MINISTRY OF EDUCATION
AND TRAINING

VIETNAM ACADEMY OF SCIENCE
AND TECHNOLOGY

GRADUATE UNIVERSITY OF SCIENCE AND TECHNOLOGY

Nguyen Thi Hue

RESEARCH ON APPROXIMATION ALGORITHMS FOR SELECTED SUBMODULAR OPTIMIZATION PROBLEMS WITH COST CONSTRAINTS

SUMMARY OF DISSERTATION ON COMPUTER

Major: Computer Science

Mã số: 9 48 01 01

Ha Noi - 2026

The dissertation is completed at: Graduate University of Science and Technology, Vietnam Academy Science and Technology

Supervisors:

1. Supervisor 1: Assoc. Prof. Dr. Pham Van Canh
2. Supervisor 2: Assoc. Prof. Dr. Nguyen Long Giang.

Referee 1:.....

Referee 2:.....

The dissertation is examined by Examination Board of Graduate University of Science and Technology, Vietnam Academy of Science and Technology at…….. (time, date……)

The dissertation can be found at:

1. Graduate University of Science and Technology Library
2. National Library of Vietnam

INTRODUCTION

1. Research Motivation

Combinatorial Optimization (CO) problems arise from the need to select optimal solutions in economic, engineering, and social activities. In the era of artificial intelligence, CO plays a central role in many data science applications, such as influence maximization, information summarization, recommender systems, and revenue maximization. Many CO problems can be modeled as set-based optimization problems in which the objective function or constraints are *submodular*. This property, also known as the *diminishing returns property*, enables the development of efficient algorithms with theoretical guarantees on solution quality. This class of problems is collectively referred to as Submodular Optimization (SO) and is particularly suitable for large-scale data applications.

In its most general form, an SO problem can be formulated as follows. Let $V = \{e_1, e_2, \ldots, e_n\}$ be a finite ground set, and let $f : 2^V \mapsto \mathbb{R}_+$[1] be a submodular objective function. That is, for every $A \subseteq B \subseteq V$ and every element $u \in V \setminus B$, we have

$$f(A \cup \{u\}) - f(A) \geq f(B \cup \{u\}) - f(B).$$

An SO problem aims to find a subset $S \subseteq V$ satisfying a given constraint $\mathcal{C}$ such that the value of f is maximized or minimized.

The SO problems considered in this dissertation are NP-hard. Since SO constitutes a relatively broad class of problems, researchers commonly divide it into the following two main categories (see Figure 0.1).

Problem 1: Minimum-Cost Submodular Cover Problem (MCSC). Each element $e \in V$ has a cost $c(e) > 0$. The objective function is the total cost $f(S) = \sum_{e \in S} c(e)$. Given a threshold $T > 0$ and a submodular utility function $g : 2^V \to \mathbb{R}$, such that, for every $A \subseteq B \subseteq V$ and $e \in V \setminus B$,

$$g(A \cup \{e\}) - g(A) \geq g(B \cup \{e\}) - g(B),$$

the problem aims to find

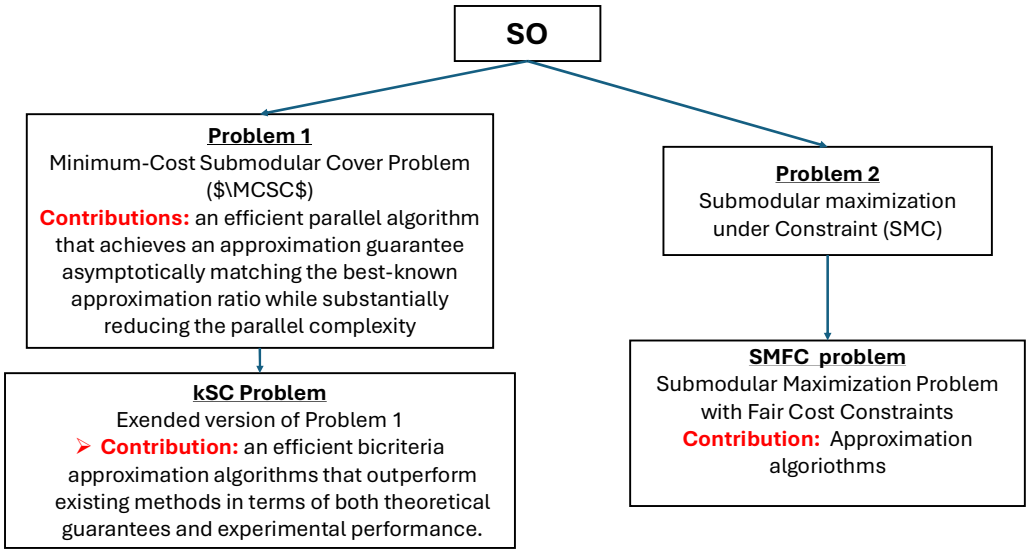
$$\begin{aligned} \arg\min_{S \subseteq V} \quad & f(S) \\ \text{subject to:} \quad & g(S) \geq T. \end{aligned}$$

[1] 2^V denotes the collection of all subsets of V, including the empty set.

Problem 2: Constrained Submodular Maximization Problem (SM)**.** Given a constraint family $\mathcal{C}$, the problem aims to find

$$\begin{aligned} \max_{S \subseteq V} \quad & f(S) \\ \text{subject to:} \quad & S \in \mathcal{C}. \end{aligned}$$

This class encompasses many representative problems in data analysis and machine learning, including influence maximization in social networks, feature selection in machine learning, revenue maximization in social networks, sensor placement, and representative subset selection to maximize the amount of information obtained under resource constraints. Common constraint families $\mathcal{C}$ in combinatorial optimization include cardinality constraints, knapsack constraints, and matroid constraints.



Hình 0.1: The main categories of SO problems and the contributions of the dissertation.

Despite substantial progress, SO problems still face several challenges in both modeling and algorithm design, as follows:

First, the rapid growth of data in modern systems creates an urgent need for algorithms that not only guarantee high-quality solutions but also process data within a practically acceptable amount of time.

Second, many practically important problem variants still lack efficient algorithms. Real-world problems often involve multiple complex constraints, such as fairness requirements among different groups, multidimensional resource limitations, or dependency structures among elements. Consequently, existing standard models cannot fully capture the requirements of practical applications. Therefore, extending existing models

and developing efficient algorithms for new problem variants remain important research directions with considerable potential for further development.

2. Research Objectives

The dissertation focuses on addressing the above challenges through the following objectives:

1. To investigate Problem 1 (MCSC) and Problem 2 (SM) under practically relevant constraint families $\mathcal{C}$.

2. To propose and analyze efficient approximation algorithms for settings in which finding exact optimal solutions is computationally infeasible because of the high computational complexity.

3. Main Contributions

To achieve the above objectives, the dissertation makes the following main contributions:

The dissertation investigates Problem 1 (MCSC) and proposes an efficient parallel algorithm that achieves an approximation guarantee asymptotically matching the best-known approximation ratio while substantially reducing the parallel complexity, as demonstrated through both theoretical analysis and experimental evaluation.

The dissertation investigates the Minimum-Cost k-Submodular Cover Problem (kSC), an extension of MCSC. It proposes efficient bicriteria approximation algorithms that outperform existing methods in terms of both theoretical guarantees and experimental performance.

For Problem 2, the dissertation investigates the *Submodular Maximization Problem with Fair Cost Constraints* (SMFC) and proposes efficient approximation algorithms for this problem.

