# Budgeted Sequence Submodular Maximization

**Xuefeng Chen**[1] , **Liang Feng**[1*] , **Xin Cao**[2] , **Yifeng Zeng**[3] , **Yaqing Hou**[4]
[1] College of Computer Science, Chongqing University, China
[2] School of Computer Science and Engineering, University of New South Wales, Australia
[3] Department of Computer and Information Sciences, Northumbria University, UK
[4] College of Computer Science and Technology, Dalian University of Technology, China
{xfchen, liangf}@cqu.edu.cn, xin.cao@unsw.edu.au, yifeng.zeng@northumbria.ac.uk, houyq@dlut.edu.cn

## Abstract

The problem of selecting a sequence of items that maximizes a given submodular function appears in many real-world applications. Existing study on the problem only considers uniform costs over items, but non-uniform costs on items are more general. Taking this cue, we study the problem of budgeted sequence submodular maximization (BSSM), which introduces non-uniform costs of items into the sequence selection. This problem can be found in a number of applications such as movie recommendation, course sequence design and so on. Non-uniform costs on items significantly increase the solution complexity and we prove that BSSM is NP-hard. To solve the problem, we first propose a greedy algorithm GBM with an error bound. We also design an anytime algorithm POBM based on Pareto optimization to improve the quality of solutions. Moreover, we prove that POBM can obtain approximate solutions in expected polynomial running time, and converges faster than a state-of-the-art algorithm POSEQSEL for sequence submodular maximization with cardinality constraints. We further introduce optimizations to speed up POBM. Experimental results on both synthetic and real-world datasets demonstrate the performance of our new algorithms.

## 1 Introduction

Submodular optimization is a fundamental optimization problem [Fujishige, 2005] which can be used in many applications. Most existing studies focus on the problem of selecting a subset of items from a whole set in order to maximize a given submodular function over the set, such as influence maximization [Kempe *et al.*, 2003] and information gathering [Leskovec *et al.*, 2007].

In many applications, the order of selecting items affects the utility of item sets, and thus it is desired to select a sequence of items instead of a subset. In Fig. 1, we use an example of movie recommendations to explain why sequence selection is meaningful and useful. Assume that a utility function over a sequence of movies measures how users are satisfied with these recommendation results. We can use the directed edges between two movies to represent that there is an additional utility in watching them by following the order, and the self-cycles to represent the utilities of watching individual movies. It can be observed that the order of watching movies affects users' experience. For example, watching $B_2$ :*Transformers 2* after $B_1$ :*Transformers 1* has a higher utility than watching the two movies in reverse order.

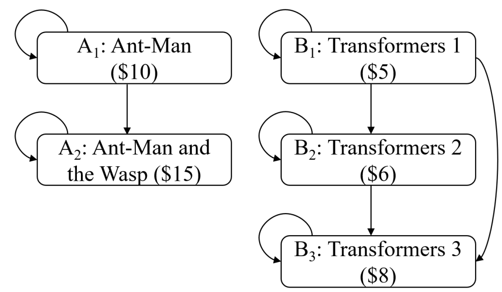



Figure 1: An example of ordered preferences for movie recommendations. A DVD price of a movie is used as its cost.

This interesting observation commonly occurs in recommender systems [McAuley *et al.*, 2015], paper reading plan [Shahaf *et al.*, 2012], course learning [Parameswaran *et al.*, 2011], etc. Thus, the problem of selecting a sequence of items that maximizes a given submodular function over the sequence has recently received increasing attention [Tschiatschek *et al.*, 2017; Qian *et al.*, 2018; Mitrovic *et al.*, 2019; Sallam *et al.*, 2020].

In many submodular optimization problems, the items have non-uniform costs [Khuller *et al.*, 1999; Bian *et al.*, 2020; Amanatidis *et al.*, 2020]. In sequence selection, non-uniform item costs are common as well. For instance, in the movie recommendation example, movies could have different costs (e.g., their DVD prices); in the course sequence design, costs (i.e., the time cost) of courses are also non-uniform. In addition, users may have a budget limit on the sequence recommended to them. However, all existing studies on sequence selection only consider uniform costs of items (i.e. cardinality constraints), although non-uniform costs of items are more general in many real-world applications.

*Liang Feng is the corresponding author.

In this paper, we study the problem of Budgeted Sequence Submodular Maximization (BSSM) by considering non-uniform costs of items, and prove the NP-hardness of the BSSM problem. The BSSM problem adopts the sequence submodular function proposed by Tschiatschek et al. [Tschiatschek *et al.*, 2017], as the function is more expressive and it can capture the effect of the order of items in the sequence on the utility of item sets. Note that the sequence submodular function of BSSM is submodular *on the sequences, but not on items*. Hence, a simple greedy algorithm that selects one item in each iteration greedily cannot return results with a guaranteed error bound [Tschiatschek *et al.*, 2017].

Due to non-uniform costs of items, the state-of-the-art algorithm OMEGA [Tschiatschek *et al.*, 2017] for sequence selection with uniform costs always obtains sequences with poor quality (as shown in the experimental study). To solve this problem, we propose a greedy algorithm GBM which picks an edge with the largest marginal cost-effective value, and we prove its error bound by exploiting the properties of the sequence submodular function.

Similar to traditional greedy algorithms, GBM often gets trapped at local optima in the search space. To address this issue and improve the algorithm effectiveness, we develop an anytime algorithm POBM based on Pareto optimization. POBM first reformulates the original constrained optimization problem as a bi-objective optimization problem that maximizes the objective function $f$ and minimizes the cost function $C$ simultaneously, then utilizes a randomized iterative method to solve it, and finally selects the best feasible solution from the maintained set of solutions. Our theoretical analysis shows that POBM not only obtains approximation solutions in expected polynomial running time, but also has chances to find optimal solutions. In addition, POBM has a faster expected rate of convergence compared to the state-of-the-art algorithm POSEQSEL [Qian *et al.*, 2018] for Sequence Submodular Maximization with Cardinality Constraints (SSMCC).

In summary, our main contributions are fourfold. Firstly, we propose the BSSM problem and show its NP-hardness. Secondly, we develop a greedy algorithm GBM, and an anytime algorithm POBM with some optimizations to solve this problem. Thirdly, we theoretically analyze the approximation ratio of GBM, and prove that POBM can achieve the same approximation guarantee as GBM in a reasonable time and can escape from the local optimum. Finally, we conduct experiments on both synthetic and real-world datasets to demonstrate the effectiveness and efficiency of our proposed algorithms. Note that the proofs of all theorems and some additional experimental results are presented in the Appendix.

## 2 Related Works

**Submodular Optimization.** Submodular optimization has been studied substantially due to its wide range of applications including viral marketing [Kempe *et al.*, 2003], information gathering [Leskovec *et al.*, 2007], deep neural network training [Joseph *et al.*, 2019], region search [Chen *et al.*, 2020] etc. A classical problem of submodular optimization is to maximize a non-negative monotone submodular function under cardinality constraints. To solve this problem, Nemhaser et al. [Nemhauser *et al.*, 1978] proposed a simple greedy algorithm with a constant factor approximation ratio of $1 - e^{-1}$ and Das et al. [Das and Kempe, 2011] further improved the approximation ratio by introducing a submodular ratio. Meanwhile, different types of generalizing submodular optimization have been the focus of many studies recently. For instance, there exist non-monotone submodular maximization [Amanatidis *et al.*, 2020], streaming submodular maximization [Halabi *et al.*, 2020], and continuous submodular maximization [Raut *et al.*, 2021], and so on. These studies aim to select a subset of items rather than a sequence.

**Sequence Submodular Maximization.** Many studies have considered sequence submodular maximization, since the sequence plays an important role in submodular optimization applications. Aliaei et al. [Alaei *et al.*, 2010] first considered sequence selection in submodular maximization by introducing non-decreasing sequence submodular functions. Zhang et al. [Zhang *et al.*, 2015] presented a string submodular function by relaxing the sequence-submodular function. These two submodular functions fail to consider the effect of the order of items in the sequence on the functions. To fill this gap, Tschiatschek et al. [Tschiatschek *et al.*, 2017] proposed a new class of sequence submodular functions on a directed graph. As this function is expressive, it has been used to study the submodularity on a hypergraph [Mitrovic *et al.*, 2018] and adaptive sequence submodularity [Mitrovic *et al.*, 2019]. To solve all the above problems of sequence submodular maximization, Qian et al. [Qian *et al.*, 2018] proposed an algorithm POSEQSEL based on Pareto optimization. Meanwhile, Sara et al. [Bernardini *et al.*, 2020] offered a unified view of sequence submodularity, and Gamal et al. [Sallam *et al.*, 2020] first studied the problem of robust sequence submodular maximization. However, these works only consider uniform costs (i.e. cardinality constraints).

**Submodular Optimization with Non-uniform Costs (Knapsack Constraint).** Although uniform costs have been used extensively in existing submodular optimization studies, non-uniform costs are more general in real-world applications. Khuller et al. [Khuller *et al.*, 1999] proposed a budgeted maximum coverage problem which first considered non-uniform costs in submodular optimization, and they developed a greedy algorithm with an error bound. Sviridenko [Sviridenko, 2004], Krause and Guestrin [Krause and Guestrin, 2005] presented efficient approximation algorithms for the budgeted maximization of nondecreasing submodular set functions. Georgios et al. [Amanatidis *et al.*, 2020] designed a fast randomized greedy algorithm that achieves a 5.83 approximation for non-monotone submodular maximization subject to a knapsack constraint. In recent years, Qian et al. [Qian *et al.*, 2017] considered a general cost constraint on a subset selection which is a type of submodular maximization, and proposed POMC algorithm based on Pareto optimization for the problem. To solve the problem more effectively, Bian et al. [Bian *et al.*, 2020] developed a new anytime algorithm EAMC with a better approximation ratio. None of the existing work considers non-uniform costs of items for sequence submodular maximization.

## 3 Problem Formulation

In this section, we formally define the problem of Budgeted Sequence Submodular Maximization (BSSM) and show its NP-hardness. Here, we adopt the sequence submodular setting [Tschiatschek *et al.*, 2017], as it considers the effect of the order of items in the sequence on the utility of item sets [Tschiatschek *et al.*, 2017; Mitrovic *et al.*, 2018]. We first present the sequence submodular setting as below.

