TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 05/14 pp1030–1040
DOI: 10.26599/TST.2022.9010061
Volume 28, Number 6, December 2023

Bicriteria Algorithms for Approximately Submodular Cover Under Streaming Model

Yijing Wang, Xiaoguang Yang*, Hongyang Zhang, and Yapu Zhang

Abstract: In this paper, we mainly investigate the optimization model that minimizes the cost function such that the cover function exceeds a required threshold in the set cover problem, where the cost function is additive linear, and the cover function is non-monotone approximately submodular. We study the problem under streaming model and propose three bicriteria approximation algorithms. Firstly, we provide an intuitive streaming algorithm under the assumption of known optimal objective value. The intuitive streaming algorithm returns a solution such that its cover function value is no less than $\alpha(1-\epsilon)$ times threshold, and the cost function is no more than $(2+\epsilon)^2/(\epsilon^2\omega^2)\cdot\kappa$, where κ is a value that we suppose for the optimal solution and α is the approximation ratio of an algorithm for unconstrained maximization problem that we can call directly. Next we present a bicriteria streaming algorithm scanning the ground set multi-pass to weak the assumption that we guess the optimal objective value in advance, and maintain the same bicriteria approximation ratio. Finally we modify the multi-pass streaming algorithm to a single-pass one without compromising the performance ratio. Additionally, we also propose some numerical experiments to test our algorithm's performance comparing with some existing methods.

Key words: approximately submodular; linear additive; streaming model; bicriteria algorithm

1 Introduction

Submodular optimization possesses great position for decades. Besides the important applications in machine learning and data mining, such as data summarization[1, 2], dictionary selection[3], monitor placement[4], influence maximization[5], and many others, there are still some classic combinatorial optimization models, including maximum cut problem, vertex cover problem, facility location problem, etc.

Subject to a variety of limited scenarios, most submodular optimization problems need to describe various restrictions. For example, we need to select a limited number of locations to place the monitors, so that the monitors can cover as much of the effective area as possible in real life. This is actually the cardinality constraint, which can be characterized as the optimization model $\max\{f(S) : S \subseteq G, |S| \leqslant k\}$, whose goal is to choose a subset S from the ground set G to maximize the cover function f. Except from cardinality constraints, there are knapsack constraints, matroid constraints, box constraints, and so on.

Nevertheless, rather than maximizing the objective function, we prefer to reduce the cost under the premise of ensuring that the coverage objective function meets sufficient requirements in many practical applications. In the case of data summarization, it is desirable that

• Yijing Wang and Xiaoguang Yang are with Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing 100190, China. E-mails: yjwang@amss.ac.cn; xgyang@iss.ac.cn.
• Hongyang Zhang is with School of Mathematics and Statistics, Ningbo University, Ningbo 315211, China. E-mail: hongyangz0708@163.com.
• Yapu Zhang is with Beijing Institute for Scientific and Engineering Computing, Beijing University of Technology, Beijing 100124, China. E-mail: zhangyapu@bjut.edu.cn.
∗ To whom correspondence should be addressed.
Manuscript received: 2022-08-02; revised: 2022-10-24; accepted: 2022-11-28

minimizing the total storage memory of the summary to extract a sufficiently good summary rather than only maximizing the summarization information. Inspired by this, scholars propose the cover maximization model. When the cover function satisfies linearity or submodularity, it is also called the linear cover or submodular cover problem. In this work, we focus on the cover problem relating to submodularity. Here we give a brief introduction to some related works.

1.1 Related Work

(1) Submodular cover

In the current field of research, variants of Submodular Cover (SC) problem have been studied when the cover function f is monotone[6–8]. Wolsey[7] provided a $(1 + \ln(\mu/\nu))$-approximation algorithm utilizing the greedy technique, where μ and ν are instance dependent parameters. Motivated by the greedy technique, one designed a $(\ln(1/\epsilon), 1 - \epsilon)$-bicriteria approximation algorithm.

(2) Constrained submodular maximization

In fact, Iyer and Bilmes[9] showed that there exists a close connection between submodular maximization and submodular cover problem. They proved that an algorithm solving submodular maximization problem can be used to solve submodular cover problem as input information, and vice versa. Particularly, a (ρ, σ)-bicriteria approximation algorithm for maximizing a submodular function limited to knapsack restricts can be called as a subroutine to solve the submodular cover problem and get a solution of $((1 + \epsilon)\sigma, \rho)$ approximation. In the meanwhile, the time complexity of the two algorithms can also be measured.

(3) Approximately submodular maximization

A large number of studies show that the optimization objective functions in many applications are not strictly submodular but approximately submodular, such as the active function in influence maximization model, the interpretation expression in deep neural networks, and the sparse regression function in data mining, and so on. Inspired by this, more and more researchers turn to the study of approximately submodularity optimization[10–13].

Harshaw et al.[14] studied an optimization model maximizing the cover function meanwhile minimizing the cost function limited to the cardinality constraints, and they finally acquired a $(1 - 1/\mathrm{e}^{\omega} - \epsilon, 1)$-bicriteria approximation solution, where ω is a parameter to perform a cover function's submodularity approximately. Instead of the greedy technique used in most works, Qian[15] took the multi-objective evolutionary technique to design algorithm for the same optimization model, and obtained a bicriteria ratio of $(1 - 1/\mathrm{e}^{\omega}, 1)$.

(4) Streaming setting

Monotone submodular cover maximization with cardinality constraint has been studied previously under the streaming setting[16]. In the streaming model, if we are given a guess of the optimal solution, we can present a streaming algorithm that outputs a $(2/\epsilon, 1 - \epsilon)$ approximation solution only scanning the streaming one pass. Alternatively, if one may scan the streaming set multi-pass, it is possible to improve the approximation ratio to $(\ln(1/\epsilon), 1 - \epsilon)$.

As far as we know, all previous studies require that the cover function is monotone. However, many applications involve non-monotone cover function. Therefore, it is meaningful and challenging to study the cover problem when the covering function is non-monotone.

The detailed non-monotone approximately submodular cover problem can be formulated as the following model: for two set functions $f : 2^G \to \mathbf{R}_{\geqslant 0}$ and $c : G \to \mathbf{R}_{\geqslant 0}$ defined on the ground set G, where f is approximately submodular but non-monotone, and function c is linear additive, our object is to choose a subset S from the ground set G such that $S := \arg\min_{S \in G}\{\sum_{x \in S} c(x), \quad \text{s.t.}, f(S) \geqslant \tau\}$, given $\tau \leqslant \max\{f(S) : S \subseteq G\}$. Let's take data summarization as an example to illustrate this model. In digital information age, data transfer in a second and it is more and more complex to store the data information. Extracting effective information from huge data is the inevitable demand in current age, which motivates the creation of data summarization. In data summarization, we wish to minimize the total storage of the summary and maintain the sufficiently effective information. This is an intuitive application of approximately submodular cover problem. The detailed definitions of approximately submodular and linear will be provided in Section 2. The common method to solve this kind of optimization model is to construct an approximation algorithm and analyze its bicriteria ratio.