4. Organization of the Dissertation

In addition to the Introduction and Conclusion, the dissertation is organized into four chapters as follows:

Chapter 1 provides an overview of combinatorial optimization problems and their roles in modern data analysis applications.

Chapter 2 presents the research results on the Minimum-Cost Submodular Cover Problem (MCSC).

Chapter 3 presents the research results on the Minimum-Cost k-Submodular Cover Problem (kSC).

Chapter 4 presents the research results on the Submodular Maximization Problem with Fair Cost Constraints (SMFC).

CHAPTER 1
OVERVIEW OF SUBMODULAR OPTIMIZATION PROBLEMS

1.1. SUBMODULAR OPTIMIZATION PROBLEMS

From a mathematical modeling perspective, Submodular Optimization (SO) problems are commonly defined based on the following concepts:
- *Ground set* $V = \{e_1, e_2, \ldots, e_n\}$: a finite set consisting of n elements, where each element e_i represents a candidate object to be selected.
- *Constraint* $\mathcal{C}$: used to model practical requirements and limitations.
- *Power set* 2^V: the collection of all possible subsets of V.
- *Feasible solution space*:

$$\Omega = \{S \subseteq V : S \text{ satisfies } \mathcal{C}\}.$$

- *Objective function f and constraint evaluation function g*: over this solution space, we define an objective function and a constraint evaluation function taking nonnegative real values:

$$f, g : 2^V \to \mathbb{R}_+,$$

where f measures the quality or utility of each subset, whereas g measures the extent to which a solution satisfies the constraint $\mathcal{C}$. The function $f : 2^V \to \mathbb{R}$ is *submodular* if, for every pair of subsets $A \subseteq B \subseteq V$ and every element $e \in V \setminus B$, the following inequality holds:

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

Furthermore, a submodular function f is called *monotone* if

$$f(A) \leq f(B), \qquad \forall A \subseteq B \subseteq V.$$

Some important concepts are defined as follows:
- *Feasible solution:* A subset $S \subseteq V$ is called a feasible solution if $S \in \Omega$.
- *Optimal solution:* A feasible solution $O \in \Omega$ is called an optimal solution if

$$O \in \arg\max_{S \in \Omega} f(S)$$

for a maximization problem, or

$$O \in \arg\min_{S \in \Omega} f(S)$$

for a minimization problem.

1.1.1. Some Representative Problems

The following are some representative SO problems that commonly arise in practical applications.

Definition 1.1 (Influence Maximization Problem (Influence Maximization–IM))**.** Given a directed graph $G = (V, E)$ modeling a social network and a fixed information diffusion model $\mathcal{M}$, the Influence Maximization problem aims to identify an initial set of nodes, also called a seed set, $S \subseteq V$, such that the influence spread from S throughout the network, measured by the influence function $f(S)$, is maximized.

Definition 1.2 (Threshold Influence Problem [1])**.** Given a social network $G = (V, E)$, an information diffusion model $\mathcal{M}$, and a threshold T, the objective is to find a seed set $S \subseteq V$ such that

$$f(S) \geq T,$$

while minimizing the required budget, such as the number of *seed nodes* $|S|$ or the total incurred cost $\sum_{v \in S} c(v)$.

1.1.2. Algorithmic Complexity

Definition 1.3 (Time Complexity of an Algorithm)**.** Consider an algorithm $\mathcal{A}$ that solves a problem with an input of size n. The time complexity of algorithm $\mathcal{A}$, denoted by $T_{\mathcal{A}}(n)$, is defined as the maximum number of elementary computational steps required by the algorithm to process any input whose size does not exceed n.

Definition 1.4 (Query Complexity)**.** Consider an algorithm $\mathcal{A}$ that solves a problem with an input of size n. Suppose that the objective function can be accessed through a prescribed method, referred to as a black box. The query complexity of algorithm $\mathcal{A}$, denoted by $Q_{\mathcal{A}}(n)$, is defined as the maximum number of queries to f through this method, equivalently, the maximum number of evaluations of the objective function f, that the algorithm must perform for any input whose size does not exceed n.

Definition 1.5 (Parallel Complexity)**.** For an objective function f, the parallel complexity of an algorithm is the minimum number of sequential rounds required, where, in each round, the algorithm performs $O(\mathrm{poly}(n))$ independent queries to the objective function, and n is the input size of the algorithm.

1.1.3. Approximation Algorithms

Definition 1.6 (Approximation Algorithm [2])**.** Consider an optimization problem $\mathcal{P}$ with an objective function f and an input of size n, where the objective may be either

maximization or minimization. An algorithm $\mathcal{A}$ is called an approximation algorithm for problem $\mathcal{P}$ with approximation ratio $\rho(n)$ if:

(i) Algorithm $\mathcal{A}$ runs in polynomial time with respect to the input size n;

(ii) Algorithm $\mathcal{A}$ always returns a feasible solution to the problem, denoted by S. Let S^* denote an optimal solution. Then:

$$f(S) \geq \rho(n) \cdot f(S^*), \quad 0 < \rho(n) \leq 1, \qquad \text{for a maximization problem,}$$
$$f(S) \leq \rho(n) \cdot f(S^*), \quad \rho(n) \geq 1, \qquad \text{for a minimization problem.}$$

In other words, the solution returned by the algorithm is guaranteed to be within a factor of $\rho(n)$ of the optimal solution.

Definition 1.7 (Bicriteria Approximation Algorithm [3])**.** For the MCSC problem, an algorithm is called a *bicriteria approximation algorithm*, or an (α, β)-approximation algorithm, if it runs in polynomial time with respect to the input size n and returns a solution S satisfying

$$g(S) \geq \alpha T \qquad \text{and} \qquad f(S) \leq \beta f(S^\star),$$

where $S^\star$ is an optimal solution, $0 < \alpha \leq 1$, and $\beta \geq 1$. Thus, α measures the extent to which the coverage requirement is satisfied, whereas β represents the approximation factor for the solution cost.

CHAPTER 2
MINIMUM-COST SUBMODULAR COVER PROBLEM

2.1. PROBLEM DEFINITION AND CONTRIBUTIONS OF THE DISSERTATION

Consider a finite ground set $V = \{e_1, \ldots, e_n\}$ and a utility function $g : 2^V \mapsto \mathbb{R}^+$ that measures the quality of a subset $S \subseteq V$. The function g is assumed to be *submodular* and monotone nondecreasing; that is,

$$g(A) \leq g(B) \qquad \text{for every } A \subseteq B \subseteq V.$$

In addition, let $c : 2^V \mapsto \mathbb{R}_+$ be a modular cost function defined by

$$c(S) = \sum_{e \in S} c(e).$$

The *Minimum-Cost Submodular Cover* problem (MCSC) aims to find a subset $S \subseteq V$ such that $g(S) \geq T$ while minimizing its total cost.