**Definition 1. Item Sequences.** *Given a set of $n$ items $V = (v_1, v_2, \cdots, v_n)$, a sequence from $V$ is represented as $s = \{(s_1, s_2, \cdots, s_k) | s_i \in V, k \in \mathbb{Z}^+\}$, and $s \in \mathcal{S}$, where $\mathcal{S}$ is the set of all possible sequences of items from $V$, and $k = 0$ represents the empty sequence $\emptyset$. For two sequences $s$ and $t$, their concatenation is denoted as $s \oplus t$.*

The utility function of sequences can be defined through a directed graph $G = (V, E)$ where vertices correspond to the items $V$, and a set of edges $E$ represents the utility in picking items in a certain order. An example of $G$ is shown in Fig. 1. Specifically, the edge $e_{ij} = (v_i, v_j)$ represents the utility in selecting $v_j$ after $v_i$, and the self-cycle $e_{ii}(v_i, v_i)$ represents the utility of selecting $v_i$ individually. We define the utility function as below.

$$f(s) = h(E(s)), \tag{1}$$

where $E(s) = \{(s_i, s_j) | (s_i, s_j) \in E, i \leq j\}$ is the set of edges induced by the sequence $s$, and $h : 2^E \rightarrow \mathbb{R}^+$ is a non-negative monotone submodular set function over the edges. It means that for an edge $e \in E$, $h(E_1 \bigcup \{e\}) - h(E_1) \geq h(E_2 \bigcup \{e\}) - h(E_2)$ if $E_1 \subseteq E_2 \subseteq E$ and $e \notin E_1$.

However, the utility function $f$ is neither a set function nor submodular on items. We use an example to explain this. As shown in the example of Fig. 1, and assume that $h(E(s))$ counts the number of edges in $E(s)$, and thus the utilities of the sequences$(A_1)$, $(A_2)$, $(A_1, A_2)$ and $(A_2, A_1)$ are computed as:

$f((A_1)) = h(\{(A_1, A_1)\}) = 1$
$f((A_2)) = h(\{(A_2, A_2)\}) = 1$
$f((A_1, A_2)) = h(\{(A_1, A_1), (A_2, A_2), (A_1, A_2)\}) = 3$
$f((A_2, A_1)) = h(\{(A_1, A_1), (A_2, A_2)\}) = 2$

Note that the order of selecting items affects the utilities of item sets, i.e., $f((A_1, A_2)) \neq f((A_2, A_1))$, and the function $f$ is not submodular over item sets, because $f((A_1)) - f(\emptyset) \ngeq f((A_1, A_2)) - f((A_2))$ although $\emptyset \subset A_2$ (i.e., it does not satisfy the diminishing returns property). On the other hand, the function $f$ is more expressive than submodular functions over items. This is because, when the graph $G$ only has self-cycles and no other edges, $f$ can express any submodular set function.

Besides the utilities of sequences, we consider the costs of sequences, and we define the cost score of a sequence as the sum of the costs on all items in the sequence, which is computed as: $C(s) = \sum_{v_i \in s} c_{v_i}$, where $c_{v_i}$ is the cost of $v_i$.

Formally, we define the BSSM problem as follows:

**Definition 2. Budgeted Sequence Submodular Maximization (BSSM).** *Given a set of $n$ items $V = (v_1, v_2, \cdots, v_n)$ with item costs $(c_{v_1}, c_{v_2}, \cdots, c_{v_n})$, a utility function $f$ over the sequences of items, and a budget constraint $\Delta$, the target is to find a sequence $s$ such that*

$$\begin{aligned} &s = argmax_{s \in \mathcal{S}} \ f(s) \\ &subject\ to\ \ C(s) \leq \Delta \end{aligned} \tag{2}$$

For example, consider the instance of movie recommendation in Fig. 1 again. Given $\Delta = 20$, the optimal sequence of the BSSM problem is $(B_1, B_2, B_3)$, as it allows the user to fully enjoy three movies under the budget constraint.

**Theorem 1.** *The BSSM problem is NP-hard.*

## 4 The GBM Algorithm

In this section, we propose a novel greedy algorithm called GBM for solving the BSSM problem and prove its error bound. Instead of picking the edge with the maximum marginal utility (which is conducted by OMEGA [Tschiatschek *et al.*, 2017]), GBM chooses the edge with the maximum marginal cost-effective value in each step until no more edges can be inserted into the solution. This can guarantee the approximation ratio of GBM.

As shown in Alg. 1, GBM starts by initializing a candidate edges set $E^{ca}$ and an edge set $E^{se}$ for storing the selected edges (line 1). After that, the algorithm iteratively and greedily extends $E^{se}$ until no more edges can be added (lines 2-6). In each iteration, the algorithm first updates $E^{ca}$ by pruning the edges whose all vertexes (i.e., items) belongs to $V(E^{se})$ or would make the cost of the selected edge set exceed the budget constraint (lines 3-4). Next, it selects the edge $e^*$ with the maximum marginal cost-effective value $\Delta_f / \Delta_C = \frac{f(RE(E^{se} \cup \{e_l\})) - f(RE(E^{se}))}{C(E^{se} \cup \{e_l\}) - C(E^{se})}$ to insert into $E^{se}$ (lines 5-6), where $C(E^{se}) = \sum_{v_i \in V(E^{se})} c_{v_i}$, $V(E^{se})$ denotes the set of items in $E^{se}$, and $RE(E^{se})$ is a function for obtaining the optimal order with the maximal value of the utility function over all possible orders of items contained in $E^{se}$ (i.e., $RE(E^{se}) = REORDER(E^{se})$, and the algorithm of implementing function $REORDER$ is shown in the previous work [Tschiatschek *et al.*, 2017]). After finishing the extension of $E^{se}$, the algorithm obtains the best sequence $s^1$ in $E^{se}$ using $RE(E^{se})$ (line 7). It also gets another sequence $s^2$ that only has one edge with the maximal utility and satisfies the

**Algorithm 1:** GBM Algorithm

**Input:** $G = (V, E)$, a utility function $f$, item costs $(c_{v_1}, c_{v_2}, \cdots, c_{v_n})$, $\Delta$
**Output:** A sequence $s$
1 $E^{ca} \leftarrow E$, $E^{se} \leftarrow \emptyset$;
2 **while** $E^{ca} != \emptyset$ **do**
3 　$E^{ca} \leftarrow E^{ca} \setminus \{e_l = (v_i, v_j) | v_i, v_j \in V(E^{se})\}$;
4 　$E^{ca} \leftarrow E^{ca} \setminus \{e_l \in E^{ca} | C(E^{se} \cup \{e_l\}) > \Delta\}$;
5 　$e^* \leftarrow argmax_{e_l \in E^{ca}} \frac{f(RE(E^{se} \cup \{e_l\})) - f(RE(E^{se}))}{C(E^{se} \cup \{e_l\}) - C(E^{se})}$;
6 　$E^{se} \leftarrow E^{se} \cup \{e^*\}$;
7 $s^1 \leftarrow RE(E^{se})$;
8 $e' \leftarrow argmax_{e_l \in E, C(e_l) \leq \Delta} f(RE(e_l))$;
9 $s^2 \leftarrow RE(\{e'\})$;
10 $s \leftarrow argmax_{s^i \in \{s^1, s^2\}} f(s^i)$;
11 **return** the sequence $s$;

budget constraint (lines 8-9). Finally, it returns the sequence with the largest utility as the solution $s$.

We obtain the approximation ratio of GBM in Theorem 2. For the sake of clarity, we let $c_{min} = min_{v_i \in V} c_{v_i}$ and introduce a parameter $\beta = 4\lfloor \frac{\Delta}{c_{min}} \rfloor$.

**Theorem 2.** *When $G$ is a DAG (not counting self-cycles), GBM offers an approximation ratio of $\frac{1}{\beta+2}(\frac{1}{\beta})^{\lfloor \frac{\Delta}{c_{min}} \rfloor}(1 - e^{-1})$.*

Next, we analyze the complexity of GBM. It requires $\lfloor \frac{\Delta}{c_{min}} \rfloor$ iterations in the worst case, and in each iteration, it takes $O(n^2)$ time to update $E^{ca}$, and costs $O(n^2 \lfloor \frac{\Delta}{c_{min}} \rfloor log \lfloor \frac{\Delta}{c_{min}} \rfloor)$ time to select the edge $e^*$ for inserting into $E^{se}$, since the number of items in $E^{se}$ is less than $\lfloor \frac{\Delta}{c_{min}} \rfloor$ and the complexity of computing $f(RE(E^{se}))$ is $O(\lfloor \frac{\Delta}{c_{min}} \rfloor log \lfloor \frac{\Delta}{c_{min}} \rfloor)$. And then, GBM costs $O(\lfloor \frac{\Delta}{c_{min}} \rfloor log \lfloor \frac{\Delta}{c_{min}} \rfloor))$ time and $O(n^2)$ time to obtain $s^1$ and $s^2$, respectively. Thus, the time complexity of GBM is $O(n^2 (\lfloor \frac{\Delta}{c_{min}} \rfloor)^2 log \lfloor \frac{\Delta}{c_{min}} \rfloor)$.

## 5 The POBM Algorithm

Although GBM can achieve an approximate solution in a reasonable time, it usually gets trapped in a local optimum due to the greedy rule. To alleviate this issue, we design an anytime algorithm POBM based on Pareto Optimization [Qian *et al.*, 2015]. For POBM, most of its steps are similar to those of POMC [Qian *et al.*, 2017] which is for the problem of subset selection with a general cost constraint. But POBM uses a different objective function and adopts the function $RE$ (which is mentioned in the above section) to obtain the best sequence from an item set. Note that the input of function $RE$ can be an item set or an edge set.