(5) (*a*, *b*)-bicriteria approximation algorithm

We call Algorithm $\mathcal{A}$ is (a, b)-bicriteria approximation if the algorithm returns a solution S such that $c(S)$ is no more than $a \cdot c(S^*)$, and $f(S)$ is no less than $b \cdot \tau$ for the optimization model $\min_{S \subseteq G}\{\sum_{x \in S} c(x) : f(S) \geqslant \tau\}$, where S^* is an optimal solution, ratios $a \geqslant 1$ and $0 < b \leqslant 1$.

1.2 Our contributions

In this paper, we mainly study the non-monotone approximately submodular cover problem, whose goal is to minimize the cost function, such that the cover function value is no less than a required threshold in set function optimization. In this optimization model, we set the cover function approximately submodular as well as non-negative non-monotone, and the cost function is linear additive. We mainly investigate the problem under streaming model and focus on designing streaming algorithms which are suitable for a huge volume of data. Motivated by this, we show three bicriteria approximation algorithms.

• Firstly, we provide an intuitive streaming algorithm under the guess of known optimal objective value. This algorithm returns an $(\alpha(1-\epsilon), (2+\epsilon)^2/(\epsilon^2\omega^2))$-bicriteria approximation solution. In other words, the cover function value of the output set S obeys $f(S) \geqslant \alpha(1-\epsilon)\cdot\tau$ and the cost function value meets $c(S) \leqslant (2+\epsilon)^2/(\epsilon^2\omega^2)\cdot\kappa$, where κ is an approximation value of the optimal solution and α is an algorithm's approximation ratio to solve the unconstrained maximization problem used as a subroutine in our algorithm frame.

• As we know, it is a challenge to guess the optimal objective value in the streaming model. In order to relax this condition, we present a modified algorithm that scans the streaming set multi-pass, and get the same bicriteria approximation ratio as that in former algorithm.

• Further, we adjust the multi-pass streaming algorithm to a single-pass one and keep the performance ratio consistently.

1.3 Paper organization

In Section 2, we introduce some basic concepts of approximately submodular function and formally restate minimum cost cover problem. In Section 3, we provide three streaming algorithms for the approximately submodular cover model. Firstly, we show a detailed stream scanning algorithm and the corresponding analysis based on the condition that we know the optimal objective value in advance in Section 3.1. Then we adapt Algorithm 1 to a multi-pass streaming algorithm in Section 3.2 to avoid the guess that the optimal value is known. Next we modify the multi-pass streaming algorithm to single pass and give a brief analysis in Section 3.3. Besides, we provide some numerical experiments to test the performance of the designed algorithm in Section 4. Finally, we give a conclusion for our work in Section 5.

2 Preliminary

Given the ground set $G = \{x_1, x_2, \ldots, x_n\}$ and a function f denoted on the subsets of G, function f is called set function such that $f : 2^G \to \mathbf{R}$. The non-monotone approximately submodular cover problem is proposed for set functions, which can be modeled as

$$\begin{aligned} &\text{minimize} && \sum_{x\in S} c(x), \\ &\text{s. t.,} && f(S) \geqslant \tau,\ S \subseteq G, \end{aligned}$$

where

• function f is approximately submodular but non-monotone characterizing the cover;

• function c is linear additive characterizing the cost;

• value τ is a required threshold.

Thus the non-monotone approximately submodular cover problem is also called minimum cost cover problem. The detailed definitions of function f and c are provided following.

Given a ground set $G = \{x_1, x_2, \ldots, x_n\}$ and a set function $m : 2^G \to \mathbf{R}$, we call

• function $m : 2^G \to \mathbf{R}$ is monotone if $m(X \cup \{x\}) \geqslant m(X)$ for any subset X and any element x;

• function $m : 2^G \to \mathbf{R}$ is normalized if $m(\varnothing) = 0$;

• function $m : 2^G \to \mathbf{R}$ is submodular if $m(x|X) \leqslant m(x|Y)$, for any $Y \subseteq X \subset G$ and $x \in G \setminus X$, where $m(x|X) := m(X \cup \{x\}) - m(X)$ represents the marginal gain when adding x into X;

• function $m : G \to \mathbf{R}$ is linear additive if $m(X) = \sum_{x\in X} m(x)$.

Benefiting from the works of Lehmann et al.[17] and Kuhnle et al.[13], the expression of approximately submodular function is shown in Definition 1.

Definition 1 For the ground set G and its set function $m : 2^G \to \mathbf{R}$, if there exists the largest parameter $0 < \omega \leqslant 1$ obeying $\omega \times m(x|V_2) \leqslant m(x|V_1)$ for any subsets $V_1 \subseteq V_2 \subset G$ and $x \in G \setminus V_2$, then we call function m is approximately submodular.

Next we restate the detailed definition of (a, b)-bicriteria approximation algorithm.

Definition 2 For the minimum cost cover problem mentioned above

$$\begin{aligned} &\text{minimize} && \sum_{x\in S} c(x), \\ &\text{s. t.,} && f(S) \geqslant \tau,\ S \subseteq G. \end{aligned}$$

Algorithm $\mathcal{A}$ is (a, b)-bicriteria approximately if it gets a subset S obeying $c(S) \leqslant a \cdot c(S^*)$ for the cost function and $f(S) \geqslant b \cdot \tau$ for the cover function, where S^* is an optimal subset for the original model, parameters $a \geqslant 1$

and $0 < b \leqslant 1$.

For an algorithm's complexity, whether in the submodular maximization problem or submodular cover problem, we usually assume there are given oracles to calculate the function value for any subset. Once one has a subset S, he can obtain the function value $f(S)$ if he queries the oracle and the complexity is the number of querying.

3 Streaming Algorithms for Approximately Submodular Cover Problem

In this part, we aim to present approximation algorithms for the minimum cost approximately submodular cover problem under streaming setting. In the streaming model, elements arrive over time. Before data selection, we need to open some buffer space. As the elements arrive, we need to check each element for additional coverage and cost to the current solution set. If it reaches a certain threshold, we store the element in the buffer space. Finally, we choose a subset which has the maximal cover value from all the buffer spaces.

As statement in the introduction, Feige et al.[18] showed that there is a close collection between the problems of submodular maximization and submodular cover. Based on the previous work, especially Crawford[19], we know the following theorem still holds. We refer the reader to the detailed proof in the work of Crawford[19].