Reference	Approximation Guarantee	Query Complexity	Adaptive Complexity
Minimum-Cost Submodular Cover Problem (MCSC)			
[4]	$1 + \log\left(\frac{\max_{e \in S} g(e)}{\beta}\right)$	$\Omega(n\lvert S\rvert)$	$\Omega(\lvert S\rvert)$
[5]	$(1 + \log \frac{T}{\epsilon}, 1 - \epsilon)$	$\Omega(n\lvert S\rvert)$	$\Omega(\lvert S\rvert)$
[6]	$\frac{H(\min\{\max_{e \in S} g(e), T\})}{1-5\epsilon}$	$O(\frac{n \log(nT) \log(T)(\log T + \log\log(nT))}{\epsilon^4})$	$O(\frac{\log(nT) \log(T)(\log T + \log\log(nT))}{\epsilon^4})$
LA	$((1+\epsilon)(1+\ln\frac{1}{\lambda}), 1-\lambda)$	$O(\frac{n^2 \log^2(n/\epsilon) \log(\frac{1}{\lambda})}{\epsilon^3})$ **or** $O(\frac{n \log(n) \log^2(n/\epsilon) \log(\frac{1}{\lambda})}{\epsilon^3})$	$O(\frac{\log(n) \log(1/\lambda)}{\epsilon^2})$ **or** $O(\frac{\log^2(n) \log(1/\lambda)}{\epsilon^2})$
Uniform-Cost Submodular Cover			
[7]	$O(\log T)$	$O(n \log(n \log T) \log T)$	$O(\log(n \log T) \log T)$
LA	$((1+\epsilon)(1+\ln\frac{1}{\lambda}), 1-\lambda)$	$O(\frac{n^2 \log^2(n) \log(\frac{1}{\lambda})}{\epsilon^3})$ **or** $O(\frac{n \log^3(n) \log(\frac{1}{\lambda})}{\epsilon^3})$	$O(\frac{\log(n) \log(1/\lambda)}{\epsilon^2})$ **or** $O(\frac{\log^2(n) \log(1/\lambda)}{\epsilon^2})$

Table 2.1: A comparison of existing algorithms for the MCSC problem and its uniform-cost variant. The notation (α, β) represents an (α, β)-bicriteria approximation guarantee. Note that the algorithms proposed in [7, 6] are applicable only when g is integer-valued.

Contributions of the dissertation to the MCSC problem and discussion of the results. Motivated by the limitations and research gaps in related work, as summarized in Table 2.1, the dissertation develops an efficient parallel algorithm for the MCSC problem. Compared with the best existing low-adaptivity algorithm, the proposed algorithm completely eliminates the dependence on $\log T$ and substantially reduces the dependence on $1/\epsilon$.

2.2. THE PARALLEL ALGORITHM AdaptMCSC

From a technical perspective, the AdaptMCSC algorithm combines the *sequential sampling* technique [8] with the *threshold-greedy* strategy introduced in [9]. To imple-

ment this idea, the dissertation proposes a subroutine denoted by THRESHSEQ, which selects a set of elements satisfying a prescribed marginal-gain-density threshold and adds high-value elements to the solution.

In the main algorithm, the threshold-greedy strategy from [9] is adapted so that only a constant number of sequential rounds is performed at each stage. In each round, the algorithm invokes THRESHSEQ as a subroutine to select a set of elements whose marginal-gain density is no smaller than an appropriate threshold. Finally, the dissertation employs a partition of the ground set V to control the ratio $c_{\max}/c_{\min}$, thereby deriving tight bounds on the query and adaptive complexities of the algorithm.

a. Algorithm Description

Based on the above ideas, the AdaptMCSC algorithm consists of the following two phases.

Phase 1: Finding a Feasible Solution with a Bound on the Optimal Cost

In the first phase, the algorithm handles the ratio $c_{\max}/c_{\min}$ and partitions the ground set V into three subsets V_1, V_0, and V'. The partition guarantees that every optimal solution excludes the elements in V_1, whereas the elements in V_0 can be added directly to the final solution at a small cost.

Phase 2: Solving the Subproblems

In the second phase, the algorithm solves the subproblem defined by

$$(g_{V_0}, V', T - g(V_0)),$$

where

$$g_{V_0}(S) = g(V_0 \cup S) - g(V_0),$$

by repeatedly invoking AdaptMCSCopt, an auxiliary procedure used when an estimate of the optimal cost is available, with different estimates of the optimal cost.

Let opt' denote the cost of an optimal solution to this subproblem. Since

$$c'_{\min} \leq \mathsf{opt}' \leq nc'_{\max},$$

a sequence of suitable estimates for opt' can be defined as

$$v_i = \frac{c'_{\min}}{(1-\epsilon)^i}.$$

For each value v_i, the algorithm invokes AdaptMCSCopt to obtain a candidate solution. Among all candidate solutions, it selects the minimum-cost solution S satisfying

$$g(S) \geq (1-\lambda)\big(T - g(V_0)\big).$$

Finally, the AdaptMCSC algorithm returns

$$S \cup V_0$$

as the output solution to the original MCSC problem. The complete procedure of the main algorithm AdaptMCSC is presented in Algorithm 2.1.

Algorithm 2.1: The AdaptMCSC Algorithm

Input: (g, V, T), $\epsilon \in (0, 1/2)$, $\lambda \in (0, 1)$, and $\delta \in (0, 1)$.

Output: A subset S

1. $S \leftarrow \emptyset$, $\epsilon' \leftarrow 4\epsilon/23$
2. Sort $V = \{u_1, u_2, \ldots, u_{|V|}\}$ in nonincreasing order of cost
3. $j \leftarrow \min\{i : g(\{u_1, u_2, \ldots, u_i\}) \geq T\}$
4. $c'_{\min} \leftarrow \epsilon' c(u_j)/n$, $c'_{\max} \leftarrow jc(u_j)$
5. $V_0 \leftarrow \{u \in V : c(u) < c'_{\min}\}$
6. $V_1 \leftarrow \{u \in V : c(u) > c'_{\max}\}$
7. $V' \leftarrow V \setminus (V_0 \cup V_1)$, $T' \leftarrow T - g(V_0)$, $g'(\cdot) \leftarrow g_{V_0}(\cdot)$
8. **for** $i = 0$ *to* $\lceil \log_{1/(1-\epsilon')}(n^3/\epsilon') \rceil$ ***in parallel*** **do**
9. $\quad v_i \leftarrow c'_{\min}/(1-\epsilon')^i$
10. $\quad S(i) \leftarrow \mathsf{AdaptMCSCopt}(V', g', T', \epsilon', \lambda, v_i)$
11. **end**
12. $S \leftarrow \arg\min\{c(S(i)) : g(S(i)) \geq T'(1-\lambda), i = 0, \ldots, \lceil \log_{1/(1-\epsilon')}(n^3/\epsilon') \rceil\}$
13. $S_{\text{final}} \leftarrow S \cup V_0$
14. **return** S_{final}

Theorem 2.1. *For $\delta \in (0, 1)$, $\lambda \in (0, 1)$, and $\epsilon \in (0, 1/2)$, Algorithm 2.1 returns a solution S satisfying*

$$g(S) \geq T(1-\lambda), \qquad \mathbb{E}[c(S)] \leq (1+\epsilon)\left(1 + \ln\left(\frac{1}{\lambda}\right)\right)\mathsf{opt},$$

with query complexity

$$O\left(\frac{n^2}{\epsilon^3}\log\left(\frac{1}{\lambda}\right)\log^2\left(\frac{n}{\epsilon}\right)\right),$$

and adaptive complexity

$$O\left(\frac{1}{\epsilon^2}\log\left(\frac{1}{\lambda}\right)\log\left(\frac{n}{\epsilon}\right)\right).$$

2.3. EXPERIMENTAL EVALUATION

The dissertation experimentally evaluates the proposed algorithm against state-of-the-art algorithms for the Influence Threshold (IT) problem using the datasets listed in Table 2.2. The evaluation considers the following criteria:

(1) Solution quality, measured by the total cost of the returned solution required to achieve the prescribed influence threshold;

(2) Adaptive complexity, measured by the number of sequential data-access rounds under the parallel computation model;

(3) The empirical running time of the algorithms.

