Stochastic Submodular Maximization with Performance-Dependent Item Costs*

Takuro Fukunaga,[1] **Takuya Konishi,**[2] **Sumio Fujita,**[3] **Ken-ichi Kawarabayashi**[2]
[1]RIKEN Advanced Intelligence Project and JST, PRESTO, takuro.fukunaga@riken.jp
[2]National Institute of Informatics, {takuya-ko,k_keniti}@nii.ac.jp
[3]Yahoo Japan Corporation, sufujita@yahoo-corp.jp

Abstract

We formulate a new stochastic submodular maximization problem by introducing the performance-dependent costs of items. In this problem, we consider selecting items for the case where the performance of each item (i.e., how much an item contributes to the objective function) is decided randomly, and the cost of an item depends on its performance. The goal of the problem is to maximize the objective function subject to a budget constraint on the costs of the selected items. We present an adaptive algorithm for this problem with a theoretical guarantee that its expected objective value is at least $(1 - 1/\sqrt[4]{e})/2$ times the maximum value attained by any adaptive algorithms. We verify the performance of the algorithm through numerical experiments.

1 Introduction

In the stochastic submodular maximization, we are given a set of items associated with a random utility function that has a certain diminishing marginal return property. The goal of the problem is to select items so as to maximize the utility function subject to constraints. To deal with decision making under uncertainty, several variants of stochastic submodular maximization are actively being considered (see Section 2), and the approaches proposed in this literature have been successfully applied to numerous decision making tasks.

The aim of this paper is to introduce the *performance-dependent costs* of items into the stochastic submodular maximization. In many decision making applications, we sometimes have to make a decision without exact information on the performance of each item (i.e., how much an item contributes to the utility function) because it varies on several uncertain factors. Moreover, it is often the case that the cost of selecting an item varies depending on its performance. To deal with this situation, we introduce random performance and performance-dependent costs of items, and consider the stochastic optimization problem of maximizing the utility function subject to a budget constraint on the costs of the selected items.

*The first author is supported by JSPS KAKENHI Grant Number JP17K00040 and JST PRESTO Grant Number JPMJPR1759. The second and the fourth authors are supported by JST ERATO Grant Number JPMJER1201.

This problem models many natural settings of decision making. Here, we present two examples.

Recommendation: Diminishing marginal return property plays a key role in designing objectives for recommender systems (Ziegler et al. 2005). For example, in news recommendation, users often browse news articles in order to cover daily news or to know the different aspects of a favorite topic. It is critical for such users to recommend diverse and unseen items, and the objective function is modeled through submodular functions (El-Arini et al. 2009; Yue and Guestrin 2011; Ahmed et al. 2012). In several practical scenarios, the performance of a recommended item is determined by some random factors, and it incurs a cost according to the performance; e.g., when a user receives recommended news articles, he/she decides to skip or (partially) read them and spends his/her own time or money by reading the selected articles. The existing formulations are not applicable to this situation because they cannot deal with the case where both the performances and the costs of items are unknown before recommending them.

Batch-mode active learning: In the batch-mode active learning, we have a set of unlabeled data and the objective is to select several data points to label subject to a budget constraint on the cost. In a previous study (Hoi et al. 2006), this task is formulated as a monotone submodular maximization problem, where the objective function represents the information amount of data. However, this existing formulation does not consider uncertainty in the performance of labeling. In many cases, time for labeling the given data is limited, and it is uncertain in advance how much data can be processed within the given time due to factors such as variation of labelers' skill and experimental conditions. Also, it is natural that the cost (e.g., the fee paid to labelers) for the labeling depends on the amount of processed data (see also Figure 1). Our problem can model this situation while it cannot be handled by the existing formulation.

We present algorithms that compute adaptive policies for this problem together with the theoretical analysis of their performances. One of our proposed algorithms is guaranteed to achieve an expected objective value of least $(1 - 1/\sqrt[4]{e})/2$ (> 0.110) times that attained by any adaptive policy. In addition to the theoretical guarantee, we evaluate the empirical performances of our algorithms through numerical experiments. The experimental results indicate that our algorithms

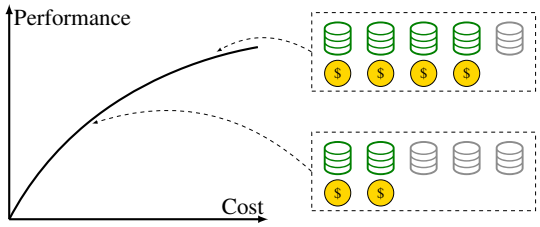

Figure 1: An illustration of the performance-dependent cost of an item in the batch-mode active learning. In this figure, an item corresponds to a group of data points and each cylinder denotes one data point in a group. Once the number of the processed data points is revealed, the corresponding expense is incurred.

perform better than baseline algorithms in many settings. For example, when the algorithms are applied to a task of the batch-mode active learning, we observe that our algorithms reduce the prediction errors of a learning algorithm compared with baseline algorithms; see Section 6 for more details.

Our algorithms extend the algorithm of Gupta et al. (2011) for a stochastic knapsack problem, which corresponds to a special case of our problem where the objective function is linear. Since the performance guarantee of Gupta et al. is $1/8 = 0.125$, our guarantee is not much worse than it although our algorithms deal with a more general problem; relationships with previous studies are discussed more in Section 2. Except for our algorithms, we are aware of no algorithms that achieve a constant approximation ratio for our problem. It is important to observe that the ratio is a constant because it means that the algorithm behaves reasonably for any instance. Indeed, numerical experiments show that the empirical performance of our algorithms is stable while baseline algorithms sometimes achieve only small objective values; in a certain instance, the scores of baseline algorithms are less than 70% of our proposed algorithms.