Let a Boolean vector $\boldsymbol{p} \in \{0,1\}^n$ represent an item set, and $C(\boldsymbol{p}) = \sum_{v_i \in \boldsymbol{p}} c_{v_i}$. POBM reformulates BSSM as a bi-objective maximization problem.

$$argmax_{\boldsymbol{p} \in \{0,1\}^n} (f_1(\boldsymbol{p}), f_2(\boldsymbol{p})),$$

where $f_1(\boldsymbol{p}) = \begin{cases} -\infty, & C(\boldsymbol{p}) \geq 2\Delta \\ f(RE(\boldsymbol{p})), & otherwise \end{cases}$,

and $f_2(\boldsymbol{p}) = -C(\boldsymbol{p})$. It means that POBM maximizes the utility function $f$ and minimizes the cost function $C$ simultaneously. By setting $f_1$ to $-\infty$, we exclude overly infeasible solutions. In the bi-objective setting, we consider both the two objective scores to compare two item sets $\boldsymbol{p}$ and $\boldsymbol{p'}$. $\boldsymbol{p}$ weakly dominates $\boldsymbol{p'}$ (i.e., $\boldsymbol{p}$ is **better** than $\boldsymbol{p'}$, denoted as $\boldsymbol{p} \succeq \boldsymbol{p'}$) if $f_1(\boldsymbol{p}) \geq f_1(\boldsymbol{p'})$ and $f_2(\boldsymbol{p}) \geq f_2(\boldsymbol{p'})$; $\boldsymbol{p}$ dominates $\boldsymbol{p'}$ (i.e., $\boldsymbol{p}$ is **strictly better** than $\boldsymbol{p'}$, denoted as $\boldsymbol{p} \succ \boldsymbol{p'}$) if $\boldsymbol{p} \succeq \boldsymbol{p'}$ and either $f_1(\boldsymbol{p}) > f_1(\boldsymbol{p'})$ or $f_2(\boldsymbol{p}) > f_2(\boldsymbol{p'})$. But if neither $\boldsymbol{p}$ is better than $\boldsymbol{p'}$ nor $\boldsymbol{p'}$ is better than $\boldsymbol{p}$, they are **incomparable**.

The procedure of POBM is presented in Alg. 2. It begins with initializing a solution set $P$ by adding the empty item set $\{0\}^n$ into it (line 1). In the following steps, it iteratively tries to improve the quality of the item sets in $P$ (lines 2-9). In each iteration, a new item set $\boldsymbol{p'}$ is generated by randomly flipping bits of an archived item set $p$ selected from the current $P$ randomly (lines 3-4); if $\boldsymbol{p'}$ is not dominated by any archived item set in $P$, it will be inserted into $P$, and meanwhile, the archived item sets which are weakly dominated by $\boldsymbol{p'}$ will be pruned from $P$ (lines 5-6). Obviously, $P$ always contains incomparable item sets. After $T$ iterations, the algorithm obtains the best feasible item set $\boldsymbol{p^{be}}$ with the maximum utility score from $P$, and then gets the best sequence $s$ from $\boldsymbol{p^{be}}$ to return as a solution (lines 8-10).

---

**Algorithm 2:** POBM Algorithm

**Input:** $G = (V, E)$, a utility function $f$, item costs $(c_{v_1}, c_{v_2}, \cdots, c_{v_n})$, $\Delta$, the number $T$ of iterations
**Output:** A sequence $s$

1 $\boldsymbol{p} \leftarrow \{0\}^n$, $P \leftarrow \{\boldsymbol{p}\}$, $t \leftarrow 0$;
2 **while** $t < T$ **do**
3     Obtain $\boldsymbol{p}$ from $P$ uniformly at random;
4     Generate $\boldsymbol{p'}$ by flipping each bit of $\boldsymbol{p}$ with prob. $\frac{1}{n}$;
5     **if** $\nexists \boldsymbol{z} \in P$ *such that* $\boldsymbol{z} \succ \boldsymbol{p'}$ **then**
6         $P \leftarrow (P \setminus \{\boldsymbol{z} \in P | \boldsymbol{p'} \succeq \boldsymbol{z}\}) \cup \{\boldsymbol{p'}\}$;
7     $t = t + 1$;
8 $\boldsymbol{p^{be}} \leftarrow argmax_{\boldsymbol{p} \in P, C(\boldsymbol{p}) \leq \Delta} f(RE(\boldsymbol{p}))$;
9 $s \leftarrow RE(\boldsymbol{p^{be}})$;
10 **return** the sequence $s$;

---

The number $T$ of iterations affects the solution quality of POBM, we analyze their relation in a theoretical way, and achieve the approximation ratio of POBM in Theorem 3, where $\mathbb{E}(T)$ denotes the expected number of iterations, $P_{max}$ denotes the largest size of $P$ during the running process of POBM, and $s^*$ denotes an optimal sequence. Generally, we set $c_{v_i} \in \mathbb{Z}^+$ for each $v_i \in V$, thus $P_{max} \leq 2\Delta$.

**Theorem 3.** *When $G$ is a DAG (not counting self-cycles), POBM with $\mathbb{E}(T) \leq \lfloor \frac{\Delta}{2c_{min}} \rfloor en^2 P_{max}$ finds a sequence $s$ with $C(s) \leq \Delta$ and $f(s) \geq \frac{1}{\beta+2}(\frac{1}{\beta})^{\lfloor \frac{\Delta}{c_{min}} \rfloor}(1 - e^{-1}) f(s^*)$.*

Theorem 3 shows that POBM can obtain the same approximation ratio as GBM. Next, we explain that POBM has chances to find a global optimum in the following theorem.

**Theorem 4.** *When $G = (V, E)$ is a DAG (not counting self-cycles), POBM with $\mathbb{E}(T) \leq en^{\lfloor \frac{\Delta}{c_{min}} \rfloor} P_{max}$ achieves $s^*$.*

Algorithm POSEQSEL [Qian *et al.*, 2018] which is proposed for SSMCC is also an anytime algorithm based on Pareto Optimization. We observe that POSEQSEL can be adopted to solve BSSM after replacing its two objective functions with those of POBM. And then, we analyze the approximation ratio of POSEQSEL in BSSM and get the following Theorem.

**Theorem 5.** *When $G$ is a DAG (not counting self-cycles), POSEQSEL with $\mathbb{E}(T) \leq \lfloor \frac{2\Delta}{c_{min}} \rfloor en^2 P_{max}$ finds a sequence $s$ with $C(s) \leq \Delta$ and $f(s) \geq \frac{1}{\beta+2}(\frac{1}{\beta})^{\lfloor \frac{\Delta}{c_{min}} \rfloor}(1 - e^{-1}) f(s^*)$.*

Theorem 5 illustrates that, compared to POSEQSEL, POBM has a better expected rate of convergence for achieving the approximate solution. We will further verify it in the experimental study.

Since POBM is an anytime randomized iterative algorithm, it would consume a lot of time when the number of iterations is large. To improve the efficiency of POBM Algorithm, we present two optimizations: speeding up computing $E(RE(\boldsymbol{p}))$ and the quick dominance check.

We first speed up computing $E(RE(\boldsymbol{p}))$. As $\boldsymbol{p'}$ is generated by flipping each bit of $\boldsymbol{p}$ with probability $\frac{1}{n}$, there is one different bit between $\boldsymbol{p'}$ and $\boldsymbol{p}$ in average. Thus, we can reuse $E(RE(\boldsymbol{p}))$ to compute $E(RE(\boldsymbol{p'}))$ quickly in two steps.

In step 1, we obtain the items in $\boldsymbol{p}$ and not in $\boldsymbol{p'}$ (i.e., $\boldsymbol{p^{de}} = \{v_i \in \boldsymbol{p} \setminus \boldsymbol{p'}\}$), and then delete all edges which has an item in $\boldsymbol{p^{de}}$ from $E(RE(\boldsymbol{p}))$ (i.e., $E^{s1} = E(RE(\boldsymbol{p})) \setminus E^{de}$ and $E^{de} = \{(v_i, v_j) | v_i \in \boldsymbol{p^{de}} \vee v_j \in \boldsymbol{p^{de}}\}$).

In step 2, we first achieve the items in $\boldsymbol{p'}$ and not in $\boldsymbol{p}$ (i.e., $\boldsymbol{p^{ad}} = \{v_i \in \boldsymbol{p'} \setminus \boldsymbol{p}\}$). After that, we get the edges among items in $\boldsymbol{p^{ad}}$, and between items in $\boldsymbol{p^{ad}}$ and items in $\boldsymbol{p} \setminus \boldsymbol{p^{de}}$ (i.e., $E^{ad} = \{(v_i, v_j) | v_i, v_j \in \boldsymbol{p^{ad}} \vee v_i \in \boldsymbol{p^{ad}}, v_j \in \boldsymbol{p} \setminus \boldsymbol{p^{de}}\}$). Finally, we can obtain $E(RE(\boldsymbol{p'}))$ by combining $E^{s1}$ and $E^{ad}$ (i.e., $E(RE(\boldsymbol{p'})) = E^{s1} \cup E^{ad}$)

Next, we describe the optimization for the quick dominance check. For each new solution, POBM needs to do a dominance check, and then delete all weakly dominated solutions if the new solution is not dominated by any solution in the archive $P$. To accelerate this process, we first sort the solutions in $P$ in ascending order of their cost scores. After that, we can only use the solution $\boldsymbol{p}$ that has the closest cost score to the new solution $\boldsymbol{p'}$ and $C(\boldsymbol{p}) \leq C(\boldsymbol{p'})$ to check whether $\boldsymbol{p'}$ is dominated by any solution in $P$. If not, we insert $\boldsymbol{p'}$ into $P$, and then check $\boldsymbol{p}$ and the solutions after $\boldsymbol{p}$ in $P$ one by one and delete the weakly dominated solutions, until the solution's utility score is larger than that of $\boldsymbol{p'}$ or all solutions are scanned.

**Remarks.** If the graph $G$ is not a DAG, GBM and POBM can still be used for BSSM, as function $RE$ can get approximate orders by computing a feedback vertex set of $G$ [Karp, 1972]. Although the theoretical guarantees cannot hold in this case, GBM and POBM can also achieve high-quality solutions, which is demonstrated in the experimental study.

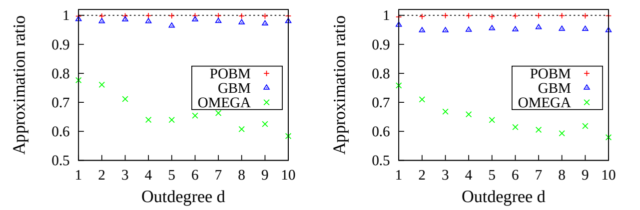
## 6 Experimental Study

In this section, we study the experimental performance of our algorithms using both synthetic and real-world datasets. We denote POBM with the optimizations of speeding up computing $E(RE(p))$ and the quick dominance check as POBMOpt. We use two state-of-the-art algorithm for SSMCC (i.e., OMEGA [Tschiatschek *et al.*, 2017] and POSEQSEL [Qian *et al.*, 2018]) as baseline methods. We implement all the algorithms in C++ on Windows 10, and run on a desktop with an Intel(R) i7-10700 2.9 GHz CPU and 32 GB memory.