Table 2.2: Statistics of the datasets used in the experiments

Network	Source	Number of nodes (n)	Number of edges (m)
EmailEuCore	SNAP	1,005	25,571
EgoFacebook	SNAP	4,039	88,234
NetHEPT	SNAP	15,233	62,774

The algorithms are implemented in C++ and parallelized using the OpenMP library. The experiments are conducted on a high-performance computing (HPC) system with the following configuration: `partition=large`, `#threads(CPU)=64`, `nodes=2`, and a maximum memory capacity of 3,073 GB. We set $\epsilon = 0.1$ for all algorithms.

The experimental results show that MINSMC and AdaptMCSC not only maintain solution quality comparable to that of the greedy method but also substantially reduce the number of required sequential rounds. In particular, the proposed AdaptMCSC algorithm achieves the smallest number of sequential rounds in most experimental settings and provides superior running-time performance, especially on large-scale datasets. These results are consistent with the theoretical analysis of the algorithm.

CHAPTER 3
MINIMUM-COST k-SUBMODULAR COVER PROBLEM

3.1. MOTIVATION AND PROBLEM FORMULATION

In many practical applications, simply adding an element to a solution S is insufficient to fully capture the underlying structure of the problem. For example, in the Influence Threshold problem, each user may initially be influenced by different topics. Similarly, in sensor placement problems, different types of sensors may measure different quantities, such as temperature, light, and humidity.

Therefore, to model these situations more naturally and flexibly, we employ the class of k-submodular functions, in which each element can be assigned to one of k different states or labels.

Let $V = \{e_1, e_2, \ldots, e_n\}$ be a ground set and let k be a positive integer. Denote $[k] = \{1, 2, \ldots, k\}$ and define

$$(k+1)^V = \{(V_1, V_2, \ldots, V_k) \mid V_i \subseteq V,\ \forall i \in [k],\ V_i \cap V_j = \emptyset,\ \forall i \neq j\}.$$

Thus, $(k+1)^V$ is the family of all tuples consisting of k pairwise disjoint subsets of V. Each such tuple is called a **k-set**.

For $\mathbf{x} = (X_1, X_2, \ldots, X_k) \in (k+1)^V$, we define

$$supp_i(\mathbf{x}) = X_i, \qquad supp(\mathbf{x}) = \bigcup_{i \in [k]} X_i.$$

The set X_i is called the **i-th component of $\mathbf{x}$**. In addition, let $\mathbf{0} = (\emptyset, \ldots, \emptyset)$ denote the empty k-set.

If $e \in X_i$, we write $\mathbf{x}(e) = i$ and say that i is the **position** or **label** of element e in $\mathbf{x}$. Conversely, if $e \notin supp(\mathbf{x})$, we define $\mathbf{x}(e) = 0$. The operation of adding an element $e \notin supp(\mathbf{x})$ to X_i is denoted by $\mathbf{x} \sqcup (e, i)$. A k-set can also be represented as $\mathbf{x} = \{(e_1, i_1), (e_2, i_2), \ldots, (e_t, i_t)\}$, where $e_j \in supp(\mathbf{x})$ and $i_j = \mathbf{x}(e_j)$ for every $1 \leq j \leq t$. If $X_i = \{e\}$ and $X_j = \emptyset$ for every $j \neq i$, then $\mathbf{x}$ is simply denoted by (e, i).

For two k-sets $\mathbf{x} = (X_1, X_2, \ldots, X_k)$ and $\mathbf{y} = (Y_1, Y_2, \ldots, Y_k)$ in $(k+1)^V$, we write $\mathbf{x} \sqsubseteq \mathbf{y}$ if and only if $X_i \subseteq Y_i$ for every $i \in [k]$. For simplicity, we assume that g is nonnegative, i.e., $g(\mathbf{x}) \geq 0$ for every $\mathbf{x} \in (k+1)^V$, and normalized, i.e., $g(\mathbf{0}) = 0$.

A function $g : (k+1)^V \mapsto \mathbb{R}_+$ is called *k-submodular* if, for every $\mathbf{x} = (X_1, X_2, \ldots, X_k)$ and $\mathbf{y} = (Y_1, Y_2, \ldots, Y_k)$ in $(k+1)^V$, we have $g(\mathbf{x}) + g(\mathbf{y}) \geq g(\mathbf{x} \sqcap \mathbf{y}) + g(\mathbf{x} \sqcup \mathbf{y})$, where

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

and

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

Suppose that each element $e \in V$ is assigned a positive cost $c(e)$. The total cost of a solution $\mathbf{x}$ is defined as $c(\mathbf{x}) = \sum_{i=1}^{k} \sum_{e \in X_i} c(e)$. In this chapter, the dissertation investigates the Minimum-Cost k-Submodular Cover problem (kSC), formally defined as follows.

Định nghĩa 3.1 (Minimum-Cost k-Submodular Cover Problem (kSC))**.** Given a threshold $T > 0$, the kSC problem aims to find a solution $\mathbf{x} = (X_1, X_2, \ldots, X_k) \in (k+1)^V$ with minimum total cost $c(\mathbf{x})$ such that $g(\mathbf{x}) \geq T$.

The kSC problem has various applications in information diffusion and revenue maximization, as described below.

Application 1: k-Topic Influence Threshold Problem (ITk). In viral marketing and product recommendation scenarios, a company may wish to conduct an advertising campaign involving k different product groups and influence at least T users in a social network. To minimize the total budget, the company must select a collection of influential users capable of generating the desired level of information diffusion. This model naturally extends classical single-topic information diffusion models, corresponding to $k = 1$, as well as the widely studied Influence Threshold problem.

Application 2: Minimum-Cost Revenue Threshold Problem (RT). Let $G = (V, E)$ be a graph, where V is the set of vertices representing customers and each edge $(u, v) \in E$ has a weight $w_{uv} \geq 0$ representing the strength of the relationship between customers u and v. The objective is to allocate k types of products to customers by determining a k-set $\mathbf{s} = (S_1, S_2, \ldots, S_k)$, where $S_i \subseteq V$ is the set of customers assigned product type i, and $S_1, \ldots, S_k$ are pairwise disjoint. Thus, each customer is assigned at most one product type.

The revenue function of an allocation $\mathbf{s}$ is defined by

$$g(\mathbf{s}) = \sum_{u \in V} \sum_{i=1}^{k} \left(\sum_{v \in S_i} w_{uv} \right)^{\alpha_{u,i}}.$$

The objective is to find a minimum-cost allocation whose revenue reaches a prescribed threshold T.

Contributions of the Dissertation. The dissertation investigates the kSC problem in two main settings: (1) the *uniform-cost* setting and (2) the *general-cost* setting.

a. Uniform-Cost k-Submodular Cover

For the uniform-cost setting, the dissertation proposes the FBiU algorithm and analyzes its theoretical bicriteria approximation guarantees. Specifically, the algorithm returns a solution $\mathbf{s}$ whose size is bounded relative to that of an optimal solution while simultaneously guaranteeing that the achieved utility is sufficiently close to the prescribed threshold.