Our algorithms are based on the contention resolution scheme, which is a general framework to design approximation algorithms for the submodular maximization. The contention resolution scheme is so useful that many efficient algorithms based on it have been proposed in the literature. However, the existing scheme is not applied to our problem because it is restricted to the submodular set-functions. In our problem, the objective function is defined over the integer lattice in order to represent the dependence of the objective on the performance levels of selected items. Thus, we extend the contention resolution scheme to lattice-submodular functions, and design our algorithms based on this extended framework. This technique is potentially useful in other contexts, and so it is of independent interest.

To summarize, the contributions of this paper can be described as follows.

- We formulate a new stochastic submodular maximization by introducing the performance-dependent costs of items, which has numerous natural applications.
- We present adaptive algorithms for the above problem, one of which has the theoretical guarantee that its expected objective value is at least $(1 - 1/\sqrt[4]{e})/2$ times the optimal value. Its empirical performance is verified through numerical experiments.
- To design the algorithms, we extend the contention resolution scheme to lattice-submodular functions, which is a new general framework of independent interests to design approximation algorithms for maximizing lattice-submodular functions.

The remainder of this paper is organized as follows. Section 2 reviews related previous studies. Section 3 formulates our problem. Our algorithms and their analysis are given in Sections 4 and 5. Since our algorithms are based on a continuous relaxation of the original problem, Section 4 formulates this continuous relaxation and discusses how to solve it. Section 5 explains how to construct an adaptive policy from a continuous solution. Section 6 reports on numerical experiments, and Section 7 concludes the paper.

2 Related Work

Since the body of previous studies on submodular maximization is huge, we here review some of those on its stochastic variants. A typical example of stochastic submodular maximization is the study of Golovin and Krause (2011) on adaptive submodularity. They proposed the notions of adaptive submodularity and adaptive monotonicity of stochastic set-functions, and presented an adaptive algorithm for maximizing adaptive monotone submodular functions. Since their study, adaptive algorithms for maximizing adaptive submodular functions have been investigated in various settings (Chen and Krause 2013; Fujii and Kashima 2016; Gabillon et al. 2013; Gabillon et al. 2014; Gotovos, Karbasi, and Krause 2015; Yu, Fang, and Tao 2016).

Another example of stochastic submodular maximization is the submodular probing problem (Adamczyk, Sviridenko, and Ward 2016; Gupta, Nagarajan, and Singla 2017). In this problem, the given submodular function is deterministic but each item takes the active or inactive state randomly. A chosen item contributes to the objective value only when it is active. The constraints consist of inner and outer constraints, where the inner constraints restrict the chosen active items, whereas the outer constraint restricts all chosen items. Thus, the inner constraints depend on the states of the chosen items, which as far as we know, is the only example where randomness in both the objective function and the constraints are correlated as our problem.

Asadpour and Nazerzadeh (2016) considered a stochastic submodular maximization with a monotone lattice-submodular function. In their setting, each chosen item has a state selected from nonnegative rational numbers. The objective function receives the vector encoding the states of chosen items as an input. This setting for the objective function is similar to that in our problem (there is a small difference that their objective function receives real-valued vectors while our objective function receives integer-valued vectors). However, Asadpour and Nazerzadeh considered only a deterministic matroid constraint. constraints do not include randomness in Asadpour and Nazerzadeh, whereas the constraint in our problem depends on the random states of items.

When the objective function is linear, our problem coincides with the stochastic knapsack problem studied by Gupta et al. (2011). Gupta et al. gave a pseudo-polynomial time algorithm with the performance guarantee of ratio $1/8$ ($= 0.125$). Observe that the ratio $(1 - 1/\sqrt[4]{e})/2$ (> 0.110) of our algorithm is not much worse than the ratio of Gupta et al. even though our algorithm is for a more general problem. Indeed, our algorithm is equivalent to the algorithm of Gupta et al. when the problem is restricted to their problem. Gupta et al. also showed that their algorithm can be converted into a polynomial-time algorithm with a constant loss of approximation ratio. This conversion can be also applied to our algorithm although we do not focus on it in this paper. The algorithm of Gupta et al. was later improved by Ma (2014), who gave a pseudo-polynomial time algorithm with ratio $1/2$. The conversion to a polynomial-time algorithm cannot be applied to the algorithm of Ma.

3 Setting

First, we introduce the lattice-submodular functions. Let $\mathbb{Z}_+$ and $\mathbb{R}_+$ denote the sets of nonnegative integers and nonnegative real numbers, respectively. For $n \in \mathbb{Z}_+$, we let $[n]$ denote $\{0, 1, \dots, n\}$. Let I be a set of items, and let $B \in \mathbb{Z}_+$. For two vectors $u, v \in [B]^I$, $u \leq v$ means that the relation holds componentwise, i.e., $u(i) \leq v(i)$ for all $i \in I$. $u \wedge v$ and $u \vee v$ are the vectors in $[B]^I$ defined by $(u \wedge v)(i) = \min\{u(i), v(i)\}$ and $(u \vee v)(i) = \max\{u(i), v(i)\}$ for all $i \in I$. For $i \in I$, let χ_i denote the vector in $[B]^I$ such that $\chi_i(i) = 1$ and $\chi_i(i') = 0$ for all $i' \in I \setminus \{i\}$. Let $f\colon [B]^I \to \mathbb{R}_+$ be a function over the integer lattice $[B]^I$. Function f is called *monotone* if $f(u) \leq f(v)$ holds for any $u, v \in [B]^I$ such that $u \leq v$, and f is called *lattice-submodular* if $f(u) + f(v) \geq f(u \wedge v) + f(u \vee v)$ holds for all $u, v \in [B]^I$. The latter condition is equivalent to $f(u \vee j\chi_i) - f(u) \geq f(v \vee j\chi_i) - f(v)$ holding for any $u, v \in [B]^I$ such that $u \leq v$, $i \in I$, and $j \in [B]$. Note that the lattice-submodularity does not imply the property called DR-submodularity, which is the diminishing marginal returns along the direction of χ_i for each $i \in I$. That is, $f(u + \chi_i) - f(u) \geq f(v + \chi_i) - f(v)$ does not necessarily hold for all $u, v \in [B]^I$ such that $u \leq v$ and $i \in I$ even if f is lattice-submodular.

