Improved Streaming Algorithm for Minimum Cost Submodular Cover Problem

Tan D. Tran[1], Canh V. Pham[2], Dung P. Trung[2], and Uyen T. Nguyen[3]

[1] Faculty of Information Technology, VNU University of Engineering and Technology, Hanoi, Vietnam
22027005@vnu.edu.vn
[2] ORLab, Faculty of Computer Science, Phenikaa University, Hanoi, Vietnam
{canh.phamvan,dung.phamtrung}@phenikaa-uni.edu.vn
[3] Institute of Engineering and Technology, Vinh University, Nghean, VietNam
uyennt@vinhuni.edu.vn

Abstract. This paper introduces an efficient streaming algorithm for a well-known problem named Minimum cost Submodular Cover (MSC). Our algorithm makes $O(\log n)$ passes over the ground set and takes $O(n \log n)$ query complexity and returns a $(1/\epsilon, 1-\epsilon)$-bicriteria approximation solution with the ground set of size n and the precise parameter $\epsilon > 0$. Therefore, it is better than the existing streaming algorithms in terms of solution guarantees and query complexity. Besides the theoretical performance, the experiment results on two applications, Revenue Threshold, and Coverage Threshold, further show the superiority of our algorithm with the state-of-the-art ones.

Keywords: Submodular Cover, Approximation Algorithm, Streaming Algorithm

1 Introduction

This paper considers the Minimum cost Submodular Cover problem (MSC). We have a finite ground set V sized n and a monotone submodular set function $f : 2^V \to \mathbb{R}^+$, an additive cost function $c : 2^V \to \mathbb{R}^+$, and a positive threshold $T \leq f(V)$, the goal is to find a subset $S \subseteq V$ with the minimum cost such that $f(S) \geq T$, where the cost of S is $c(S) = \sum_{v \in S} c(v)$.

The problem has a wide range of applications in the areas of artificial intelligence, data mining, and combination optimization such as recommending systems [6], data summarization [18,12], social influence [10,16] and revenue maximization in social networks [9], maximum coverage [15,9], revenue threshold, coverage threshold, etc. The primary objective of revenue and coverage threshold differs from typical maximization goals. Rather than solely seeking to maximize the solution, revenue and coverage threshold optimization focuses on identifying a solution in which the object value surpasses a specified threshold while minimizing costs. Although there has been a lot of work focused on

designing efficient algorithms for the MSC problem [20,19,13,4], the exponential growth of input data requires more efficient algorithms for both running time and memory usage. In this context, the streaming algorithm is an effective method to solve the above issues because it goes through the entire stream only once or a few times and uses a small amount of memory to get the solution with a theoretical bound.

Unfortunately, the proposed streaming algorithms for this problem still have weaknesses. The first one of Norouzi-Fard *et al.*[14] only focused on a special problem with uniform cost. Therefore, it could not be applied to the general MSC. The authors in [3] introduced the state-of-the-art with a non-monotone utility function. It returned a $(O(1/\epsilon^2), (1-\epsilon)/2)$-bicriteria approximation solution but ran in $O(n \log \mathsf{opt})/c_{min})$ query complexity, where opt was the cost of optimal solution and c_{min} was the smallest cost of an element. Note that the query of this algorithm was not a polynomial since $1/c_{min}$ was not a constant, and it might become arbitrarily large.

Motivated by the above phenomena, in this paper, we propose an improved streaming algorithm that provides better approximation guarantees and query complexity. **Our contributions** are as following:

- We propose a streaming algorithm for the MSC problem (StrMSC) that provides a $(\frac{1}{\epsilon}, 1-\epsilon)$ bicriteria approximation solution that takes $O(\frac{1}{\epsilon}\log\frac{n}{\epsilon})$ passes over the ground set, $O(n)$ memories and $O(\frac{n}{\epsilon}\log\frac{n}{\epsilon})$ query complexity, where $\epsilon > 0$ is a pre-defined accuracy parameter. Therefore, our algorithm is the first multi-pass and near-linear query complexity for MSC.
- We assess the practical performance of the StrMSC algorithm through an extensive experiment in two applications: Revenue Threshold and Coverage Threshold. Our comparison includes the algorithms, MULTI and SINGLE, developed by Crawford *et al.* [3], as well as the GREEDY algorithm by Goyal *et al.* [7]. Our algorithm consistently outperforms these benchmarked algorithms across various evaluation criteria, encompassing objective value, the number of queries, and memory utilization.