Table 3.1 compares the proposed FBiU algorithm with the streaming algorithm introduced in [10]. The results show that FBiU provides a substantially stronger bicriteria approximation guarantee. More precisely, it returns a solution satisfying $|supp(\mathbf{s})| \leq (1+\epsilon)\left(1+\ln\left(\frac{1}{\delta}\right)\right)\,\mathsf{opt}$ and $g(\mathbf{s}) \geq \frac{1-\delta}{2}T$, where opt denotes the size of an optimal solution. Compared with the previous streaming algorithm, this result substantially improves the theoretical guarantees while maintaining polynomial query complexity.

Table 3.1: Algorithms for the kSC problem. A pair (x, y) represents a bicriteria approximation guarantee, meaning that the algorithm returns a solution $\mathbf{s}$ satisfying $|supp(\mathbf{s})| \leq x \cdot \mathsf{opt}$ and $g(\mathbf{s}) \geq y \cdot T$, where $x, y > 0$. Compared with the streaming algorithm in [10], the proposed FBiU algorithm achieves a substantially stronger bicriteria approximation guarantee.

Algorithm	**Approximation Guarantee**	**Query Complexity**
FBiU (Algorithm 3.1, Ours)	$\left((1+\epsilon)(1+\ln(\frac{1}{\delta})), \frac{1-\delta}{2}\right)$	$O\left(\frac{nk\log^2(n)}{\epsilon}\log\left(\frac{n}{\delta}\right)\right)$
Streaming algorithm [10]	$\left(\frac{1-\epsilon}{2\epsilon}, \frac{1-\epsilon^2}{2}\right)$	$O\left(nk\frac{\log(n)}{\epsilon}\right)$

b. General-Cost k-Submodular Cover

For the general-cost setting, the dissertation proposes two polynomial-time bicriteria approximation algorithms for the kSC problem: the *Extended Greedy* algorithm and the FBi algorithm. These algorithms are designed to improve the theoretical guarantees on solution quality while reducing the query complexity compared with existing methods.

Specifically, the *Extended Greedy* algorithm has query complexity $O(n^2k)$ and achieves a bicriteria approximation guarantee of

$$\left(\frac{1}{2}\left(\ln\left(\frac{1}{\epsilon}\right)+\frac{1+\epsilon}{\epsilon}\right), \frac{1-\epsilon}{2}\right)$$

when the utility function g is monotone, and

$$\left(\frac{1}{3}\left(\ln\left(\frac{1}{\epsilon}\right)+\frac{2+\epsilon}{\epsilon}\right), \frac{1-\epsilon}{3}\right)$$

when g is non-monotone.

In addition, the dissertation proposes the FBi algorithm for the monotone setting. It has query complexity $O\left(\frac{nk}{\epsilon^2}\log\left(\frac{n}{\epsilon}\right)\right)$ and achieves a bicriteria approximation guarantee comparable to that of *Extended Greedy*.

Table 3.2 compares the proposed algorithms with the current state-of-the-art algorithms for the kSC problem. The proposed algorithms provide stronger cost guarantees than the streaming algorithms in [11], while removing the dependence of their query complexities on numerical parameters such as the cost ratio and the threshold.

Algorithm	Approximation Guarantee	Query Complexity	Algorithm Type
Monotone Functions			
Extended Greedy (Ours)	$\left(\frac{1}{2}(\log(\frac{1}{\epsilon})+\frac{1+\epsilon}{\epsilon}), \frac{1-\epsilon}{2}\right)$	$O(n^2k)$	Offline
FBi (Ours)	$\left(\frac{1}{2}(1+\epsilon)(\ln(\frac{1}{\delta})+\frac{\delta+1}{\delta}), \frac{1-\delta}{2}\right)$	$O(\frac{nk}{\epsilon^2}\log(\frac{n}{\epsilon}))$	Offline
Algorithm 2 [11]	$\left(\frac{3-\epsilon}{2\epsilon(1-\epsilon)}, \frac{1-\epsilon}{2}\right)$	$O\left(nk\frac{\log(\frac{c_{\min}}{nc_{\max}})}{\log(1-\epsilon)}\right)$	Streaming
Algorithm 3 [11]	$\left(\frac{3-\epsilon}{2\epsilon(1-\epsilon)}, \frac{1-\epsilon}{2}\right)$	$O\left(nk\frac{\log(\epsilon T(B\max_{e\in V, i\in[k]} g((e,i)))^{-1})}{\log(1-\epsilon)}\right)$	Streaming
Non-Monotone Functions			
Extended Greedy (Ours)	$\left(\frac{1}{3}(\log(\frac{1}{\epsilon})+\frac{2+\epsilon}{\epsilon}), \frac{1-\epsilon}{3}\right)$	$O(n^2k)$	Offline
Algorithm 2 [11]	$\left(\frac{4-\epsilon}{3\epsilon(1-\epsilon)}, \frac{1-\epsilon}{3}\right)$	$O\left(nk\frac{\log(\frac{c_{\min}}{nc_{\max}})}{\log(1-\epsilon)}\right)$	Streaming
Algorithm 3 [11]	$\left(\frac{4-\epsilon}{4\epsilon(1-\epsilon)}, \frac{1-\epsilon}{3}\right)$	$O\left(nk\frac{\log(\epsilon T(B\max_{e\in V, i\in[k]} g((e,i)))^{-1})}{\log(1-\epsilon)}\right)$	Streaming

Table 3.2: Algorithms for the kSC problem. A pair (x, y) represents an (x, y)-bicriteria approximation guarantee. The algorithms proposed in this dissertation, namely Extended Greedy and FBi, are **offline** algorithms and therefore assume full access to the ground set. In contrast, the algorithms of Wang *et al.* (2023) are streaming algorithms that process elements sequentially using limited memory and a limited number of passes.

3.2. ALGORITHM FOR THE UNIFORM-COST SETTING

The FBiU algorithm for the uniform-cost setting is presented in Algorithm 3.1.

Algorithm 3.1: The FBiU Algorithm

Input: An instance (V, g, k, T), where g is monotone; $\epsilon \in (0, 1/2)$ and $\delta \in (0, 1)$.

Output: A solution $\mathbf{s}$.

$O \leftarrow \{(1+\epsilon)^i : i \in \mathbb{N},\ 1 \le (1+\epsilon)^i \le n\}$

foreach $v \in O$ **do**

  $\mathbf{s}_v \leftarrow \mathsf{FBiUOpt}(V, g, k, v, \delta)$

end

$\mathbf{s} \leftarrow \arg\max\left\{g(\mathbf{s}_v) : |supp(\mathbf{s}_v)| \le v\left(1+\ln(\frac{1}{\delta})\right),\ v \in O\right\}$

return s

The FBiU algorithm first constructs the candidate set

$$O = \{(1+\epsilon)^i : 1 \leq (1+\epsilon)^i \leq n,\ i \in \mathbb{N}\}$$

of possible estimates v. For each $v \in O$, the algorithm invokes the subroutine FBiUOpt(V, g, k, v, δ) to compute a candidate solution $\mathbf{s}_v$. After generating all candidate solutions, FBiU selects the solution with the largest objective value among those satisfying $|supp(\mathbf{s}_v)| \leq v\left(1 + \ln\left(\frac{1}{\delta}\right)\right)$ (Line 5). Finally, the algorithm returns the selected solution $\mathbf{s}$.