### 6.1 Synthetic Datasets

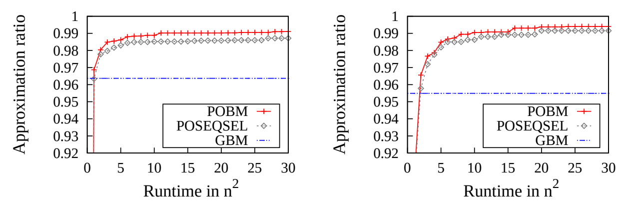
**Datasets and Parameter Settings**

We generate the synthetic datasets following the previous work [Tschiatschek *et al.*, 2017; Qian *et al.*, 2018] for the problem of sequence selection. We first construct the graph $G = (V, E)$ as follows: for each item $v_i \in V$, select a subset of size $min\{d, n - i\}$ uniformly at random from $\{v_{i+1}, \cdots, v_n\}$, where $d$ is the maximum out-going degree of the graph, and then build an edge from $v_i$ to each item in the selected subset and to itself (self-cycles). To assign a utility $w_{i,j}$ to each edge $(v_i, v_j)$, we consider two sets of functions $h : 2^E \to \mathbb{R}^+$, one is modular with $h(E(s)) = \sum_{(v_i,v_j)\in E(s)} w_{i,j}$, and the other one is submodular with $h(E(s)) = \sum_{v_j \in V(E(s))}[1 - \prod_{(v_i,v_j)\in E(s)}(1 - w_{i,j})]$. For the modular $h$, we get each utility $w_{i,j}$ by sampling from $[0, 1]$ randomly. For the submodular $h$, we get each utility $w_{i,j}$ with $i < j$ by sampling from $[0, 1]$ randomly, and each utility $w_{i,i}$ by sampling from $[0, 0.1]$ randomly. We obtain the cost $c_{v_i}$ for each item $v_i \in V$ by sampling the values from $\{1, 2, 3, 4, 5\}$ randomly. To compute the approximation ratios of algorithms, we use an exhaustive enumeration to find the optimal solutions and set $n = 50$, $B = 10$ by default. For each experiment, we generate 50 problem instances randomly and report the average results.




(a) Modular $h$ (b) Submodular $h$

Figure 2: Comparison of algorithms for modular and submodular utility functions over the edges with varying maximum outdegree $d$.




(a) Modular $h$ (b) Submodular $h$

Figure 3: The effect of $T$ on the solution quality for POBM and POSEQSEL over modular and submodular utility functions.

**Performance of Our Methods**

We first compare GBM and POBM with OMEGA in terms of the solution quality by varying maximum outdegree $d$ from 1 to 10, as the proposed optimizations do not affect the solution quality. We set the number $T$ of iterations of POBM as $2\Delta\lfloor\frac{\Delta}{2c_{min}}\rfloor en^2$, that is suggested by Theorem 3. The approximation ratios of the three algorithms are shown in Fig. 2. It illustrates that GBM and POBM can achieve high-quality solutions, while the solution quality of OMEGA is poor, as OMEGA is designed for the uniform cost setting. POBM outperforms other algorithms and almost finds the optimum.

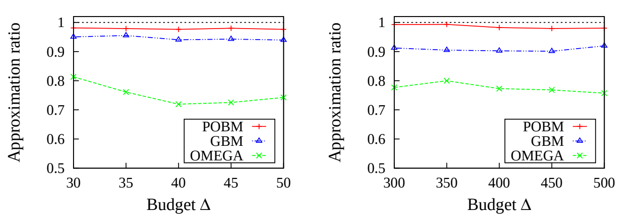
Next, we investigate the effect of $T$ on the solution quality for POBM and POSEQSEL in both modular and submodular utility functions with $d = 5$. Fig. 3 shows the curve of the approximation ratio over time for POBM and POSEQSEL by using GBM as the baseline. It demonstrates that POBM and POSEQSEL can quickly find a better solution whose approximation ratio is more than 95%, and POBM converges faster than POSEQSEL, which is consistent with the theoretical analysis. Note that the results of GBM keep unchanged, as they are not affected by $T$. According to this result, we set $T = 10n^2$ for POBM by default. Note that the result of examining the acceleration of our optimizations on POBM is reported in the Appendix.

## 6.2 Two Real-world Datasets

**Datasets and Parameter Settings**

We use two real-world datasets, one is the Movielens 1M (MOV) dataset [Harper and Konstan, 2015] and the other one is the XuetangX (XTX) dataset [Feng *et al.*, 2019]. MOV contains $1,000,209$ time-stamped ratings made by $6,040$ users for $3,706$ different movies in MovieLens platform, and XTX has the tracking log files that records the $772,887$ users' learning behavior over $1,629$ courses in XuetangX platform from August 2015 to August 2017. They will be used to do a movie recommendation and a course sequence design task, respectively. In order for our data to be representative of the general population, referring to the work [Mitrovic *et al.*, 2018], we preprocess those datasets and then obtain $412,222$ ratings made by $2,549$ users for 882 different movies in MOV, and the tracking log files made by $238,834$ users' over 956 courses in XTX.

Following the work [Tschiatschek *et al.*, 2017], we use the utility function $h(E(s)) = \sum_{v_j \in V(E(s))}[1 - \prod_{(v_i,v_j)\in E(s)}(1 - p_{j|i})]$, where $p_{j|i}$ associated on the edge $(v_i, v_j)$ is the conditional probability that a user rates movie (or enrolls course) $v_j$ given that she has rated movie (or enrolled course) $v_i$ before, and $p_{i|i}$ associated on self-cycles is the item frequency $p_i$. We next construct the graph $G = (V, E)$ to compute $p_{j|i}$ referring to the work [Tschiatschek *et al.*, 2017]. We also obtain the cost $c_{v_i}$ for each movie by crawling the purchase price from Amazon's website, and extract the costs of courses from the XTX dataset directly.



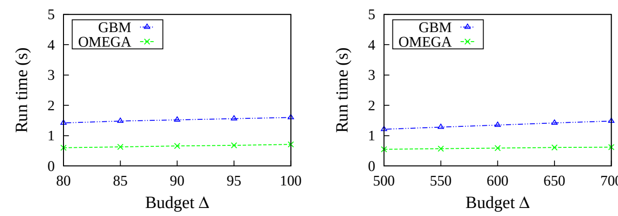

(a) MOV (b) XTX

Figure 4: Comparison of algorithms' solution quality by varying $\Delta$ on MOV and XTX datasets.

**Performance of Our Methods**

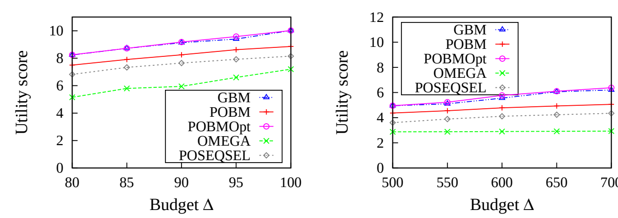
We first compare GBM and POBM with OMEGA in terms of the solution quality on both MOV and XTX datasets by varying the budget constraint. To compute the approximation ratios of algorithms, we use an exhaustive enumeration to find the optimal solutions and sample 50 movies (or courses) randomly for each instance (i.e., $n = 50$). And we generate 50 problem instances randomly and report the average results. As shown in Fig. 4, although the worst-case error bound of GBM is poor, GBM can obtain high-quality solutions, and POBM nearly achieves the optimal solutions.

To further compare the performance of these algorithms, we run them on the entire MOV and XTX datasets with larger $\Delta$ to solve the problem. The run time of GBM and OMEGA on both datasets is shown in Fig. 5. As POBM, POBMOpt and POSEQSEL require a large number of iterations when the number of items $n$ is large, we set a time limit $TimLim = 30s$ for them to compare their solution quality with that of GBM and OMEGA. As shown in Fig. 6, in terms of the solution quality, POBMOpt $>$ POBM $>$ POSEQSEL, it demonstrates that, our optimization can speed up POBM well, and POBM has a faster rate of convergence than POSEQSEL. Note that GBM can achieve high-quality solutions within 2s when $\Delta$ is large, and the solution quality of POBMOpt is always the best on both datasets.




(a) MOV (b) XTX

Figure 5: Comparison of the run time for GBM and OMEGA on MOV and XTX datasets.




(a) MOV (b) XTX

Figure 6: Comparison of algorithms' utility scores on MOV and XTX datasets.

# 7 Conclusion

We propose the BSSM problem which aims to find the optimal sequence such that it maximizes the utility of the sequence computed by a sequence submodular function under a given budget constraint. The problem is proved to be NP-hard. To solve the BSSM problem, we propose a greedy algorithm GBM and an anytime algorithm POBM based on Pareto optimization. We also analyze the approximation ratios of GBM and POBM, and present some optimizations to speed up POBM. The results of empirical studies on both synthetic and real-world datasets verify the theoretical analysis and show that our proposed algorithm can perform well in practice. In future work, would like to focus on the theoretical development of a parallel POBM.


# Acknowledgments

This work was supported in part by NSFC Grants (No. 61876025, 61836005, 62176225, and 61906032), the Fundamental Research Funds for the Central Universities under Grant DUT21TD107, and ARC DE190100663.

## References


[Alaei *et al.*, 2010] Saeed Alaei, Ali Makhdoumi, and Azarakhsh Malekian. Maximizing sequence-submodular functions and its application to online advertising. *arXiv preprint arXiv:1009.4153*, 2010.

[Amanatidis *et al.*, 2020] Georgios Amanatidis, Federico Fusco, Philip Lazos, Stefano Leonardi, and Rebecca Reiffenhäuser. Fast adaptive non-monotone submodular maximization subject to a knapsack constraint. In *NeurIPS*, 2020.

[Bernardini *et al.*, 2020] Sara Bernardini, Fabio Fagnani, and Chiara Piacentini. Through the lens of sequence submodularity. In *ICAPS*, pages 38–47, 2020.

[Bian *et al.*, 2020] Chao Bian, Chao Feng, Chao Qian, and Yang Yu. An efficient evolutionary algorithm for subset selection with general cost constraints. In *AAAI*, pages 3267–3274, 2020.

[Chen *et al.*, 2020] Xuefeng Chen, Xin Cao, Yifeng Zeng, Yixiang Fang, and Bin Yao. Optimal region search with submodular maximization. In *IJCAI*, pages 1216–1222, 2020.

[Cormen *et al.*, 2009] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2009.

[Das and Kempe, 2011] Abhimanyu Das and David Kempe. Submodular meets spectral: Greedy algorithms for subset selection, sparse approximation and dictionary selection. In *ICML*, pages 1057–1064, 2011.

