) \rceil$
6.  for $i \leftarrow 1$ to $i_{max}$ do
7.      $S \leftarrow ThresholdGreedy(\mathcal{R}_1, \epsilon)$
8.      $\delta_1 = \delta_2 = \delta/3i_{max}$
9.      compute $f_l(S)$ and $f_u(S^*)$ by 4.13) and (4.14
10.     $\epsilon^{'} \leftarrow f_l(S)/f_u(S^*)$
11.     if $\epsilon^{'} \geq (1/2 - 2\epsilon)$ or $i == i_{max}$ then
12.         return $S$.
13.     $\Gamma_0 \leftarrow 2\Gamma_0$ and generate $\Gamma_0$ new random RR sets for $\mathcal{R}_1, \mathcal{R}_2$.
```

Table 4.2: Statistics of datasets.

Dataset	#Nodes	#Edges	#Communities	Avg. degree	Type
Epinions	131,828	841,372	6,359	13.4	Directed
Pokec	1,632,803	30,622,564	1,284	37.5	Directed
Live-journal	3,997,962	34,681,189	5,000	28.5	Directed
Orkut	3,072,441	117,185,083	4,745	76.3	Undirected

4.5 Experiment and Result Evaluation

We conducted some experiments on the FBIM problem in our work. This section describes the experiment process, including datasets, algorithms for comparison, parameter setting, discussion, and evaluation of the results.

4.5.1 Experiment setting

4.5.1.1 Datasets

For a comprehensive experiment, we choose four datasets on SNAP [64]. These datasets have medium to big, diverse numbers of edges, nodes, and communities. They include the Epinions social network (*Epinions*), the Pokec social network (*Pokec*), the LiveJournal social network and ground-truth communities (*Live-journal*), and the Orkut social network and ground-truth communities (*Orkut*). These datasets have been used commonly to find the seed set with Influence Maximization. Table 4.2 presents the information of the datasets.

4.5.1.2 Environment

We conducted our experiments on a Linux machine with Intel Xeon Gold 6154 (720) @ 3.700GHz CPUs and 3TB RAM. Our implementation is written in C/C++ language and compiled with g++ 11.

4.5.1.3 Algorithm Comparison

To the best of our knowledge, there is no existing algorithm that solves the problem of fairness influence maximization under constraints with a pair of lower and upper bound budgets for communities. Therefore, we experimented to compare FBIM1 to DSSA, three algorithms of FBIM to OPIMC because they are similar in implementation. As such, this study's experiment does four algorithms, including OPIMC, FBIM1, FBIM2, and FBIM3, with different sets of input parameters to analyze and evaluate the effectiveness of these algorithms. Briefly, we call our three proposed algorithms (FBIM1, FBIM2, and FBIM3) FBIM's algorithms. As a result, we perform two separate experiments as follows:

- **Experiment A**. To compare FBIM1 to DSSA on two datasets, (Epinions and Orkut) under the LT model.
- **Experiment B**. To compare FBIM's algorithms to OPIMC on four datasets (Epinions, Pokec, Live-journal, Orkut), under both information diffusion models (IC and LT). Although the implementation method of FBIM1 differs from FBIM2 and FBIM3, we want to evaluate their effect when experimenting on the same set of parameters and dataset. Therefore, FBIM1 is also present in this experiment.

As mentioned above, FBIM's algorithms obtain S and almost $|S| \simeq k$ so that $f(S)$ reaches maximum. Furthermore, our algorithms find S to satisfy the fairness constraint, while DSSA and OPIMC do not. Therefore, we compare factors such as influence $f(S)$, running time, memory usage, approximation ratio $\frac{f_l(S)}{f_u(S^*)}$, and coverage ratio S across the target communities. This ratio is the number of communities with elements selected into S that satisfy the constraints of the lower and upper bounds, compared with the original total number of communities (K).

4.5.1.4 Parameter Setting

We conducted two experiments, which are called Experiment A and Experiment B, Each experiment has two sub-experiments with the following sets of parameter settings.

1. **Experiment A. (FBIM1 & DSSA, under the LT model)**
   - **Experiment A1.** We set $|C'| = K$ with $K \in [50, 1000]$, $k \in [1000, 20000]$, the pair $(k_l^i; k_u^i)$ of C_i are $(0.3; 0.5)$, called FBIM-1.1 and $(0.1; 0.9)$, called FBIM-1.2.
   - **Experiment A2.** We used the same k as Experiment A1, but we did not set a specific K. The upper bound was fixed at 0.5 while the lower bound would be 0.1, 0.01, and 0.001 for FBIM-1.3, FBIM-1.4, and FBIM-1.5 , respectively. We will explain these changes in the experiment results section.
2. **Experiment B. (FBIM's algorithm & OPIMC, under the LT and IC models)**
   - **Experiment B1**. We set $k \in [1000, 10000]$, $|C'| = K$ with $K = 20\%k$, the pair $(k_i^l; k_i^u)$ of each C_i are $(0.3; 0.5)$, called $\langle AlgorithmName \rangle$ .1 (such as FBIM1.1, FBIM2.1, FBIM3.1); and $(k_i^l; k_i^u)$ equals $(0.1; 0.9)$, called $\langle AlgorithmName \rangle$ .2 (such as FBIM1.2, FBIM2.2, FBIM3.2).
   - **Experiment B2**. We use the same k as *Experiment B1* and $K = k$. In this case, K is no longer limited to the problem. Because it is already set to the maximum value, that is, in the ultimate case, if the number of communities covered is exactly K (for $K = k$), each community of K communities chooses precisely one element

for input S. For the rest parameters, the upper bound k_i^u is assigned a fixed value of 0.5 while the lower bound k_i^l varies by $0.1, 0.01$ and 0.001, respectively for $\langle AlgorithmName \rangle$ .3, $\langle AlgorithmName \rangle$ .4, and $\langle AlgorithmName \rangle$ .5. In the discussion of the experimental result, we will explain these changes.

Finally, we set the parameters $\epsilon = 0.1, \delta = \frac{1}{n}$ as the default setting.

4.5.2 Discussion and evaluation of experimental results

In this section, we first analyze two objective factors that affect the efficiency of FBIM's algorithms. Later, we discuss and evaluate the experimental results to clarify the strengths and weaknesses of the algorithms. The results are clearly shown in Figures 4.3 - 4.6.

4.5.2.1 Objective factors affect the efficiency of algorithms

As clarified in the theoretical analysis of the algorithms, our algorithms guarantee the theoretical optimization probability. However, in the experimental process, some objective factors affect the performance of the algorithms. They are the data preprocessing and the sampling generation by the RIS framework.

1. Data preprocessing

   (a) *Communities detection strategy.* We used the *Directed Louvain* method [93] to detect communities in dataset. This method applies the idea of simulating the Monte Carlo approach's randomness. It has been proven to be able to achieve a more promising result at extracting communities in case of a directed graph, but it still has its flaws. In short, our algorithms' efficiency is affected by the random factor during the communities detection.

   (b) *Communities selection strategy.* To ensure objectivity, selecting K communities for the input of the problem FBIM works by randomly choosing, as long as they have the probability of satisfying the condition for a valid result within the upper/lower bounds of the fairness constraint. The selection of the combination of communities is entirely random. This selection will be repeated many times if the previous combinations do not satisfy the conditions of the problem FBIM. In a nutshell, our algorithms are affected by the randomness factor in choosing the initial K communities. However, if we apply these algorithms to real problems, the communities selection could depend on a constraint. Thus, the user can improve in selecting elements of S on communities to achieve a better result.

2. RIS framework

   Moreover, FBIM's algorithms depend on the generation of sample graphs of the RIS framework. In the first few iterations of the algorithms, if the vertices intersection set's size of the sample graphs and the selected communities is empty or small, the algorithm must generate more sample graphs to increase this value (i.e., increase the number of vertex degrees). After this necessary condition is satisfied, the algorithm finds a seed set S that meets the fairness constraint. Therefore, our algorithms may take extra time to generate sample graphs. Briefly, our algorithms depend on the random factor in creating the sample graph and the number of sample graphs.

In summary, if the above factors are not good, the algorithm takes more time to run until it chooses the set S that meets the problem requirements. These weaknesses are heading in one of our improvement directions for future studies.

4.5.2.2 Experimental Result Evaluation

- **Experiment A.**

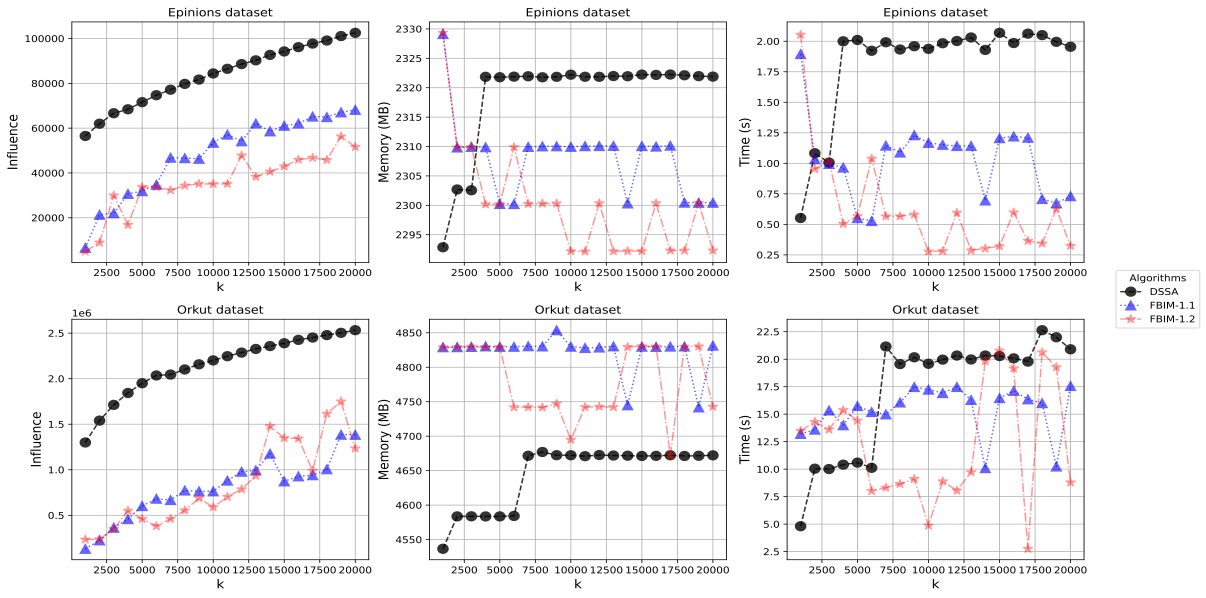


Figure 4.1: Influence, memory usage and runtime of the Experiment A1 under the LT model.

1. **Experiment A1.** For the this parameter setting, we want to show how FBIM1 perform in a general setting, which include a large range $[k_l^i, k_u^i]$ and a small K. The results in Figure 4.1 shows that FBIM1 almost runs faster than DSSA, even in some cases, FBIM1 is 2 to 4 times faster than DSSA. However, the influence $f(S)$ of FBIM is less than $f(S)$ of DSSA because of the small value of K. When K is small, we have fewer communities to cover. It leads to faster computation while still guaranteeing the fairness constrain. Nevertheless, this is a double-edged sword.

Because the K communities are randomly chosen, we have a smaller range to choose the seeds from, although we take less time to ensure fairness. Thus, it does not guarantee that the algorithm chooses nodes with the greatest influence according to the greedy strategy. Moreover, FBIM1 also takes more memory usage because the dataset contains large communities (as Orkut). This cause is understandable because FBIM1 must take an additional step to store and process communities information. In summary, in this case, although FBIM1 obtains S with an influence spread $f(S)$ 2 to 3 times smaller than $f(S)$ of DSSA, the running time of FBIM is faster, and one important thing is it solves fairness constraint.

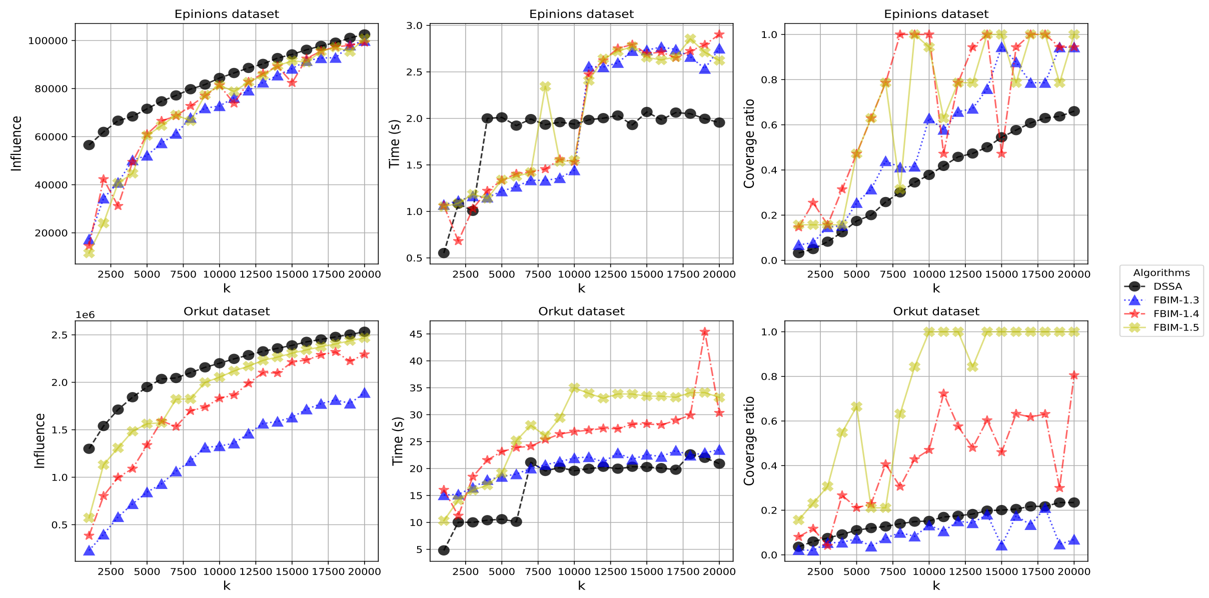


Figure 4.2: Running time, influence, and Coverage ratio of the Experiment A2 under the LT model.

2. **Experiment A2.** For the second parameter setting, we want to show the value of K to select a k-size seed set S, which can cover most of the target communities. Therefore, we do not set a specific value for K. In this case, Figure 4.2 shows that the lower bound impacted the results of FBIM1 strongly. We can see in Figure 4.2, the influence of FBIM-1.4 and FBIM-1.5 increase significantly compared to FBIM-1.3 (FBIM-1.3 has the same lower bound setting as FBIM1-2), even in some cases, they were equal to the influence of DSSA. The communities coverage ratio also changed drastically in FBIM-1.4 and FBIM-1.5. At times, it even reached 100%. On the other hand, the set S covers the desired communities, so the influenced individuals are the target ones we want to influence.

In summary, we can control the quality of the seed set of FBIM1 by holding the lower bound k_l^i and k. But the drawback is the long-running time. The cause is to find S, DSSA only needs to see the first k seeds that satisfy its requirements, while FBIM1 must find k seeds such that *(1) satisfy the exact requirements as DSSA's* and (2) *satisfy the requirements so that S is extendable.* The more communities for S to cover, the more extensive the list of nodes that needs computing to ensure the fairness constraint. For the convenience of the readers, we summarize the experimental results of the FBIM1 DSSA algorithm in Table 4.3.

Table 4.3: Statistical comparison of experimental results of FBIM1 and DSSA

Experiment A1	Runtime	2 to 4 times faster
	Memory	0.96 to 1.01 times less
	Influence	2 to 3 times less
Experiment A2	Runtime	0.5 to 1.75 times faster
	Influence	1 to 3 times less
	Coverage ratio	1.6 to 5 times greater (for Orkut, even FBIM1 can reach 100% while DSSA achieves 22%; for Epinion, even FBIM1 can reach 100% while DSSA achieves 63%)

- **Experiment B**

1. **Experiment B1.** For this setting case, the goal of the experiment is to evaluate the algorithm's performance by choosing two narrow and wide bound ranges along with the number of small communities. This setting simulates the general requirement when it is necessary to select a seed set from several specific finite communities. Experiment B1 will evaluate the algorithm's feasibility when performing compared to the algorithms in terms of runtime and resources. How disparate are they?
Hence, for Experiment B1, we want to show how the FBIM' algorithms perform in a general setting, including a large range $[k_i^l, k_i^u]$ and a small K. The results in the figures 4.3 and 4.4 show that most of the FBIM's algorithms run faster than OPIMC, even in some cases, the FBIM's algorithms are 4 to 12 times faster than OPIMC, especially when k increases. Because, for the essence of these algorithms, OPIMC uses the greedy strategy, which scans all elements on V to find S on the $\mathcal{R}$ set; meanwhile, FBIM2 and FBIM3 use an improved greedy approach, which scans all elements in communities of $\mathcal{C}'$ $(|\bigcup_{C_i \in \mathcal{C}'}(C_i)| \leq n)$. However, there are some cases in which the runtime of FBIM's algorithms is extended more than OPIMC since they are affected by the above objective factors. Intuitively, the instances of the FBIM's algorithm can run longer than OPIMC when k is small. Furthermore, FBIM1 sometimes runs longer than OPIMC because DSSA (the base method of

FBIM1) generates more sample graphs than OPIMC. This assertion was made clear by Tang *et al.* in [15].

Although the runtime of FBIM's algorithms is usually faster than OPIMC, the influence $f(S)$ of FBIM's algorithms is less than $f(S)$ of OPIMC. The disparity ranges from 1.5 to 10 times. The reason is the small value of K, that is, when K is small, we have fewer communities to cover. It leads to faster computation while still guaranteeing the fairness constraint. However, this is a double-edged sword. Because the K communities are randomly chosen, we have a smaller domain to choose the seeds, although we take less time to ensure fairness. Thus, it does not guarantee that the algorithm chooses nodes with the greatest influence according to the improved-greedy strategy.

2. **Experiment B2.** The goal of this experiment is to evaluate the theoretical potential of the algorithm with the requirement to cover as many communities as possible, so the K is not explicitly set. The lower bound changes, while the upper bound remains the same. The lower bound determines the minimum number of elements in the seed set that must be selected from each community to satisfy the extendable property of S. The upper bound is merely a constraint on the maximum number of elements selected from each community.

   Therefore, for this setting case, we want to show what is the possible value of K to select a k-size seed set S so that S can cover most of the target communities? Therefore, we do not set a specific value for K. In this case, the figures 4.5 and 4.6 show that the lower bound strongly affected the results of the algorithms FBIM. In these figures, the influence of $\langle AlgorithmName \rangle$ .4 and $\langle AlgorithmName \rangle$ .5 increased significantly compared to $\langle AlgorithmName \rangle$ .3 ($\langle AlgorithmName \rangle$ .3 and $\langle AlgorithmName \rangle$ .2 have the same lower bound setting). The coverage ratio of FBIM's algorithms is greater than OPIMC's ratio, from 2x to 10x. Even in some cases, the coverage ratio of OPIMC is only 1.5% (for Pokec), but the ratio of FBIM's algorithms can achieve $33.9\% - 43.9\%$. For the Orkut dataset, the most extensive dataset in four, FBIM1 achieves 100%. Furthermore, the value of the coverage ratio and the disparity of these ratios between FBIM's algorithms and OPIMC are proportional to k. Especially, because $\langle AlgorithmName \rangle$ .5 has the smallest lower bound in the lower bounds of the experiments, the algorithm's coverage ratios are better and the difference distance of the $f(S)$ value from OPIMC is shortened.