Theorem 3.1. *For every $\epsilon, \delta \in (0,1)$, with probability at least $1-\delta$, Algorithm 3.1 returns a solution with bicriteria approximation guarantee $\left((1+\epsilon)\left(1+\ln\left(\frac{1}{\delta}\right)\right), \frac{1-\delta}{2}\right)$ and query complexity $O\left(\frac{nk\log^2(n)}{\epsilon}\log\left(\frac{n}{\delta}\right)\right)$.*

3.3. ALGORITHMS FOR THE GENERAL-COST SETTING

In this section, the dissertation proposes algorithms for the general-cost kSC problem, including: (i) the Extended Greedy algorithm; and (ii) the FBi algorithm, which improves the running time of Extended Greedy for monotone functions.

3.3.1. The Extended Greedy Algorithm

The proposed algorithm extends the classical greedy strategy to handle a k-submodular objective function. The complete procedure is presented in Algorithm 3.2.

Algorithm 3.2: Extended Greedy for the kSC Problem

Input: An instance (V, g, k, T) and a parameter $\epsilon \in (0,1)$.

Output: A solution $\mathbf{s}$.

1: $\mathbf{s} \leftarrow \mathbf{0}$; $\quad \alpha \leftarrow \begin{cases} \frac{1}{2}, & \text{if } g \text{ is monotone,} \\ \frac{1}{3}, & \text{if } g \text{ is non-monotone.} \end{cases}$

2: Define $h : (k+1)^V \mapsto \mathbb{R}_+$ by $h(\cdot) \leftarrow \min\{g(\cdot), T\}$

3: **while** $h(\boldsymbol{s}) < \alpha(1-\epsilon)T$ **do**

4: $\quad (e,i) \leftarrow \arg\max_{e \in V \setminus supp(\mathbf{s}),\, i \in [k]} \frac{\Delta_{(e,i)} h(\mathbf{s})}{c(e)}$

5: $\quad \mathbf{s} \leftarrow \mathbf{s} \sqcup (e,i)$

6: **end**

7: **return s**

Theorem 3.2. *For every $\epsilon \in (0,1)$, Algorithm 3.2 has query complexity $O(n^2 k)$ and achieves the following bicriteria approximation guarantees:*

- *If g is monotone, the algorithm achieves* $\left(\frac{1}{2}\left(\ln\left(\frac{1}{\epsilon}\right)+\frac{1+\epsilon}{\epsilon}\right), \frac{1-\epsilon}{2}\right)$.

- *If g is non-monotone, the algorithm achieves* $\left(\frac{1}{3}\left(\ln\left(\frac{1}{\epsilon}\right)+\frac{2+\epsilon}{\epsilon}\right), \frac{1-\epsilon}{3}\right)$.

3.3.2. The Fast Bicriteria Algorithm FBi

The algorithm consists of the following two phases.

Phase 1: Ground-set reduction. The algorithm constructs two subsets V_0 and V' from V to retain a sufficiently rich search space while controlling the dependence of the complexity on the ratio $c_{\max}/c_{\min}$.

Phase 2: Candidate generation and selection. The algorithm repeatedly invokes FBiOpt, a proposed subroutine that assumes an estimate of the optimal solution cost, on V' using different estimates v_i. It obtains candidate solutions $\mathbf{s}_i$ and returns the best candidate combined with $\mathbf{x}_0$.

Theorem 3.3. *For every $\epsilon \in (0, 1/2)$ and $\delta \in (0, 1)$, suppose that g is a monotone k-submodular function. Then Algorithm 3.3 returns a solution with bicriteria approximation guarantee* $\left(\frac{1}{2}(1+\epsilon)\left(\ln\left(\frac{1}{\delta}\right)+\frac{1+\delta}{\delta}\right), \frac{1-\delta}{2}\right)$ *and query complexity* $O\left(\frac{nk}{\epsilon^2}\log\left(\frac{n}{\epsilon}\right)\right)$.

3.3.3. Experimental Evaluation

To evaluate the effectiveness of the proposed algorithms, namely Extended Greedy and FBi, the dissertation compares them with the state-of-the-art algorithms listed in Table 3.2. The comparison is based on three important criteria: (i) the quality of the returned solutions; (ii) the number of queries to the objective function g, which reflects the computational cost of the algorithms; and (iii) the empirical running time.

The dissertation compares Extended Greedy and FBi with existing state-of-the-art methods on two representative applications of the kSC problem: (i) the *k-Topic Influence Threshold problem (ITk)*; and (ii) the *Revenue Threshold problem (RT)*.
The baseline algorithms are selected from the best-performing methods reported in the literature, as summarized in Table 3.2. Four standard social-network datasets from SNAP are used[1].

The results on multiple real-world and synthetic datasets show that the proposed algorithms consistently maintain solution quality comparable to that of existing state-of-the-art algorithms. At the same time, the proposed methods substantially reduce the number of function queries and the running time compared with the algorithms of Wang *et al.*, demonstrating superior computational efficiency and strong potential for large-scale applications.

[1] `https://snap.stanford.edu`

Algorithm 3.3: The FBi Algorithm (Fast Bicriteria)

Input: An instance (V, g, k, T), where g is monotone; $\epsilon \in (0, 1/2)$ and $\delta \in (0, 1)$.

Output: A solution $\mathbf{s}$.

$\mathbf{x} \leftarrow \mathbf{0}$; $\quad \epsilon' \leftarrow \epsilon/8$; $\quad \alpha \leftarrow 1/2$; $\quad \mathbf{x}_0 \leftarrow \mathbf{0}$

$\mathbf{x} \leftarrow$ a solution to $\mathbf{u}kS$ on (g, V, k) obtained using the $1/2$-approximation algorithm in [12]

foreach $e \in V \setminus supp(\boldsymbol{x})$ **do**

  Choose $i \in [k]$ uniformly at random and set $\mathbf{x} \leftarrow \mathbf{x} \sqcup (e, i)$

end

Sort the elements of $supp(\mathbf{x}) = \{e_1, \ldots, e_n\}$ in nondecreasing order of cost

$j \leftarrow \min\left\{i : g\big(\{(e_1, \mathbf{x}(e_1)), \ldots, (e_i, \mathbf{x}(e_i))\}\big) \geq T\right\}$

$c_m \leftarrow \epsilon' c(e_j)/n$; $\quad c_M \leftarrow j\, c(e_j)$

$V_0 \leftarrow \{e \in V : c(e) < c_m\}$; $\quad V' \leftarrow \{e \in V : c_m \leq c(e) \leq c_M\}$

$\mathbf{x}_0 \leftarrow$ a solution to $\mathbf{u}kS$ on (g, V_0, k) obtained using the $1/2$-approximation algorithm in [12]

$T_0 \leftarrow g(\mathbf{x}_0)$; $\quad$ Define $g'(\cdot) \leftarrow g(\mathbf{x}_0 \sqcup \cdot) - g(\mathbf{x}_0)$

$T' \leftarrow T - T_0/\alpha$

for $i = 0$ **to** $\left\lceil \log_{\frac{1}{1-\epsilon'}} \left(\frac{jn^2}{\epsilon'}\right) \right\rceil$ **do**

  $v_i \leftarrow c_m/(1-\epsilon')^i$

  $\mathbf{s}_i \leftarrow$ the solution returned by FBiOpt with input $(V', g', k, T', \epsilon', \delta, v_i)$

end

$\mathbf{s} \leftarrow \arg\min\left\{c(\mathbf{s}_i) : g(\mathbf{s}_i \sqcup \mathbf{x}_0) \geq (1-\delta)T/2\right\}$

return $\mathbf{s} \sqcup \mathbf{x}_0$

CHAPTER 4
SUBMODULAR MAXIMIZATION WITH FAIR COST CONSTRAINTS

4.1. PROBLEM FORMULATION AND CONTRIBUTIONS OF THE DISSERTATION

In many practical applications, particularly *influence maximization in social networks*, the elements of the ground set can be partitioned into different user groups according to characteristics such as geographical location, interests, age, or demographic attributes. In such settings, selecting a seed set is subject not only to an overall budget constraint but also to fairness requirements among different groups. The Submodular Maximization problem with Fair Cost Constraints (SMFC) is formally defined as follows.