The core idea of our algorithm includes two components. The first component adapts the technique of splitting the ground set by [17] to find a major set within one pass over the ground set. The second one finds a good solution over the major set. For each pass, it establishes a threshold and limits the cost to select: (1) a new element to the partial solution if density gain is at least the threshold, and (2) a good singleton without violating the limited cost. Each pass returns the best candidate solution between the above candidates, and the algorithm terminates after at most $O(\log n)$ passes or the value of the utility function is at least $(1-\epsilon)T$.

Organization. The rest of the paper is structured as follows. Section 2 provides the literature review on MSC problem. Section 3 presents notations and proprieties of studied problem. Section 4 introduces the proposed algorithm and theoretical analysis. Experimental computation is provided in Section 5. Finally, we conclude this work in Section 6.

2 Related Works

The MSC problem is an NP-hard problem [8]. One common approach to solving the problem is using greedy, which sequentially selects elements with maximal marginal gain into the partial solution and takes advantage of the submodular to derive the approximation bound. Wolsey *et al.* [20] first introduced a greedy version that had an approximation ratio of $1+\ln(\alpha/\beta)$, where α was the maximum value of the objective over all the singletons, and β was the smallest non-zero marginal gain of the greedy algorithm. Then, Wan *et al.* [19] showed the greedy algorithm could return an approximation ratio of $1+\ln(\alpha/\beta)$ for a special case of MSC problem. Another work of [7] proposed a greedy algorithm with a different stopping condition. It returned a $(\log(\frac{T}{\epsilon}), 1-\epsilon)$-bicriteria approximation solution. Recently, Crawford *et al.* [4] proved that a greedy algorithm could provide another bound under noise models when one could only estimate the value of the utility function within a bias error. In general, greedy algorithms have $O(n^2)$ query complexity and make $O(n)$-passes through the data. Thus, they may be infeasible for some applications with large data. Besides, several works have focused on developing evolutionary algorithm [5] or parallel algorithm [17] for MSC. However, they needed expensive query complexity and required a polynomial number of passes through the data.

Streaming algorithms are effective for submodular optimization problems, especially in big data. Norouzi-Fard *et al.* [14] showed that a single pass streaming algorithm for MSC with an approximation ratio better than $n/2$ must use at least $O(n)$ memory. In the seminal work, they proposed an efficient algorithm for the Submodular Cover problem, a special case of MSC with uni-cost. Their algorithm made a single-pass, used M memories and provided $(2\ln(1/\epsilon), 1-\frac{1}{\ln(1/\epsilon)})$-bicriteria approximation solution, where M was the required memory. More recently, Crawford *et al.*[3] proposed two bicriteria algorithms for solving MSC for the non-monotone case. The first one, called MULTI, returned $((1+\epsilon)(1+\frac{4}{\epsilon^2}), \frac{1-\epsilon}{2})$ bicriteria approximation ratio, made $O(\log(\mathsf{opt})/c_{min})$ passes, wasted $O(\frac{n}{\epsilon}\log(\mathsf{opt}/c_{min}))$ query complexity, and used $O(\mathsf{opt}/\epsilon^2)$ space complexity. The query complexity of MULTI depended on $1/c_{min}$ that was not a constant and might be arbitrarily large. Their second one, SINGLE, made a single pass and returned the same approximation bound but had a larger query complexity of $\Omega(n^2/c_{min})$.

3 Preliminaries

Given a ground set $V = \{e_1, \ldots, e_n\}$ of size n, the submodular set function $f : 2^V \mapsto \mathbb{R}^+$ measures the quality of a subset $S \subseteq V$. It is assumed that there exists an oracle query, which, when queried with the set S, returns the value $f(S)$. We assume f normalized, i.e., $f(\emptyset) = 0$. The marginal gain of an element e to a set $S \subseteq V$ is defined $f(e|S) = f(S \cup \{e\}) - f(S)$. We also define $f(S|X) = f(S \cup X) - f(X)$ for any set $X \subseteq V$. We simplify $f(\{e\})$ to $f(e)$.

The function f is monotone if for $A \subseteq B \subseteq V$, we have $f(A) \leq f(B)$. f is submodular if for any $A \subseteq B \subseteq V$, $e \in V \setminus B$, $f(e|A) \geq f(e|B)$.

An instance of the problem MSC is presented by a tuple (V, f, T). Given an instance of MSC with cost function c additive, i.e., $c(S) = \sum_{e \in S} c(e)$, we define $c_{min} = \min_{e \in V} c(e)$, $c_{max} = \max_{e \in V} c(e)$, and O is the optimal solution and the optimal cost $\mathsf{opt} = c(O)$.

We call an algorithm is a (x, y)-**bicriteria approximation** for MSC problem if it returns a solution S satisfying $f(S) \geq y \cdot T$ and $c(S) \leq x \cdot \mathsf{opt}$, where $x, y > 0$.

