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 clue, 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 recommendation 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 a reverse order.

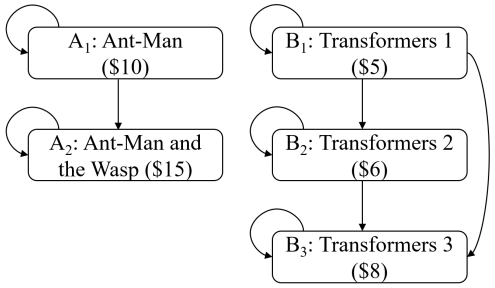


Figure 1: An example of ordered preferences for movie recommendations. A DVD price of a movie is used as its cost.

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 adapts 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. Due to space limitations, the proofs of all theorems and some additional experimental results are presented in the extended version of this paper[1].

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] offer a unified view of sequence submodularity, and Gamal et al. [Sallam *et al.*, 2020] first study 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

[1] https://xincao-unsw.github.io/BSSM-Extended.pdf

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 adapt 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 $E(s) = \{(v_i, v_j) | (v_i, v_j) \in E, i < j\}$ 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 follow:

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 sequence of items, and a budget constraint Δ, 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 films under the budget constraint.

Theorem 1. *The problem of solving 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 with non-uniform costs 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 $C(\mathcal{E}^{se}) = \sum_{v_i \in V(\mathcal{E}^{se})} c_{v_i}$, $V(\mathcal{E}^{se})$ denotes the set of items in $\mathcal{E}^{se}$, and $RE(\mathcal{E}^{se})$ is a function for obtaining the optimal order with the maximal value of the utility function over all possible orders of items contained in $\mathcal{E}^{se}$ (i.e., $RE(\mathcal{E}^{se}) = REORDER(\mathcal{E}^{se})$, and the algorithm of implementing function $REORDER$ is shown in the previous work [Tschiatschek *et al.*, 2017]). After finishing the

Algorithm 1: GBM Algorithm

Input: V, a utility function f, item costs $(c_{v_1}, c_{v_2}, \cdots, c_{v_n})$, Δ
Output: A sequence s

1 $E \leftarrow \{(v_i, v_j) | v_i \in V, v_j \in V, i \neq j\}$, $E^{ca} \leftarrow E$, $E^{se} \leftarrow \emptyset$;
2 **while** $E^{ca} != \emptyset$ **do**
3 $\quad E^{ca} \leftarrow E^{ca} \setminus \{e_l = (v_i, v_j) | v_i, v_j \in V(E^{se})\}$;
4 $\quad E^{ca} \leftarrow E^{ca} \setminus \{e_l \in E^{ca} | C(E^{se} \cup \{e_l\}) > \Delta\}$;
5 $\quad e^* \leftarrow argmax_{e_l \in E^{ca}} \frac{f(RE(E^{se} \cup \{e_l\})) - f(RE(E^{se}))}{C(E^{se} \cup \{e_l\}) - C(E^{se})}$;
6 $\quad E^{se} \leftarrow E^{se} \cup \{e^*\}$;
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;

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 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 $\lfloor \frac{\Delta}{c_{min}} \rfloor - 1$ iterations in the worst case, and in each iteration, it takes $O(n^2)$ time to update $\mathcal{E}^{se}$, and costs $O(n^2 \lfloor \frac{\Delta}{c_{min}} \rfloor log \lfloor \frac{\Delta}{c_{min}} \rfloor)$ time to select the edge $\tilde{e}^*$ for inserting into $\mathcal{E}^{se}$, since the number of items in $\mathcal{E}^{se}$ is less than $\lfloor \frac{\Delta}{c_{min}} \rfloor$ and the complexity of computing $f(REO(\mathcal{E}^{se}))$ is $O(\lfloor \frac{\Delta}{c_{min}} \rfloor log \lfloor \frac{\Delta}{c_{min}} \rfloor)$. Meanwhile, GBM costs $O(\lfloor \frac{\Delta}{c_{min}} \rfloor log \lfloor \frac{\Delta}{c_{min}} \rfloor))$ time, $O(n)$ time and $O(n^2)$ time to obtain s^1, s^2 and s^3 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 p denote any item set, and $C(p) = \sum_{v_i \in p} c_{v_i}$. POBM reformulates BSSM as a bi-objective maximization problem.