Definition 4.1 (Submodular Maximization with Fair Cost Constraints (SMFC))**.** Let V be a ground set, let $f : 2^V \to \mathbb{R}_+$ be a normalized monotone submodular function, let $B > 0$ be an overall budget, and let $\mathcal{C} = \{C_1, C_2, \ldots, C_K\}$ be a partition of V into K groups. Let $\alpha \in (0, 1)$ be a fairness parameter. Each element $u \in V$ has a positive cost $c(u)$.

The objective is to find a subset $S \subseteq V$ that maximizes $f(S)$ subject to the following constraints:

1. The total cost of the selected elements in each group does not exceed αB: $c(S \cap C_i) \leq \alpha B, \qquad \forall i \in [K]$.
2. The total cost of the selected set does not exceed the overall budget: $c(S) \leq B$.

For an element $e \in V$, let $C(e)$ denote the unique group containing e.

Application to Influence Maximization. In many practical applications of viral marketing and information diffusion in social networks, users are commonly divided into different groups based on geographical location, age, interests, or demographic attributes. In advertising campaigns or public information dissemination, organizers aim not only to maximize the spread of information but also to ensure that the campaign budget is allocated fairly among different user groups.

Therefore, the problem is to select a seed set that maximizes the expected information spread in the social network while ensuring an appropriate and fair allocation of the available budget among user groups.

Contributions of the Dissertation. The dissertation investigates the SMFC problem in two settings: *(i) the uniform-cost setting*, in which every element has unit cost,

and *(ii) the general-cost setting*, in which each element may have an arbitrary positive cost.

Table 4.1: Summary of the main results of the dissertation for the Submodular Maximization problem with Fair Cost Constraints (SMFC)

Algorithm	Setting	Approximation Ratio	Query Complexity
TGFastU	Uniform cost ($c(e) = 1$)	$1/2 - \epsilon$	$O\left(\frac{n}{\epsilon} \log \frac{k}{\epsilon}\right)$
TGFast	General cost ($c(e) > 0$)	$\frac{\alpha}{\alpha+2} - \epsilon$	$O\left(\frac{n}{\epsilon} \log \left(\frac{n(\alpha+2)}{2\alpha}\right)\right)$

In the above table, $n = |V|$ denotes the size of the ground set, k is the cardinality budget in the uniform-cost setting, $\alpha \in (0, 1)$ is the fairness parameter, and $\epsilon > 0$ is an accuracy parameter controlling the trade-off between solution quality and computational complexity.

4.2. PROPOSED ALGORITHMS

4.2.1. Algorithm for the Uniform-Cost Setting

For the uniform-cost setting, the dissertation proposes the *Threshold Greedy* algorithm, denoted by TGFastU. The algorithm adapts the threshold-greedy strategy to select elements with large marginal contributions while ensuring that the group-allocation constraints are not violated. In this setting, these constraints can be modeled as a matroid constraint. The complete pseudocode of TGFastU is presented in Algorithm 4.1.

Theorem 4.1. *Algorithm 4.1 has query complexity* $O\left(\frac{n}{\epsilon} \log \frac{k}{\epsilon}\right)$ *and returns a* $(1/2 - \epsilon)$*-approximate solution to the* SMFC *problem in the uniform-cost setting.*

4.2.2. Algorithm for the General-Cost Setting

For the general-cost setting, the dissertation proposes the TGFast algorithm, which achieves an approximation ratio of $\frac{\alpha}{\alpha+2} - \epsilon$, where $\epsilon \in (0, 1)$ is an arbitrarily small constant. The TGFast algorithm extends the uniform-cost algorithm TGFastU by selecting elements according to their *marginal-gain density*, defined as the ratio between the marginal contribution of an element and its cost. This approach enables the algorithm to select elements with large contributions while ensuring that neither the overall budget constraint nor the group-level cost constraints are violated.

The algorithm initializes the solution as $S = \emptyset$ and first identifies an individually feasible element $e_{\max} = \arg\max_{e \in V_{\text{feas}}} f(\{e\})$ with the largest singleton value. It then sets $M = f(\{e_{\max}\})$. The main procedure employs an outer loop controlled by a density threshold θ. Initially, θ is set proportionally to M, and its value is reduced by a factor of

Algorithm 4.1: The TGFastU Algorithm

Input: A monotone submodular function f, a collection of groups $\mathcal{C} = \{C_1, C_2, \ldots, C_K\}$, a fairness parameter $\alpha \in (0, 1)$, a cardinality budget k, and an accuracy parameter $\epsilon > 0$.

Output: A solution set S.

$S \leftarrow \emptyset$;
$M \leftarrow \max_{e \in V} f(\{e\})$;
$\theta \leftarrow M$;
while $\theta \geq \epsilon M/k$ **do**
  foreach $e \in V \setminus S$ **do**
    if $|S \cap C(e)| + 1 \leq \alpha k$ ***and*** $|S| + 1 \leq k$ **then**
      if $f(S \cup \{e\}) - f(S) \geq \theta$ **then**
        $S \leftarrow S \cup \{e\}$;
      end
    end
  end
  $\theta \leftarrow (1 - \epsilon)\theta$;
end
return S;

$(1 - \epsilon)$ after each outer iteration.

During each iteration, the algorithm scans the elements in $V_{\text{feas}} \setminus S$. An element e is added to S if its addition violates neither the overall budget constraint nor the group-level cost constraint and if it satisfies

$$\frac{f(e \mid S)}{c(e)} \geq \theta,$$

where

$$f(e \mid S) = f(S \cup \{e\}) - f(S).$$

The algorithm terminates when θ falls below a prescribed lower bound. Finally, it returns the better solution between the constructed set S and the singleton solution $\{e_{\max}\}$.

Theorem 4.2. *Algorithm 4.2 has query complexity* $O\left(\frac{n}{\epsilon} \log\left(\frac{n(\alpha+2)}{2\alpha}\right)\right)$ *and returns a feasible solution with approximation ratio* $\frac{\alpha}{\alpha+2} - \epsilon$ *for the* SMFC *problem, where* $\alpha \in (0, 1)$ *is the fairness parameter and the accuracy parameter* ϵ *satisfies* $0 < \epsilon \leq \frac{\alpha}{\alpha+2}$.

Algorithm 4.2: The TGFast Algorithm

Input: A ground set V, a monotone submodular function $f : 2^V \to \mathbb{R}_+$, a collection of groups $\mathcal{C} = \{C_1, \ldots, C_K\}$, a fairness parameter $\alpha \in (0, 1)$, an overall budget B, and an accuracy parameter $\epsilon > 0$.

Output: A solution set $S \subseteq V$.