We now formulate our new submodular optimization problem, which we call the *correlated stochastic submodular maximization problem* (CSSMP). In CSSMP, we assume that each item $i \in I$ has a state $\theta(i) \in \{1, \dots, B\}$, which is independently determined at random. Let $p_i(j)$ denote the probability that the level of an item i is j, where we assume without loss of generality that $p_i(j) > 0$ holds for all $j \in \{1, \dots, B\}$. The state of an item represents its performance level; as $\theta(i)$ is larger, the performance of item i is better. The performance level of an item determines its cost and contribution to the objective value. We let $c_i(j)$ denote the cost of an item i when the state of i is j. We assume that c_i is monotone as follows, indicating that the cost of item i is larger as the performance of i is better.

Assumption 1. *$0 \leq c_i(1) \leq c_i(2) \leq \cdots \leq c_i(B) \leq C$ holds for any $i \in I$, where C is the given budget.*

For $S \subseteq I$, let θ_S denote the vector in $[B]^I$ such that $\theta_S(i) = \theta(i)$ if $i \in S$, and $\theta_S(i) = 0$ otherwise. The objective function of our problem is a monotone lattice-submodular function $f\colon [B]^I \to \mathbb{R}_+$, and the objective value of our choice S of items is defined as $f(\theta_S)$. Thus, if a chosen item performs well, then a better objective value is achieved.

Summarizing, the inputs of CSSMP are a set I of items, a monotone lattice-submodular function $f\colon [B]^I \to \mathbb{R}_+$, a budget $C \in \mathbb{Z}_+$, and the costs $c_i\colon \{1, \dots, B\} \to \mathbb{Z}_+$ and the probabilities $p_i\colon \{1, \dots, B\} \to [0, 1]$ associated with each item $i \in I$. The objective of CSSMP is to find $S \subseteq I$ that maximizes $f(\theta_S)$ subject to $\sum_{i \in S} c_i(\theta(i)) \leq C$. Recall that θ is a random vector decided by the probabilities $\{p_i\colon i \in I\}$.

Adaptive policy Our aim is to compute an efficient policy for CSSMP. A policy chooses items sequentially and receives feedback by observing the performance of the chosen items. We will now explain more concretely. We assume that the distributions determining the states (or performances) of items are known to the policy but that the realizations of the states are not given beforehand. When a policy chooses an item $i \in I$, its state $\theta(i)$ (which the policy observes) is determined according to the probability distribution p_i. The subsequent behavior of an *adaptive* policy can depend on the observations made up to that point, whereas a policy is said to be *non-adaptive* when the subsequent behavior is independent of observations.

In this paper, we consider the situation where the cancellation of items is prohibited. That is, once a policy chooses an item, this choice is irrevocable. In addition, we also prohibit the total costs of selected items from exceeding the budget. This means that we assume the following.

Assumption 2. *If a policy has already selected a set S of items and $c_i(B) > C - \sum_{i' \in S} c_{i'}(\theta(i'))$ for some item $i \in I \setminus S$, then the policy cannot select item i.*

We use a vector $r \in [B]^I$ to describe the behavior of a policy. If an item $i \in I$ is chosen by the policy and the state of i is realized as $\theta(i)$, then $r(i) = \theta(i)$. If $i \in I$ has not been chosen yet, then $r(i) = 0$. We call this vector a *realization vector*. For a policy π and a realization vector $r \in [B]^I$, let $\omega_\pi(r) \in [0, 1]$ denote the probability that the final state of the policy π is r (i.e., π chooses each item $i \in \{i' \in I\colon r(i') > 0\}$, and the state of that item i is realized as $r(i)$). We note that $\omega_\pi(r)$ reflects the randomness of both π and the states of items when π is a randomized policy. Let $f_{\mathrm{avg}}(\pi)$ denote $\sum_{r \in [B]^I} \omega_\pi(r) f(r)$, i.e., the average objective value attained by π. We evaluate the performance of a policy π by $f_{\mathrm{avg}}(\pi)$. Our aim is to find a policy π for which $f_{\mathrm{avg}}(\pi)$ is as high as possible. We say that π is an *α-approximation policy* for $\alpha \in [0, 1]$ if $f_{\mathrm{avg}}(\pi) \geq \alpha f_{\mathrm{avg}}(\pi^*)$ holds for any policy π^*.

4 Continuous Optimization Phase

Our algorithm for CSSMP consists of two phases, the continuous optimization phase (which solves a continuous optimization problem) and the rounding phase (which constructs an adaptive policy from the computed solution for the continuous optimization problem). This section explains the continuous optimization phase.