Theorem 1 For the problem of non-negative non-monotone approximately submodular cover on the ground set $G = \{x_1, x_2, \ldots, x_n\}$, if Algorithm $\mathcal{A}$ including adaptive algorithm and possibly randomized algorithm can return a subset S, such that the expected cover function value satisfies

$$E[f(S)] \geqslant (1/2 + \epsilon)\tau,$$

for any instance and any parameter $\epsilon > 0$, the oracle query is at least $\Omega(\ln(1+\epsilon)\mathrm{e}^{\epsilon^2 n}/\ln(n))$.

In the next section, we start with a streaming algorithm under the guess of the known optimal objectie value. Then we weak the assumption and design a multi-pass streaming algorithm to maintain the same bicriteria approximation ratio. Next we improve the multi-pass streaming algorithm to single-pass streaming algorithm without compromising the performance ratio.

3.1 Intuitive streaming algorithm for cover problem

In this section, we mainly focus on designing a streaming algorithm and analyzing its performance guarantee. Before describing the specific algorithm execution, we need to obtain some assumptions in advance, which are listed below:

- Initially take a parameter ϵ from $(0, 1)$;
- Guess a possible function value of the optimal solution κ in advance;
- Open some buffer space and break into $2/\epsilon$ disjoint subsets, $S_1, S_2, \ldots, S_{2/\epsilon}$.

Next we give a description of the streaming algorithm high levelly. There are mainly two important algorithmic operations and one comparison.

- As each element x arrives with the streaming set, we check the marginal gain of x to the cover function f relating to the current subset as well as its cost function. If the marginal gain and its cost satisfy some certain threshold, we add the element to the corresponding buffer subset.
- Once scan the whole streaming set or come to the break conditions, we call for an unconstrained maximization algorithm $\mathcal{A}(\alpha)$ which maintains α-approximation on the union of the buffer subsets $S_1, S_2, \ldots, S_{2/\epsilon}$, and output an additional subset S_0.
- Compare all the function values of subsets $S_0, S_1, \ldots, S_{2/\epsilon}$, and select the one which has the maximal cover value.

The detailed pseudocode of Algorithm 1 (Stream-Known-κ) is as follows.

Before we expand our analysis of Algorithm 1, we need some preparatory work. For any random subset

Algorithm 1 Stream-Known-κ

```
Step 1: Give a streaming set $G$, a cover function $f$ satisfying weakly submodularity, a linear cost function $c$, a parameter $\epsilon \in (0,1)$, and an optimal objective value $\kappa$.
Step 2: Construct a series of buffer space $S_1, S_2, \ldots, S_{2/\epsilon}$. Initially set $S_1 := \varnothing, S_2 := \varnothing, \ldots, S_{2/\epsilon} := \varnothing$.
Step 3: For each arriving element $x$, if $c(x) \leqslant \kappa$ and for buffer space sets $S_1, S_2, \ldots, S_{2/\epsilon}$, there exists $S_l$, such that
            $\frac{f(x|S_l)}{c(x)} \geqslant \frac{\epsilon\tau}{2\kappa} \cdot \omega^2$,
        set $S_l := S_l \cup \{x\}$.
Step 4: Repeat Step 3 until there is some subset $S_l$ such that $c(S_l) > 2\kappa/(\epsilon \cdot \omega^2)$. Stop scanning the streaming element and output buffer sets $S_1, S_2, \ldots, S_{2/\epsilon}$.
Step 5: Running an unconstrained maximization algorithm $\mathcal{A}(\alpha)$ which maintains $\alpha$-approximation on the union of the storage subsets $S_1, S_2, \ldots, S_{2/\epsilon}$, and output a subset $S_0$.
Step 6: Output a subset $S_j$ from $\{S_0, S_1, \ldots, S_{2/\epsilon}\}$ which has the maximal cover value, i.e.,
            $S_j := \arg\max\{f(S_0), f(S_1), \ldots, f(S_{2/\epsilon})\}$.
```

S chosed from the ground set G with probability p for each element, we can analyze its expected function value under an approximately submodular function, as shown in Lemma 1.

Lemma 1 For the ground set G and its set function $m : 2^G \to \mathbf{R}$ which satisfies approximately submodularity, we can get a random subset $R \subseteq G$ under the condition that each element is selected with probability p from the ground set G, and denote it by $R(p)$. Then the expectation function value of the random set $R(p)$ is lower bounded by $(1-p)\cdot m(\varnothing)+\omega\cdot p\cdot m(R)$.

Proof We expand the proof by induction with respect to $|R|$.

• Firstly, it is intuitive to hold the lower bound when R is empty set.

• Then, we assume that the value of $R(p)$ keeps the relation when the cardinality of random set R is strictly more than zero. That is,

$$E[m(R'(p))] \geqslant (1-p)\cdot m(\varnothing) + \omega \cdot p \cdot m(R').$$

• Assume another subset $R = R' \cup \{x\}$, and element x is not contained by the random set R'. In this case, we know $R(p) \cap R' \subseteq R'$, in which each element appears under the probability of p and denote $R'(p) = R(p) \cap R'$ for short. Then by the definition of approximately submodularity, we get

$$m(R(p)) - m(R'(p)) \geqslant \\ \omega \cdot (m(R(p) \cup R') - m(R')).$$

Reformulating the inequality, there is

$$m(R(p)) \geqslant \\ m(R'(p)) + \omega \cdot (m(R(p) \cup R') - m(R')).$$

The expectation value of $m(R(p))$ satisfies

$$E[m(R(p))] \geqslant \\ E[m(R'(p)) + \omega \cdot (m(R(p) \cup R') - m(R'))] = \\ E[m(R'(p))] + \omega \cdot E[(m(R(p) \cup R') - m(R'))] = \\ E[m(R'(p))] + \omega \cdot [p \cdot (m(R) - m(R'))+ \\ (1-p)\cdot(m(R') - m(R'))] = \\ E[m(R'(p))] + \omega \cdot p \cdot (m(R) - m(R')) \geqslant \\ (1-p)\cdot m(\varnothing) + \omega \cdot p \cdot m(R') + \omega \cdot p \cdot m(R)- \\ \omega \cdot p \cdot m(R') = \\ (1-p)\cdot m(\varnothing) + \omega \cdot p \cdot m(R).$$

Thus, we prove the lemma. ■

From Lemma 1, we can get a proposition which is useful in the analysis of performance guarantee.

Proposition 1 For subsets $S_1, S_2, \ldots, S_l$, and T in ground set G satisfying $S_i \cap S_j = \varnothing, i \neq j, i, j \in [l]$, and approximately submodular function $m : 2^G \to \mathbf{R}$, there is $i \in \{1, 2, \ldots, l\}$, such that

$$m(S_i \cup T) \geqslant (1 - \frac{1}{l})m(T).$$

Proof Construct set function $g(V) = m(T \cup V)$. Based on the property of function m, we acquire function g is non-negative approximately submodular.