Streaming algorithm. A streaming algorithm is an approximation algorithm that processes the data stream in which the input is presented as a sequence of elements and can be examined in only one or a few passes. These algorithms are designed to operate with limited memory, generally logarithmic in the size of the stream or in the maximum value in the stream.

4 Proposed Algorithm

This section introduces our streaming version StrMSC that provides a $(1-\epsilon, \frac{1}{\epsilon})$-bicriteria approximation solution for MSC within $O(\log n)$ passes over the ground set V.

Algorithm description. StrMSC consists of two phases. The first phase (lines 2-9) adapts a strategy of dividing the set V into reasonable subsets by [17], within one pass over the ground set V, to ensure the algorithm takes at most near-linear query complexity. Accordingly, it first sorts $V = \{u_1, u_2, \ldots, u_n\}$ in non-decreasing order and then finds the smallest j so that $f(u_1, u_2, \ldots, u_j) \geq T$. The algorithm divides V into three subsets: the first subset V_0 contains elements with the cost less than $c'_{min} \leftarrow \epsilon c(u_j)/n$, the second one V_1 contains elements with the cost greater than $c'_{max} \leftarrow jc(u_j)$, and the last one V' contains the rest. We call the set V' as the major set, and one may find the near-optimal solution of the problem over the ground V' instead of V. By the above division strategy, one can bound the ratio of $\frac{\max_{e \in V'} c(e)}{\min_{e \in V'} c(e)} = O(n^3)$ that helps the algorithm finds the solution in near-linear query complexity.

The second phase (lines 9-21) takes at most $O(\log_{1+\epsilon} n)$ passes over the major set V'. In each pass, the algorithm finds a candidate solution S_v by adding a new element e with the density gain, i.e., the ratio between $f'(e|S_v)$ and the cost $c(e)$, satisfies the condition in the line 13, where $f(\cdot) = f(\cdot|V_0)$. The algorithm then updates the element e_v with the highest utility value with the cost is at most $(1+\epsilon)^{v+1}$. At the end of each pass, it finds the best solution among S_v and $\{e_v\}$ and terminates this phase if $f'(S'_v) \geq \frac{\alpha T'}{2}$.

Finally, the algorithm returns which set has a lower cost between candidate ones $S'_v \cup V_0$ and S^0.

Theoretical Analysis. In the following, we show the theoretical guarantees of StrMSC in Theorem 1.

Theorem 1. *Algorithm 1 is multi-pass streaming algorithm that*

- *Returns a solution S with $f(S) \geq (1-\epsilon)T$ and $c(S) \leq \frac{1}{\epsilon}\mathsf{opt}$.*

Algorithm 1: Algorithm

Input: An instance (V, f, T), parametter $\epsilon > 0$
Output: A solution S
// Phase 1: Pre-processing
1: Within one passe over ground set V do:
2: Sort $V = \{u_1, u_2, \ldots, u_{|V|}\}$ in non-decreasing cost order
3: Find $j \leftarrow \min\{i : f(\{u_1, u_2, \ldots, u_i\}) \geq T\}$, $S^0 \leftarrow \{u_1, \ldots, u_j\}$
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: $V' \leftarrow V \setminus (V_0 \cup V_1)$, $T' \leftarrow T - f(V_0)$, $f'(\cdot) \leftarrow f(\cdot|V_0)$
8: $c_0 \leftarrow \min_{e \in V'} c(e)$, $U = \{v \in [n] : c_0 \leq (1+\epsilon)^v \leq c(V')\}$
9: $\alpha = 2(1-\epsilon), \beta = \frac{(1+\epsilon)\alpha}{2-\alpha}$
// Phase 2: Main Streams
10: **foreach** $v \in U$ **do**
11: $S_v \leftarrow \emptyset, e_v \leftarrow \emptyset$
12: **foreach** $e \in V'$ **do**
13: **if** $\frac{f'(e|S_v)}{c(e)} \geq \frac{\alpha T'}{\beta(1+\epsilon)^v}$ *and* $c(S_v \cup \{e\}) \leq \beta(1+\epsilon)^v$ **then**
14: $S_v \leftarrow S_v \cup \{e\}$
15: **if** $c(e_v) \leq (1+\epsilon)^{v+1}$ **then**
16: $e_v \leftarrow \arg\max_{x \in \{e, e_v\}} f'(x)$
17: $S'_v \leftarrow \arg\max_{X \in \{S_v, e_v\}} f'(X)$
18: **if** $f'(S'_v) \geq \frac{\alpha T'}{2}$ **then**
19: **break**
20: **else**
21: Delete S'_v, e_v
22: **return** $\arg\min_{X \in \{S'_v \cup V_0, S^0\}} c(X)$.