4.1 Formulation of Continuous Optimization Problem

In this subsection, we define our continuous optimization problem. Before presenting it, we first define a submodular set-function $\bar{f}\colon 2^I \to \mathbb{R}_+$ from a lattice-submodular function $f\colon [B]^I \to \mathbb{R}_+$. Let $S \subseteq I$. We randomly sample a vector $r \in [B]^I$ as follows. The components of r are determined independently. If $i \in S$, then the corresponding component $r(i)$ takes a value from $\{1, \dots, B\}$, and $r(i) = j$ with probability $p_i(j)$. Otherwise, $r(i) = 0$. We let $p_S(r)$ denote the probability that $r \in [B]^I$ is sampled. We let $r \sim p_S$ denote that vector r is sampled according to the distribution p_S. Then, we define $\bar{f}(S) = \mathbb{E}_{r\sim p_S}[f(r)]$ for any $S \subseteq I$. If f is monotone lattice-submodular, then $\bar{f}$ is monotone set-submodular (Asadpour and Nazerzadeh 2016).

The continuous optimization problem is based on the concept of time. We assume that, if an item i is selected, then this selection uses time $c_i(\theta(i))$. In other words, if item i is selected at time t, then the processing of i continues until time $t + c_i(\theta(i))$, after which the policy can choose the next item. At time 0, the policy has selected no items. We regard the budget C as the time limit. Because of Assumption 2, the policy cannot choose item i if the processing of i may not finish before time C.

A variable $x(i,t) \in [0,1]$ is defined for each $i \in I$ and $t \in [C]$. This variable indicates whether item i is selected at time t. Let $\bar{x}$ denote the vector in $\mathbb{R}_+^I$ defined by $\bar{x}(i) = \sum_{t\in[C-c_i(B)]} x(i,t)$. Our continuous optimization problem includes a constraint $\bar{x}(i) \le 1$ for each $i \in I$. Define $\bar{F}\colon [0,1]^I \to \mathbb{R}_+$ as the multilinear extension of $\bar{f}$, i.e., $\bar{F}(y) = \sum_{S\subseteq I} \prod_{i\in S} y(i) \prod_{i'\notin S}(1 - y(i'))\bar{f}(S)$ for any $y \in [0,1]^I$. Let P be the set of $x \in [0,1]^{I\times[C]}$ satisfying $\bar{x}(i) \le 1$ for all $i \in I$, and

$$\sum_{i\in I} \mathbb{E}[\min\{c_i(\theta(i)), t\}] \sum_{t'\in[t]} x(i,t') \le 2t \qquad (1)$$

for all $t \in \{1, \dots, C\}$. Here, the expectation $\mathbb{E}[\min\{c_i(\theta(i)), t\}]$ is taken with respect to distribution p_i. In other words, $\mathbb{E}[\min\{c_i(\theta(i)), t\}] = \sum_{j=1}^{B} p_i(j) \min\{c_i(j), t\}$. Then, our continuous optimization problem can be written as

$$\max\{\bar{F}(\bar{x})\colon x \in P\}. \qquad (2)$$

Although we define $x(i,t)$ even for $t > C - c_i(B)$, this is only for notational convenience. Since it does not contribute to the objective function, the objective value does not decrease even if it is set to 0. Thus we assume $x(i,t) = 0$ for $t > C - c_i(B)$ in the remainder of this paper.

Note that (2) contains $\Omega(|I|C)$ variables and $\Omega(|I| + C)$ constraints. Since the budget C is encoded in $O(\log C)$ bits, the formulation size of (2) is not polynomial but pseudo-polynomial on the input size. Due to this, our algorithm based on (2) is a pseudo-polynomial time algorithm. We can convert the algorithm into a polynomial-time algorithm with a constant loss of approximation ratio by applying the technique used in (Gupta et al. 2011) but we instead focus on the current form. Evaluating the function $\bar{F}$ can be also done by sampling in polynomial time with an error of factor $1 + \epsilon$ for any constant $\epsilon > 0$. This is standard in the submodular maximization, and see e.g., (Asadpour and Nazerzadeh 2016) and (Călinescu et al. 2011). In the rest of this section, we assume that F can be evaluated exactly for ease of discussion.

In the following theorem, we relate the optimal value of the continuous optimization problem to the maximum expected objective value attained by any adaptive policy. This enables us to compare adaptive policy based on this continuous optimization problem with an optimal adaptive policy.

Theorem 1. *The optimal value of* (2) *is at least* $(1 - 1/e) \cdot f_{\mathrm{avg}}(\pi^*)$ *for any adaptive policy* π^*.

This theorem is proven by the validity of constraints proven by Gupta et al. (2011) and the relationship between $\bar{F}$ and the expected objective values achieved by adaptive policies given by Asadpour and Nazerzadeh (2016).

4.2 Algorithms for the Continuous Optimization Problem

To solve the continuous optimization problem (2), we have two choices: one is the continuous greedy algorithm, which was proposed by Călinescu et al. (2011) and was slightly extended by Feldman (2013); the other is the stochastic continuous greedy algorithm proposed by Asadpour and Nazerzadeh (2016).

Continuous Greedy This algorithm is for maximizing the multilinear extension G of a monotone set-submodular function g over a solvable downward-closed polytope. Here, a polytope Q is said to be *solvable* if there is an algorithm that optimizes linear functions over it, and is said to be *downward-closed* if $0 \le y \le y' \in Q$ imply $y \in Q$. The algorithm is controlled by a parameter called stopping time. Feldman (2013) observed that, if the continuous greedy algorithm with stopping time $b > 0$ is applied to the problem with a solvable downward-closed polytope Q, then the algorithm outputs a solution x such that $x/b \in Q$ and $G(x) \ge (1 - e^{-b} - O(n^3\delta)) \max_{y\in Q} G(y)$ hold, where n is the size of the set over which g is defined and δ is the step size used in the algorithm. Here Q is assumed to include the characteristic vector of every singleton set. Stopping time b should be set to 1 for computing a feasible continuous solution attaining a better objective value. However, when the continues greedy algorithm is combined with a rounding algorithm, we sometimes obtain a better performance guarantee by setting b to a value smaller than 1.