   On the other hand, the set S covers the desired communities, so the influenced individuals are the target ones we want to influence. In summary, we can control the quality of the seed set of FBIM's algorithms by holding the lower bound k_i^l and k. However, the drawback is the long runtime. The cause is to find S, OPIMC only needs to see the first k seeds that satisfy its requirements, while FBIM's algorithms must find k seeds such that *(1) satisfy the exact requirements as* OPIMC*'s* and *(2) satisfy the requirement so that S is extendable.* Therefore, the more communities S covers, the more vertex lists must be computed to ensure the fairness constraint. This assertion is clearly shown in the algorithms' experiments in both the LT and IC propagation models.

For the term of memory usage of these algorithms, in two parameters setting under both IC and LT models, the FBIM's algorithms almost take more memory usage because the dataset contains large communities (as Epinions Live-journal, Orkut). It is inevitable due to the FBIM's algorithms must take an additional step to store and process communities information. Briefly, although FBIM's algorithms obtain S with an influence spread $f(S)$ smaller than $f(S)$ of OPIMC, the running time of our algorithms is usually faster and the most important thing is that they solve the fairness constraint. For the convenience of the readers, we summarize the experimental results of the FBIM's algorithms and OPIMC algorithm in Table 4.4.

Table 4.4: Statistical comparison of experimental results of FBIMs' algorithms and OPIMC

Experiment B1	Runtime	4 to 12 times faster
	Memory	1 to 2.7 times greater
	Influence	1.5 to 10 times less
Experiment B2	Runtime	0.5 to 6 times faster
	Influence	2 to 6 times less
	Coverage ratio	1 to 5 times greater (for Orkut, even FBIM1 can reach 100% ; for Pokec, FBIMs can achieve from 33.9% to 43.9% while OPIMC only achieves 1.5%).

4.6 Concluding remarks

We studied and proposed three algorithms to resolve the FBIM problem. The main result of the study is that our approximation algorithms achieve a $(1/2-\epsilon)$-approximation to the optimum solution and require $O\left(kT\log\left((8+2\epsilon)n^{\frac{\ln\frac{2}{\delta}+\ln\binom{n}{k}}{\epsilon^2}}\right)\right)$, $O\left(kT\log\frac{n}{\epsilon^2 k}\right)$, and $O\left(\frac{T}{\epsilon}\log\frac{k}{\epsilon}\log\frac{n}{\epsilon^2 k}\right)$ complexities, respectively. We compare our algorithms with the state-of-the-art algorithms (DSSA and OPIMC) by conducting some experiments under both the information diffusion models, LT and IC. The experiments help to confirm our proven theoretical results. Concurrently, we have presented the algorithms' strengths and weaknesses in analyzing and evaluating experimental results. The results indicate that our algorithms are highly scalable, achieve results that satisfy theoretical assurance and approximation solutions, and are feasible and

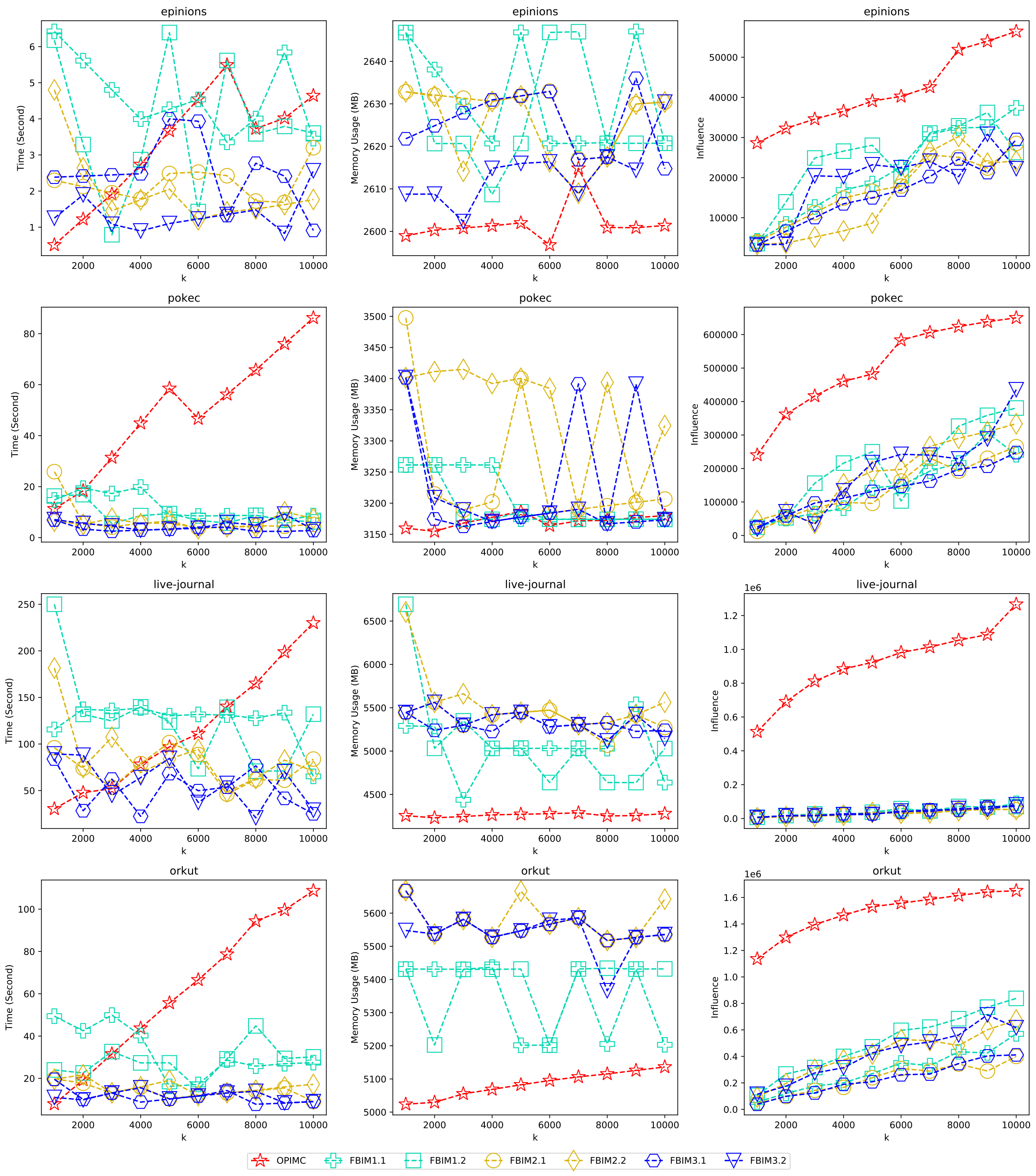


Figure 4.3: Running time, memory usage and influence of the Experiment B1 under the LT model.

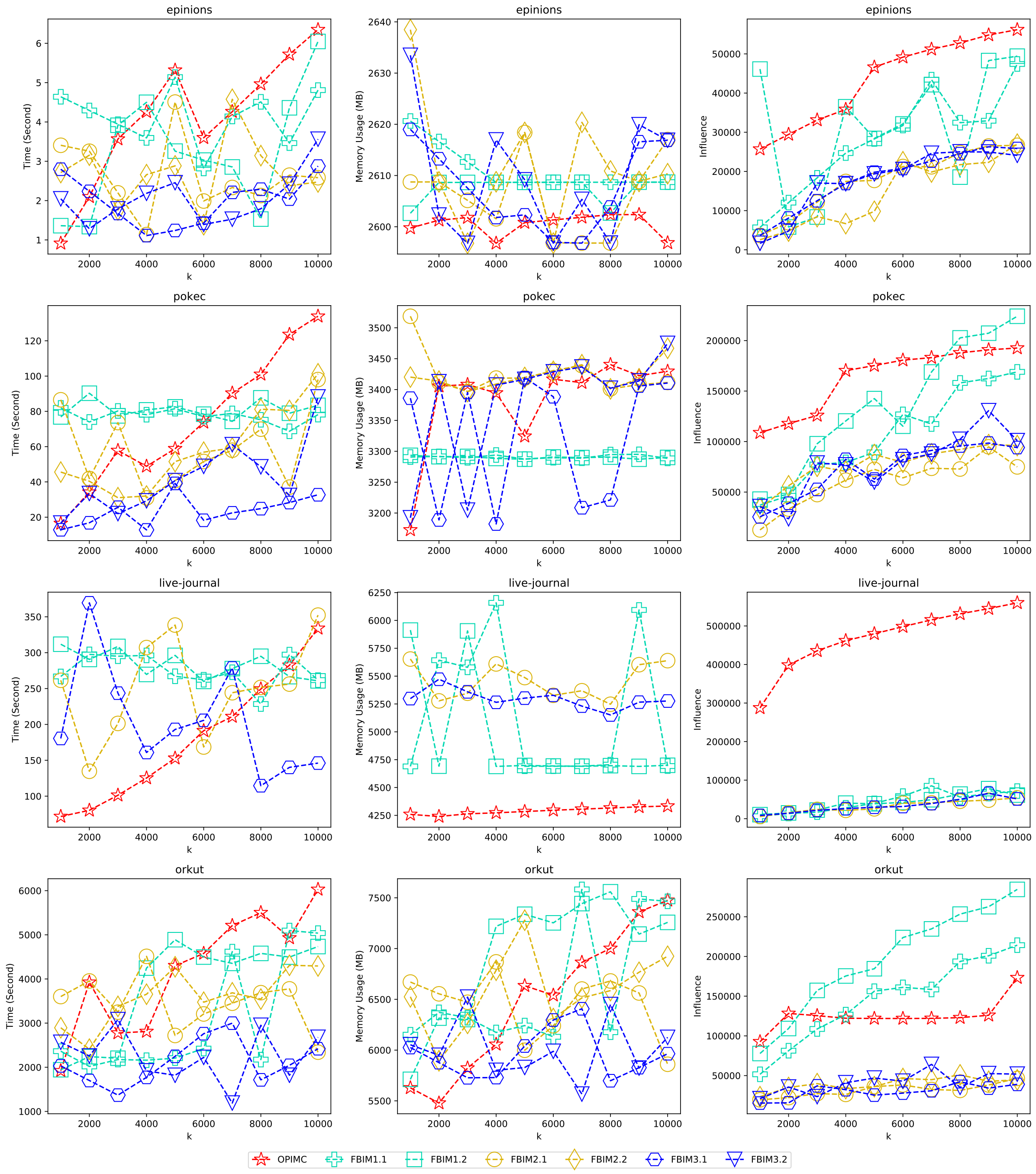


Figure 4.4: Running time, memory usage and influence of the Experiment B1 under the IC model.

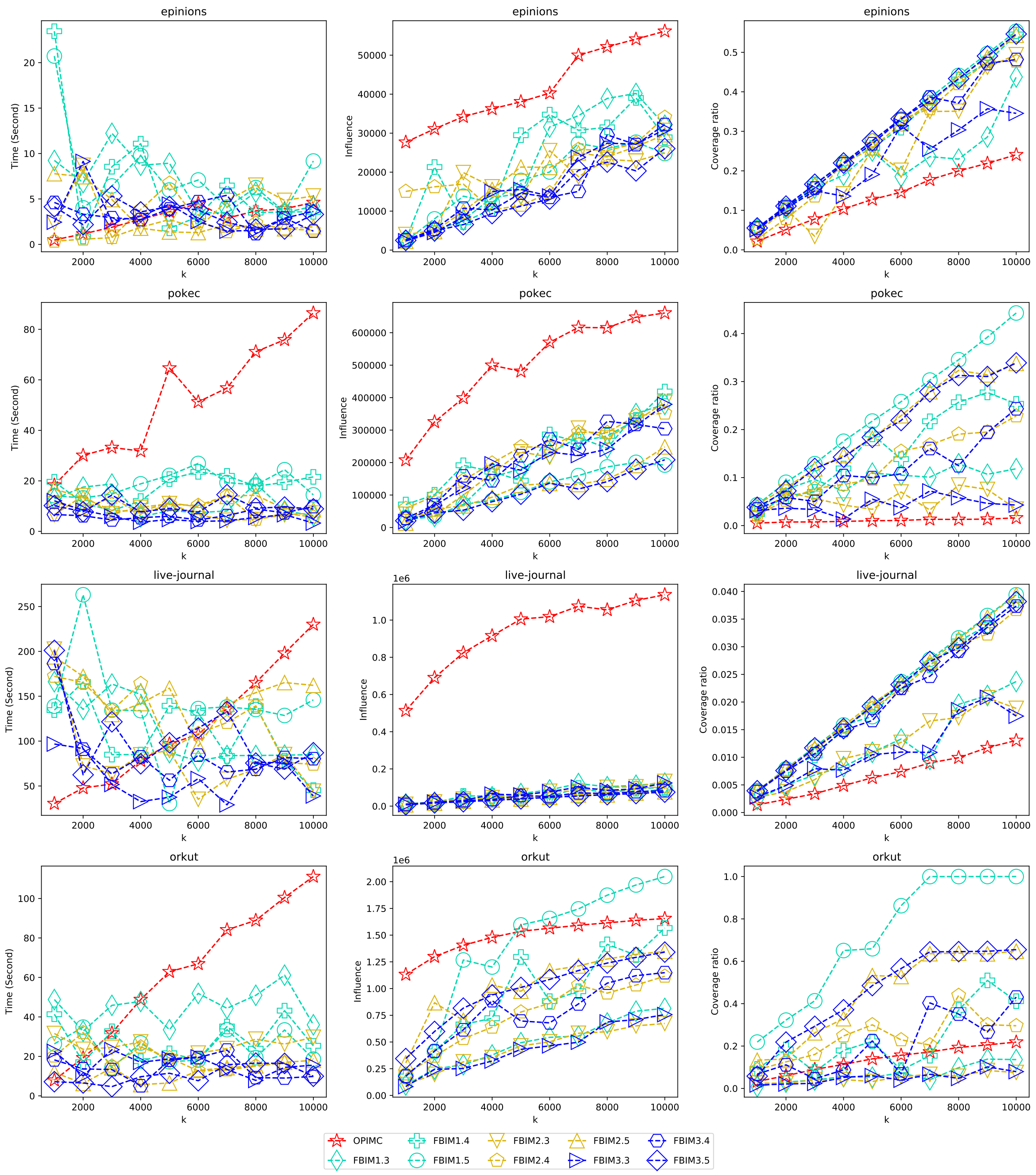


Figure 4.5: Running time, influence, and Coverage ratio of the Experiment B2 under the LT model.

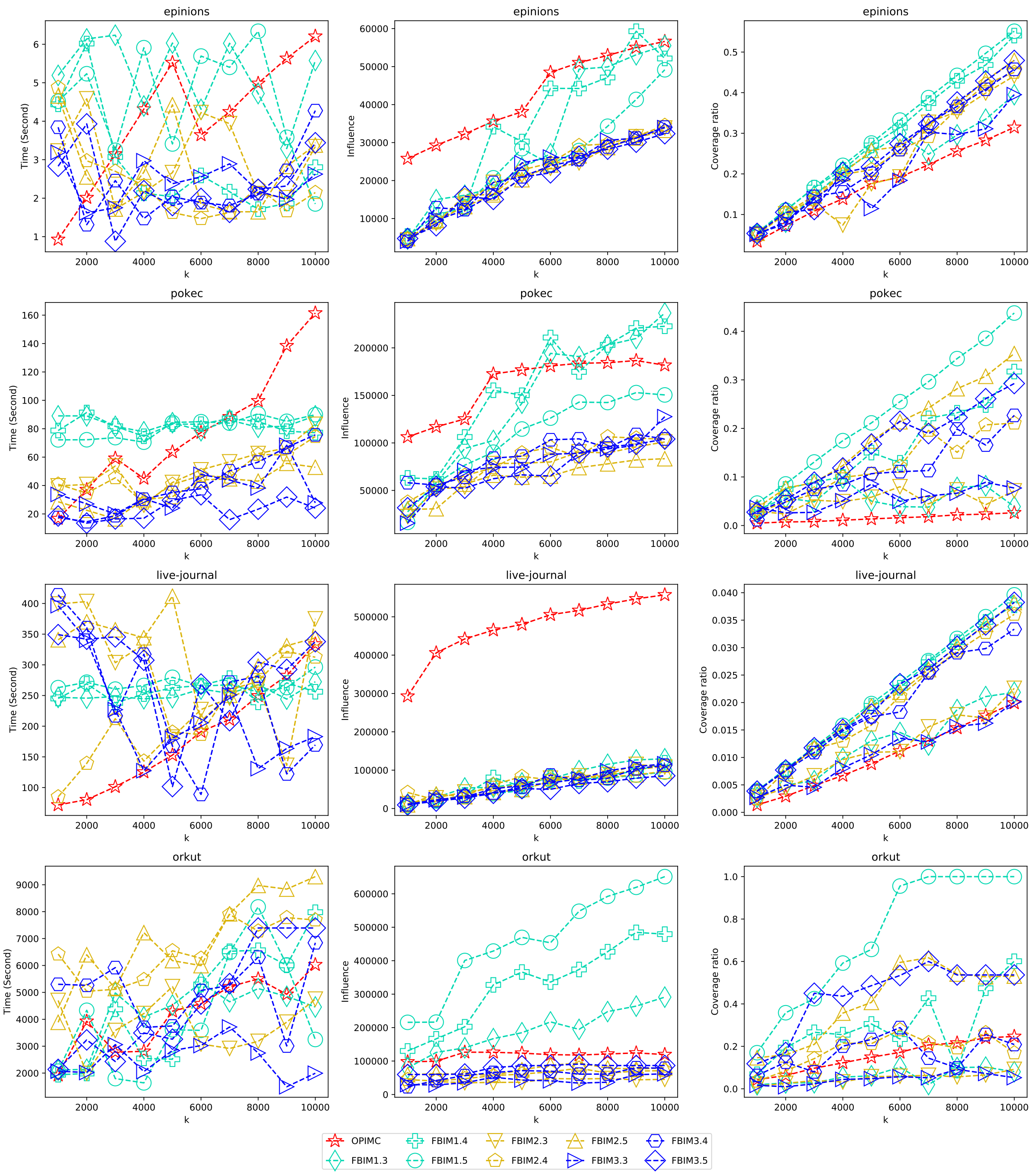


Figure 4.6: Running time, influence, and Coverage ratio of the Experiment B2 under the IC model.

effective even with big data. In future work, we plan to improve the objective factors that affect the efficiency of these algorithms to obtain a shorter runtime and a more significant influence spread.

Chapter 5

Problem 3. Maximizing DR-submodular function on the integer lattice

This chapter describes the problem of maximizing DR-submodular function on the integer lattice in detail, including the motivation to study the problem, problem definition, related work, proposed algorithms, experiment and result evaluation. Table 5.1 summarizes the usually used notations in this chapter.

5.1 Introduction

Submodular function maximization problems have recently received a lot of interest in the research community. A satisfactory explanation for this state is the prevalence of optimization problems related to submodular functions in many real-world applications as mentioned in Chapter 1.

Given a ground set E, a function $f : 2^E \to \mathbb{R}_{\geq 0}$ is submodular function. The *submodularity* of a monotone 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.

$$f(A \cup e) - f(A) \geq f(B \cup e) - f(B) \tag{5.1}$$

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

Most of the existing studies of this problem are in the area that considers submodular functions identified over a set-submodular functions. It means that the problem has the input is a subset of the ground set and returns an actual value. However, there are real-world situations where 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

Table 5.1: Table of the usually used notations in Chapter 5.

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 $\mathbf{x}$
v	an optimal value of the object function, $(1-\epsilon)\mathsf{OPT} \leq v \leq \mathsf{OPT}$ 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 $\mathbf{y}$
$(\mathbf{x} \wedge \mathbf{y})(e)$	$\mathbf{x} \vee \mathbf{y} := min\{\mathbf{x}(e), \mathbf{y}(e)\}$
$\mathbf{x} + \mathbf{y}$	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 $\mathbf{x}$
$f(\mathbf{x}\\|\mathbf{y})$	$f(\mathbf{x}\\|\mathbf{y}) = f(\mathbf{x} + \mathbf{y}) - f(\mathbf{y})$

functions over a multiset, under the name *submodular function on the integer lattice* [94]. 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 [95], document summarization and sensor placement [96], the submodular welfare problem [97], and maximization of the spread of influence with partial incentives [98]. The definition of such a submodular function is as follows:
A function $f : \mathbb{Z}_+^E \to \mathbb{R}$ is an *integer-lattice submodular* function if for all $\mathbf{x}, \mathbf{y} \in \mathbb{Z}_+^E$

$$f(\mathbf{x}) + f(\mathbf{y}) \geq f(\mathbf{x} \vee \mathbf{y}) + f(\mathbf{x} \wedge \mathbf{y}) \tag{5.2}$$

where $\mathbf{x} \wedge \mathbf{y}$ and $\mathbf{x} \vee \mathbf{y}$ denote the coordinate-wise min and max operations, respectively. A function $f : \mathbb{Z}_+^E \to \mathbb{R}$ is called *diminishing return submodular (DR-submodular)*, if for all $\mathbf{x}, \mathbf{y} \in \mathbb{Z}_+^E$ with $\mathbf{x} \leq \mathbf{y}$

$$f(\mathbf{x} + \chi_e) - f(\mathbf{x}) \geq f(\mathbf{y} + \chi_e) - f(\mathbf{y}) \tag{5.3}$$

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

The submodularity defined on the integer lattice differs from 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. [94]. 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 [95, 96]. Nevertheless, the diminishing return property of the DR-submodular function maximization problem often plays a fundamental role in some practical problems [95, 96, 97].

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 and streaming algorithms. Many 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. Understandably, the greedy method always scans many times on data to find the best. However, this cause makes its algorithms take a long runtime; even it cannot apply 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 [99].

Attracted by the usefulness of the maximizing DR-submodular function on 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 using a fast double greedy algorithm for maximization of the nonmonotone DR-submodular function [100], using a threshold greedy algorithm for maximization of the monotone DR-submodular constraint knapsack over an integer lattice [101], combining the threshold greedy algorithm with a partial element enumeration technique for maximization of the monotone DR submodular knapsack constraint over an integer lattice [94], using a streaming method to maximize DR-submodular functions with d-knapsack constraints [102], using an one-pass streaming algorithm for DR-submodular maximization with a knapsack constraint over the integer lattice [103], and using streaming algorithms for maximizing monotone DR-submodular function with a cardinality constraint on the integer lattice [104].

In this study, we focus on *the maximization of monotone DR-submodular function under cardinality constraint on the integer lattice* (the MDRSCa problem in Definition 20). In reviewing the literature, there are two novel methods for this problem. First, Soma *et al.* [94] 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.* [104] first devised a streaming algorithm based on Sieve streaming [105]. 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 [104], 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 investigate 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. Later, we provide two 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 $\mathsf{OPT} \in [m, 2km]$ where $m = \max_{e \in E}\{f(\chi_e)\}$. Based on the estimated OPT, the StrDRS1, an 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 a single-pass streaming algorithm (Algorithm 12), which has an 1/4 approximation solution. 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, CaDRS [94], SieveStr$++$ [104], StrDRS1, and StrDRS2, 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.

For the convenience of comparing the number of scans, the approximation ratio, and the complexity of the current state-of-the-art algorithms for MDRSCa problem, we summarize them in Table 5.2.

Table 5.2: State-of-the-art algorithms for MDRSCa problem in terms of the number of scans, approximation ratio, and 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	$O(\frac{1}{\epsilon})$	$1-1/e-\epsilon$	$O(\frac{n}{\epsilon}\log B)$

5.2 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.* [106] are pioneers in studying this problem 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 [107] develops an improved greedy algorithm, which achieves a $(1-1/e)$-approximation with $O(n^5)$ time complexity for a knapsack constraint. Subsequently, Calinescu *et al.* [108] first devised a $(1-1/e)$-approximation algorithm with 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.* [92] design a $(1-1/e-\epsilon)$-approximation algorithm with any fixed constraint $\epsilon > 0$, which takes $O(\frac{n}{\epsilon}\log\frac{n}{\epsilon})$ time complexity for the cardinality constraint.

Several studies have recently begun to investigate the maximization of DR submodular functions on the integer lattice under various constraints. For a knapsack constraint, Soma *et al.* [96] propose a $(1-1/e)$-approximation algorithm that takes a pseudo-polynomial time complexity. Next, Soma *et al.* [94] continue to give algorithms under a cardinality constraint, a knapsack constraint, and a polymatroid constraint on the integer lattice, respectively. These algorithms have polynomial time and achieve a $(1-1/e)$-approximation ratio. Subsequently, Gu *et al.* [100] study the problem of maximizing non-monotone DR-submodular function on the bounded integer lattice. They propose a fast double greedy algorithm that improves the runtime. Their result achieves a 1/2-approximation algorithm with a $O(n \log B)$ time complexity. Liu *et al.* [102] develop two streaming algorithms for this problem under the d-knapsack constraint. 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 memory complexity to $O(\frac{d}{\beta\epsilon})$ with an unchanged approximation ratio and query complexity. Most recently, Tan *et al.* [103] design an online algorithm for this problem with a knapsack constraint, called DynamicMRT, which achieves a $(1/3-\epsilon)$-approximation ratio, a memory complexity $O(K \log \frac{K}{\epsilon})$ and query complexity $O(\log^2 \frac{K}{\epsilon})$ per element for the knapsack constraint K. Meanwhile, Gong *et al.* [101] consider the knapsack constraint maximization problem of a non-negative monotone DR-submodular function f over a bounded integer lattice. They present a deterministic algorithm and theoretically reduce its runtime to a new record, $O((\frac{1}{\epsilon}^{O(1/\epsilon^5)}.n \log \frac{1}{c_{min}} \log B))$, (where $c_{min} = min_{e\in E} c(e)$ and $c(.)$ is a cost function defined in E) with the approximate ratio of $(1-1/e-O(\epsilon))$.

5.3 Problem definition

This section introduces the definition of the MDRSCa problem and its associated notations.

5.3.1 Notation

For a positive integer $k \in \mathbb{N}$, $[k]$ denotes the set $\{1, \ldots, k\}$. Given a ground set $E = \{e_1, \ldots, e_n\}$, 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 $supp^+(\mathbf{x}) = \{e \in E | \mathbf{x}(e) > 0\}$. 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 $\mathbf{x}, \mathbf{y} \in \mathbb{Z}_+^E$, $\mathbf{x} \le \mathbf{y}$ signifies $\forall e \in E$ then $\mathbf{x}(e) \le \mathbf{y}(e)$. 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 $\mathbf{x}-\mathbf{y} = \mathbf{x}+(-\mathbf{y})$.

5.3.2 Definition

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

Definition 19 (DR-submodular function) *A function $f : \mathbb{Z}_+^E \to \mathbb{R}_+$ is monotone if $f(\boldsymbol{x}) \le f(\boldsymbol{y})$ for all $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{Z}_+^E$ with $\boldsymbol{x} \le \boldsymbol{y}$ and f is said to be diminishing return submodular (DR-submodular), if*

$$f(\boldsymbol{x}+\chi_e) - f(\boldsymbol{x}) \ge f(\boldsymbol{y}+\chi_e) - f(\boldsymbol{y}) \tag{5.4}$$

Definition 20 (MDRSCa problem) *Let $\boldsymbol{B} \in \mathbb{Z}_+^E$, $B = \|\boldsymbol{B}\|_\infty$ and an integer $k > 0$, we consider the DR-submodular function under cardinality constraint as follows*

$$\textit{maximize: } f(\boldsymbol{x}) \tag{5.5}$$

$$\textit{subject to: } \boldsymbol{0} \le \boldsymbol{x} \le \boldsymbol{B}, \boldsymbol{x}(E) \le k \tag{5.6}$$

5.4 Proposed algorithms

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

5.4.1 Streaming algorithm with approximation ratio of $(1/2 - \epsilon)$

First, we propose a single-pass streaming algorithm, called StrOpt (in Algorithm 10), for the MDRSCa problem on the assumption that the optimal value of the objective function is known. Afterwards, we base on the traditional method to estimate the optimal value and devise the one-pass streaming algorithm called StrDRS1 (in Algorithm 11). This is the first main algorithm in our study for the MDRSCa problem.

5.4.1.1 Algorithm with knowing optimal value

We assume that the optimal value OPT of the objective function of MDRSCa is already known. Algorithm 10 is created to find vector $\mathbf{x}$ by utilizing this OPT. Given a known optimal value v that satisfies $(1-\epsilon)\mathsf{OPT} \le v \le \mathsf{OPT}$ 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 $f(k_e|\mathbf{x})/k_e < \frac{v}{2k}$. 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.

Lemma 3, Theorem 9, and their proofs demonstrate the theoretical solution guarantee of Algorithm 10. On the basis of that, we devise the first main streaming algorithm for the MDRSCa problem.

Algorithm 10: StrOpt$(f, \mathbf{B}, k, \epsilon, v)$