– *Takes $O(n)$ memories and makes at most $O(\frac{1}{\epsilon}\log(\frac{n}{\epsilon}))$-pass through V.*
– *Takes $O(\frac{n}{\epsilon}\log(\frac{n}{\epsilon}))$ query complexity.*

Proof. For ease of following, we first recap the proof of showing the bound c'_{min} and c'_{max} in [17]. Supposing that j is an integer number the algorithm finds in line 2 and $v_t = \max_{e \in O} c(e)$. If $c(v_t) < c(v_j)$, then $O = \{v_1, v_2, \ldots, v_t\}$. By the monotoncity of f, we have $f(O) \geq T = f(\{v_1, v_2, \ldots, v_t\})$ which contracts to the definition of j. Thus, $c(v_j) \leq c(v_t) \leq c(O)$. On the other hand, since the set $\{v_1, v_2, \ldots, v_j\}$ is a feasible solution of (V, f, T), we have $c(O) \leq c(\{v_1, v_2, \ldots, v_j\}) \leq jc(v_j)$.

Prove approximation guarantees. By the selection of V_0, we have $c(V_0) \leq |V_0|c'_{min} \leq \epsilon c(v_j) \leq \epsilon\mathsf{opt}$. Denote by O' an optimal solution of the instance (V', f', T') and $\mathsf{opt}' = c(O')$, where $f'(\cdot) = f(\cdot|V_0)$ and $T' = T - f(V_0)$. It is easy to see that $f'(\cdot)$ is a monotone and submodular function. Since $f'(O) = f(O \cup V_0) - f(V_0) \geq T - f(V_0)$, O is a feasible solution of the problem under the

instance (V', f', T') and thus $\mathsf{opt}' \leq \mathsf{opt}$. On the other hand, since $c'_{min} \leq c_0 \leq (1+\epsilon)^v \leq c(V')$, there exists an integer v so that $\frac{\mathsf{opt}'}{1+\epsilon} < u = (1+\epsilon)^v \leq \mathsf{opt}$.

We provide the theoretical bound of S'_v and use it to obtain the proof. We consider two following cases:

Case 1. There exists an element $o \in O' \setminus S_v$ so that $c(S_v) + c(o) > \beta(1+\epsilon)^v$ and $\frac{f'(o|S_v)}{c(o)} \geq \frac{T'}{\beta(1+\epsilon)^v}$. By the selection of S_v, we have $f'(S_v) \geq \frac{c(S_v)\alpha T'}{\beta u}$. Therefore

$$f'(S_v \cup \{o\}) \geq f'(S_v) + \frac{\alpha c(o) T'}{\beta u} \geq \frac{c(S_v)\alpha T'}{\beta u} + \frac{\alpha c(o) T'}{\beta u} \tag{1}$$

$$= (c(S) + c(o))\frac{\alpha T'}{\beta v} > \alpha T'. \tag{2}$$

By the selection rule of e_v, we have $c(e_v) \leq (1+\epsilon)u > \mathsf{opt}'$, so $f(e_v) \geq \max_{o \in O'} f'(o)$. Combine this with the submodularity of f', we have

$$f'(S'_v) \geq \max\{f'(S_v), f(e_v)\} \geq \max\{f'(S_v), f'(o)\} \geq \frac{f'(S_v \cup \{o\})}{2} > \frac{\alpha T'}{2} \tag{3}$$

and $c(S'_v) \leq \max\{c(S'_v), c(e_v)\} = \max\{\beta u, (1+\epsilon)u\} \leq \beta u \leq \beta \mathsf{opt}'$.

Case 2. There is no such element $o \in O' \setminus S_v$, i.e., $c(S_v) + c(o) \leq \beta(1+\epsilon)^v$ and $\frac{f'(o|S_v)}{c(o)} \geq \frac{T'}{\beta(1+\epsilon)^v}$ for all $o \in O' \setminus S_v$. In this case, we also have $c(S'_v) \leq \beta \mathsf{opt}'$. By the monotoncity and submodularity of f' with a note that $u = (1+\epsilon)^v > \frac{\mathsf{opt}'}{1+\epsilon}$ we have:

$$f(O') - f(S_v) \leq f(O' \cup S_v) - f(S_v) \tag{4}$$

$$\leq \sum_{o \in O' \setminus S_v} f(o|S_v) \tag{5}$$

$$\leq \sum_{o \in O' \setminus S_u} c(o)\frac{\alpha T'}{\beta(1+\epsilon)^v} \leq \mathsf{opt}'\frac{\alpha T'}{\beta u} \tag{6}$$

$$< \frac{\alpha(1+\epsilon)T'}{\beta} \tag{7}$$

which implies that

$$f(S_v) \geq f(O') - \frac{\alpha(1+\epsilon)T'}{\beta} \geq T' - \frac{\alpha(1+\epsilon)T'}{\beta} = (1 - \frac{(1+\epsilon)\alpha}{\beta})T'. \tag{8}$$