[Feng *et al.*, 2019] Wenzheng Feng, Jie Tang, and Tracy Xiao Liu. Understanding dropouts in moocs. In *AAAI*, pages 517–524, 2019.

[Fujishige, 2005] Satoru Fujishige. *Submodular functions and optimization*. Elsevier, 2005.

[Halabi *et al.*, 2020] Marwa El Halabi, Slobodan Mitrović, Ashkan Norouzi-Fard, Jakab Tardos, and Jakub Tarnawski. Fairness in streaming submodular maximization: Algorithms and hardness. In *NeurIPS*, 2020.

[Harper and Konstan, 2015] F Maxwell Harper and Joseph A Konstan. The movielens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems*, 5(4):1–19, 2015.

[Joseph *et al.*, 2019] KJ Joseph, Krishnakant Singh, and Vineeth N Balasubramanian. Submodular batch selection for training deep neural networks. In *IJCAI*, pages 2677–2683, 2019.

[Karp, 1972] Richard M Karp. Reducibility among combinatorial problems. In *Complexity of computer computations*, pages 85–103. Springer, 1972.

[Kempe *et al.*, 2003] David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. In *KDD*, pages 137–146. ACM, 2003.

[Khuller *et al.*, 1999] Samir Khuller, Anna Moss, and Joseph Seffi Naor. The budgeted maximum coverage problem. *Information processing letters*, 70(1):39–45, 1999.

[Krause and Guestrin, 2005] Andreas Krause and Carlos Guestrin. *A note on the budgeted maximization of submodular functions*. Carnegie Mellon University. Center for Automated Learning and Discovery, 2005.

[Leskovec *et al.*, 2007] Jure Leskovec, Andreas Krause, Carlos Guestrin, Christos Faloutsos, Jeanne VanBriesen, and Natalie Glance. Cost-effective outbreak detection in networks. In *KDD*, pages 420–429, 2007.

[McAuley *et al.*, 2015] Julian McAuley, Rahul Pandey, and Jure Leskovec. Inferring networks of substitutable and complementary products. In *KDD*, pages 785–794, 2015.

[Mitrovic *et al.*, 2018] Marko Mitrovic, Moran Feldman, Andreas Krause, and Amin Karbasi. Submodularity on hypergraphs: From sets to sequences. In *AISTATS*, pages 1177–1184, 2018.

[Mitrovic *et al.*, 2019] Marko Mitrovic, Ehsan Kazemi, Moran Feldman, Andreas Krause, and Amin Karbasi. Adaptive sequence submodularity. In *NeurIPS*, pages 5352–5363, 2019.

[Nemhauser *et al.*, 1978] George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. An analysis of approximations for maximizing submodular set functions. *Mathematical programming*, 14(1):265–294, 1978.

[Parameswaran *et al.*, 2011] Aditya Parameswaran, Petros Venetis, and Hector Garcia-Molina. Recommendation systems with complex constraints: A course recommendation perspective. *ACM Transactions on Information Systems*, 29(4):1–33, 2011.

[Qian *et al.*, 2015] Chao Qian, Yang Yu, and Zhi-Hua Zhou. Subset selection by pareto optimization. In *NeurIPS*, pages 1774–1782, 2015.

[Qian *et al.*, 2017] Chao Qian, Jing-Cheng Shi, Yang Yu, and Ke Tang. On subset selection with general cost constraints. In *IJCAI*, volume 17, pages 2613–2619, 2017.

[Qian *et al.*, 2018] Chao Qian, Chao Feng, and Ke Tang. Sequence selection by pareto optimization. In *IJCAI*, pages 1485–1491, 2018.

[Raut *et al.*, 2021] Prasanna Raut, Omid Sadeghi, and Maryam Fazel. Online dr-submodular maximization: Minimizing regret and constraint violation. In *AAAI*, pages 9395–9402, 2021.

[Sallam *et al.*, 2020] Gamal Sallam, Zizhan Zheng, Jie Wu, and Bo Ji. Robust sequence submodular maximization. In *NeurIPS*, 2020.

[Shahaf *et al.*, 2012] Dafna Shahaf, Carlos Guestrin, and Eric Horvitz. Metro maps of science. In *KDD*, pages 1122–1130, 2012.

[Sviridenko, 2004] Maxim Sviridenko. A note on maximizing a submodular set function subject to a knapsack constraint. *Operations Research Letters*, 32(1):41–43, 2004.

[Tschiatschek *et al.*, 2017] Sebastian Tschiatschek, Adish Singla, and Andreas Krause. Selecting sequences of items via submodular maximization. In *AAAI*, pages 2667–2673, 2017.

[Zhang *et al.*, 2015] Zhenliang Zhang, Edwin KP Chong, Ali Pezeshki, and William Moran. String submodular functions with curvature constraints. *IEEE Transactions on Automatic Control*, 61(3):601–616, 2015.

# 8 Appendix

## 8.1 Proof of Theorem 1

**Proof**. We prove the NP-hardness of the problem by reducing the well-known *Knapsack* problem [Cormen *et al.*, 2009]. Given a set of $n$ items $V = (v_1, v_2, \cdots, v_n)$ with item weights $(w_1, w_2, \cdots, w_n)$, item value $(x_1, x_2, \cdots, x_n)$, and a maximum weight capacity $W$, the knapsack problem aims to find a collection of items $s'$ whose total weights is less than or equal to $W$ and the total value is maximal.

Given an instance of the *Knapsack* problem $\varphi$, we can construct a BSSM instance $\omega$ as follows: we build a graph $G$ that only contains $n$ nodes and their self-cycles, in which each node corresponds to an item in $V$, and set $\Delta = W$, $f(s) = h(E(s)) = \sum_{v_i \in E(s)} x_i$, and $C(s) = \sum_{v_i \in s} w_i$.

Given this mapping, if $s'$ is the optimal collection of $\varphi$, any sequence of $s'$ is the optimal sequence of $\omega$, and vice versa. As the *Knapsack* problem has been proved to be NP-hard, the BSSM problem is NP-hard as well. ■

## 8.2 Proof of Theorem 2

To prove Theorem 2, we first present a definition and some important lemmas. Let $s^*$ denote an optimal sequence.

**Definition 3. Function** $g$**.** *In an auxiliary set function* $g : 2^E \to \mathbb{R}$*, for any edge set* $Y \subseteq E$*,* $g(Y) = h(E(RE(Y)))$*.*

Note that, Definition 3 indicates that $g(Y) = f(RE(Y))$.

**Lemma 1.** *When* $G$ *is a DAG (not counting self-cycles), for any edge set* $Y \subseteq E$*, let* $e^* = argmax_{e_l \in E \setminus Y} \frac{f(RE(Y \cup \{e_l\})) - f(RE(Y))}{C(Y \cup \{e_l\}) - C(Y)}$*, it holds that* $f(RE(Y \cup \{e^*\})) - f(RE(Y)) \geq \frac{C(Y \cup \{e^*\}) - C(Y)}{\beta \Delta}(f(s^*) - f(RE(Y)))$*.*

**Proof**. We denote the edge set containing all possible item pairs as $\mathcal{E} = \{< v_i, v_j > | v_i \in V, v_j \in V, i \neq j\}$, and use $\tilde{e}$ to represent the pair in $\mathcal{E}$ in this article. Next, we first prove that, when $G$ is a DAG (not counting self-cycles), for any pair set $Y \subseteq \mathcal{E}$, let $\tilde{e}^* = argmax_{\tilde{e}_l \in \mathcal{E} \setminus Y} \frac{f(RE(Y \cup \{\tilde{e}_l\})) - f(RE(Y))}{C(Y \cup \{\tilde{e}_l\}) - C(Y)}$, it holds that $f(RE(Y \cup \{\tilde{e}^*\})) - f(RE(Y)) \geq \frac{C(Y \cup \{\tilde{e}^*\}) - C(Y)}{\beta \Delta}(f(s^*) - f(RE(Y)))$.

We construct a node set $Y^*$ by setting edges from the items with the minimal cost to other items in $V(s^*)$. Assume that $Y^* \setminus Y = \{\tilde{e}_1, \tilde{e}_2, \cdots, \tilde{e}_m\}$, and the edges are in ascending order of their costs , it is obvious that $V(Y) = V(s^*)$, $m \leq |Y^*| \leq |V(s^*)| - 1$ and $\sum_{i=1}^{m} C(\{\tilde{e}_i\}) \leq 2\Delta$. According to Definition 3, $f(s^*) = g(Y^*)$, $f(RE(Y \cup \{\tilde{e}^*\})) = g(Y \cup \{\tilde{e}^*\})$ and $f(RE(Y)) = g(Y)$. For simplify, we use function $g$ in the following proof. We can get

$$g(Y^*) - g(Y) \leq g(Y^* \cup Y) - g(Y) = g(Y^* \setminus Y \cup Y) - g(Y)$$
$$= \sum_{i=1}^{m} (g(\{\tilde{e}_1, \tilde{e}_2, \cdots, \tilde{e}_i\} \cup Y) - g(\{\tilde{e}_1, \tilde{e}_2, \cdots, \tilde{e}_{i-1}\} \cup Y))$$

Let $z_k = g(\{\tilde{e}_1, \tilde{e}_2, \cdots, \tilde{e}_k\} \cup Y) - g(\{\tilde{e}_1, \tilde{e}_2, \cdots, \tilde{e}_{k-1}\} \cup Y)$. Next, we prove that when $z_k > 0$,

$$\frac{z_k}{C(Y \cup \{\tilde{e}_k\}) - c(Y)} \leq 2(k+1) \frac{g(Y \cup \{\tilde{e}^*\}) - g(Y)}{C(Y \cup \{\tilde{e}^*\}) - C(Y)} \quad (3)$$

It is easy to see when $k = 1$, Inequality 3 holds according to the definition of $\tilde{e}^*$.
When $k > 1$, let $Y_{k-1} = \{\tilde{e}_1, \tilde{e}_2, \cdots, \tilde{e}_{k-1}\} \cup Y$, based on the definition of $Y^*$, $|V(Y_{k-1} \cup \{\tilde{e}_k\})| - |V(Y_{k-1})| \leq 1$. As $z_k > 0$, $|V(Y_{k-1} \cup \{\tilde{e}_k\})| - |V(Y_{k-1})| = 1$. Thus, we can let $V(Y_{k-1} \cup \{\tilde{e}_k\}) \setminus V(Y_{k-1}) = \{v_k\}$.