```
    Input: $f : \mathbb{Z}_+^E \to \mathbb{R}_+$, $\mathbf{B}$, $k$, $\epsilon$, a guess of optimal value $v$
    Output: A vector $\mathbf{x}$
1:  $\mathbf{x} \leftarrow \mathbf{0}$
2:  foreach $e \in E$ do
3:      $I \leftarrow \{i_1, i_2, \ldots, i_{|I|}\} : i_1 < i_2 \ldots < i_{|I|}\} \leftarrow \{\lceil \mathbf{B}(e)(1-\epsilon)^i \rceil : i \in \mathbb{Z}, 1 \leq \mathbf{B}(e)(1-\epsilon)^i \leq \mathbf{B}(e)\}$
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 $\mathbf{x}$
```

Lemma 3 *We have* $f(k_e\chi_e|\boldsymbol{x}) \geq (1-\epsilon)k_e\frac{v}{2k}$

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.

$$f(k_e\chi_e|\mathbf{x}) \geq f(i_{j-1}\chi_e|\mathbf{x}) \geq i_{j-1}\frac{v}{2k} \geq (1-\epsilon)k_e\frac{v}{2k} \tag{5.7}$$

The proof is complete. ■

Theorem 9 *For any $\epsilon \in (0, \frac{1}{2})$ and $(1-\epsilon)\mathsf{OPT} \leq v \leq \mathsf{OPT}$, the Algorithm 10 takes $O(n \log(\frac{1}{\epsilon} \log B))$ queries and returns a solution $\boldsymbol{x}$ satisfying $f(\boldsymbol{x}) \geq (1-\epsilon)v/2$.*

Proof The Algorithm 10 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 $k_i \chi_{e_i}$ are the solution at the beginning of the iteration i and the additional vector into current solution at iteration i, respectively. We consider two following cases:
Case 1. If $\|\mathbf{x}\|_1 = k$, we have $k_1 + k_2 + \ldots + k_n = k$ thus:

$$f(\mathbf{x}) = \sum_{i=1}^{n} f(k_i \chi_{e_i} | \mathbf{x}_i) \geq \sum_{i=1}^{n} (1-\epsilon) k_i \frac{v}{2k} = \frac{(1-\epsilon)v}{2} \tag{5.8}$$

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 on 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 $f(\mathbf{x}) \geq \mathsf{OPT} - v/2 \geq v/2$. The proof is complete. ■

5.4.1.2 $(1/2 - \epsilon)$-approximation streaming algorithm - StrDRS1 algorithm

Based on the analysis of Algorithm 10, and the working frame of the Sieve streaming algorithm [109], we design the StrDRS1 algorithm (in Algorithm 11) 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.

Algorithm 11: 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:     $I \leftarrow \{i_1, i_2, \ldots, i_{|I|}\} : i_1 < i_2 \ldots < i_{|I|}\} \leftarrow \{\lceil \mathbf{B}(e)(1-\epsilon)^i \rceil : i \in \mathbb{Z}, 1 \leq$
       $\mathbf{B}(e)(1-\epsilon)^i \leq \mathbf{B}(e)\}$
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 $\arg\max_{\mathbf{x}^v, v \in O} f(\mathbf{x}^v)$
```