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

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

and $f_2(p) = -C(p)$. 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 p and p'. p weakly dominates p' (i.e., p is **better** than p', denotes as $p \succeq p'$) if $f_1(p) \geq f_1(p')$ and $f_2(p) \geq f_2(p')$; p dominates p' (i.e., p is **strictly better** than p', denotes as $p \succ p'$) if $f_1(p) > f_1(p')$ and $f_2(p) > f_2(p')$. But if neither p is better than p' nor p' is better than 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 p' is generated by randomly

Algorithm 2: POBM Algorithm

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

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

flipping bits of an archived item set p selected from the current P randomly (lines 3-4); if 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 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 p^{be} with the maximum utility score from P, and then get the best sequence s from p^{be} to return as a solution (lines 8-10).

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 $\mathbb{E}(T)$ denote the expected number of iterations, and P_{max} denote the largest size of P during the running process of POBM, s^* is denoted 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 $E(RE(p))$ and the quick dominance check.

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

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

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

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 p that has the closest cost score to the new solution p' and $|p| \leq |p'|$ to check whether p' is dominated by any solution in P. If not, we insert p' into P, and then check p and the solutions after p in P one by one and delete the weakly dominated solutions, until the solution's utility score is larger than that of 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.

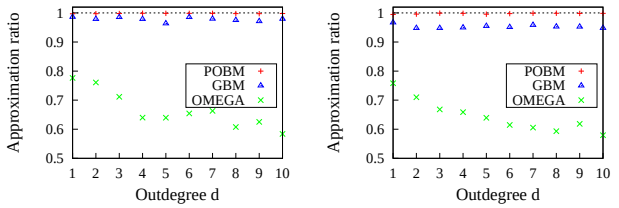
6 Experimental Study

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

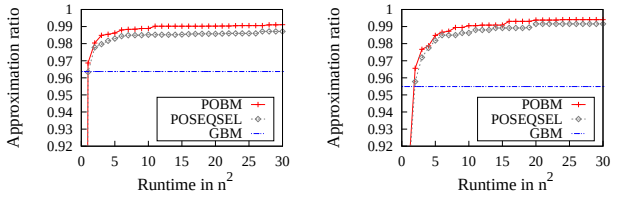
Datasets and Parameter Settings

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 $\{v_{i+1}, \cdots, v_n\}$, and 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 the maximum out-going degree 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 costs 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

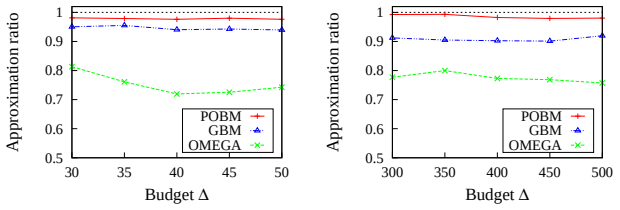
faster than POSEQSEL, which is consistent with the theoretical analysis. Note that the results of GBM keep unchanged, as they are not affected by T. According to this result, we set $T = 10n^2$ for POBM by default. Note that the result of examining the acceleration of our optimizations on POBM is reported in the extended version of this paper.

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.



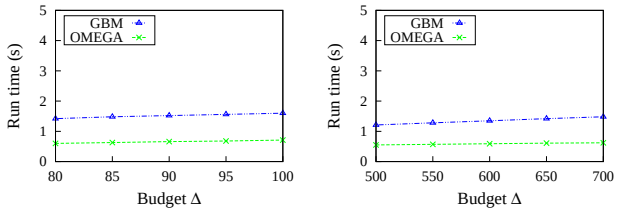
(a) MOV (b) XTX

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

Performance of Our Methods

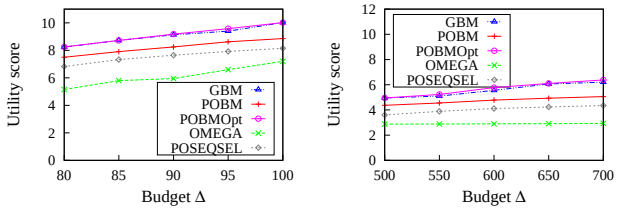
We first compare GBM and POBM with OMEGA in terms of the solution quality on both MOV and XTX datasets by varying the budget constraint. To compute the approximation ratios of algorithms, we use an exhaustive enumeration to find the optimal solutions and sample 50 movies (or courses) randomly for each 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 Δ 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 Δ 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 any time 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.

Acknowledgment

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.

[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.