It is not obvious that the continuous greedy algorithm can be applied to our continuous optimization problem (2). This is because the objective function $\bar{F}(\bar{x})$ (defined over the domain $[0,1]^{I\times[C]}$) does not seem to be the multilinear extension of a submodular set-function on $I \times [C]$. Nevertheless, we can claim that this earlier analysis is valid even for our problem (2), because the problem is equivalent to maximizing $\bar{F}(y)$ subject to $y \in Q := \{y' \in [0,1]^I\colon \exists x \in P, y' = \bar{x}\}$. Notice that Q is a downward-closed solvable polytope in $|I|$-dimensional space. Therefore, combining Theorem 1 and the analysis of Feldman (2013) gives the following theorem.

Theorem 2. *If the continuous greedy algorithm with stopping time $b \in (0,1]$ and step size $\delta = o(|I|^{-3})$ is applied to Problem* (2)*, then it outputs a solution $x \in bP$ such that $\bar{F}(\bar{x}) \geq (1-1/e)(1-e^{-b}-o(1))f_{\mathrm{avg}}(\pi^*)$ for any adaptive policy π^*.*

Let us sketch how the continuous greedy algorithm computes a solution for (2). The algorithm first initializes all variables in the temporary solution x to 0 and then updates them by repeating the following two steps:

Step (i): find a vector $d_{\max} \in P$ that maximizes $w^\top d_{\max}$ for the weight vector $w \in \mathbb{R}_+^{I\times[C]}$ defined by $w(i,t) = \bar{F}(\bar{x} \vee \chi_i) - \bar{F}(\bar{x})$ for each $(i,t) \in I \times [C]$;

Step (ii): move the current solution x in direction $d_{\max}$ by step size $\delta \in (0,1]$ (i.e., x is updated to $x + \delta d_{\max}$).

When the stopping time is $b \in (0,1]$, the algorithm outputs x after repeating these steps b/δ times.

Stochastic Continuous Greedy Theorem 2 states that using the continuous greedy algorithm to solve (2) imposes two factors, one coming from the gap between (2) and the optimal objective value, and the other being due to the performance of the continuous greedy algorithm. By using the stochastic continuous greedy algorithm, we can save the former factor.

The difference between the continuous greedy and the stochastic continuous greedy algorithms is the weight vector used for deciding $d_{\max}$. Here, we define the weight vector w' used in the stochastic continuous greedy algorithm. Let $r \in [B]^I$ be a random vector such that $r(i) = j \in \{1,\dots,B\}$ with probability $\bar{x}(i)p_i(j)$ and $r(i) = 0$ with probability $1 - \bar{x}(i)$ for each $i \in I$, where the different components are determined independently. Moreover, for each $i \in I$, choose one integer j randomly from $\{1,\dots,B\}$ according to probability $p_i(j)$, and define the vector $r'_i \in [B]^I$ by $r'_i(i) = \max\{r(i), j\}$ and $r'_i(i') = r(i')$ for each $i' \in I \setminus \{i\}$. The value of $w'(i,t)$ is defined as $\mathbb{E}[f(r'_i) - f(r)]$ for each $(i,t) \in I \times [C]$; $w'(i,t)$ takes the same value for all $t \in [C]$. We note that, if $r'_i(i)$ is defined as j, then this weight vector coincides with w used in the continuous greedy algorithm. The stochastic continuous greedy algorithm uses w' instead of w in Step (i) of each iteration. The other part of the algorithm is the same as the continuous greedy algorithm.

Asadpour and Nazerzadeh (2016) proved that the stochastic continuous greedy algorithm with stopping time $b = 1$ outputs a solution of value at least $(1 - e^{-1} - o(1))f_{\mathrm{avg}}(\pi^*)$. Note that this bound is better than that of the continuous greedy algorithm by the factor $1 - e^{-1}$. Their analysis can be extended to an arbitrary value of the stopping time as follows (we skip the proof because the extension is straightforward).

Theorem 3. *If the stochastic continuous greedy algorithm with stopping time $b \in (0,1]$ and step size $\delta = o(|I|^{-3})$ is applied to Problem* (2)*, then the algorithm outputs a solution $x \in bP$ such that $\bar{F}(\bar{x}) \geq (1-e^{-b}-o(1))f_{\mathrm{avg}}(\pi^*)$ for any adaptive policy π^*.*

5 Rounding Phase

This section presents an algorithm that outputs an adaptive policy achieving at least half the objective value of the continuous solution. For this, we introduce the contention resolution scheme for lattice-submodular functions. The existing contention resolution scheme is a general framework that provides a rounding algorithm for maximizing set-submodular functions (Călinescu et al. 2011; Feldman 2013; Feldman, Naor, and Schwartz 2011). We extend this scheme to the lattice-submodular functions, and show that the solution constructed by our policy coincides with the solution output by a contention resolution scheme. Since the contention resolution scheme for set-submodular functions is widely used in the literature, our extension is of independent interest.