Theorem 10 *Algorithm 11 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

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

By applying the proof of Theorem 9, and the working frame of the Sieve streaming algorithm in [109], we obtain:

$$f(\mathbf{x}^v) \geq \frac{(1-\epsilon)}{2} v \geq \frac{(1-\epsilon)^2}{2}\mathsf{OPT} \geq (\frac{1}{2} - \epsilon)\mathsf{OPT} \tag{5.9}$$

The proof is complete. ■

5.4.2 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 (StrDRS2 algorithm) in our study.

5.4.2.1 (1/4)-approximation streaming algorithm (Stepping-Stone algorithm)

We design the Stepping-Stone algorithm (in Algorithm 12), which is a (1/4)-approximation streaming algorithm. 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 (Algorithm 11), that is, Stepping-Stone algorithm (Algorithm 12) 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 Algorithm 11, 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 $\|\mathbf{x}\|_1 = k$.

Lemma 4, 5, 6, and Theorem 11 clarify the theoretical analysis of Algorithm 12.

Algorithm 12: 1/4-approximation algorithm (Stepping-Stone algorithm)

```
Input: $f : \mathbb{Z}_+^E \to \mathbb{R}_+$, $\mathbf{B}$, $k$, $\epsilon$
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 $k_e \leftarrow \arg\max\{i_t - 1 : i_t \in I, f(\chi_e|\mathbf{x} + (i_t - 1)\chi_e) < f(\mathbf{x} + i_{t-1}\chi_e)/k$ 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 $\mathbf{x}$
```

Lemma 4 *After each iteration of the Algorithm 12, we have* $f(k_e\chi_e|\boldsymbol{x}) \geq (1-\epsilon)k_e f(\boldsymbol{x})/k$

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 4, 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

$$f(i_j|\mathbf{x} + i_{j-1}\chi_e) = \sum_{l=i_{j-1}+1}^{i_j} f(\chi_e|\mathbf{x} + i_{j-1}\chi_e) \geq \sum_{l=i_{j-1}+1}^{i_j} f(\mathbf{x} + i_{j-1}\chi_e)/k \tag{5.10}$$

$$\geq (i_j - i_{j-1}) f(\mathbf{x} + i_{j-1}\chi_e)/k \tag{5.11}$$

Therefore:

$$f(k_e\chi_e|\mathbf{x}) \geq f(i_{t-1}\cdot\chi_e|\mathbf{x}) \tag{5.12}$$
$$\geq \sum_{j=1}^{t-1}(i_j - i_{j-1})f(\chi_e|\mathbf{x}+i_{j-1}\cdot\chi_e) \tag{5.13}$$
$$= i_{t-1}f(\mathbf{x})/k \geq (1-\epsilon)k_e f(\mathbf{x})/k \tag{5.14}$$

The proof is completed. ∎

Lemma 5 *After the main loop of Algorithm 12, we have* $2f(\boldsymbol{x}) \geq \mathsf{OPT}$

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

$$f(\mathbf{o}) - f(\mathbf{x}) = f(\mathbf{o}\vee\mathbf{x}) - f(\mathbf{x}) \tag{5.15}$$
$$= \sum_{e\in\{\mathbf{o}\vee\mathbf{x}-\mathbf{x}\}} f(\chi_e|\mathbf{x}) \tag{5.16}$$
$$= \sum_{e\in\{\mathbf{o}-\mathbf{o}\wedge\mathbf{x}\}} f(\chi_e|\mathbf{x}) \tag{5.17}$$
$$\leq \sum_{e\in\{\mathbf{o}-\mathbf{o}\wedge\mathbf{x}\}} f(\chi_e|\mathbf{x}_{(e)}) \tag{5.18}$$
$$< \sum_{e\in\{\mathbf{o}-\mathbf{o}\wedge\mathbf{x}\}} \frac{f(\mathbf{x}_{(e)})}{k} \leq f(\mathbf{x}) \tag{5.19}$$

which implies the proof. ∎

Lemma 6 *At the end of the Algorithm 12, we have* $f(\boldsymbol{x}') \geq \frac{1-3\epsilon}{2-3\epsilon}f(\boldsymbol{x})$.