Combine two case with choosing α, β in the algorithm, we have $f'(S'_v) \geq (1-\epsilon)T'$ and $c(S'_v) \leq \frac{1-\epsilon^2}{\epsilon}\mathsf{opt}'$. Therefore, the algorithm must meet the condition in line 13 and return the final solution after at most v iterations. If the algorithm returns S'_v, recap that $c(V_0) \leq \epsilon \mathsf{opt}$ and $\mathsf{opt}' \leq \mathsf{opt}$, we have:

$$c(S \cup V_0) \leq (\beta + \epsilon)\mathsf{opt} \leq \frac{\mathsf{opt}}{\epsilon}. \tag{9}$$

On the other hand,

$$f(S'_v \cup V_0) = f(S'_v \cup V_0) - f(V_0) + f(V_0) \tag{10}$$
$$= f'(S'_v) + f(V_0) \geq (1-\epsilon)(T - f(V_0)) + f(V_0) \tag{11}$$
$$= (1-\epsilon)T + \epsilon f(V_0) \geq (1-\epsilon)T. \tag{12}$$

If the algorithm meets the condition in line 13 at the iteration $k < v$, we have $f(S'_k) \geq (1-\epsilon)T'$ and thus $f(S'_k \cup V_0) \geq T$. Besides, $c(S'_k) \leq \beta(1+\epsilon)^k < \beta(1+\epsilon)^v = \beta\mathsf{opt}'$, the approximation guarantees holds.

Prove complexities. The algorithm makes one pass to finish the first phase and needs n memories to find S^0. For the second phase, the number of passes through V' is at most

$$\log_{1+\epsilon}(\frac{c(V')}{c_0}) < \log_{1+\epsilon}(\frac{nc'_{max}}{c'_{min}}) \leq \log_{1+\epsilon}(\frac{n^3}{\epsilon}) \tag{13}$$
$$= \frac{\log(\frac{n^3}{\epsilon})}{\log(1+\epsilon)} \leq \frac{1}{\epsilon}\log(\frac{n^3}{\epsilon}) = O(\frac{1}{\epsilon}\log(\frac{n}{\epsilon})), \tag{14}$$

in which each pass takes at most n queries. Therefore the memories needed, the total number of passes and the query complexity are $O(n)$, $O(\frac{1}{\epsilon}\log(\frac{n}{\epsilon}))$ and $O(\frac{n}{\epsilon}\log(\frac{n}{\epsilon}))$, respectively. □

5 Experimental Evaluation

In this section, we provide a comparative analysis of our algorithm alongside MULTI, SINGLE [3] and GREEDY [7] for the MSC problem. We evaluate their performance on two specific applications: Revenue Threshold and Coverage Threshold. Our assessment primarily centers around four essential metrics: the oracle value of the objective function, the number of queries, cost value, and memory usage.

5.1 Applications and Datasets

Revenue Threshold. Given a social network represented by a graph $G = (V, E)$, where V denotes the set of users and E represents the set of user connections. Each edge (u, v) is assigned a weight $w_{(u,v)}$ that is non-negative. We follow [12] to define the advertising revenue of any node set $S \subseteq V$ as $f(S) = \sum_{u \in V} R_u(S)$. For this evaluation, we choose $R_u(S) = \left(\sum_{v \in S} w_{uv}\right)^{\alpha_u}$, where α_u is chosen independently for each u uniformly in $(0, 1)$. The revenue objective $f(.)$ is monotone and submodular [9]. Different from the Revenue Maximization application defined by Kuhnle [9], the goal of Revenue Threshold is a solution S such that $f(S)$ exceeds a given threshold T such that the cost $c(S)$ is minimized. In this application, we utilized the ego Facebook dataset from [11]. This dataset consists of over 4K nodes and over 88K edges.

Coverage Threshold. Based on the Maximum Coverage described in [9], the Coverage Threshold can be described as follows: Considering a graph $G = (V, E)$, for any given subset $S \subseteq V$, we define S^I as the set comprising all vertices that share an incident edge with any vertex in S. Subsequently, we define the function $f(S)$ as the cardinality of S^I. It is worth noting that this objective function exhibits the properties of monotonicity and submodularity, as demonstrated in the work by Kuhnle [9]. The Coverage Threshold aims to find a solution S such that $f(S)$ exceeds a given threshold T such that the cost $c(S)$ is minimized. In our practical application, we employed datasets that consisted of an Erdős-Rényi (ER) random graph with 5000 nodes and an edge probability of 0.2. Additionally, the cost associated with each node, denoted as $c(u)$, was selected randomly and uniformly from the range between 0 and 1, following the methodology outlined in the study by Amanatidis *et al.*[1].