If $G = (V, E)$ is a DAG (excluding self-cycles), for any two distinct items $v_i$ and $v_j$, if there is an edge $e_l = (v_i, v_j) \in E$, $e_{l+1} = (v_j, v_i) \notin E$, and vice versa. Then, for any edge $e_l = (v_i, v_j) \in E$, $v_i \in V(Y)$ and $v_j \in V(Y)$, if and only if $v_i$ precedes $v_j$ in a sequence $s$, $e_l \in E(s)$. Thus, we can always find a sequence contains all edges existing in all possible sequences of $V(Y)$, i.e., $E(RE(Y)) = \{(v_i, v_i) | v_i \in V(Y), (v_i, v_i) \in E\} \cup \{(v_i, v_j) | v_i, v_j \in V(Y), (v_i, v_j) \in E\}$. We have

$$g(Y_{k-1} \cup \{\tilde{e}_k\}) = h(E(RE(Y_{k-1})) \cup E(RE(\{\tilde{e}_k\}))$$
$$\cup \{(v_i, v_j) | (v_i \in V(Y_{k-1}), v_j \in \{v_k\} \vee v_i \in \{v_k\}, v_j \in V(Y)),$$
$$(v_i, v_j) \in E\})$$

We break down the third part in the above function $h$ as follows. Let

$$V_{di} = V(Y_{k-1}) \setminus V(Y),$$
$$E_{k-1} = \{e_1, e_2, \cdots, e_r\} = \{(v_i, v_j) | (v_i \in V_{di}, v_j \in \{v_k\}$$
$$\vee v_i \in \{v_k\}, v_j \in V_{di}), (v_i, v_j) \in E\},$$
$$E_{sa} = \{(v_i, v_j) | (v_i \in V(Y), v_j \in \{v_k\} \vee v_i \in \{v_k\}, v_j \in V(Y)),$$
$$(v_i, v_j) \in E\},$$
$$H_t = h(E(RE(Y_{k-1})) \cup E(RE(\{\tilde{e}_k\})) \cup E_{sa} \cup \{e_1, e_2, \cdots, e_t\})$$
$$- h(E(RE(Y_{k-1})) \cup E(RE(\{\tilde{e}_k\})) \cup E_{sa} \cup \{e_1, e_2, \cdots, e_{t-1}\})$$

As $h$ is submodualr and $E(RE(Y)) \subseteq E(RE(Y_{k-1}))$, we have

$$g(Y_{k-1} \cup \{\tilde{e}_k\})$$
$$\leq h(E(RE(Y_{k-1})) \cup E(RE(\{\tilde{e}_k\})) \cup E_{sa}) + \sum_{t=1}^{r} H_t$$
$$\leq h(E(RE(Y_{k-1})) \cup E(RE(\{\tilde{e}_k\})) \cup E_{sa}) +$$
$$\sum_{t=1}^{r} (h(E(RE(Y)) \cup \{e_t\}) - h(E(RE(Y))))$$

Considering that the edges in $\{\tilde{e}_1, \tilde{e}_2, \cdots, \tilde{e}_m\}$ is in ascending order of their costs, $2c_{v_k} \geq C(e_t)$ for each edge $e_t \in$

$E_{k-1}$, and $2c_{v_k} \geq C(\tilde{e}_k)$. Thus, for $z_k > 0$, we can get

$$\frac{g(Y_{k-1}\cup\{\tilde{e}_k\})-g(Y_{k-1})}{C(Y_{k-1}\cup\{\tilde{e}_k\})-C(Y_{k-1})} = \frac{z_k}{c_{v_k}}$$
$$\leq \frac{h(E(RE(Y_{k-1}))\cup E(RE(\{\tilde{e}_k\}))\cup E_{sa})-h(E(RE(Y_{k-1})))}{c_{v_k}}+$$
$$\frac{\sum_{t=1}^{r} h(E(RE(Y))\cup\{e_t\})-h(E(RE(Y)))}{c_{v_k}}$$
$$\leq \frac{2h(E(RE(Y))\cup E(RE(\{\tilde{e}_k\}))\cup E_{sa})-h(E(RE(Y)))}{C(\{\tilde{e}_k\})}+$$
$$2\sum_{t=1}^{r}\frac{h(E(RE(Y))\cup\{e_t\})-h(E(RE(Y)))}{C(\{e_t\})}$$
$$\leq \frac{2h(E(RE(Y))\cup E(RE(\{\tilde{e}_k\}))\cup E_{sa})-h(E(RE(Y)))}{C(Y\cup\{\tilde{e}_k\})-C(Y)}+$$
$$2\sum_{t=1}^{r}\frac{h(E(RE(Y))\cup\{e_t\})-h(E(RE(Y)))}{C(Y\cup\{e_t\})-C(Y)}$$
$$\leq 2(r+1)\frac{g(Y\cup\{\tilde{e}^*\})-g(Y)}{C(Y\cup\{\tilde{e}^*\})-C(Y)}$$

where the last inequality holds due to the definition of $\tilde{e}^*$. According to the definition of $Y^*$, $|V_{di}|=k$, $r\leq k$, Inequality 3 holds when $k>1$.
And $k\leq m\leq \lfloor\frac{\Delta}{c_{min}}\rfloor-1$, we have

$$g(Y^*)-g(Y)\leq\sum_{k=1}^{m} z_k$$
$$\leq\sum_{k=1}^{m} 2\lfloor\frac{\Delta}{c_{min}}\rfloor(C(Y_{k-1}\cup\{\tilde{e}_k\})-C(Y_{k-1}))\frac{g(Y\cup\{\tilde{e}^*\})-g(Y)}{C(Y\cup\{\tilde{e}^*\})-c(Y)}$$
$$\leq\sum_{k=1}^{m} 2\lfloor\frac{\Delta}{c_{min}}\rfloor C(\{\tilde{e}_k\})\frac{g(Y\cup\{\tilde{e}^*\})-g(Y)}{C(Y\cup\{\tilde{e}^*\})-c(Y)}$$
$$\leq 4\Delta\lfloor\frac{\Delta}{c_{min}}\rfloor\frac{g(Y\cup\{\tilde{e}^*\})-g(Y)}{C(Y\cup\{\tilde{e}^*\})-c(Y)}$$

Thus, the conclution (i.e., $f(RE(Y\cup\{\tilde{e}^*\}))-f(RE(Y))\geq\frac{C(Y\cup\{\tilde{e}^*\})-C(Y)}{\beta\Delta}(f(s^*)-f(RE(Y))))$ is hold for $\tilde{e}^*$.

At the last step, we prove that $e^*$ is a special case of $\tilde{e}^*$ in two cases. Assume that $\tilde{e}^*=<v_i,v_j>$.

In case 1, there exist an edge between $v_i$ and $v_j$, $e^*=(v_i,v_j)$ or $(v_j,v_i)$, $e^*$ and $\tilde{e}^*$ are equivalent.

In case 2, there is no any edge between $v_i$ and $v_j$, it means that $v_i$ and $v_j$ are two independent items. $e^*$ exists if and only if there exists at least one edge between $v_i$(or $v_j$) and one item in $V(Y)$. When $v_i$(or $v_j$) has no edges connecting with items in $V(Y)$, $\frac{f(RE(Y\cup\{\tilde{e}^*\}))-f(RE(Y))}{C(Y\cup\{\tilde{e}^*\})-c(Y)}=\frac{f(RE(V(Y)\cup\{v_i\}))-f(RE(Y))}{C(V(Y)\cup\{v_i\})-c(Y)}$ (or replacing $v_i$ by $v_j$), $e^*$ and $\tilde{e}^*$ are equivalent. When both $v_i$ and $v_j$ has edges connecting with items in $V(Y)$, $\frac{f(RE(Y\cup\{\tilde{e}^*\}))-f(RE(Y))}{C(Y\cup\{\tilde{e}^*\})-c(Y)}=\frac{f(RE(V(Y)\cup\{v_i\}))-f(RE(Y))}{C(V(Y)\cup\{v_i\})-c(Y)}=\frac{f(RE(V(Y)\cup\{v_j\}))-f(RE(Y))}{C(V(Y)\cup\{v_j\})-c(Y)}$, $e^*$ is a special case of $\tilde{e}^*$.

Therefore, we can conclude the proof. ■

Suppose that GBM starts with an empty edge set $E_0^{se}=\emptyset$, and then adds an edge into the edge set in each iteration using the greedy rule. It generates intermediate edge sets $E_1^{se},E_2^{se},\cdots,E_l^{se}$ in sequence, and stops with $E_{l+1}^{se}$ which violates the budget constraint.

**Lemma 2.** *When $G$ is a DAG (not counting self-cycles), for $i=1,2,\cdots,l+1$, it holds that $f(RE(E_i^{se}))\geq(\frac{1}{\beta})^i[1-\prod_{j=1}^{i}(1-\frac{C(E_j^{se})-C(E_{j-1}^{se})}{\Delta})]f(s^*)$.*

**Proof**. We prove this lemma by Mathematical induction. For $i=1$, from Lemma 1, it is straightforward that $f(RE(E_1^{se}))\geq\frac{C(E_1^{se})-C(E_0^{se})}{\beta\Delta}f(s^*)$, it shows that the lemma holds. Assume that the lemma also holds when $i=t$, for $i=t+1$, we can get

$$f(RE(E_{t+1}^{se}))=f(RE(E_t^{se}))+(f(RE(E_{t+1}^{se}))-f(RE(E_t^{se})))$$
$$\geq f(RE(E_t^{se}))+\frac{C(E_{t+1}^{se})-C(E_t^{se})}{\beta\Delta}(f(s^*)-f(RE(E_t^{se})))$$
$$\geq(1-\frac{C(E_{t+1}^{se})-C(E_t^{se})}{\beta\Delta})f(RE(E_t^{se}))$$
$$+\frac{C(E_{t+1}^{se})-C(E_t^{se})}{\beta\Delta}f(s^*)$$
$$\geq(\frac{1}{\beta})^{t+1}\big(1-\prod_{j=1}^{t}(1-\frac{C(E_{t+1}^{se})-C(E_t^{se})}{\Delta})\big)f(s^*)+$$
$$\frac{1}{\beta}\frac{C(E_{t+1}^{se})-C(E_t^{se})}{\Delta}f(s^*)$$
$$\geq(\frac{1}{\beta})^{t+1}\big(1-\prod_{j=1}^{t+1}(1-\frac{C(E_{t+1}^{se})-C(E_t^{se})}{\Delta})\big)f(s^*)$$

where the first inequality is from Lemma 1. Thus, the lemma holds when $i=t+1$. ■