Choosing a subset S uniformly randomly from the disjoint subsets $S_1, S_2, \ldots, S_l$, such that each element of G has probability at most $1/l$ of being in S. Then by Lemma 1, we get

$$\frac{1}{l}\sum_{i=1}^{l} m(T \cup S_i) = \\ \frac{1}{l}\sum_{i=1}^{l} g(S_i) = E[g(S)] \geqslant \\ (1-\frac{1}{l})g(\varnothing) + \omega \cdot \frac{1}{l} \cdot g(S) \geqslant \\ (1-\frac{1}{l})g(\varnothing) = (1-\frac{1}{l}) \cdot m(T).$$ ■

By means of Lemma 1 and Proposition 1, we can acquire the lower bound of function value for the subset S returned by Algorithm 1.

Lemma 2 Suppose that Algorithm 1 runs with input parameter $\epsilon \in (0, 1)$ and κ is no less than the optimal value OPT. Then the output set S returned by Algorithm 1 satisfies $f(S) \geqslant \alpha \cdot (1-\epsilon) \cdot \tau$, where $0 < \alpha < 1$.

Proof From the pseudocode of Algorithm 1, we know there are two break conditions in the routine:

• Condition 1: The cost function value of some buffer subset comes to the limited bound in Step 4;

• Condition 2: There is no element meeting the selection rules in Step 3.

Next we analyze the two cases in detail.

Case 1: Assume an optimal solution set is S^* for the non-negative non-monotone approximately submodular cover problem, and the output solution set of Algorithm 1 is S_l after the l-th iteration. Then by the selection rules in Step 3 and the fact that function f is non-negative, we get the cover value of output set S_l which satisfies the following bound:

$$\begin{aligned}
f(S_l) \geqslant & f(S_l) - f(\varnothing) = \\
& \sum_{\nu=1}^{|S_l|} f(S_l(\nu)) - f(S_l(\nu-1)) = \\
& \sum_{\nu=1}^{|S_l|} (f(S_l(\nu)) - f(S_l(\nu-1))) \geqslant \\
& \sum_{x \in S_l} f(x|S_l(\nu-1)) \geqslant \\
& \sum_{x \in S_l} c(x) \cdot \frac{\epsilon\tau}{2k} \cdot \omega^2 = \\
& c(S_l) \cdot \frac{\epsilon\tau}{2\kappa} \cdot \omega^2 \geqslant \\
& \frac{2\kappa}{\epsilon \cdot \omega^2} \cdot \frac{\epsilon\tau}{2\kappa} \cdot \omega^2 \geqslant \tau.
\end{aligned}$$

By the return rule in Step 6 in Algorithm 1, we choose an appropriate subset S from the subset sequence which has the maximal cover value. Then it is easy to find that the final output value meets

$$\begin{aligned}
f(S) := & \max\{f(S_0), f(S_1), \ldots, f(S_{2/\epsilon})\} \geqslant \\
& f(S_l) \geqslant \tau.
\end{aligned}$$

That is, the cover value of output subset S in break Condition 1 is at least τ.

Case 2: Based on the definitions of optimal solution S^* and the buffer sequence $S_1, S_2, \ldots, S_{2/\epsilon}$, we define two partition sets P_1 and P_2, $P_1 = S^* \cap (\cup_{i=1}^{2/\epsilon} S_i)$ and $P_2 = S^* \setminus P_1$. By Proposition 1, we find some index t in sequence set $\{1, 2, \ldots, 2/\epsilon\}$, such that

$$f(S_t \cup S^*) \geqslant (1 - \frac{\epsilon}{2})\tau.$$

Expanding the expression of function $f(S_t \cup S^*)$, there is

$$\begin{aligned}
& f(S_t \cup S^*) = \\
& f(P_1 \cup S_t) + f(S_t \cup S^*) - f(P_1 \cup S_t).
\end{aligned}$$

By the setting of partition sets P_1 and P_2, and the work of Wang et al.[20], we can rearrange expression $f(S_t \cup S^*) - f(P_1 \cup S_t)$ as follows:

$$\begin{aligned}
& f(S_t \cup S^*) - f(P_1 \cup S_t) = \\
& f(S_t \cup (P_1 \cup P_2)) - f(P_1 \cup S_t) = \\
& f(P_2 | P_1 \cup S_t) \leqslant \frac{1}{\omega} \cdot f(P_2|S_t) \leqslant \\
& \frac{1}{\omega} \cdot \frac{1}{\omega} \sum_{x \in P_2} f(x|S_t) = \frac{1}{\omega^2} \sum_{x \in P_2} f(x|S_t).
\end{aligned}$$

Combining the selection rules in Step 3 with the above inequality, the expression of $f(S_t \cup S^*)$ can be reformulated as

$$\begin{aligned}
& f(S_t \cup S^*) \leqslant \\
& f(P_1 \cup S_t) + \frac{1}{\omega^2} \cdot \sum_{x \in P_2} f(x|S_t) \leqslant \\
& f(P_1 \cup S_t) + \frac{1}{\omega^2} \cdot \frac{\epsilon\tau}{2} \cdot \omega^2 = \\
& f(P_1 \cup S_t) + \frac{\epsilon\tau}{2}.
\end{aligned}$$

Then we get

$$f(P_1 \cup S_t) \geqslant f(S_t \cup S^*) - \frac{\epsilon\tau}{2}.$$

In the previous inequality,

$$f(S_t \cup S^*) \geqslant (1 - \frac{\epsilon}{2}) \cdot \tau,$$

so we can arrange the following inequality:

$$\begin{aligned}
& f(P_1 \cup S_t) \geqslant f(S_t \cup S^*) - \frac{\epsilon\tau}{2} \geqslant \\
& (1 - \frac{\epsilon}{2}) \cdot \tau - \frac{\epsilon\tau}{2} = (1 - \epsilon)\tau.
\end{aligned}$$

From the construct relation, it is obvious to find $P_1 \cup S_t$ is a subset of the union of the sequence sets $S_1, S_2, \ldots, S_{2/\epsilon}$. Then for the cover function value, there must be

$$f(P_1 \cup S_t) \leqslant \max_{Z \subseteq \cup_{i=1}^{2/\epsilon} S_i} f(Z).$$

After calling for the unconstrained maximization algorithm $\mathcal{A}(\alpha)$ which maintains an α-approximation, we obtain a subset S, such that

$$f(S) \geqslant \alpha \cdot \max_{Z \subseteq \cup_{i=1}^{2/\epsilon} S_i} f(Z).$$

Then we can get

$$\begin{aligned}
& f(S) \geqslant \alpha \cdot \max_{Z \subseteq \cup_{i=1}^{2/\epsilon} S_i} f(Z) \geqslant \\
& \alpha \cdot f(P_1 \cup S_t) \geqslant \alpha \cdot (1 - \epsilon)\tau.
\end{aligned}$$

In other words, the cover value of returned subset S in Case 2 is no less than $\alpha \cdot (1 - \epsilon)\tau$, that is,

$$f(S) \geqslant \alpha \cdot (1 - \epsilon)\tau.$$

Combining the two results both in Case 1 and Case 2, we get the final cover value of the Stream-Known-κ Algorithm,

$$f(S) \geqslant \min\{\alpha \cdot (1 - \epsilon)\tau, \tau\} = \alpha \cdot (1 - \epsilon)\tau.$$

■

To the best of our knowledge, the memory occupied by the elements is independent of the properties of the cover function during the execution of Algorithm 1. Then utilizing the work of Crawford[19], we get the following lemma for the cost function. The proof idea is similar with that of Lemma 2, and we present a brief proof.