Experiment settings. We compare our algorithms with the applicable state-of-the-art algorithms listed below:

- **MULTI**: The streaming algorithm, as presented in [3], boasts a bicriteria approximation guarantee of $((1+\epsilon)(1+\frac{4}{\epsilon^2}), \frac{1-\epsilon}{2})$. This algorithm conducts $O(\log(\mathsf{opt})/c_{min})$ passes through the universe V, simultaneously retains elements with a total cost of $O(\mathsf{opt})$, and makes $O(\frac{n}{\epsilon}\log(\mathsf{opt}/c_{min}))$ queries to the function f when leveraging a linear-time algorithm for Unconstrained Submodular Maximization (USM) as a subroutine.
- **SINGLE**: The streaming algorithm, as introduced in [3], conducts a single pass through the universe V in an arbitrary order and provides an identical bicriteria approximation guarantee as MULTI. Nonetheless, it is worth noting that SINGLE requires a total number of queries to the function f that is within the order of $\Omega(n^2/c_{min})$.
- **GREEDY**: The greedy algorithm, as outlined in [7], achieves a bicriteria approximation guarantee of $(\log(\frac{T}{\epsilon}), 1-\epsilon)$ for the minimum target set selection problem (MINTSS).

In each experiment, our analysis begins with the execution of the double greedy algorithm, as developed by [2] and denoted as USM, to establish an initial benchmark for comparison. We use the following symbols to represent the characteristics of the USM algorithm: c_0 for cost, f_0 for the f-value, q_0 for the number of queries, and m_0 for memory usage. In all the generated plots, the y-axis is dedicated to normalized f-values relative to the threshold T, while cost values are normalized with respect to c_0, memory usage is normalized with respect to m_0, and the threshold T is normalized in the relation to f_0. Within our experimental framework, we systematically vary the threshold within the range of 0.1 to 0.5, relative to the reference value f_0, following the configuration specified in [3]. Additionally, we maintain a consistent setting of $\epsilon = 0.1$ for all algorithms used in these experiments.

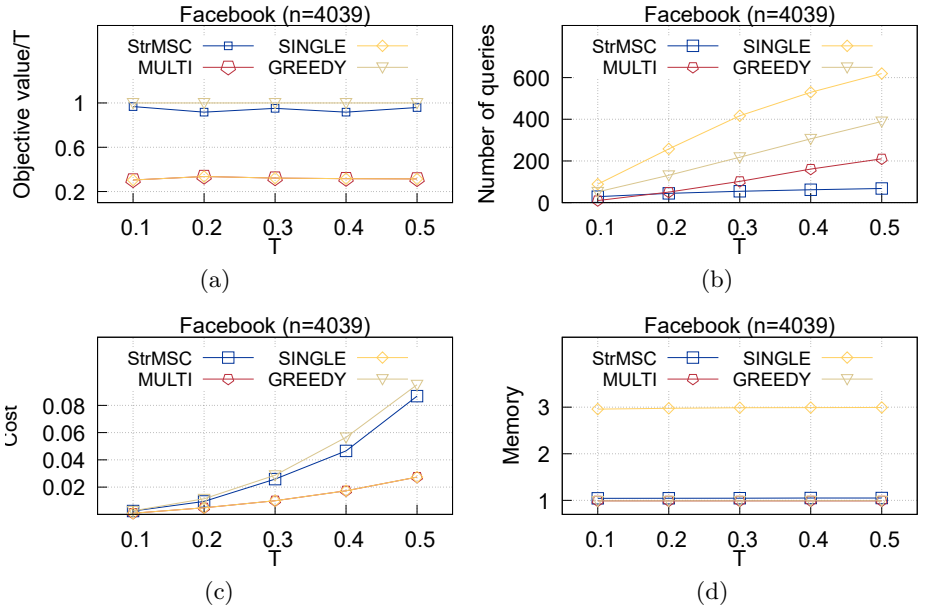

Fig. 1. Performance of algorithms for MSC on Revenue Threshold: (a) The objective values, (b) The number of queries (c) The cost values (d) The allocated memory

5.2 Experiment Results

The experimental results are depicted in Figures 1 and 2. Specifically, Figure 1 illustrates the results for the Revenue Threshold application, while Figure 2 presents the results for Coverage Threshold.