Proof If $\|\mathbf{x}'\|_1 < k$, then $\mathbf{x}' = \mathbf{x}$ and Lemma 6 holds. We consider the case $\|\mathbf{x}'\|_1 = k$. Assume that $supp(\mathbf{x}) = \{e_1, e_2, \ldots, e_l\}$, $\mathbf{x}_i = \sum_{j=1}^{i}\mathbf{x}(e_j)\cdot\chi_{e_j}$, $supp(\mathbf{x}') = \{e_p, e_{p+1}, \ldots, e_l\}$ 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 $k = \sum_{i=p}^{l} k_{e_p}$ and

$$f(\mathbf{x}) - f(\mathbf{x}^1) = \sum_{i=p}^{l} f(k_{e_i}\chi_{e_i}|\mathbf{x}_i) \geq \sum_{i=p}^{l}(1-\epsilon)k_{e_i}\frac{f(\mathbf{x}_i)}{k} \quad \text{(Lemma 4)} \tag{5.20}$$
$$\geq (1-\epsilon)\sum_{i=p}^{l}k_{e_i}\frac{f(\mathbf{x}^1)}{k} = (1-\epsilon)f(\mathbf{x}^1) \tag{5.21}$$

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 $k = q + \sum_{i=p+1}^{l} k_{e_p}$ and $i_{j-1} < \mathbf{x}_1(e_p) \leq c = i_j$. Similarly to the proof of Lemma 4, 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 $\mathbf{x}^1_l = \mathbf{x}^1 + l\chi_{e_p}$, then

$$f(q\chi_e|\mathbf{x}^1) \geq \sum_{l=c+1}^{i_{t-1}} \frac{f(\chi_{e_p}|\mathbf{x}^1 + (l-1)k_{e_p})}{k} \tag{5.22}$$

$$\geq (k_{e_p}(1-\epsilon) - c)\frac{f(\mathbf{x}^1_c)}{k} = (k_{e_p}(1-\epsilon) - (\mathbf{x}_1(e_p) + \epsilon k_{e_p}))\frac{f(\mathbf{x}^1)}{k} \tag{5.23}$$

$$\geq (q - 2\epsilon k_{e_p})\frac{f(\mathbf{x}^1)}{k} \tag{5.24}$$

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

$$f(\mathbf{x}) - f(\mathbf{x}^1) = f(q\chi_e|\mathbf{x}^1) + \sum_{i=p+1}^{l} f(k_{e_i}\chi_{e_i}|\mathbf{x}_i) \tag{5.25}$$

$$\geq (q - 2\epsilon k_{e_p})\frac{f(\mathbf{x}^1)}{k} + \sum_{i=p+1}^{l}(1-\epsilon)k_{e_i}\frac{f(\mathbf{x}_i)}{k} \quad \text{(Lemma 4)} \tag{5.26}$$

$$\geq (q + \sum_{i=p+1}^{l} k_{e_i} - 2\epsilon k_{e_p} - \epsilon\sum_{i=p+1}^{l} k_{e_i})\frac{f(\mathbf{x}^1)}{k} \tag{5.27}$$

$$\geq (k - 3\epsilon k)\frac{f(\mathbf{x}^1)}{k} = (1-3\epsilon)f(\mathbf{x}^1) \tag{5.28}$$

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. ■

Theorem 11 *Algorithm 12 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 scan only one time over the ground set E and each element e, it calculate $f(\chi_e|\mathbf{x} + (i_j - 1)\chi_e)$ 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 approximation ratio, by using Lemmas 5 and (6), we have:

$$f(\mathbf{x}') \geq \frac{1-3\epsilon}{2-3\epsilon}f(\mathbf{x}) \geq \frac{1-3\epsilon}{2(2-3\epsilon)}\mathsf{OPT} \geq (\frac{1}{4} - \frac{3}{4}\epsilon)\mathsf{OPT} \tag{5.29}$$

The proof is completed. ■

5.4.2.2 $(1-1/e-\epsilon)$-approximation streaming algorithm - StrDRS2 algorithm

StrDRS2 (in Algorithm 13) is a multi-pass streaming algorithm and is based on the output of Algorithm 12 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 $i, i \in \{1, 2, \ldots, \mathbf{B}(e)\}$, so that $f(i\chi_e|\mathbf{x})/i < \theta$. The threshold θ decreases $(1-\epsilon)$ times after each iteration.

Lemma 7 and Theorem 12 clearly demonstrate the theoretical solution-ability guarantee of Algorithm 13.

Algorithm 13: $(1 - 1/e - \epsilon)$-approximation algorithm (StrDRS2)

```
   Input: $f : \mathbb{Z}_+^E \to \mathbb{R}_+$, $\mathbf{B}$, $k$, $\epsilon$
   Output: A vector $\mathbf{x}$
1: $\mathbf{x}_0 \leftarrow$ Result of Stepping-Stone algorithm, $\Gamma \leftarrow f(\mathbf{x}_0)$
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 $k_e \leftarrow \arg\min\{i-1 : i \in \{1, 2, \ldots, \mathbf{B}(e)\}, f(i\chi_e|\mathbf{x})/i < \theta\}$ 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 $\mathbf{x}$
```

Lemma 7 *In Algorithm 13, at any iteration of the outer loop, we have:*

$$f(k'_e\chi_e|\boldsymbol{x}) \geq \frac{(1-\epsilon)k_e}{k}(\mathsf{OPT} - f(\boldsymbol{x})) \tag{5.30}$$

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

$$f(\mathbf{o}) - f(\mathbf{x}) = k\frac{\mathsf{OPT}}{k} \leq k\frac{(4-3\epsilon)\Gamma}{(1-3\epsilon)k} < \frac{k\theta}{1-\epsilon} \leq \frac{f(k'_e\chi_e|\mathbf{x}_i)}{(1-\epsilon)k'_e} \tag{5.31}$$

Thus, $f(k'_e\chi_e|\mathbf{x}) \geq \frac{(1-\epsilon)k'_e}{k}(\mathsf{OPT} - f(\mathbf{x}))$, Lemma 7 holds. 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

$$f(\mathbf{o}) - f(\mathbf{x}_i) \leq f(\mathbf{o} \vee \mathbf{x}) - f(\mathbf{x}) \tag{5.32}$$
$$= \sum_{e \in \{\mathbf{o} \vee \mathbf{x} - \mathbf{x}\}} f(\chi_e|\mathbf{x}) \tag{5.33}$$
$$= \sum_{e \in \{\mathbf{o} - \mathbf{o} \wedge \mathbf{x}\}} f(\chi_e|\mathbf{x}) \tag{5.34}$$
$$\leq k\frac{\theta}{1-\epsilon} \leq \frac{f(k'_e \chi_e|\mathbf{x}_i)}{(1-\epsilon)k'_e} \tag{5.35}$$

The proof is complete. ■

Theorem 12 *The Algorithm 13 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:

$$f(\mathbf{o}) - f(\mathbf{x}) \leq f(\mathbf{o} \vee \mathbf{x}) - f(\mathbf{x}) \tag{5.36}$$
$$= \sum_{e \in \{\mathbf{o} \vee \mathbf{x} - \mathbf{x}\}} f(\chi_e|\mathbf{x}) \tag{5.37}$$
$$= \sum_{e \in \{\mathbf{o} - \mathbf{o} \wedge \mathbf{x}\}} f(\chi_e|\mathbf{x}) \tag{5.38}$$
$$\leq k\theta_{min} \leq k(1-\epsilon)\frac{\Gamma}{4k} \leq (1-\epsilon)\frac{\mathsf{OPT}}{4} \tag{5.39}$$

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

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 i-th updated and the final solution $\mathbf{x} = x_l$, Lemma 7 gives

$$f(\mathbf{x}_{i+1}) - f(\mathbf{x}_i) = f(k'_{e_{i+1}}\chi_{e_{i+1}}|\mathbf{x}_i) \geq \frac{(1-\epsilon)k'_{e_{i+1}}}{k}(\mathsf{OPT} - f(\mathbf{x}_i)) \tag{5.40}$$

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

$$\mathsf{OPT} - f(\mathbf{x}_l) \leq (1 - \frac{(1-\epsilon)k'_{e_l}}{k})(\mathsf{OPT} - f(\mathbf{x}_{l-1})) \tag{5.41}$$

$$\leq e^{-\frac{(1-\epsilon)k'_{e_l}}{k}}(\mathsf{OPT} - f(\mathbf{x}_{l-1})) \tag{5.42}$$

$$\ldots \leq e^{-\sum_{j=1}^{l} \frac{(1-\epsilon)k'_{e_j}}{k}}\mathsf{OPT} \tag{5.43}$$

$$e^{-(1-\epsilon)}\mathsf{OPT} \leq (\frac{1}{e} + \epsilon)\mathsf{OPT} \tag{5.44}$$

Hence $f(\mathbf{x}) \geq (1 - 1/e - \epsilon)\mathsf{OPT}$, the proof is completed. ■

5.5 Experiment and Result Evaluation

We conducted experiments based on the *budget allocation problem over the bipartite influence model* [110]. This problem is an instance of the *monotone submodular function maximization problem over integer lattice under a constraint* [7]. As mentioned above, we consider the problem under a cardinality constraint.

Suppose we consider the context of the algorithmic marketing approach. The budget allocation problem can be explained as follows. In marketing strategy, one of the crucial choices is deciding how much of a given budget to spend on different media, including TV, 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 bi-partition $(V_1;V_2)$ 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 budget of the ad source corresponding to v_1. Each edge $v_1v_2 \in E$ is associated with a probability $p(v_1v_2) \in [0;1]$, which means that putting an advertisement to a slot of v_1 activates customer v_2 with probability $p(v_1v_2)$. Each source node v_1 will be allocated a budget $\mathbf{x}(v_1) \in \{0, 1, ..., B_{v_1}\}$ 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 $\mathbf{x}$[7], is defined as follows.

$$f : \mathbb{Z}_+^V \to \mathbb{R}_+ \text{ as } f(\mathbf{x}) = \sum_{v_2 \in V_2} \left(1 - \prod_{v_1v_2 \in E} (1 - p(v_1v_2))^{\mathbf{x}(v_1)} \right) \tag{5.45}$$

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

Table 5.3: Statistics of datasets. All datasets have type of bipartite, and undirected.

Dataset	**#Nodes**	**#Edges**	**Node meaning ($n_1; n_2$)**	**Edge meaning**
FilmTrust	3,579	35,494	(user, film) (1,508;2,071)	rating
NIPS	13,875	1,932,365	(doc, word) (1,500;12,375)	occurrence

5.5.1 Experimental Setting

5.5.1.1 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* from KONECT[1] project [111]: the network of the *FilmTrust ratings* project and the *NIPS* is doc-word dataset of NIPS full papers. The weighted of the rating datasets is 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 5.3.

5.5.1.2 Environment

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

5.5.1.3 Parameter Setting

We set the parameters as follows: $\epsilon = 0.1$, $B = 5$ for all experiments. Because FilmTrust has a small set of nodes, so $k \in \{60, 70, 80, 90, 100\}$. 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 datasets, which refers to the probability $p(v_1 v_2)$. For FilmTrust, the edge-weighted is the ratio of the rated value and the maximum rated value $(\frac{rated\ value}{maximum\ rated\ value})$. While, it is the ratio of the number of the word's occurrences in the document and the number of words in the document $(\frac{number\ of\ the\ word's\ occurrences\ in\ the\ document}{number\ of\ words\ in\ the\ document})$ for NIPS.

5.5.2 Experimental Results

This section discusses the experimental results to clarify the algorithms' benefits and drawbacks 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*

[1]http://konect.cc

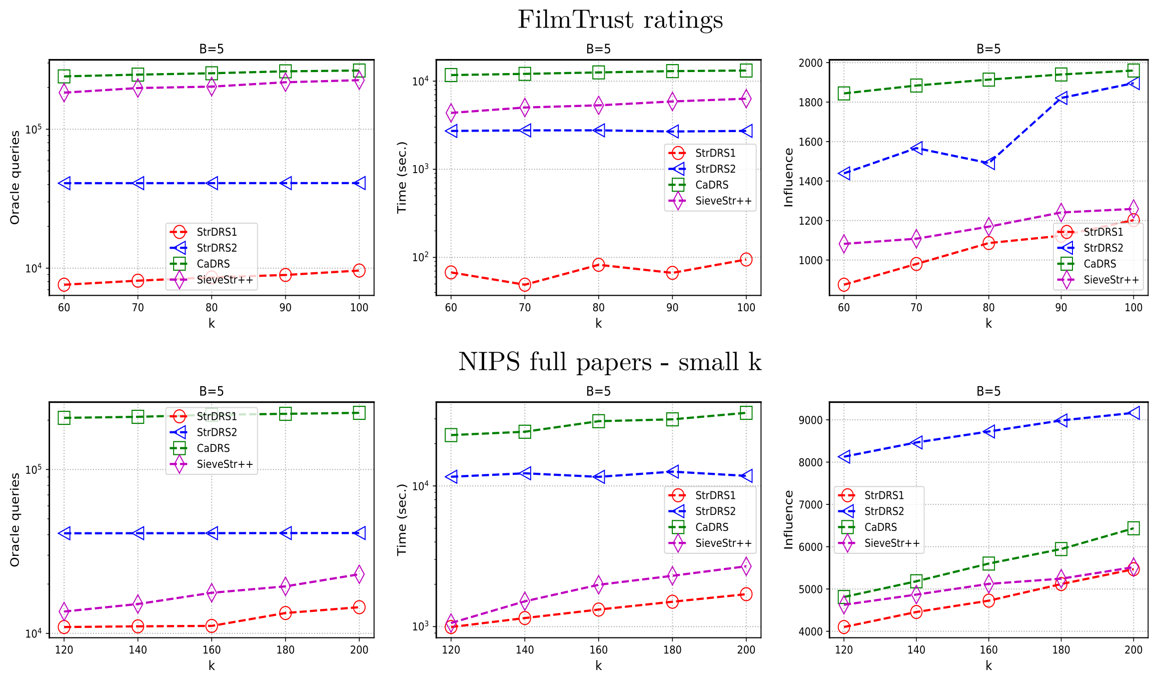


Figure 5.1: The results of the experimental comparison of algorithms on the datasets.

to or greater than that of SieveStr ++ *and* CaDRS *if we suitably set parameters B and k for the dataset.* Figure 5.1 clearly shows the achievement results.