Lemma 3 The cost function of the output subset S returned by Algorithm 1 is no more than $\frac{(2+\epsilon)^2}{\epsilon^2\omega^2}\kappa$.

Proof Consider the break condition of Algorithm 1, there are two corresponding cases.

Case 1: When the current element x has been scanned but it is not added into any subset, then by the execution process, we know $c(S_l) \leqslant \frac{2\kappa}{\epsilon\omega^2}$ for all index $l \in \{1, 2, \ldots, 2/\epsilon\}$.

Case 2: When the current element x has been scanned and it is added into some subset S_l, then by the condition in Step 3, we know $c(x) \leqslant \kappa$. After adding element x, the cost of subset S_l implies that $c(S_l) \leqslant (\frac{2\kappa}{\epsilon\omega^2} + \kappa)$.

After running the unconstrained maximization algorithm $\mathcal{A}(\alpha)$, we get a subset S_0, which is selected from the union of $S_1, S_2, \ldots, S_{2/\epsilon}$. Thus, after any iteration of Algorithm 1, there is

$$
\begin{aligned}
c(\cup_{l=0}^{2/\epsilon} S_l) \leqslant \sum_{l=0}^{2/\epsilon} c(S_l) \leqslant \\
\sum_{l=0}^{2/\epsilon} \max\{\frac{2\kappa}{\epsilon\omega^2}, \frac{2\kappa}{\epsilon\omega^2} + \kappa\} = \\
\sum_{l=0}^{2/\epsilon} (\frac{2\kappa}{\epsilon\omega^2} + \kappa) = \\
(\frac{2}{\epsilon\omega^2} + 1)(\frac{2}{\epsilon} + 1)\kappa = \\
\frac{(2+\epsilon\omega^2)(2+\epsilon)}{\epsilon^2\omega^2} \cdot \kappa \leqslant \\
\frac{(2+\epsilon)^2}{\epsilon^2\omega^2}\kappa.
\end{aligned}
$$

Thus the cost function value of the output subset is no more than $(2+\epsilon)^2/(\epsilon^2\omega^2) \cdot \kappa$. ■

On the basis of Lemmas 2 and 3, and the results, we come to the final conclusion.

Theorem 2 When executing the Stream-Known-κ Algorithm for any instance of non-monotone approximately submodular cover problem, we can acquire the following conclusions:

- The cover function value of output set S is no less than $\alpha \cdot (1-\epsilon)\tau$;
- The cost function value is no more than $(2+\epsilon)^2/(\epsilon^2\omega^2) \cdot \kappa$.

3.2 Multi-pass algorithm for cover problem

From Algorithm 1, we know there is a guess value of optimal solution. However, it is a huge challenge in the streaming model. In order to avoid this assumption, we set the minimum cost value of single element to the initial optimal value, which needs to scan the streaming one pass. Next we need to take a parameter $\epsilon \in (0,1)$, and run the sieve-streaming algorithm as a subroutine under the lower bound. If the output set S satisfies that the cover value is no less than $\alpha(1-\epsilon)$ times threshold, we return the current subset. Otherwise, we increase the guess value κ to $(1+\epsilon)\kappa$ and execute the sieve-streaming subroutine sequentially. The detailed description is shown in the Stream-Multi-Pass Algorithm.

By analyzing Algorithm 2, we get the following performance guarantee.

Theorem 3 When executing the multi-pass streaming algorithm for an instance of non-monotone approximately submodular cover problem, we can acquire the following conclusions:

- The cover function value of output set S is no less than $\alpha \cdot (1-\epsilon)\tau$;
- The cost function value is no more than $(2+\epsilon)^2/(\epsilon^2\omega^2) \cdot (1+\epsilon)OPT$.

Proof From the updating operation for κ in Step 4 in Algorithm 2, we can find the unique index $u \in \mathbf{Z}_+$, such that OPT is strictly lager than $(1+\epsilon)^{u-1}c_{\min}$, but no more than $(1+\epsilon)^u c_{\min}$.

Benefitting from Lemma 2, once the value of κ comes to the optimal value OPT, the Stream-Known-κ algorithm will output a subset S whose cover value is no less than $\alpha(1-\epsilon)\tau$. It consists with the bound of Theorem 2.

With the benefit of Lemma 3, we acquire the cost function satisfies

$$c(S) \leqslant (2+\epsilon)^2/(\epsilon^2\omega^2)\kappa.$$

Since the current κ is no more than $(1+\epsilon)^u c_{\min}$, it is no more than $(1+\epsilon)OPT$ absolutely. Further we obtain

$$c(S) \leqslant (2+\epsilon)^2/(\epsilon^2\omega^2) \cdot (1+\epsilon)OPT.$$

Thus, we complete the proof. ■

3.3 Single-pass algorithm for cover problem

In this section, we design a single-pass streaming

Algorithm 2 Stream-Multi-Pass

```
Step 1: Give a parameter $\epsilon \in (0,1)$ and the corresponding
    information in Algorithm 1 (Stream-Known-$\kappa$).
Step 2: Scan the streaming set $G$ to find the smallest cost
    value $c_{\min} = \min_{x \in G} c(x)$.
    Initially set $\kappa := c_{\min}$.
Step 3: Run Algorithm 1 (Stream-Known-$\kappa$) for the current
    pair $(\epsilon, \kappa)$ and output subset $S$.
Step 4: If the cover value $f(S) \geqslant \alpha(1-\epsilon)\tau$, then output $S$.
    Otherwise, update $\kappa := (1+\epsilon)\kappa$ and go back to Step 3.
```

algorithm to reduce the storage cost that scans the streaming multi-pass. In the single-pass algorithm, besides the input $\epsilon \in (0,1)$, we also take a non-negative real number U as the upper bound value. Similar with the multi-pass streaming algorithm, the single-pass streaming algorithm also takes the sieve-streaming algorithm as a subroutine under the assumption of optimal value. Specifically, the sieve-streaming subroutine is executed in parallel in the single-pass streaming algorithm once a bound is given. Instead of guessing the optimal value sequentially, the single-pass streaming algorithm maintains a set with respect to the guess values. As element arrives, the lower bound of guess is updated correspondingly. After running the sieve-streaming subroutine in parallel, we can get a series of subsets. Next we will run an unconstrained maximization algorithm $\mathcal{A}(\alpha)$ which maintains α-approximation on the union of output subsets and get an additional subset. Finally we select a subset which has the maximal cover value. The detailed description is shown in Algorithm 3.