5.1 Contention Resolution Scheme for Lattice-Submodular Functions

The considered setting is defined as follows. Let $f\colon [B]^I \to \mathbb{R}_+$ be a monotone lattice-submodular function and the probability distribution $q_i\colon [B] \to [0,1]$ on $[B]$ be given for each $i \in I$. We write $v \sim q$ if $v \in [B]^I$ is a random vector such that, for each $i \in I$, the corresponding component $v(i)$ is determined independently as $j \in [B]$ with probability $q_i(j)$. Let $\mathcal{F} \subseteq [B]^I$ be a downward-closed subset of $[B]^I$ (i.e., if $u \leq v \in \mathcal{F}$, then $u \in \mathcal{F}$), and let $\alpha \in [0,1]$. A mapping $\psi\colon [B]^I \to \mathcal{F}$ is referred to as an *α-contention resolution scheme* (α-CRS) with regards to q if it satisfies the following two conditions:

(i) $\psi(v)(i) \in \{v(i), 0\}$ for each $i \in I$;

(ii) if $v \sim q$, then $\Pr[\psi(v)(i) = j \mid v(i) = j] \geq \alpha$ holds for each $i \in I$ and $j \in B$, where the probability considers the randomness of v as well as that of ψ when ψ is a random mapping.

An α-CRS ψ is said to be *monotone* if, for each $u, v \in [B]^I$ such that $u(i) = v(i)$ and $u \leq v$, $\Pr[\psi(u)(i) = u(i)] \geq \Pr[\psi(v)(i) = v(i)]$ holds, where the probability here considers only the randomness of ψ.

We prove that a monotone α-CRS maps $v \in [B]^I$ into a vector in $\mathcal{F}$, for which the value of f is at least α times that for v in expectation. The proof requires the following preliminary lemma, known as the FKG inequality.

Lemma 1 (Fortuin, Kasteleyn, and Ginibre (1971))**.** *Let $\mu\colon L \to [0,1]$ be a log-supermodular probability distribution on a distributive lattice L (i.e., $\mu(u)\cdot\mu(v) \leq \mu(u\wedge v)\cdot\mu(u\vee v)$ for any $u, v \in L$). Then, for any non-increasing functions $h, l\colon L \to \mathbb{R}_+$, we have*

$$\mathbb{E}_{u\sim\mu}[h(u)] \cdot \mathbb{E}_{v\sim\mu}[l(v)] \leq \mathbb{E}_{v\sim\mu}[h(v)\cdot l(v)].$$

We note that, if μ is a distribution on $[B]^I$ such that the components of a random vector $v \sim \mu$ are decided independently, then μ is log-supermodular.

Theorem 4. *If ψ is a monotone α-CRS with respect to q, then $\mathbb{E}_{v\sim q}[f(\psi(v))] \geq \alpha\mathbb{E}_{v\sim q}[f(v)]$.*

Proof. For notational convenience, let $I = \{1,\dots,n\}$ and let u denote $\psi(v)$. For a vector $y \in [B]^I$ and $i \in I$, we let $\hat{y}_i$ denote the vector defined by $\hat{y}_i(i') = y(i')$ for $i' \in [i]$, and $\hat{y}_i(i') = 0$ for $i' \in \{i+1,\dots,n\}$. In this proof,

expectations are with regard to the randomness of v (sampled with distribution q) and that of ψ unless stated otherwise.

We prove that

$$\mathbb{E}[f(\hat{u}_i) - f(\hat{u}_{i-1})] \geq \alpha \mathbb{E}[f(\hat{v}_i) - f(\hat{v}_{i-1})] \qquad (3)$$

holds for all $i \in I$. The theorem is proven by summing this z1inequality over all $i \in I$.

Let us prove (3) for $i \in I$. We let $\Delta(x, y)$ denote $f(y) - f(x)$ for $x, y \in [B]^I$ and $\mathbb{I}$ be the indicator function of events; i.e., $\mathbb{I}[E] = 1$ if an event E occurs, and $\mathbb{I}[E] = 0$ otherwise. The left-hand side of (3) is bounded as

$$\begin{aligned}
&\mathbb{E}[f(\hat{u}_i) - f(\hat{u}_{i-1})] \\
&= \mathbb{E}[\mathbb{I}[u(i) > 0]\Delta(\hat{u}_{i-1}, \hat{u}_i)] \\
&= \sum_{j=1}^{B} \Pr[v(i) = j] \cdot \mathbb{E}[\mathbb{I}[u(i) > 0]\Delta(\hat{u}_{i-1}, \hat{u}_i) \mid v(i) = j] \\
&= \sum_{j=1}^{B} \Pr[v(i) = j] \cdot \mathbb{E}[\mathbb{I}[u(i) = j]\Delta(\hat{u}_{i-1}, \hat{u}_i) \mid v(i) = j] \\
&\geq \sum_{j=1}^{B} \Pr[v(i) = j] \cdot \mathbb{E}[\mathbb{I}[u(i) = j]\Delta(\hat{v}_{i-1}, \hat{v}_i) \mid v(i) = j],
\end{aligned}$$

where the inequality is obtained from the lattice-submodularity of f and $\hat{u}_{i-1} \leq \hat{v}_{i-1}$.

We give a lower-bound on $\mathbb{E}[\mathbb{I}[u(i) = j]\Delta(\hat{v}_{i-1}, \hat{v}_i) \mid v(i) = j]$ for a fixed $j \in [B]$. Let L denote the sub-lattice $\{x \in [B]^I : x(i) = j\}$ of $[B]^I$. We define two functions $h, l : L \to \mathbb{R}_+$ by $h(v) = \Pr[\psi(v)(i) = j]$ and $l(v) = \Delta(\hat{v}_{i-1}, \hat{v}_i)$ for each $v \in L$. Then, both h and l are non-increasing; indeed, the non-increasingness of h follows from the monotonicity of ψ, and that of l follows from the lattice-submodularity of f. Moreover,

$$\begin{aligned}
&\mathbb{E}[\mathbb{I}[u(i) = j]\Delta(\hat{v}_{i-1}, \hat{v}_i) \mid v(i) = j] \\
&= \mathbb{E}[h(v)l(v) \mid v(i) = j] \\
&\geq \mathbb{E}[h(v) \mid v(i) = j] \cdot \mathbb{E}[l(v) \mid v(i) = j], \\
&= \Pr[\psi(v)(i) = j \mid v(i) = j] \cdot \mathbb{E}[\Delta(\hat{v}_{i-1}, \hat{v}_i) \mid v(i) = j] \\
&\geq \alpha \mathbb{E}[\Delta(\hat{v}_{i-1}, \hat{v}_i) \mid v(i) = j]
\end{aligned}$$

where the first inequality follows from the FKG inequality and the second inequality follows from the definition of α-CRS.