**Lemma 3.** *When $G$ is a DAG (not counting self-cycles), for the edge $e'=argmax_{e_l\in E,C(e_l)\leq\Delta}f(RE(e_l))$, an edge set $E_{fea}$ with $C(E_{fea})\leq\Delta$ and an edge $e_l\in E$, it holds that $f(RE(e'))\geq\frac{1}{2\lfloor\frac{\Delta}{c_{min}}\rfloor+1}f(RE(E_{fea}\cup\{e_l\}))-f(RE(E_{fea}))$.*

**Proof**. From the proof of Lemma 1, we can get

$$g(E_{fea}\cup\{e_l\})-g(E_{fea})\leq h(E(REO(\{e_l\}))\cup E(REO(E_{fea})))$$
$$-h(E(REO(E_{fea})))+\sum_{e_i\in E_2}h(\{e_i\})$$
$$\leq h(E(REO(\{e'\})))+\sum_{e_i\in E_2}h(E(REO(\{e'\})))$$
$$\leq(2\lfloor\frac{\Delta}{c_{min}}\rfloor+1)g(\{e'\})$$

where $E_2=\{(v_i,v_j)|(v_i\in V(E_{fea}),v_j\in V(\{e_l\})\vee v_i\in V(\{e_l\}),v_j\in V(E_{fea})),(v_i,v_j)\in E\}$, the first inequality holds due to the submodularity of $h$, and the second inequality holds as the definition of $e'$, $|V(E_{fea})|\leq\lfloor\frac{\Delta}{c_{min}}\rfloor$, $|E_2|\leq 2|V(E_{fea})|$. ■

**Next, based on the above lemmas, we prove Theorem 2 as below.**

**Proof**. We denote the sequence returned by GBM as $s^{gbm}$, $s^{gbm}=argmax_{s^i\in\{s^1,s^2\}}f(s^i)$, $f(s^{gbm})\leq f(s^*)$.
We first consider the fact that for $a_1,a_2,\cdots,a_n\in\mathbb{R}^+$ and

$\sum_{i=1}^{n} a_i = A$, $1 - \prod_{i=1}^{n}(1 - \frac{a_i}{A})$ obtains its minimum value when $a_1 = a_2 = \cdots = a_n = \frac{A}{n}$. Following it, we can get

$$f(RE(E_{t+1}^{se})) \geq (\frac{1}{\beta})^{l+1}\big(1 - \prod_{j=1}^{l+1}(1 - \frac{C(E_j^{se}) - C(E_{j-1}^{se})}{\Delta})\big)f(s^*)$$
$$\geq (\frac{1}{\beta})^{l+1}\big(1 - \prod_{j=1}^{l+1}(1 - \frac{C(E_j^{se}) - C(E_{j-1}^{se})}{C(E_{l+1}^{se})})\big)f(s^*)$$
$$\geq (\frac{1}{\beta})^{l+1}\big(1 - (1 - \frac{1}{l+1})^{l+1}\big)f(s^*) \geq (\frac{1}{\beta})^{l+1}(1 - e^{-1})f(s^*)$$

where the first inequality is from Lemma 2 and the second inequality due to $C(E_{l+1}^{se}) > \Delta$ in the definition of $E_{l+1}^{se}$. Let $E_{l+1}^{se} = E_l^{se} \cup e_{l+1}^*$, from Lemma 3, we have

$$(\frac{1}{\beta})^{l+1}(1 - e^{-1})f(s^*) \leq f(RE(E_{t+1}^{se}))$$
$$\leq f(RE(E_t^{se})) + (f(RE(E_t^{se} \cup e_{l+1}^*)) - f(RE(E_t^{se})))$$
$$\leq f(RE(E_t^{se})) + (2\lfloor\frac{\Delta}{c_{min}}\rfloor + 1)f(RE(e'))$$

where $e'$ is defined in Algorithm 1, and $s^2 = RE(\{e'\})$. Thus, we can get

$$f(s^{gbm}) \geq max\{f(s^1), f(s^2)\} \geq max\{f(RE(E_l^{se})), f(RE(e'))\}$$
$$\geq \frac{1}{4\lfloor\frac{\Delta}{c_{min}}\rfloor + 2}(\frac{1}{\beta})^{l+1}(1 - e^{-1})f(s^*)$$

Meanwhile, $|s^*| \leq \lfloor\frac{\Delta}{c_{min}}\rfloor$, and GBM inserts two items into $E^{se}$ in the first iteration, and then adds at least one item into $E^{se}$ in each of the following iterations. Then, GBM has at most $l = \lfloor\frac{\Delta}{c_{min}}\rfloor - 1$ iterations, we have $f(s^{gbm}) \geq \frac{1}{\beta+2}(\frac{1}{\beta})^{\lfloor\frac{\Delta}{c_{min}}\rfloor}(1 - e^{-1})f(s^*)$. Therefore, we can obtain the conclusion by combining the above two cases . ■

## 8.3 Proof of Theorem 3

**Proof**. We conduct the proof by analyzing the increase of a quantity $J_{max}$, which is denoted as the the maximum value of $j \geq 0$ such that in the archived item set $P$ of POBM, there exist an item set $\boldsymbol{p}$ with $C(\boldsymbol{p}) \leq j$ and $f(RE(\boldsymbol{p})) \geq (\frac{1}{\beta})^k\big(1 - (1 - \frac{j}{k\Delta})^k\big)f(s^*)$ for some $k$.
Based on the proof of Lemma 2, we can always get $J_{max}$. Next, we analyze the increase of $J_{max}$.

The initial value of $J_{max}$ is 0, as POBM begins with the empty item set. Suppose that currently $J_{max} = i < \Delta$, and let $\boldsymbol{p}$ be a corresponding item set with $C(\boldsymbol{p}) \leq i$, and $f(RE(\boldsymbol{p})) \geq (\frac{1}{\beta})^k\big(1 - (1 - \frac{i}{k\Delta})^k\big)f(s^*)$ for some $k$.

We first illustrate that $J_{max}$ cannot decrease. If $\boldsymbol{p}$ is maintained in $P$, $J_{max}$ obviously will not drop. If $\boldsymbol{p}$ is pruned from $P$, the newly added item set $\boldsymbol{p'}$ must weakly dominate $\boldsymbol{p}$ (line 6 of Algorithm 2), thus $f_1(\boldsymbol{p'}) \geq f_1(\boldsymbol{p})$ and $C(\boldsymbol{p'}) \leq C(\boldsymbol{p}) = i$. According to the definition of $J_{max}$, this case do not change $J_{max}$.

We next analyze the increase of $J_{max}$. We recall Lemma 1, and get the following lemma straightforwardly. When $G$ is a DAG (not counting self-cycles), for any item set $\boldsymbol{p} \subseteq V$, there always exists one item $v_i \in V \setminus \boldsymbol{p}$ or two distinct items $v_i, v_j \in V \setminus \boldsymbol{p}$, such that adding $v_i$ or $v_i, v_j$ into $\boldsymbol{p}$ can generate a new item set $\boldsymbol{p'}$ which satisfies that $f(RE(\boldsymbol{p'})) - f(RE(\boldsymbol{p})) \geq \frac{C(\boldsymbol{p'})-C(\boldsymbol{p})}{\beta\Delta}(f(s^*) - f(RE(\boldsymbol{p})))$
Based on above two inequalities of $\boldsymbol{p}$, $\beta \geq 4$ when $|s^*| \geq 1$, $C(\boldsymbol{p'}) \leq 2\Delta$, we have

$$f(RE(\boldsymbol{p'})) \geq (1 - \frac{C(\boldsymbol{p'}) - C(\boldsymbol{p})}{\beta\Delta})f(RE(\boldsymbol{p})) + \frac{C(\boldsymbol{p'}) - C(\boldsymbol{p})}{\beta\Delta}f(s^*)$$
$$\geq (\frac{1}{\beta})^{k+1}\big(1 - (1 - \frac{i}{k\Delta})^k(1 - \frac{C(\boldsymbol{p'}) - C(\boldsymbol{p})}{\Delta})\big)f(s^*)$$
$$\geq (\frac{1}{\beta})^{k+1}\big(1 - (1 - \frac{i + C(\boldsymbol{p'}) - C(\boldsymbol{p})}{(k+1)\Delta})^{k+1}\big)f(s^*)$$

where the last inequality holds by using the AM-GM inequality. As the function $f(RE())$ is monotone and $C(\boldsymbol{p'}) > C(\boldsymbol{p})$, $f(\boldsymbol{p'}) \geq f(\boldsymbol{p})$, $\boldsymbol{p'}$ must be inserted into $P$, it leads to an increase of $J_{max}$.
After POBM obtains $\boldsymbol{p}$ with the probability at least $\frac{1}{P_{max}}$, there are 2 cases for generating $\boldsymbol{p'}$.
Case1: $|\boldsymbol{p'}| = |\boldsymbol{p}| + 1$, in which POBM generates $\boldsymbol{p'}$ with the probability at least $\frac{1}{n}(1 - \frac{1}{n})^{n-1}$ by flipping a specific bit of $\boldsymbol{p}$ and unchanging other bits. Hence, $J_{max}$ can increase by at least $c_{min}$ in one iteration with the probability $\frac{1}{enP_{max}}$.
Case2: $|\boldsymbol{p'}| = |\boldsymbol{p}| + 2$, where POBM generates $\boldsymbol{p'}$ with the probability $(\frac{1}{n})^2(1 - \frac{1}{n})^{n-2}$ $(\geq \frac{1}{n}\big(\frac{1}{n}(1 - \frac{1}{n})^{n-1}\big) \geq \frac{1}{en^2})$ by flipping two specific bits of $\boldsymbol{p}$ and maintaining other bits unchanged. Thus, $J_{max}$ can grow by $2c_{min}$ in one iteration with the probability at least $\frac{1}{en^2P_{max}}$.

Let $\boldsymbol{p_2^*} = argmax_{\boldsymbol{p_2^i} \subseteq V \setminus \boldsymbol{p}, |\boldsymbol{p_2^i}| \leq 2}\frac{f(RE(\boldsymbol{p} \cup \boldsymbol{p_2^i})) - f(RE(\boldsymbol{p}))}{C(\boldsymbol{p} \cup \boldsymbol{p_2^i}) - C(\boldsymbol{p})}$, and $J_{max} + C(\boldsymbol{p} \cup \boldsymbol{p_2^*}) - C(\boldsymbol{p}) \geq \Delta$.