Similar to the proof of multi-pass streaming algorithm, we can also get the performance guarantee of Algorithm 3. Since the idea and logic of the proof are highly consistent with those in Algorithms 1 and 2, we can keep the same performance guarantee.

Theorem 4 When executing the Stream-Single-Pass algorithm for an instance of non-monotone approximately submodular cover problem, we can acquire the following conclusions:

- The cover function value of output set S is no less than $\alpha \cdot (1-\epsilon)\tau$;
- The cost function value is no more than $(2+\epsilon)^2/(\epsilon^2\omega^2) \cdot (1+\epsilon)OPT$.

4 Numerical Experiment

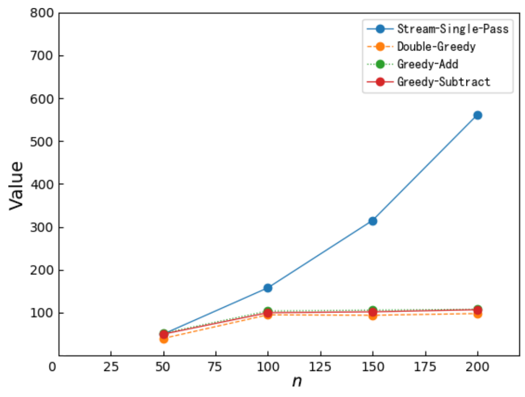
In this section, we apply the Stream-Single-Pass Algorithm to the maximum cut problem of graph theory to test its performance guarantee. For an undirected graph $T=(V,E)$ with weight, each vertex $v \in V$ has weight $w(v)$, and the cut defined on the graph T with respect to vertex set S is the edges set, where one vertex is contained in S while another vertex is in $V \setminus S$. We denote the cut function $f(S)=|C(S)|$ and the weight function $w(S)=\sum_{v\in S} w(v)$. It is obvious that the cut function is non-monotone submodular and the weight function is additive linear, which satisfy the properties required for the input functions in our algorithm. Therefore, utilizing the Stream-Single-Pass Algorithm to solve the maximum cut problem is reasonable and feasible to test the performance.

Algorithm 3 Stream-Single-Pass

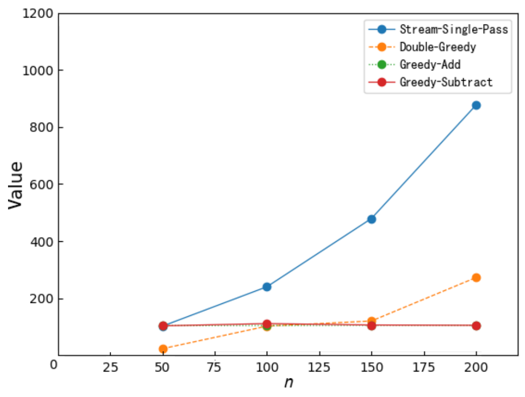
```
Step 1: Give parameters $\epsilon \in (0,1)$, upper bound value $U \in \mathbf{R}_{\geqslant 0}$,
        and the corresponding information in Algorithm 1 (Stream-Known-$\kappa$).
Step 2: Construct a series of buffer subsets $S_{(1+\epsilon)^i,j}$ for any
        integer index $i$ and index $j$ in $\{0,1,2,\ldots,2/\epsilon\}$, and set
        $S_{(1+\epsilon)^i,j} := \varnothing$ and $m := 0$ initially.
Step 3: For each arriving element $x$, if $\frac{f(x)}{c(x)} > m$, update
            $m := \max\{m, \frac{f(x)}{c(x)}\}$.
        For each $\kappa$ enumerated in the set $\{(1+\epsilon)^i : i \in \mathbf{Z}, \frac{\epsilon\tau}{2m} \leqslant$
        $(1+\epsilon)^i \leqslant U\}$, if there is some $i \in \{1,2,\ldots,2/\epsilon\}$, such that
        $|S_{\kappa,i}| < \frac{2\kappa}{\epsilon}$ and $f(x|S_{\kappa,i}) \geqslant \frac{c(x)\epsilon\tau}{2\kappa} \cdot \omega^2$, then update
        $S_{\kappa,i} := S_{\kappa,i} \cup \{x\}$.
Step 4: Repeat Step 3 and return a series of storage subsets $S_{\kappa,i}$,
        $i = \{1,2,\ldots,2/\epsilon\}$.
Step 5: Run an unconstrained maximization algorithm $\mathcal{A}(\alpha)$
        which maintains $\alpha$-approximation on the union set
        $\cup_{i=1}^{2/\epsilon} S_{\kappa,i}$ and output a set $S_{\kappa,0}$.
Step 6: If the value of $\max\{f(S_{\kappa,i}) : i \in \{0,1,\ldots,2/\epsilon\}\}$ is
        no smaller than $\alpha(1-\epsilon)\tau$, then set $U := \kappa$ and update
        $S_{(1+\epsilon)^i,j} := \varnothing$ for each integer index $i$ and index $j$ in
        $\{0,1,\ldots,2/\epsilon\}$ and $U < (1+\epsilon)^i$.
Step 7: Go through Steps 3–6, and output subset
            $S := \arg\max\{f(S_{\kappa,i}) : \kappa \leqslant U, i \in \{0,1,\ldots,2/\epsilon\}\}$.
```

In order to express the performance of Stream-Single-Pass Algorithm, we compare it with Double-Greedy Algorithm (selecting elements double greedily), Greedy-Add Algorithm (adding elements greedily), and Greedy-Substract Algorithm (deleting elements greedily), which are also common methods for solving these problems. For the convenience of statement, we take 200 data points to test the algorithms' performance, which are constructed randomly or obtained from movielens set. We first present the results of the four algorithms on coverage function values, under the condition of data size and low bound threshold with respect to random set and movielens set. The detailed numerical experiments are shown as follows.