Hence we have

$$\begin{aligned}
&\sum_{j=1}^{B} \Pr[v(i) = j] \cdot \mathbb{E}[\mathbb{I}[u(i) = j]\Delta(\hat{v}_{i-1}, \hat{v}_i) \mid v(i) = j] \\
&\geq \alpha \sum_{j=1}^{B} \Pr[v(i) = j] \cdot \mathbb{E}[\Delta(\hat{v}_{i-1}, \hat{v}_i) \mid v(i) = j] \\
&= \alpha \mathbb{E}[\Delta(\hat{v}_{i-1}, \hat{v}_i)].
\end{aligned}$$

This completes the proof of (3). □

5.2 Proposed Algorithm

Our algorithm is given as Algorithm 1. It consists of two parts. The first part is pre-processing. The algorithm is given the problem instance and computes an ordering of items. The second part corresponds to an adaptive policy. The algorithm sequentially chooses items according to the ordering computed in the first part, and then observes their states. The output of the algorithm is the realization vector representing the final states of items.

Algorithm 1 Pre-processing and adaptive policy

Input: set I of items, monotone lattice-submodular function $f : [B]^I \to \mathbb{R}_+$, budget $C \in \mathbb{Z}_+$, costs $c_i : [1, B] \to \mathbb{Z}_+$, and probabilities $p_i : [1, B] \to [0, 1]$ $(i \in I)$
Output: $r \in [B]^I$

```
// Pre-processing
```
compute a solution x for (2) by the continuous greedy or the stochastic continuous greedy algorithm with stopping time $1/4$
$r \longleftarrow 0$, $I' \longleftarrow \emptyset$, $C' \longleftarrow 0$
for $i \in I$ **do**
  sample a number t from $[C - c_i(B)]$ with probability $x(i, t)$ (or do nothing with probability $1 - \sum_{t \in [C - c_i(B)]} x(i, t) = 1 - \bar{x}(i)$)
  if some number is chosen in the previous step **then** call it $t(i)$ and update $I' \longleftarrow I' \cup \{i\}$
if $I' = \emptyset$ **then** output r and terminate
$\Pi \longleftarrow$ sequence of items in I' obtained by sorting in a non-decreasing order of $t(i)$, breaking ties arbitrarily
```
// Adaptive Policy
```
for $i = 1, \ldots, |I'|$ **do**
  if $C' \leq t(\Pi_i)$ **then**
    observe $\theta(\Pi_i)$
    $r(\Pi_i) \longleftarrow \theta(\Pi_i)$
    $C' \longleftarrow C' + c_{\Pi_i}(\theta(\Pi_i))$
output r and terminate

Theorem 5. *Let π denote Algorithm 1, and x denote the solution for* (2) *computed in Algorithm 1. Then $f_{\mathrm{avg}}(\pi) \geq \bar{F}(\bar{x})/2$ holds.*

The following corollary is derived from Theorems 2, 3, and 5.

Corollary 1. *Let π denote Algorithm 1. If π uses the continuous greedy algorithm with $\delta = o(|I|^{-3})$ to compute x, then $f_{\mathrm{avg}}(\pi) \geq (1 - 1/e)(1 - 1/\sqrt[4]{e} - O(1))/2 \cdot f_{\mathrm{avg}}(\pi^*)$ holds for any adaptive policy π^*. If π uses the stochastic continuous greedy algorithm with $\delta = o(|I|^{-3})$, then $f_{\mathrm{avg}}(\pi) \geq (1 - 1/\sqrt[4]{e} - o(1))/2 \cdot f_{\mathrm{avg}}(\pi^*)$ holds for any adaptive policy π^*.*

The remainder of this subsection is the proof of Theorem 5. To analyze Algorithm 1, we present two mappings $\sigma : P \to [B]^I$ and $\tau : [B]^I \to [B]^I$ such that the output of Algorithm 1 is bounded by $\tau(\sigma(x))$ from below, where x is the solution for (2) computed in the first step. $\sigma(x)$ is a random vector which is given to a CRS, and τ is a CRS. Our proof of Theorem 5 shows these correspondences. Below, we first present the definitions of σ and τ.

The mapping $\sigma(x)$ returns a random vector $v \in [B]^I$ from $x \in P$ as follows. Let $i \in I$. The component $v(i)$ corresponding to i is $j \in [B]$ with probability $p_i(j)\bar{x}(i)$, and otherwise

(probability $1 - \bar{x}(i)$) is $v(i) = 0$. Each component of v is determined independently. We note that the construction v corresponds to the construction of I' in Algorithm 1; the probability that $v(i)$ is set to $j > 0$ is equal to the one that the state of i is realized as j and i is included in I'.