5.5.2.1 Oracle queries and Runtime

Because most of the execution time of the algorithms is consumed by the number of queries for computing 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 when compared with the other algorithms. This cause makes it possible for us to mistake them for constants when looking at the charts. Table 5.4 clearly shows the variation of the number of queries.

5.5.2.2 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

Table 5.4: Statistics of the number of queries

k	StrDRS1	StrDRS2	CaDRS	SieveStr ++
FilmTrust rating				
60	7601	**40895**	240202	183453
70	8126	**40917**	247335	198063
80	8582	**40939**	252978	202716
90	8933	**40957**	261258	217947
100	9613	**40979**	264730	225569
NIPS full papers				
120	10934	**40807**	206599	13595
140	11040	**40847**	209606	15106
160	11108	**40887**	215624	17690
180	13313	**40927**	218639	19362
200	14446	**40963**	221672	22958

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. In contrast, the remaining three algorithms use streaming techniques, so it is difficult to achieve the same influence as CaDRS's. Nevertheless, 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.

5.6 Concluding remarks

We studies the maximization of monotone DR-submodular functions with a cardinality constraint on the integer lattice. We proposed 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 polymatroid constraint and knapsack constraint. In another direction, we consider the maximization of the nonmonotone DR-submodular function under cardinality constraint.

Chapter 6

Conclusions

6.1 Summary

This study contributes to clarifying the essential role of submodular function optimization in combinatorial optimization problems, which has many applications in economic, social network analysis, viral marketing, machine learning, game theorem, and other fields. In other words, this thesis proposes efficient approaches to solve three variants of the submodular optimization problem, including the problem of minimizing cost submodular cover under noises (Problem 1), the fairness budget distribution for the influence maximization problem (Problem 2), and the problem of maximizing DR-submodular function on the integer lattice (Problem 3). At the same time, we perform experiments on these algorithms to investigate their efficiency. The experimental results show that our algorithms work and outperform the state-of-the-art methods in runtime and number of queries. Our algorithms especially perform even for big data. Specifically, our contributions are as follows.

- **Problem 1.** We propose approaches that use the streaming strategy to minimize cost submodular cover under noise models (Str-SCN-M, Str-SCN-A). These approaches compute object function values near the optimal solution and satisfy the threshold constraint, but they spend a low cost compared to the state-of-the-art. This work was recognized through publications at the RIVF2021 international conference (Scopus) [**Publication 1**] and the Computer Standards and Interfaces journal (Elsevier, SCIE) [**Publication 8**].

- **Problem 2**. We combine the threshold greedy algorithm, the dynamic stop-and-start technique, and the RIS framework to solve the problem of maximizing a submodular function under fairness constraints (FBIM1, FBIM2, and FBIM3). The obtained results are guaranteed to have a good approximation solution through theoretical analysis and experimental results. Significantly, the seed set's distribution guarantees a high coverage ratio, which is an expression of ensuring the fairness constraint. This contribution is recognized in publications at the ICABDE2021 international conference (Scopus) [**Publication 2**] and the Mathematics journal (MDPI, SCIE) [**Publication 9**].

- **Problem 3.** We use the streaming strategy and the threshold greedy methods to design two effective streaming algorithms for the problem of maximizing the DR-submodular function with a cardinality constraint on the integer lattice. This contribution is recognized in publication in the Mathematics journal (MDPI, SCIE) [**Publication 10**].

In summary, the obtained results of the thesis help to deal with some aspects of the challenges of the submodular optimization problem, which are mentioned in Chapter 1 (Motivation section).

6.2 Future directions

As mentioned in the introduction chapter, submodular function optimization problems still have many interesting and challenging things for the conquest of the research community on combinatorial optimization problems. Here are some research directions that our team is and will study:

- Currently, the efficiency of the algorithms of the FBIM problem is affected by two objective factors: the data preprocessing and generating sampling using the RIS framework. Therefore, we are working to improve or raise constraints to reduce this affection. Specifically, we are working on a community detection method that costs better than the one we use (Directed Louvain method [93]). Besides, instead of randomly choosing the target communities, we consider adding more parameters to select the input set of the target communities with a better probability. In other words, we select the set of target communities that satisfies the fairness condition of the FBIM problem, helping to minimize the repetition when this set does not meet the requirements of FBIM.

- For the problem of maximizing a monotone DR-submodular function under the Knapsack constraint over the integer lattice, Gong *et al.* [101] proposed the novel deterministic algorithm, which used the threshold greedy strategy. We are studying a streaming algorithm for this problem to improve the runtime with the same or better approximation.

- Another exciting research direction that we intend to study is considering the submodular function of the simplicial complex and comparing the submodular function on the simplicial complex to the graph and integer lattice.

Bibliography

1. VONDRÁK, Jan. Submodularity in combinatorial optimization. 2007. Available also from: `https://theory.stanford.edu/~jvondrak/data/KAM_thesis.pdf`.
2. NGUYEN, Hung T.; THAI, My T.; DINH, Thang N. Stop-and-Stare: Optimal Sampling Algorithms for Viral Marketing in Billion-scale Networks. In: *Proceedings of the 2016 International Conference on Management of Data, SIGMOD Conference 2016, San Francisco, CA, USA, June 26 - July 01, 2016*. ACM, 2016, pp. 695–710. Available from DOI: `10.1145/2882903.2915207`.
3. GALHOTRA, Sainyam; ARORA, Akhil; VIRINCHI, Srinivas; ROY, Shourya. ASIM: A Scalable Algorithm for Influence Maximization under the Independent Cascade Model. In: *Proceedings of the 24th International Conference on World Wide Web*. Florence, Italy: Association for Computing Machinery, 2015, pp. 35–36. WWW '15 Companion. ISBN 9781450334730. Available from DOI: `10.1145/2740908.2742725`.
4. KAPRALOV, Michael; POST, Ian; VONDRÁK, Jan. Online Submodular Welfare Maximization: Greedy is Optimal. In: *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*. SIAM, 2013, pp. 1216–1225. Available from DOI: `10.1137/1.9781611973105.88`.
5. KORULA, Nitish; MIRROKNI, Vahab S.; ZADIMOGHADDAM, Morteza. Online Submodular Welfare Maximization: Greedy Beats 1/2 in Random Order. *SIAM J. Comput.* 2018, vol. 47, no. 3, pp. 1056–1086. Available from DOI: `10.1137/15M1051142`.
6. CLARK, Andrew; POOVENDRAN, Radha. A submodular optimization framework for leader selection in linear multi-agent systems. In: *2011 50th IEEE Conference on Decision and Control and European Control Conference*. IEEE, 2011, pp. 3614–3621.
7. SOMA, Tasuku; YOSHIDA, Yuichi. A Generalization of Submodular Cover via the Diminishing Return Property on the Integer Lattice. In: *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems*. 2015, pp. 847–855.

8. GOEMANS, Michel X.; WILLIAMSON, David P. Improved Approximation Algorithms for Maximum Cut and Satisfiability Problems Using Semidefinite Programming. *J. ACM.* 1995, vol. 42, no. 6, pp. 1115–1145. Available from DOI: `10.1145/227683.227684`.
9. BACH, Francis R. Learning with Submodular Functions: A Convex Optimization Perspective. *Found. Trends Mach. Learn.* 2013, vol. 6, no. 2-3, pp. 145–373. Available from DOI: `10.1561/2200000039`.
10. DAS, Abhimanyu; KEMPE, David. Approximate Submodularity and its Applications: Subset Selection, Sparse Approximation and Dictionary Selection. *Journal of Machine Learning Research.* 2018, vol. 19, no. 3, pp. 1–34.
11. IYER, Rishabh K.; BILMES, Jeff A. Submodular Optimization with Submodular Cover and Submodular Knapsack Constraints. In: *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.* 2013, pp. 2436–2444.
12. HASSIDIM, Avinatan; SINGER, Yaron. Submodular Optimization under Noise. In: *Proceedings of the 30th Conference on Learning Theory, COLT 2017, Amsterdam, The Netherlands, 7-10 July 2017.* PMLR, 2017, vol. 65, pp. 1069–1122. Proceedings of Machine Learning Research.
13. BADANIDIYURU, Ashwinkumar; MIRZASOLEIMAN, Baharan; KARBASI, Amin; KRAUSE, Andreas. Streaming Submodular Maximization: Massive Data Summarization on the Fly. In: *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* New York, New York, USA: Association for Computing Machinery, 2014, pp. 671–680. KDD '14. ISBN 9781450329569.
14. MIRZASOLEIMAN, Baharan; KARBASI, Amin; KRAUSE, Andreas. Deletion-Robust Submodular Maximization: Data Summarization with "the Right to be Forgotten". In: *Proceedings of the 34th International Conference on Machine Learning, ICML.* PMLR, 2017, vol. 70, pp. 2449–2458. Proceedings of Machine Learning Research.
15. TANG, Jing; TANG, Xueyan; XIAO, Xiaokui; YUAN, Junsong. Online Processing Algorithms for Influence Maximization. In: *Proceedings of the 2018 International Conference on Management of Data.* Houston, TX, USA: Association for Computing Machinery, 2018, pp. 991–1005. SIGMOD '18. ISBN 9781450347037. Available from DOI: `10.1145/3183713.3183749`.
16. PHAM, Canh V.; PHAM, Dung V.; BUI, Bao Q.; NGUYEN, Anh V. Minimum budget for misinformation detection in online social networks with provable guarantees. *Optimization Letters.* 2021. Available from DOI: `10.1007/s11590-021-01733-0`.

17. YANG, Ruiqi; XU, Dachuan; CHENG, Yukun; GAO, Chuangen; DU, Ding-Zhu. Streaming Submodular Maximization Under Noises. In: *39th IEEE International Conference on Distributed Computing Systems, ICDCS 2019, Dallas, TX, USA, July 7-10, 2019*. IEEE, 2019, pp. 348–357.
18. GUO, Jianxiong; CHEN, Tiantian; WU, Weili. Continuous Activity Maximization in Online Social Networks. *IEEE Transactions on Network Science and Engineering*. 2020, vol. 7, no. 4, pp. 2775–2786. Available from DOI: `10.1109/TNSE.2020.2993042`.
19. GRIMSMAN, David; SEATON, Joshua H.; MARDEN, Jason R.; BROWN., Philip N. The Cost of Denied Observation in Multiagent Submodular Optimization. In: *59th IEEE Conference on Decision and Control (CDC)*. 2020, pp. 1666–1671. Available from DOI: `10.1109/CDC42340.2020.9304054`.
20. JALEEL, Hassan; SHAMMA, Jeff S. Distributed Optimization for Robot Networks: From Real-Time Convex Optimization to Game-Theoretic Self-Organization. *Proceedings of the IEEE*. 2020, vol. 108, no. 11, pp. 1953–1967. Available from DOI: `10.1109/JPROC.2020.3028295`.
21. FUJISHIGE, Satoru. Submodular Functions and Optimization. In: 2005, pp. 71–104.
22. CRAWFORD, Victoria G.; KUHNLE, Alan; THAI, My T. Submodular Cost Submodular Cover with an Approximate Oracle. In: *Proceedings of the 36th International Conference on Machine Learning, ICML 2019*. PMLR, 2019, vol. 97, pp. 1426–1435.
23. KEMPE, David; KLEINBERG, Jon M.; TARDOS, Éva. Maximizing the spread of influence through a social network. In: *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 24 - 27, 2003*. 2003, pp. 137–146.
24. BORGS, Christian; BRAUTBAR, Michael; CHAYES, Jennifer T.; LUCIER, Brendan. Maximizing Social Influence in Nearly Optimal Time. In: *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*. SIAM, 2014, pp. 946–957. Available from DOI: `10.1137/1.9781611973402.70`.
25. TANG, Jing; TANG, Xueyan; YUAN, Junsong. Profit Maximization for Viral Marketing in Online Social Networks: Algorithms and Analysis. *IEEE Transactions on Knowledge and Data Engineering*. 2018, vol. 30, no. 6, pp. 1095–1108. Available from DOI: `10.1109/TKDE.2017.2787757`.
26. LIU, Paul; VONDRÁK, Jan. Submodular Optimization in the MapReduce Model. In: FINEMAN, Jeremy T.; MITZENMACHER, Michael (eds.). *2nd Symposium on Simplicity in Algorithms, SOSA*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019, vol. 69, 18:1–18:10. OASIcs.

27. AMIR, Rabah. Supermodularity and Complementarity in Economics: An Elementary Survey. *Southern Economic Journal*. 2005, vol. 71, no. 3, pp. 636–660. ISSN 00384038.

28. ABEDI, Vahideh Sadat; BERMAN, Oded; KRASS, Dmitry. Supporting New Product or Service Introductions: Location, Marketing, and Word of Mouth. *Operations Research*. 2014, vol. 62, no. 5, pp. 994–1013.

29. VONDRAK, Jan. Optimal Approximation for the Submodular Welfare Problem in the Value Oracle Model. In: *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*. Victoria, British Columbia, Canada: Association for Computing Machinery, 2008, pp. 67–74. STOC '08. ISBN 9781605580470.

30. BUCHBINDER, Niv; FELDMAN, Moran; NAOR, Joseph; SCHWARTZ, Roy. A Tight Linear Time (1/2)-Approximation for Unconstrained Submodular Maximization. *SIAM J. Comput.* 2015, vol. 44, no. 5, pp. 1384–1402. Available from DOI: 10.1137/130929205.

31. ELENBERG, Ethan R.; DIMAKIS, Alexandros G.; FELDMAN, Moran; KARBASI, Amin. Streaming Weak Submodularity: Interpreting Neural Networks on the Fly. In: GUYON, Isabelle; LUXBURG, Ulrike von; BENGIO, Samy; WALLACH, Hanna M.; FERGUS, Rob; VISHWANATHAN, S. V. N.; GARNETT, Roman (eds.). *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, USA*. 2017, pp. 4044–4054.