As can be seen from Fig. 1, we find the function coverage value of Stream-Single-Pass Algorithm is significantly better than those of Double-Greedy Algorithm, Greedy-Add Algorithm, and Greedy-Substract Algorithm for both random dataset and movielens dataset. For the movielens dataset in Fig. 1b, Double-Greedy Algorithm performs better than Greedy-

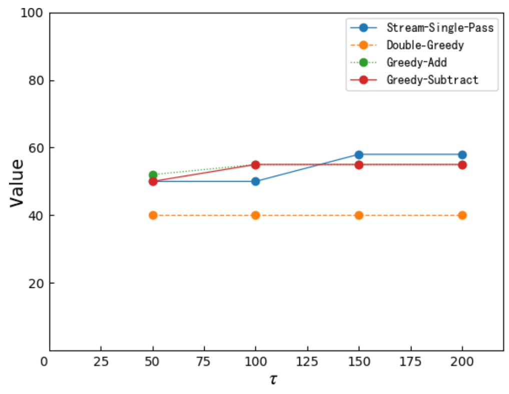


(a) Function value with respect to n on random dataset

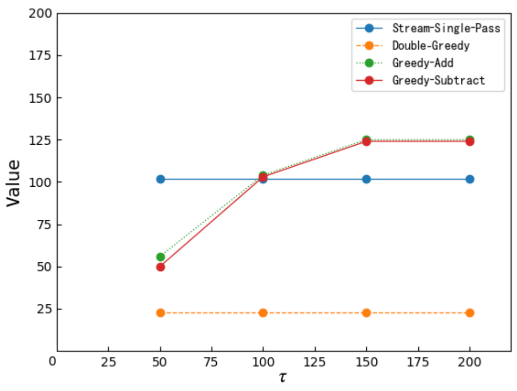


(b) Function value with respect to n on movielens dataset

Fig. 1 Performance comparison of cover function with n.



(a) Function value with respect to τ on random dataset



(b) Function value with respect to τ on movielens dataset

Fig. 2 Performance comparison of cover function with τ.

Add Algorithm and Greedy-Substract Algorithm when $n \geqslant 125$. It is absolutely possible to maintain this trend for massive dataset.

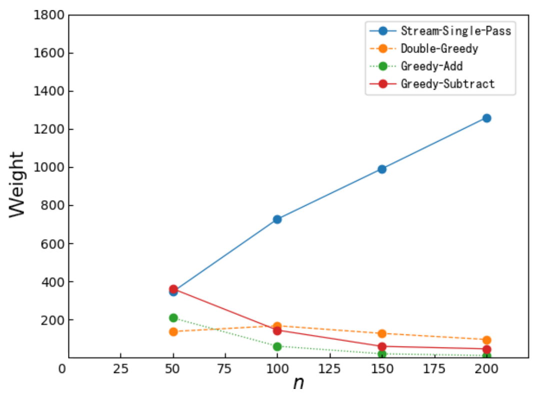
As shown in Fig. 2, Stream-Single-Pass Algorithm presents the best performance for dataset generated randomly when τ value is no less than 135. For the dataset received from movielens dataset, we find the function cover value obtained by Stream-Single-Pass Algorithm is the largest when τ is no more than 100. There is no doubt that the function coverage values on these two datasets are significantly better than that of Double-Greedy Algorithm.

Based on the optimization model, we not only compare the advantages of the four algorithms from the aspect of cover function value, but also further discuss them from the view of cost function. We compare the performance of the weight function with respect to the data size n and τ value on the two kinds of datasets. For both datasets generated randomly and extracted from movielens dataset, we find the cost of Stream-Single-Pass Algorithm increases with the increasing of data scale, which is also larger than other algorithms in Fig. 3.

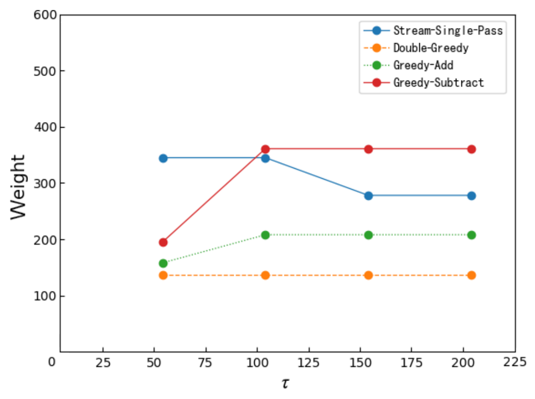
Corresponding to Figs. 1 and 2, we also investigate the weight with respect to τ on the random set and movielens dataset. As can be seen from Fig. 4, the weight required by Stream-Single-Pass Algorithm for random dataset is less than that of Greedy-Substract Algorithm when τ value is no less than 100.

5 Conclusion

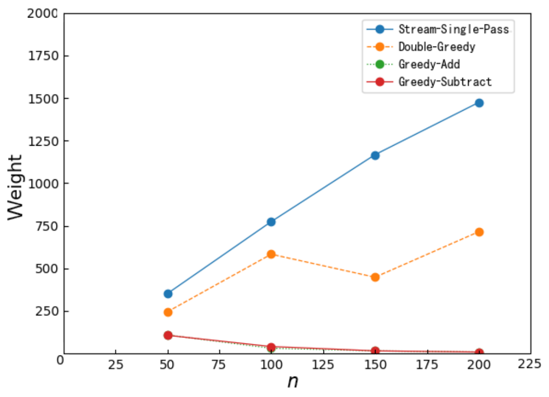
In this work, we aim to consider the problem of minimizing the cost function while maintaining the cover function exceeding a required threshold under the streaming model. It is also called approximately submodular cover problem. To solve the non-monotone approximately submodular cover problem, an effective method is bicriteria approximation algorithm. We propose three bicriteria approximation algorithms. Firstly, we provide a streaming algorithm under the guess of known optimal objective value. By analyzing the Stream-Known-κ algorithm, we obtain the bicriteria ratio $(\alpha(1-\epsilon), (2+\epsilon)^2/(\epsilon^2\omega^2))$. That is, the cover value of the output set is no smaller than $\alpha \cdot (1-\epsilon)\tau$, and the cost value is no larger than $(2+\epsilon)^2/(\epsilon^2\omega^2) \cdot \kappa$. In order to avoid the assumption that the optimal value κ is known in advance, we provide a multi-pass streaming



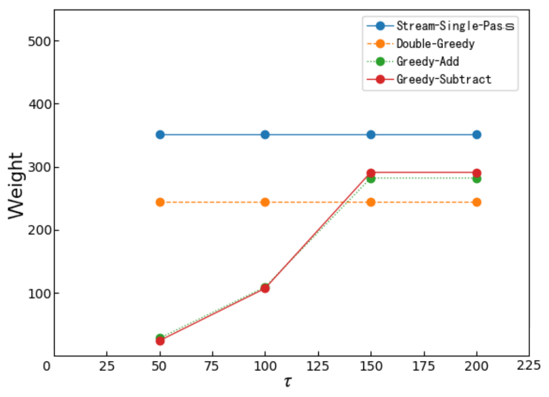
(a) Weight with respect to n on random dataset



(a) Weight with respect to τ on random dataset



(b) Weight with respect to n on movielens dataset



(b) Weight with respect to τ on movielens dataset

Fig. 3 Performance comparison of cost function with n.