Firstly, our algorithm surpasses SINGLE and MULTI algorithms in terms of the objective value in Revenue Threshold (Figure 1.a) and Coverage Threshold (Figure 2.a) applications. This accomplishment is significant as our objective value closely approaches the asymptotic objective value of the GREEDY and converges in proximity to the threshold T. In stark contrast, the objective values of the SINGLE and MULTI algorithms are confined to a mere one-third of the threshold T.

Secondly, our algorithm demonstrates exceptional query efficiency (Figure 1.b, Figure 2.b), with the number of queries consistently ranking among the lowest compared to other algorithms. A notable attribute is its capacity to maintain a stable number of queries even as the threshold T increases, while other algorithms tend to experience significant increases in the number of queries under similar conditions.

Thirdly, with regard to the cost value analysis (Figure 1.c, Figure 2.c), our algorithm exhibits a higher cost value than the SINGLE and MULTI algorithms, yet it remains lower than that of the GREEDY. This cost disparity can be attributed to our algorithm's superior capacity to achieve a higher objective value.

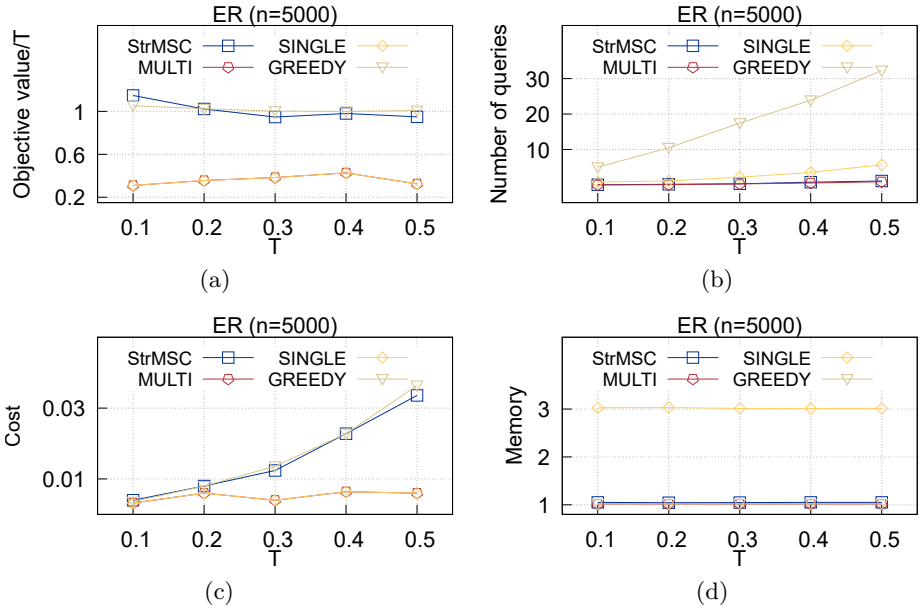

Fig. 2. Performance of algorithms for MSC on Coverage Threshold: (a) The objective values, (b) The number of queries (c) The cost values (d) The allocated memory

Nevertheless, it is crucial to emphasize that the assurance of a valid solution justifies this increased cost.

Lastly, concerning memory utilization (Figure 1.d, Figure 2.d), our StrMSC algorithm, along with the GREEDY and MULTI algorithms, exhibits similar memory usage patterns, closely aligning with the memory consumption of the USM algorithm. On the contrary, the SINGLE algorithm stands out due to its notably larger memory footprint, approximately three times that of the other algorithms.

In summary, our StrMSC algorithm excels in various evaluation criteria compared to the benchmarked algorithms. These criteria include objective value, number of queries, and memory utilization, all evaluated in the contexts of both Revenue Threshold and Coverage Threshold applications. While the cost value surpasses that of SINGLE and MULTI algorithms, it remains lower than that of GREEDY algorithm, which aligns with the observed differences in objective values.

6 Conclusion

In conclusion, this paper introduces a novel streaming algorithm designed to address the MSC problem. This algorithm yields a $(1-\epsilon, \frac{1}{\epsilon})$-bicriteria approximation solution for MSC while maintaining computational efficiency with only

$O(\log n)$ passes over the ground set V in the monotone case. To evaluate our algorithmic solutions, we conducted comprehensive experiments encompassing two diverse applications: Revenue Threshold and Coverage Threshold. The experimental outcomes unequivocally demonstrate the superior performance of our algorithms across various evaluation metrics when compared to MULTI, SINGLE, and GREEDY algorithms. Nevertheless, several open questions persist, igniting the spark for future research endeavors. Prominent among these questions is the pursuit of strategies to further minimize the solution's associated cost.

In summary, the algorithms proposed in this paper offer both efficiency and effectiveness in addressing the MSC problem. Our forthcoming research efforts will be dedicated to refining these algorithms and undertaking the formidable challenges that lie ahead in this domain.