32. JEGELKA, Stefanie; BILMES, Jeff A. Submodularity beyond submodular energies: Coupling edges in graph cuts. In: *The 24th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2011, Colorado Springs, CO, USA, 20-25 June 2011*. IEEE Computer Society, 2011, pp. 1897–1904. Available from DOI: 10.1109/CVPR.2011.5995589.

33. LIU, Yuzong; WEI, Kai; KIRCHHOFF, Katrin; SONG, Yisong; BILMES, Jeff. Submodular feature selection for high-dimensional acoustic score spaces. In: *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. 2013, pp. 7184–7188. Available from DOI: 10.1109/ICASSP.2013.6639057.

34. LIN, Hui; BILMES, Jeff A. A Class of Submodular Functions for Document Summarization. In: *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference, 19-24 June, 2011, Portland, Oregon, USA*. The Association for Computer Linguistics, 2011, pp. 510–520.

35. STREETER, Matthew; GOLOVIN, Daniel. An Online Algorithm for Maximizing Submodular Functions. In: *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2009.

36. PHAM, Dung V; NGUYEN, Giang L; NGUYEN, Tu N; PHAM, Canh V; NGUYEN, Anh V. Multi-topic misinformation blocking with budget constraint on online social networks. *IEEE Access*. 2020, vol. 8, pp. 78879–78889.

37. PHAM, Canh V; PHU, Quat V; HOANG, Huan X; PEI, Jun; THAI, My T. Minimum budget for misinformation blocking in online social networks. *Journal of Combinatorial Optimization*. 2019, vol. 38, no. 4, pp. 1101–1127.

38. PHAM, Canh V; THAI, My T; DUONG, Hieu V; BUI, Bao Q; HOANG, Huan X. Maximizing misinformation restriction within time and budget constraints. *Journal of Combinatorial Optimization*. 2018, vol. 35, no. 4, pp. 1202–1240.

39. NGUYEN, Bich-Ngan T.; PHAM, Phuong N. H.; TRAN, Loi H.; PHAM, Canh V.; SNÁŠEL, Václav. Fairness Budget Distribution for Influence Maximization in Online Social Networks. In: *The 2021 International Conference on Artificial Intelligence and Big Data in Digital Era" (ICABDE 2021)*. 2021.

40. PHAM, Canh V; DUONG, Hieu V; HOANG, Huan X; THAI, My T. Competitive influence maximization within time and budget constraints in online social networks: an algorithmic approach. *Applied Sciences*. 2019, vol. 9, no. 11, p. 2274.

41. CHEN, Wei; WANG, Yajun; YANG, Siyu. Efficient Influence Maximization in Social Networks. In: *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Paris, France: Association for Computing Machinery, 2009, pp. 199–208. KDD '09. ISBN 9781605584959.

42. GOYAL, Amit; LU, Wei; LAKSHMANAN, Laks V.S. CELF++: Optimizing the Greedy Algorithm for Influence Maximization in Social Networks. In: *Proceedings of the 20th International Conference Companion on World Wide Web*. Hyderabad, India: Association for Computing Machinery, 2011, pp. 47–48. WWW '11. ISBN 9781450306379.

43. KHAJEHNEJAD, Moein; REZAEI, Ahmad Asgharian; BABAEI, Mahmoudreza; HOFFMANN, Jessica; JALILI, Mahdi; WELLER, Adrian. Adversarial Graph Embeddings for Fair Influence Maximization over Social Networks. *CoRR*. 2020, vol. abs/2005.04074. Available from arXiv: `2005.04074`.

44. RAHMATTALABI, Aida; JABBARI, Shahin; LAKKARAJU, Himabindu; VAYANOS, Phebe; IZENBERG, Max; BROWN, Ryan; RICE, Eric; TAMBE, Milind. Fair Influence Maximization: a Welfare Optimization Approach. In: *Thirty-Fifth AAAI Conference on Artificial Intelligence*. AAAI Press, 2021, pp. 11630–11638.

45. NGUYEN, Bich-Ngan T.; PHAM, Phuong N. H.; PHAM, Canh V.; SU, Anh N.; SNÁŠEL, Václav. Streaming Algorithm for Submodular Cover Problem Under Noise. In: *2021 RIVF International Conference on Computing and Communication Technologies (RIVF)*. 2021, pp. 1–6. Available from DOI: `10.1109/RIVF51545.2021.9642118`.

46. CHEN, Wei; WANG, Chi; WANG, Yajun. Scalable Influence Maximization for Prevalent Viral Marketing in Large-Scale Social Networks. In: *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. USA: Association for Computing Machinery, 2010, pp. 1029–1038. ISBN 9781450300551.

47. KEMPE, David; KLEINBERG, Jon; TARDOS, Éva. Maximizing the Spread of Influence through a Social Network. *Theory of Computing*. 2015, vol. 11, no. 4, pp. 105–147. Available from DOI: 10.4086/toc.2015.v011a004.

48. CHEN, Yu; WANG, Wei; FENG, Jinping; LU, Ying; GONG, Xinqi. Maximizing multiple influences and fair seed allocation on multilayer social networks. *PLOS ONE*. 2020-03, vol. 15, no. 3, pp. 1–19. Available from DOI: 10.1371/journal.pone.0229201.

49. BANERJEE, Abhijit; G A, Arun; DUFLO, Esther; JACKSON, Matthew. The Diffusion of Microfinance. *Science (New York, N.Y.)* 2013-07, vol. 341, p. 1236498. Available from DOI: 10.1126/science.1236498.

50. YADAV, Amulya; WILDER, Bryan; RICE, Eric; PETERING, Robin; CRADDOCK, Jaih; YOSHIOKA-MAXWELL, Amanda; HEMLER, Mary; ONASCH-VERA, Laura; TAMBE, Milind; WOO, Darlene. Bridging the Gap Between Theory and Practice in Influence Maximization: Raising Awareness about HIV among Homeless Youth. In: *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*. International Joint Conferences on Artificial Intelligence Organization, 2018-07, pp. 5399–5403. Available from DOI: 10.24963/ijcai.2018/761.

51. MIRZASOLEIMAN, Baharan; BABAEI, Mahmoudreza; JALILI, Mahdi. Immunizing complex networks with limited budget. *EPL (Europhysics Letters)*. 2012-05, vol. 98, no. 3, p. 38004. Available from DOI: 10.1209/0295-5075/98/38004.

52. MIRZASOLEIMAN, Baharan; KARBASI, Amin; BADANIDIYURU, Ashwinkumar; KRAUSE, Andreas. Distributed Submodular Cover: Succinctly Summarizing Massive Data. In: *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*. Montreal, Canada: MIT Press, 2015, pp. 2881–2889. NIPS'15.

53. MIRZASOLEIMAN, Baharan; ZADIMOGHADDAM, Morteza; KARBASI, Amin. Fast Distributed Submodular Cover: Public-Private Data Summarization. In: *Proceedings of the 30th International Conference on Neural Information Processing Systems*. Barcelona, Spain: Curran Associates Inc., 2016, pp. 3601–3609. NIPS'16. ISBN 9781510838819.

54. SOMA, Tasuku; YOSHIDA, Yuichi. A Generalization of Submodular Cover via the Diminishing Return Property on the Integer Lattice. In: *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2015, vol. 28.

55. GOYAL, Amit; BONCHI, Francesco; LAKSHMANAN, V. S. Laks; VENKATASUBRAMANIAN, Suresh. On minimizing budget and time in influence propagation over social networks. *Social Netw. Analys. Mining*. 2013, pp. 179–192.

56. KUHNLE, Alan; PAN, Tianyi; ALIM, Md Abdul; THAI, My T. Scalable Bicriteria Algorithms for the Threshold Activation Problem in Online Social Networks. In: *IEEE Conference on Computer Communications*. 2017. Available from DOI: `10.1109/INFOCOM.2017.8057068`.

57. NOROUZI-FARD, Ashkan; BAZZI, Abbas; BOGUNOVIC, Ilija; HALABI, Marwa El; HSIEH, Ya-Ping; CEVHER, Volkan. An Efficient Streaming Algorithm for the Submodular Cover Problem. In: *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*. 2016, pp. 4493–4501.

58. WOLSEY, Laurence A. An analysis of the greedy algorithm for the submodular set covering problem. *Comb.* 1982, vol. 2, no. 4, pp. 385–393. Available from DOI: `10.1007/BF02579435`.

59. WAN, P.; DU, D.; PARDALOS, P.; WU, W. Greedy approximations for minimum submodular cover with submodular cost. *Computational Optimization and Applications*. 2010, vol. 45, pp. 463–474.

60. CHEN, Wei; WANG, Chi; WANG, Yajun. Scalable Influence Maximization for Prevalent Viral Marketing inLarge-Scale Social Networks. In: *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY*. 2010, pp. 1029–1038.

61. CHAO, Qian.; JING-CHENG, Shi.; YANG, Yu.; KE, Tang.; ZHI-HUA, Zhou. Subset Selection under Noise. In: *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2017, vol. 30.

62. COHEN, Edith; DELLING, Daniel; PAJOR, Thomas; WERNECK, Renato F. Sketch-based Influence Maximization and Computation: Scaling up with Guarantees. In: *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, CIKM 2014, Shanghai, China, November 3-7, 2014*. ACM, 2014, pp. 629–638.

63. CHEN, Wei; LAKSHMANAN, Laks V. S.; CASTILLO, Carlos. *Information and Influence Propagation in Social Networks*. Morgan & Claypool Publishers, pp. 35-66, 2014. Synthesis Lectures on Data Management.

64. LESKOVEC, Jure; KREVL, Andrej. *SNAP Datasets: Stanford Large Network Dataset Collection* [`http://snap.stanford.edu/data`]. 2014-06.

65. NGUYEN, Lan; THAI, My T. Streaming k-Submodular Maximization under Noise subject to Size Constraint. In: *Proceedings of the 37th International Conference on Machine Learning*. PMLR, 2020-07, vol. 119, pp. 7338–7347. Proceedings of Machine Learning Research.

66. HEIDEMANN, Julia; KLIER, Mathias; PROBST, Florian. Online social networks: A survey of a global phenomenon. *Computer Networks*. 2012, vol. 56, no. 18, pp. 3866–3878. ISSN 1389-1286.

67. BANERJEE, Suman; JENAMANI, Mamata; PRATIHAR, Dilip Kumar. A survey on influence maximization in a social network. *Knowledge and Information Systems*. 2020, vol. 62, no. 9, pp. 3417–3455.

68. DU, Nan; SONG, Le; GOMEZ RODRIGUEZ, Manuel; ZHA, Hongyuan. Scalable influence estimation in continuous-time diffusion networks. In: 2013, vol. 26.

69. LI, Jianxin; CAI, Taotao; MIAN, Ajmal; LI, Rong-Hua; SELLIS, Timos; YU, Jeffrey Xu. Holistic influence maximization for targeted advertisements in spatial social networks. In: *34th International Conference on Data Engineering (ICDE)*. IEEE, 2018, pp. 1340–1343.

70. PHAM, Canh V.; HA, Dung K. T.; VU, Quang C.; SU, Anh N.; HOANG, Huan X. Influence Maximization with Priority in Online Social Networks. *Algorithms*. 2020, vol. 13, no. 8.

71. SUN, Gengxin; CHEN, Chih-Cheng. Influence Maximization Algorithm Based on Reverse Reachable Set. *Mathematical Problems in Engineering*. 2021-07, vol. 2021, pp. 1–12.

72. RICHARDSON, Matthew; DOMINGOS, Pedro M. Mining knowledge-sharing sites for viral marketing. In: *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, July 23-26, 2002, Edmonton, Alberta, Canada*. ACM, 2002, pp. 61–70. Available from DOI: `10.1145/775047.775057`.

73. BECKER, Ruben; D'ANGELO, Gianlorenzo; GHOBADI, Sajjad; GILBERT, Hugo. Fairness in influence maximization through randomization. *Journal of Artificial Intelligence Research*. 2022, vol. 73, pp. 1251–1283.

74. UDWANI, Rajan. Multi-Objective Maximization of Monotone Submodular Functions with Cardinality Constraint. *CoRR*. 2017, vol. abs/1711.06428. Available from arXiv: `1711.06428`.

75. LI, Yuchen; FAN, Ju; WANG, Yanhao; TAN, Kian-Lee. Influence Maximization on Social Graphs: A Survey. *IEEE Transactions on Knowledge and Data Engineering*. 2018, vol. 30, no. 10, pp. 1852–1872. Available from DOI: `10.1109/TKDE.2018.2807843`.

76. LESKOVEC, Jure; KRAUSE, Andreas; GUESTRIN, Carlos; FALOUTSOS, Christos; VANBRIESEN, Jeanne; GLANCE, Natalie. Cost-Effective Outbreak Detection in Networks. In: *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. San Jose, California, USA: Association for Computing Machinery, 2007, pp. 420–429. KDD '07. ISBN 9781595936097.

77. ZHOU, Chuan; ZHANG, Peng; ZANG, Wenyu; GUO, Li. On the upper bounds of spread for greedy algorithms in social network influence maximization. *IEEE Transactions on Knowledge and Data Engineering*. 2015, vol. 27, no. 10, pp. 2770–2783.

78. GOYAL, Amit; LU, Wei; LAKSHMANAN, Laks V.S. SIMPATH: An Efficient Algorithm for Influence Maximization under the Linear Threshold Model. In: *2011 IEEE 11th International Conference on Data Mining*. 2011, pp. 211–220. Available from DOI: `10.1109/ICDM.2011.132`.

79. HE, Xinran; KEMPE, David. Stability of Influence Maximization. In: *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, New York, USA: Association for Computing Machinery, 2014, pp. 1256–1265. KDD '14. ISBN 9781450329569.