Fig. 4 Performance comparison of cost function with τ.

algorithm which runs the Stream-Known-κ algorithm as a subroutine. Further, we adjust the multi-pass streaming algorithm to a single-pass one. Both the multi-pass streaming algorithm and single-pass streaming algorithm maintain the same bicriteria approximation ratio as that of the Stream-Known-κ algorithm. In addition, we present some numerical experiments to express the single-pass streaming algorithm's performance comparing with some existing ones.

Acknowledgment

This work was supported by the National Natural Science Foundation of China (Nos. 72192804, 72192800, and 12201619) and the China Postdoctoral Science Foundation (No. 2022M723333).

References

[1] E. J. Barezi, I. D. Wood, P. Fung, and H. R. Rabiee, A submodular feature-aware framework for label subset selection in extreme classification problems, in *Proc. 2019 Conf. North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Minneapolis, MN, USA, 2019, pp. 1009–1018.

[2] J. Xu, L. Mukherjee, Y. Li, J. Warner, J. M. Rehg, and V. Singh, Gaze-enabled egocentric video summarization via constrained submodular maximization, in *Proc. 2015 IEEE Conf. Computer Vision and Pattern Recognition*, Boston, MA, USA, 2015, pp. 2235–2244.

[3] A. Das and D. Kempe, Submodular meets spectral: greedy algorithms for subset selection, sparse approximation and dictionary selection, in *Proc. 28th Int. Conf. Int. Conf. Machine Learning*, Bellevue, WA, USA, 2011, pp. 1057–1064.

[4] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. Glance, Cost-effective outbreak detection in networks, in *Proc. 13th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, San Jose, CA, USA, 2007, pp. 420–429.

[5] D. Kempe, J. Kleinberg, and É. Tardos, Maximizing the spread of influence through a social network, in *Proc. 9th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, Washington, DC, USA, 2003, pp. 137–146.

[6] V. G. Crawford, A. Kuhnle, and M. T. Thai, Submodular cost submodular cover with an approximate oracle, in *Proc. 36th Int. Conf. Machine Learning*, Long Beach, CA, USA, 2019, pp. 1426–1435.

[7] L. A. Wolsey, An analysis of the greedy algorithm for the submodular set covering problem, *Combinatorica*, vol. 2, no. 4, pp. 385–393, 1982.

[8] P. J. Wan, D. Z. Du, P. Pardalos, and W. Wu, Greedy approximations for minimum submodular cover with

submodular cost, *Comput. Optim. Appl.*, vol. 45, no. 2, pp. 463–474, 2010.

[9] R. Iyer and J. Bilmes, Submodular optimization with submodular cover and submodular knapsack constraints, in *Proc. of the 26th International Conference on Neural Information Processing Systems*, Lake Tahoe, UV, USA, pp. 2436–2444, 2013.

[10] A. A. Bian, J. M. Buhmann, A. Krause, and S. Tschiatschek, Guarantees for greedy maximization of non-submodular functions with applications, in *Proc. 34th Int. Conf. Machine Learning*, Sydney, Australia, 2017, pp. 498–507.

[11] T. Friedrich, A. Göbel, F. Neumann, F. Quinzan, and R. Rothenberger, Greedy maximization of functions with bounded curvature under partition matroid constraints, in *Proc. 33rd AAAI Conf. Artificial Intelligence*, Honolulu, HI, USA, 2019, p. 281.

[12] S. Gong, Q. Nong, W. Liu, and Q. Fang, Parametric monotone function maximization with matroid constraints, *J. Glob. Optim.*, vol. 75, no. 3, pp. 833–849, 2019.

[13] A. Kuhnle, J. D. Smith, V. G. Crawford, and M. T. Thai, Fast maximization of non-submodular, monotonic functions on the integer lattice, in *Proc. 35th Int. Conf. Machine Learning*, Stockholm, Sweden, 2018, pp. 2791–2800.

[14] C. Harshaw, M. Feldman, J. Ward, and A. Karbasi, Submodular maximization beyond non-negativity: Guarantees, fast algorithms, and applications, in *Proc. 36th Int. Conf. Machine Learning*, Long Beach, CA, USA, 2019, pp. 2634–2643.

[15] C. Qian, Multi-objective evolutionary algorithms are still good: Maximizing monotone approximately submodular minus modular functions, arXiv preprint arXiv: 1910.05492, 2019.

[16] A. Norouzi-Fard, A. Bazzi, M. El Halabi, I. Bogunovic, Y. P. Hsieh, and V. Cevher, An efficient streaming algorithm for the submodular cover problem, in *Proc. 30th Int. Conf. on Neural Information Processing Systems*, Barcelona, Spain, 2016, pp. 4500–4508.

[17] B. Lehmann, D. Lehmann, and N. Nisan, Combinatorial auctions with decreasing marginal utilities, *Games Econ. Behav.*, vol. 55, no. 2, pp. 270–296, 2006.

[18] U. Feige, V. S. Mirrokni, and J. Vondrák, Maximizing non-monotone submodular functions, *SIAM J. Comput.*, vol. 40, no. 4, pp. 1133–1153, 2011.

[19] V. G. Crawford, Scalable bicriteria algorithms for non-monotone submodular cover, arXiv preprint arXiv: 2112.09985, 2021.

[20] Y. Wang, D. Du, D, Y. Jiang, and X. Zhang, Non-submodular maximization with matroid and knapsack constraints, *Asia-Pacific Journal of Operational Research*, vol. 38, no, 5, p. 2140001, 2021.

Yijing Wang received the PhD degree from Beijing University of Technology, China in 2021. She is currently a postdoctoral researcher at Academy of Mathematics and Systems Science, Chinese Academy of Sciences, China. Her research interests include combinatorial optimization, approximation algorithms, and submodular optimization.

Xiaoguang Yang received the PhD degree from Tsinghua University, China in 1993. He is currently a professor at the Academy of Mathematics and Systems Science, Chinese Academy of Sciences, China. His research focuses on financial economics, financial mathematics, combinatorial optimization, and approximation algorithms.

Hongyang Zhang received the MS degree from Beijing University of Technology, China in 2022. He is currently a PhD candidate at Ningbo University, China. His research interests include combinatorial optimization, approximation algorithm, and submodular optimization.

Yapu Zhang received the BS degree in mathematics and applied mathematics from Northwest University, China in 2016, and the PhD degree in operational research from the University of Chinese Academy of Sciences, China in 2021. She is currently working at Beijing University of Technology, Beijing, China. Her research interests include social networks, combinatorial optimization, and algorithm design.