References

1. Amanatidis, G., Fusco, F., Lazos, P., Leonardi, S., Reiffenhäuser, R.: Fast adaptive non-monotone submodular maximization subject to a knapsack constraint. In: Annual Conference on Neural Information Processing Systems (2020)
2. Buchbinder, N., Feldman, M., Seffi, J., Schwartz, R.: A tight linear time (1/2)-approximation for unconstrained submodular maximization. SIAM Journal on Computing **44**(5), 1384–1402 (2015)
3. Crawford, V.: Scalable bicriteria algorithms for non-monotone submodular cover. In: International Conference on Artificial Intelligence and Statistics. pp. 9517–9537. PMLR (2023)
4. Crawford, V., Kuhnle, A., Thai, M.: Submodular cost submodular cover with an approximate oracle. In: International Conference on Machine Learning. pp. 1426–1435. PMLR (2019)
5. Crawford, V.G.: Faster guarantees of evolutionary algorithms for maximization of monotone submodular functions. In: Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021. pp. 1661–1667. ijcai.org (2021)
6. El-Arini, K., Guestrin, C.: Beyond keyword search: discovering relevant scientific literature. In: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, CA, USA, August 21-24, 2011. pp. 439–447 (2011)
7. Goyal, A., Bonchi, F., Lakshmanan, L.V.S., Venkatasubramanian, S.: On minimizing budget and time in influence propagation over social networks. Social Netw. Analys. Mining **3**(2), 179–192 (2013)
8. Iwata, S.: Submodular function minimization. Mathematical Programming **112**, 45–64 (2008)
9. Kuhnle, A.: Quick streaming algorithms for maximization of monotone submodular functions in linear time. In: International Conference on Artificial Intelligence and Statistics. pp. 1360–1368. PMLR (2021)
10. Kuhnle, A., Crawford, V.G., Thai, M.T.: Scalable and adaptive algorithms for the triangle interdiction problem on billion-scale networks. In: 2017 IEEE International Conference on Data Mining, ICDM 2017, New Orleans, LA, USA, November 18-21, 2017. pp. 237–246 (2017)

11. Leskovec, J., Krause, A., Guestrin, C., Faloutsos, C., VanBriesen, J.M., Glance, N.S.: Cost-effective outbreak detection in networks. In: Proc. of the 13th ACM SIGKDD Conf., 2007. pp. 420–429 (2007)
12. Mirzasoleiman, B., Badanidiyuru, A., Karbasi, A.: Fast constrained submodular maximization: Personalized data summarization. In: International Conference on Machine Learning. JMLR Workshop and Conf. Proc., vol. 48, pp. 1358–1367 (2016)
13. Mitrovic, M., Kazemi, E., Zadimoghaddam, M., Karbasi, A.: Data summarization at scale: A two-stage submodular approach. In: ICML. p. 3593–3602 (2016)
14. Norouzi-Fard, A., Bazzi, A., Bogunovic, I., El Halabi, M., Hsieh, Y.P., Cevher, V.: An efficient streaming algorithm for the submodular cover problem. Advances in Neural Information Processing Systems **29** (2016)
15. Norouzi-Fard, A., Tarnawski, J., Mitrovic, S., Zandieh, A., Mousavifar, A., Svensson, O.: Beyond 1/2-approximation for submodular maximization on massive data streams. In: Proc. of the International Conference on Machine Learning. vol. 80, pp. 3826–3835 (2018)
16. Pham, C.V., Pham, D.V., Bui, B.Q., Nguyen, A.V.: Minimum budget for misinformation detection in online social networks with provable guarantees. Optimization Letters pp. 1–30 (2021)
17. Ran, Y., Zhang, Z., Tang, S.: Improved parallel algorithm for minimum cost submodular cover problem. In: Loh, P., Raginsky, M. (eds.) Conference on Learning Theory, 2-5 July 2022, London, UK. Proceedings of Machine Learning Research, vol. 178, pp. 3490–3502. PMLR (2022), `https://proceedings.mlr.press/v178/ran22a.html`
18. Tschiatschek, S., Iyer, R.K., Wei, H., Bilmes, J.A.: Learning mixtures of submodular functions for image collection summarization. In: Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada. pp. 1413–1421 (2014)
19. Wan, P.J., Du, D.Z., Pardalos, P., Wu, W.: Greedy approximations for minimum submodular cover with submodular cost. Computational Optimization and Applications **45**(2), 463–474 (2010)
20. Wolsey, L.A.: An analysis of the greedy algorithm for the submodular set covering problem. Combinatorica **2**(4), 385–393 (1982)