80. LIU, Qi; XIANG, Biao; CHEN, Enhong; XIONG, Hui; TANG, Fangshuang; YU, Jeffrey Xu. Influence Maximization over Large-Scale Social Networks: A Bounded Linear Approach. In: *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*. Shanghai, China: Association for Computing Machinery, 2014, pp. 171–180. CIKM '14. ISBN 9781450325981.

81. PAGE, Lawrence; BRIN, Sergey; MOTWANI, Rajeev; WINOGRAD, Terry. *The PageRank Citation Ranking: Bringing Order to the Web*. Stanford InfoLab, 1999-11. Technical Report, 1999-66. Stanford InfoLab. Previous number = SIDL-WP-1999-0120.

82. GALHOTRA, Sainyam; ARORA, Akhil; ROY, Shourya. Holistic influence maximization: Combining scalability and efficiency with opinion-aware models. In: *Proceedings of the 2016 International Conference on Management of Data*. 2016, pp. 743–758.

83. TANG, Youze; XIAO, Xiaokui; SHI, Yanchen. Influence maximization: near-optimal time complexity meets practical efficiency. In: DYRESON, Curtis E.; LI, Feifei; ÖZSU, M. Tamer (eds.). *International Conference on Management of Data, SIGMOD*. ACM, 2014, pp. 75–86.

84. TANG, Youze; SHI, Yanchen; XIAO, Xiaokui. Influence Maximization in Near-Linear Time: A Martingale Approach. In: SELLIS, Timos K.; DAVIDSON, Susan B.; IVES, Zachary G. (eds.). *Proceedings of the 2015 SIGMOD International Conference on Management of Data*. ACM, 2015, pp. 1539–1554.

85. HUANG, Keke; WANG, Sibo; BEVILACQUA, Glenn; XIAO, Xiaokui; LAKSHMANAN, Laks V. S. Revisiting the Stop-and-Stare Algorithms for Influence Maximization. *Proc. VLDB Endow.* 2017-05, vol. 10, no. 9, pp. 913–924. ISSN 2150-8097.

86. NGUYEN, Hung T.; DINH, Thang N.; THAI, My T. Revisiting of Revisiting the Stop-and-Stare Algorithms for Influence Maximization. In: CHEN, Xuemin; SEN, Arunabha; LI, Wei Wayne; THAI, My T. (eds.). *Computational Data and Social Networks.* Cham: Springer International Publishing, 2018, pp. 273–285.

87. TSANG, Alan; WILDER, Bryan; RICE, Eric; TAMBE, Milind; ZICK, Yair. Group-Fairness in Influence Maximization. In: *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19.* International Joint Conferences on Artificial Intelligence Organization, 2019-07, pp. 5997–6005.

88. STOICA, Ana-Andreea; HAN, Jessy Xinyi; CHAINTREAU, Augustin. Seeding network influence in biased networks and the benefits of diversity. In: *Proceedings of The Web Conference.* 2020, pp. 2089–2098.

89. HALABI, Marwa El; MITROVIC, Slobodan; NOROUZI-FARD, Ashkan; TARDOS, Jakab; TARNAWSKI, Jakub. Fairness in Streaming Submodular Maximization: Algorithms and Hardness. In: *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual.* 2020.

90. ALI, Junaid; BABAEI, Mahmoudreza; CHAKRABORTY, Abhijnan; MIRZASOLEIMAN, Baharan; GUMMADI, Krishna P.; SINGLA, Adish. On the Fairness of Time-Critical Influence Maximization in Social Networks. *CoRR.* 2019, vol. abs/1905.06618.

91. RAZAGHI, Behnam; ROAYAEI, Mehdy; CHARKARI, Nasrollah Moghadam. On the Group-Fairness-Aware Influence Maximization in Social Networks. *IEEE Transactions on Computational Social Systems.* 2022.

92. BADANIDIYURU, Ashwinkumar; VONDRÁK, Jan. Fast algorithms for maximizing submodular functions. In: *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA.* SIAM, 2014, pp. 1497–1514.

93. DUGUÉ, Nicolas; PEREZ, Anthony. *Directed Louvain: maximizing modularity in directed networks.* 2015. PhD thesis. Université d'Orléans.

94. SOMA, Tasuku; YOSHIDA, Yuichi. Maximizing monotone submodular functions over the integer lattice. *Mathematical Programming.* 2018, vol. 172, no. 1, pp. 539–563.

95. ALON, Noga; GAMZU, Iftah; TENNENHOLTZ, Moshe. Optimizing budget allocation among channels and influencers. In: *Proceedings of the 21st international conference on World Wide Web.* 2012, pp. 381–388.

96. SOMA, Tasuku; KAKIMURA, Naonori; INABA, Kazuhiro; KAWARABAYASHI, Ken-ichi. Optimal budget allocation: Theoretical guarantee and efficient algorithm. In: *International Conference on Machine Learning*. PMLR, 2014, pp. 351–359.

97. KAPRALOV, Michael; POST, Ian; VONDRÁK, Jan. Online submodular welfare maximization: Greedy is optimal. In: *Proceedings of the twenty-fourth annual ACM-SIAM symposium on Discrete algorithms*. SIAM, 2013, pp. 1216–1225.

98. DEMAINE, Erik D; HAJIAGHAYI, MohammadTaghi; MAHINI, Hamid; MALEC, David L; RAGHAVAN, S; SAWANT, Anshul; ZADIMOGHADAM, Morteza. How to influence people with partial incentives. In: *Proceedings of the 23rd international conference on World wide web*. 2014, pp. 937–948.

99. MITROVIC, Slobodan; BOGUNOVIC, Ilija; NOROUZI-FARD, Ashkan; TARNAWSKI, Jakub; CEVHER, Volkan. Streaming Robust Submodular Maximization: A Partitioned Thresholding Approach. In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017*. 2017, pp. 4557–4566.

100. GU, Shuyang; SHI, Ganquan; WU, Weili; LU, Changhong. A fast double greedy algorithm for non-monotone DR-submodular function maximization. *Discrete Mathematics, Algorithms and Applications*. 2020, vol. 12, no. 01, p. 2050007.

101. GONG, Suning; NONG, Qingqin; BAO, Shuyu; FANG, Qizhi; DU, Ding-Zhu. A fast and deterministic algorithm for Knapsack-constrained monotone DR-submodular maximization over an integer lattice. *Journal of Global Optimization*. 2022, pp. 1–24.

102. LIU, Bin; CHEN, Zihan; DU, Hongmin W. Streaming Algorithms for Maximizing DR-Submodular Functions with d-Knapsack Constraints. In: *Algorithmic Aspects in Information and Management - 15th International Conference, AAIM*. Springer, 2021, vol. 13153, pp. 159–169.

103. TAN, Jingjing; ZHANG, Dongmei; ZHANG, Hongyang; ZHANG, Zhenning. One-pass streaming algorithm for DR-submodular maximization with a knapsack constraint over the integer lattice. *Comput. Electr. Eng.* 2022, vol. 99, p. 107766.

104. ZHANG, Zhenning; GUO, Longkun; WANG, Yishui; XU, Dachuan; ZHANG, Dongmei. Streaming Algorithms for Maximizing Monotone DR-Submodular Functions with a Cardinality Constraint on the Integer Lattice. *Asia Pac. J. Oper. Res.* 2021, vol. 38, no. 5, 2140004:1–2140004:14.

105. KAZEMI, Ehsan; MITROVIC, Marko; ZADIMOGHADDAM, Morteza; LATTANZI, Silvio; KARBASI, Amin. 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, 2019, vol. 97, pp. 3311–3320.

106. NEMHAUSER, George L.; WOLSEY, Laurence A.; FISHER, Marshall L. An analysis of approximations for maximizing submodular set functions - I. *Math. Program.* 1978, vol. 14, no. 1, pp. 265–294.

107. SVIRIDENKO, Maxim. A note on maximizing a submodular set function subject to a knapsack constraint. *Oper. Res. Lett.* 2004, vol. 32, no. 1, pp. 41–43.

108. CĂLINESCU, Gruia; CHEKURI, Chandra; PÁL, Martin; VONDRÁK, Jan. Maximizing a Monotone Submodular Function Subject to a Matroid Constraint. *SIAM J. Comput.* 2011, vol. 40, no. 6, pp. 1740–1766.

109. BADANIDIYURU, Ashwinkumar; MIRZASOLEIMAN, Baharan; KARBASI, Amin; KRAUSE, Andreas. Streaming Submodular Maximization: Massive Data Summarization on the Fly. In: *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Association for Computing Machinery, 2014, pp. 671–680. KDD '14.

110. HATANO, Daisuke; FUKUNAGA, Takuro; MAEHARA, Takanori; KAWARABAYASHI, Ken-ichi. Lagrangian Decomposition Algorithm for Allocating Marketing Channels. In: *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*. AAAI Press, 2015, pp. 1144–1150.

111. KUNEGIS, Jérôme. KONECT: the Koblenz network collection. In: *22nd International World Wide Web Conference, WWW '13*. ACM, 2013, pp. 1343–1350.

List of own publication activities

This section shows the list of own publications.
My ORCID : Bich-Ngan T. Nguyen (https://orcid.org/ 0000-0002-5297-6262)

A. Publications by Author

1. **NGUYEN, Bich-Ngan T.**; PHAM, Phuong N. H.; PHAM, Canh V.; SU, Anh N. Su; SNÁŠEL, Václav. Streaming Algorithm for Submodular Cover Problem Under Noise. In: *2021 RIVF International Conference on Computing and Communication Technologies (RIVF)*, 2021, pp. 1-6. Available from DOI: 10.1109/RIVF51545.2021.9642118.

2. **NGUYEN, Bich-Ngan T.**; PHAM, Phuong N. H.; TRAN, Loi H.; PHAM, Canh V.; SNÁŠEL, Václav. Fairness Budget Distribution for Influence Maximization in Online Social Networks. In *the 2021 International Conference on Artificial Intelligence and Big Data in Digital Era" (ICABDE)*, 2021. Available from DOI: 10.1007/978-3-030-97610-1_19.

3. **NGUYEN, Bich-Ngan T.**; PHAM, Phuong N. H.; VU, Thanh Nguyen; PHAN, Quoc Viet; LE, Dinh Tuan; SNÁŠEL, Václav. Py_ape: Text Data Acquiring, Extracting, Cleaning and Schema Matching in Python. In: *2020 FDSE International Conference on Future Data and Security Engineering. Big Data, Security and Privacy, Smart City and Industry 4.0 Applications (FDSE)*, 2020, pp. 78–89. Available from DOI: 10.1007/978-981-33-4370-2_6.

4. PHAM, Phuong N. H.; **NGUYEN, Bich-Ngan T.**; CO, Quy T. N.; SNÁŠEL, Václav. Multiple Benefit Thresholds Problem in Online Social Networks: An algorithmic approach. In *Mathematics*, 2022, vol. 10, no. 6. Available from DOI: 10.3390/math10060876.

5. PHAM, Phuong N. H.; **NGUYEN, Bich-Ngan T.**; PHAM, Canh V.; NGHIA, Nghia D.; SNÁŠEL, Václav. Efficient Algorithm for Multiple Benefit Thresholds Problem in Online Social Networks. In: *2021 RIVF International Conference on Computing and Communication Technologies (RIVF)*, 2021, pp 138-143. Available from DOI: 10.1109/RIVF51545.2021.9642099.

6. PHAM, Phuong N. H.; **NGUYEN, Bich-Ngan T.**; CO, Quy T. N.; PHAM, Canh; SNÁŠEL, Václav. Threshold Benefit for Groups Influence in Online Social Networks. In: *2021 FDSE International Conference on Future Data and Security Engineering. Big Data, Security and Privacy, Smart City and Industry 4.0 Applications (FDSE)*, 2021, pp 53-67. Available from DOI: 10.1007/978-3-030-91387-8_4.

7. PHAM, Phuong N. H.; **NGUYEN, Bich-Ngan T.**; CO, Quy T. N.; NGUYEN, Tu Nguye; TRAN, Phuoc; SNÁŠEL, Václav. An Efficient Hybrid Algorithm for Community Structure Detection in Complex Networks Based on Node Influence. In: *ICIC Express Letters Part B: Applications*, 2021, 10, 899-908.
Available from DOI: http://dx.doi.org/10.24507/icicelb.12.10.899.

B. Publications by Author Which Are Currently Under Review

8. **NGUYEN, Bich-Ngan T.**; PHAM, Phuong N. H.; PHAM, Canh V.; SNÁŠEL, Václav. Fast Streaming Algorithms Submodular Cover under Noises.
Submitted it to the *Computer Standards and Interfaces* journal (Elsevier), ISSN: 0920-5489, SCIE, Q1, IF:3.721.
We submitted it on June 2021; resubmitted the minor revise (round 2) on September 2022, and pending acceptance.

9. **NGUYEN, Bich-Ngan T.**; PHAM, Phuong N. H.; LE, Van-Vang; SNÁŠEL, Václav. Influence Maximization under Fairness Budget Distribution in Online Social Networks. Submitted it to the *Mathematics* journal (MDPI), ISSN: 2227-7390, SCIE, Q1, IF: 2.592. We submitted it on July 2022, resubmitted the minor revision September 2022, accepted and pending publication.

10. **NGUYEN, Bich-Ngan T.**; PHAM, Phuong N. H.; LE, Van-Vang; NGHIEM, Xuan Dung; SNÁŠEL, Václav. Efficient Streaming Algorithms for Maximizing Monotone DR-Submodular Function on the Integer Lattice.
Submitted it to the *Mathematics* journal (MDPI), ISSN: 2227-7390, SCIE, Q1, IF: 2.592. We submitted it in early September 2022; we are revising to resubmit the revised manuscript.

11. LE, Van-Vang; TRAN, Toai K.; **NGUYEN, Bich-Ngan T.**; NGUYEN, Quoc-Dung; SNASEL, Vaclav. Network alignment using a combination of multiple embedding techniques.
Submitted it to the Computational Intelligence and Neuroscience journal (Hindawi), ISSN: 1687-5273, SCIE, Q1, IF: 3.120.
We submitted in on June 2022, it's currently under review.