In terms of the worst complexity of POBM, $J_{max}$ always grows in case 2. It is easy to see that after $\lfloor\frac{\Delta}{2c_{min}}\rfloor en^2P_{max}$ iterations, POBM is expected to achieve $\boldsymbol{p} \cup \boldsymbol{p_2^*}$.

In terms of the solution quality, $J_{max} + C(\boldsymbol{p} \cup \boldsymbol{p_2^*}) - C(\boldsymbol{p}) \geq \Delta$ indicates that there is one item set $\boldsymbol{p} \in P$ satisfying that $C(\boldsymbol{p}) \leq J_{max} < \Delta$ and

$$f(RE(\boldsymbol{p} \cup \boldsymbol{p_2^*}))$$
$$\geq (\frac{1}{\beta})^{k+1}\big(1 - (1 - \frac{J_{max} + C(\boldsymbol{p} \cup \boldsymbol{p_2^i}) - C(\boldsymbol{p})}{\Delta(k+1)})^{k+1}\big)f(s^*)$$
$$\geq (\frac{1}{\beta})^{k+1}\big(1 - (1 - \frac{\Delta}{\Delta(k+1)})^{k+1}\big)f(s^*) \geq (\frac{1}{\beta})^{k+1}(1 - e^{-1})f(s^*)$$

Let $\boldsymbol{p_2'} = argmax_{\boldsymbol{p_2^i} \subseteq V, |\boldsymbol{p_2^i}| \leq 2, C(\boldsymbol{p_2^i}) \leq \Delta}f(RE(\boldsymbol{p_2^i}))$, considering Lemma 3, we have

$$f(RE(\boldsymbol{p} \cup \boldsymbol{p_2^*})) = f(RE(\boldsymbol{p})) + (f(RE(\boldsymbol{p} \cup \boldsymbol{p_2^*})) - f(RE(\boldsymbol{p})))$$
$$\leq f(RE(\boldsymbol{p})) + (2\lfloor\frac{\Delta}{c_{min}}\rfloor + 1)f(RE(\boldsymbol{p_2'}))$$

And then, we can get $max\{f(RE(\boldsymbol{p})), f(RE(\boldsymbol{p_2'}))\} \geq \frac{1}{\beta+2}(\frac{1}{\beta})^{k+1}(1 - e^{-1})f(s^*)$. As $J_{max} < \Delta$, $k + 1 \leq \lfloor\frac{\Delta}{c_{min}}\rfloor$. Meanwhile, POBM can generate $\boldsymbol{p_2'}$ by obtaining $\{0\}^n$ with probability $\frac{1}{enP_{max}}$.

Therefore, POBM with $\mathbb{E}(T) \leq \lfloor\frac{\Delta}{2c_{min}}\rfloor en^2P_{max}$ can find a sequence $s$ such that $f(s) \geq max\{f(RE(\boldsymbol{p})), f(RE(\boldsymbol{p_2'}))\} \geq \frac{1}{\beta+2}(\frac{1}{\beta})^{\lfloor\frac{\Delta}{c_{min}}\rfloor}(1 - e^{-1})f(s^*)$. ■

### 8.4 Proof of Theorem 4

**Proof**. Let $p^* = V(s^*)$. As $|p^*| \leq \lfloor \frac{\Delta}{c_{min}} \rfloor$, at any iteration, POBM picks $\{0\}^n$ with the probability $\frac{1}{P_{max}}$, and then generates $p^*$ with the probability $(\frac{1}{n})^{\lfloor \frac{\Delta}{c_{min}} \rfloor}(1-\frac{1}{n})^{n-\lfloor \frac{\Delta}{c_{min}} \rfloor}$ through flipping specific $\lfloor \frac{\Delta}{c_{min}} \rfloor$ bits of $\{0\}^n$ and unchanging other bits. We have $(\frac{1}{n})^{\lfloor \frac{\Delta}{c_{min}} \rfloor}(1-\frac{1}{n})^{n-\lfloor \frac{\Delta}{c_{min}} \rfloor} \geq (\frac{1}{n})^{\lfloor \frac{\Delta}{c_{min}} \rfloor -1}(\frac{1}{n}(1-\frac{1}{n})^{n-1}) \geq (\frac{1}{n})^{\lfloor \frac{\Delta}{c_{min}} \rfloor -1}\frac{1}{en}$, and $RE(p^*) = s^*$ when $G = (V, E)$ is a DAG (not counting self-cycles), Therefore, POBM with $\mathbb{E}(T) \leq en^{\lfloor \frac{\Delta}{c_{min}} \rfloor} P_{max}$ finds $s^*$. ■

### 8.5 Proof of Theorem 5

**Proof**. The proof is similar to that of Theorem 3. But after POSEQSEL obtains $\boldsymbol{p}$ with the probability $\frac{1}{P_{max}}$, the 2 cases for generating $\boldsymbol{p'}$ are different.

Case1: $|\boldsymbol{p'}| = |\boldsymbol{p}| + 1$, in this case, POSEQSEL generates $\boldsymbol{p'}$ with the probability at least $\frac{1}{2} * \frac{1}{e} * \frac{1}{n}$ by only flipping a specific bit of $\boldsymbol{p}$. Thus, $J_{max}$ can grow by at least $c_{min}$ in one iteration with the probability $\frac{1}{2enP_{max}}$.

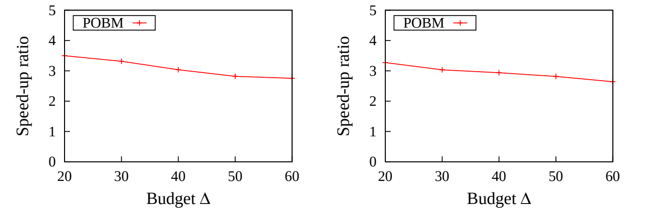
Case2: $|\boldsymbol{p'}| = |\boldsymbol{p}| + 2$, in which POSEQSEL generates $\boldsymbol{p'}$ with the probability at least $\frac{1}{2^2} * \frac{1}{2e} * \frac{2}{(n-|\boldsymbol{p}|)(n-|\boldsymbol{p}|-1)}$ ($\geq \frac{1}{4en^2}$ ) by flipping two specific bits of $\boldsymbol{p}$ and unchanging other bits. Thus, $J_{max}$ can grow by $2c_{min}$ in one iteration with the probability at least $\frac{1}{4en^2P_{max}}$.

After that, following the proof of Theorem 3, we can achieve that POSEQSEL with $\mathbb{E}(T) \leq \lfloor \frac{2\Delta}{c_{min}} \rfloor en^2 P_{max}$ can obtain a sequence $s$ such that $f(s) \geq \frac{1}{\beta+2}(\frac{1}{\beta})^{\lfloor \frac{\Delta}{c_{min}} \rfloor}(1-e^{-1})f(s^*)$. ■

### 8.6 The Details of Preprocessing Two Real-world Datasets in Section 6.2

We use two real-world datasets, one is the Movielens 1M (MOV) dataset [Harper and Konstan, 2015; Tschiatschek *et al.*, 2017] and another one is XuetangX (XTX) dataset [Feng *et al.*, 2019]. MOV dataset contains $1,000,209$ time-stamped ratings made by $6,040$ users for $3,706$ different movies in MovieLens platform, and XTX dataset has the tracking log files that records the $772,887$ users' learning behavior over $1,629$ courses in XuetangX platform from August 2015 to August 2017. They will be used to do a movie recommendation task and a course recommendation task respectively. In order for our data to be representative of the general population, referring to the work [Mitrovic *et al.*, 2018], for the MOV dataset, we remove all users who have rated fewer than 100 movies or more than 500 movies. We also remove all movies with fewer than 20 reviews or more than 2,000 reviews. This leaves us with $412,222$ ratings made by $2,549$ users for 882 different movies. For the XTX dataset, we first remove all users who have enrolled in fewer than 3 courses or more than 20 courses, and then remove all courses which were enrolled by fewer than 200 users. After this processing, we obtain the tracking log files made by $238,834$ users' over 956 courses.

Following the work [Tschiatschek *et al.*, 2017], we use the utility function $h(E(s)) = \sum_{v_j \in V(E(s))}[1 - \prod_{(v_i,v_j)\in E(s)}(1-p_{j|i})]$, swhere $p_{j|i}$ associated on the edge $(v_i, v_j)$ is the conditional probability that a user rates movie (or enrolls course) $v_j$ given that she has rated movie (or enrolled course) $v_i$ before, and $p_{i|i}$ associated on self-cycles is the item frequency $p_i$. We construct the graph $G = (V, E)$ referring to the work [Tschiatschek *et al.*, 2017]. For each user, there is a sequence $s = (s_1, s_2, \cdots, s_k)$ that she rated movies (or enrolled courses). For two distinct movies (or movies) $s_i$ and $s_j$ in $s$, if the distance $d_s = j - i, i < j$ is larger, the dependency between $s_i$ and $s_j$ is less. So that we only consider dependencies between the last $z$ items in $s$, and we can get the number of users who rated movie (or enrolled course) $s_j$ after rating movie (or enrolling course) $s_i$ (denoted as $n(i,j)$). For each movie (or course) $s_i$, we only consider the top-$z$ movies (or courses) as its follow-up movies (or courses) by descending order of $\{n(i,j)|j \in [1,n]\}$ (i.e., $n(i,j) = 0$ for other movies or courses), and compute the conditional probability on the edge $(v_i, v_j)$ as $p_{j|i} = \frac{n(i,j)}{\sum_{t\in[1,n]} n(i,t)}$. We also obtain the cost $c_{v_i}$ for each movie by crawling the purchase price from Amazon's website, and extract the costs of courses from the XTX dataset directly. We use $z = 2$ as a default setting, as the experimental results of $z = 2$ are similar to these of $z = 3, 4, 5$.




(a) Modular $h$ (b) Submodular $h$

Figure 7: The speed-up ratios of our optimizations on POBM over modular and submodular utility functions.

### 8.7 The Acceleration Effect of Our Optimization on POBM on Synthetic Datasets

We investigate the acceleration effect of our optimizations on POBM on synthetic datasets. Fig. 7 shows the speed-up ratios of our optimizations on POBM. It demonstrates that our optimizations can speed up POBM more than twice. With the increase of the budget score, the speed-up on POBM drops slightly. This is because, in POBM, when the budget score is larger, the operations that cannot be accelerated by our optimizations cost too much time.