$S \leftarrow \emptyset$;
$V_{\text{feas}} \leftarrow \{e \in V : c(e) \leq B,\ c(e) \leq \alpha B\}$;
$e_{\max} \leftarrow \arg\max_{e \in V_{\text{feas}}} f(\{e\})$;
$M \leftarrow f(\{e_{\max}\})$;
$\beta \leftarrow \frac{\alpha}{\alpha+2}$;
$\theta \leftarrow \frac{nM}{2\alpha B}$;
while $\theta \geq \frac{\beta M}{\alpha B}$ **do**
    foreach $e \in V_{\text{feas}} \setminus S$ **do**
        if $c(S) + c(e) \leq B$ **then**
            if $c(S \cap C(e)) + c(e) \leq \alpha B$ **then**
                if $\frac{f(S \cup \{e\}) - f(S)}{c(e)} \geq \theta$ **then**
                    $S \leftarrow S \cup \{e\}$;
                end
            end
        end
    end
    $\theta \leftarrow (1 - \epsilon)\theta$;
end
$S' \leftarrow \arg\max_{X \in \{S, \{e_{\max}\}\}} f(X)$;
return S'

4.3. EXPERIMENTAL EVALUATION

In this section, the dissertation experimentally evaluates the proposed algorithms TGFastU and TGFast for the SMFC problem using Influence Maximization as the application. Specifically, the proposed algorithms are compared with existing methods according to the following two criteria: (i) *solution quality*, measured by the achieved influence spread; and (ii) the *running time* of the algorithms.

The experiments compare the proposed algorithms with OPIM, a state-of-the-art algorithm for the Influence Maximization problem (IM). The comparison is conducted in terms of solution quality, measured by influence spread, and running time. The dissertation uses publicly available social-network datasets from SNAP[1].

In the **uniform-cost setting**, OPIM achieves a slightly higher influence spread than the other algorithms. However, the difference between OPIM and TGFastU is relatively small. In contrast, TGFastU maintains a substantially lower running time, which grows almost linearly with the budget and remains significantly smaller than that of OPIM.

In the **general-cost setting**, TGFast generally achieves the highest objective value. Its running time changes only slightly as the budget increases and remains substantially lower than that of OPIM. Overall, the experimental results demonstrate that the proposed algorithms achieve competitive or superior solution quality while maintaining substantially lower running times. In particular, the advantage of TGFast becomes more pronounced as the dataset size increases. This improvement primarily results from the efficient threshold-based selection mechanism employed by TGFast, which is considerably faster than existing greedy approaches.

[1] https://snap.stanford.edu/data/

CONCLUSIONS AND RECOMMENDATIONS

This dissertation focuses on a class of submodular optimization (SO) problems encompassing the following three problems: (1) Minimum-Cost Submodular Cover (MCSC), (2) Minimum-Cost k-Submodular Cover (kSC), and (3) Submodular Maximization with Fair Costs (SMFC). This class of problems can flexibly model a wide range of real-world applications, including influence maximization in social networks, feature selection, recommender systems, marketing budget allocation, and resource allocation under fairness constraints. Studying these models is not only theoretically significant but also provides a foundation for designing modern data analytics systems. The main results of the dissertation are summarized as follows:

1. **For the** MCSC **problem:** The dissertation proposes an efficient parallel algorithm for the Minimum-Cost Submodular Cover problem. The algorithm achieves an approximation ratio asymptotically matching the best-known results while significantly improving adaptive complexity and parallelizability. Theoretical analyses establish the solution quality of the proposed algorithm, whereas experiments on real-world datasets demonstrate clear advantages in terms of running time and scalability. This result contributes to narrowing the gap between theoretical guarantees and practical implementation requirements.

2. **For the** kSC **problem:** The dissertation extends the submodular cover model to the k-submodular domain and proposes bicriteria approximation algorithms for both uniform and general costs. The proposed algorithms achieve better approximation guarantees than previous results while substantially reducing the objective-function query complexity. Experimental results show that the proposed methods maintain high solution quality while significantly reducing computational costs, thereby improving their applicability to large-scale data problems.

3. **For the** SMFC **problem:** The dissertation investigates a new optimization model that combines optimization efficiency with fairness requirements in cost allocation. Two approximation algorithms are proposed for the uniform-cost and general-cost settings, respectively, together with theoretical guarantees on their approximation ratios. These results extend the scope of submodular optimization to systems with fairness constraints, thereby contributing to addressing increasingly important requirements in modern data and social network applications.

Despite the significant theoretical and experimental results obtained, the studies

presented in this dissertation still have several limitations. The proposed algorithms are mainly developed and analyzed under static optimization settings and do not fully address the characteristics of extremely large-scale or time-varying data. In addition, the investigated models focus only on several basic forms of cost and fairness constraints, whereas many real-world applications often involve more complex constraints.

In future work, the results of this dissertation can be extended in the following major research directions.

First, submodular and k-submodular optimization algorithms should be developed for large-scale data-processing models, including streaming, parallel, and distributed settings. In addition, techniques for reducing the number of objective-function queries should be investigated to improve the applicability of these algorithms to extremely large datasets.

Second, the current optimization models should be extended to accommodate more complex forms of constraints, such as multiple simultaneous budget constraints, multi-level fairness requirements, and combined constraints arising in real-world applications.

Third, optimization problems in dynamic and uncertain environments should be investigated, where data, costs, or system parameters may change over time. This research direction has considerable application potential in social networks, recommender systems, and modern data platforms.

LIST OF THE PUBLICATIONS RELATED TO THE DISSERTATION

1. **Hue T. Nguyen**, Dung T.K. Ha, Canh V. Pham. *Efficient Parallel Algorithm for Minimum Cost Submodular Cover Problem with Lower Adaptive Complexity*. Asia-Pacific Journal of Operational Research , 41(6), 2450005 (2024) (**SCIE**).

2. **Hue T. Nguyen**, Nguyen Long Giang, Canh V. Pham. *Efficient Deterministic Bicriteria Approximation Algorithms for k-Submodular Cover Problem*. Asia-Pacific Journal of Operational Research, (2026) (**SCIE**).

3. **Hue T. Nguyen**, Nguyen Long Giang, Tran D. Tran, Canh V. Pham. *Thuật toán song song hiệu quả cho bài toán Phủ Submodular với chi phí tối thiểu và ứng dụng*. Kỷ yếu Hội thảo Quốc gia lần thứ XXVII Một số vấn đề chọn lọc về Công nghệ thông tin và Truyền thông, Tr. 622-634, 2024.

4. **Hue T. Nguyen**, Bac D Pham, Uyen T. Tran, Nguyen Long Giang, Canh V. Pham. *Influence Maximization with Fairness Allocation Constraint*. In: proceeding of the 13rd International Symposium on Information and Communication Technology (SoICT), 401-414 (**SCOPUS**).

5. **Hue T. Nguyen**, Bac D Pham, Dung TK Ha, Nguyen Long Giang, Canh V. Pham. *Influence Maximization with Fairness Cost on Groups in Online Social Networks*. In: proceeding of the 17th Asian Conference Intelligent Information and Database Systems (ACIIDS) 2025, Kitakyushu, Japan, April 23-25, 2025 (**SCOPUS, Rank B**).

6. **Hue T. Nguyen**, Tan D. Tran, Nguyen Long Giang, Canh V. Pham. *Fast Stochastic Greedy Algorithm for k-Submodular Cover Problem*. In: proceeding of the 14rd International Symposium on Information and Communication Technology (SoICT), pp 173–184 (**SCOPUS**).