The mapping τ maps $v \in [B]^I$ to $y \in [B]^I$ as follows. Let $S = \{i \in I \colon v(i) \neq 0\}$. For each $i \in S$, we choose an integer $t(i)$ from $[C - c_i(B)]$ with probability $x(i, t(i))/\bar{x}(i)$. We sort the members of S into non-decreasing order of $t(i)$. We assume without loss of generality that $S = \{1, \dots, k\}$ and $t(1) \leq t(2) \leq \cdots \leq t(k)$. Vector y is defined as follows. For $i \in I \setminus S$, $y(i)$ is set to 0. For $i \in S$, $y(i) = v(i)$ if

$$\sum_{i'=1}^{i-1} c_{i'}(v(i')) \leq t(i), \tag{4}$$

and $y(i) = 0$ otherwise. We notice that setting $y(i)$ to $v(i) > 0$ corresponds to π choosing item i. However, condition (4) is slightly stronger than the condition for π to choose i, which is represented as $\sum_{i'=1,\dots,i-1:y(i')>0} c_{i'}(v(i')) \leq t(i)$.

We can observe that Algorithm 1 outputs r such that $r \geq \tau(\sigma(x))$ if realizations of randomness coincide between Algorithm 1 and mappings σ and τ; r may not be equal to $\tau(\sigma(x))$ because condition (4) for $y(i) = v(i)$ in the definition of τ is stronger than the corresponding condition for choosing item i in π. This implies that $\mathbb{E}[f(r)] \geq \mathbb{E}[f(\tau(\sigma(x)))]$, by the monotonicity of f. Hence, it suffices for proving Theorem 5 to show $\mathbb{E}[f(\tau(\sigma(x)))] \geq \bar{F}(\bar{x})/2$. For proving this relationship, we use the contention resolution scheme for lattice-submodular functions. For this, we first observe the following lemma.

Lemma 2. *$\mathbb{E}[f(\sigma(x))] = \bar{F}(\bar{x})$ holds for any $x \in P$.*

Proof. Let S be a random subset of I such that each $i \in I$ is included in S independently with probability $\bar{x}(i)$. By the definition of $\bar{F}$, we have $\bar{F}(\bar{x}) = \mathbb{E}[\bar{f}(S)]$. Recall that $\bar{f}(S) = \mathbb{E}_{u \sim p_S}[f(u)]$, and hence $\bar{F}(\bar{x}) = \mathbb{E}_{u \sim p_S}[f(u)]$. Further, notice that $\sigma(x) \sim p_S$, and hence $\mathbb{E}_{u \sim p_S}[f(u)] = \mathbb{E}[f(\sigma(x))]$. □

Lemma 3. *Let $q \colon [B]^I \to [0, 1]$ be the probability distribution over $[B]^I$ such that, if $v \sim q$, then $v(i) = j$ with probability $p_i(j)\bar{x}(i)$ for each $j \in \{1, \dots, B\}$, $v(i) = 0$ with probability $1 - \bar{x}(i)$, and different components of v are determined independently. Let $\mathcal{I} = \{y \in [B]^I \colon \sum_{i \in I : y(i) > 0} c_i(y(i)) \leq C\}$. Then, $\tau(v) \in \mathcal{I}$ holds for any $v \in [B]^I$. Moreover, τ is a monotone $1/2$-CRS with respect to q.*

Proof. First, let us observe that $\tau(v) \in \mathcal{I}$ for any $v \in [B]^I$. We denote $\tau(v)$ by y. We also use the notation $S = \{1, \dots, k\}$ and $t(1), \dots, t(k)$ used in the definition of τ. If $y(i) = 0$ for all $i \in S$, then obviously $y \in \mathcal{I}$. Hence we suppose the other case, and let i denote the largest member of S such that $y(i) > 0$. Then, $\sum_{i' \in I : y(i') > 0} c_{i'}(y(i')) = c_i(y_i) + \sum_{i' < i : y(i') > 0} c_{i'}(y(i')) \leq c_i(y(i)) + t(i) \leq C$ holds, where the equality follows from the definition of i, the first inequality follows from $y(i) > 0$ and (4), and the second inequality follows from the fact that $t(i) \leq C - c_i(B) \leq C - c_i(y(i))$.

Next, we show that τ is a $1/2$-CRS. Here, define v as a random vector with $v \sim q$. Let $v(i) = j > 0$. The probability that $y(i) = j$ holds is

$$\begin{aligned} &\Pr\left[\sum_{i'=1}^{i-1} c_{i'}(v(i')) \leq t(i)\right] \\ &= \Pr\left[\sum_{i'=1}^{i-1} \min\{c_{i'}(v(i')), t(i)\} \leq t(i)\right]. \end{aligned} \tag{5}$$

Each $i' \in I$ belongs to S (i.e., $v(i') > 0$) and then chooses $t(i')$ from $[t(i)]$ with probability $\sum_{t \in [t(i)]} x(i', t)$. Conditioned on $v(i') > 0$, the probability that $v(i') = j$ is $p_{i'}(j)$. Hence, we have

$$\begin{aligned} &\mathbb{E}\left[\sum_{i'=1}^{i-1} \min\{c_{i'}(v(i')), t(i)\}\right] \\ &= \sum_{i' \in I} \mathbb{E}[\min\{c_{i'}(\theta(i')), t(i)\}] \sum_{t \in [t(i)]} x(i', t). \end{aligned}$$

The right-hand side of this equation is at most $t(i)/2$ because $4x \in P$. Hence, by Markov's inequality, (5) is at least 1/2.

Lastly, let us show that τ is monotone. Suppose that vectors $u, v \in [B]^I$ satisfy $u \leq v$ and $u(i) = v(i) = j > 0$. It suffices to show $\Pr[\tau(u)(i) = j] \geq \Pr[\tau(v)(i) = j]$. In this case, $i \in S$. Since the choices of $t(i')$, $i' \in S$, depend on only x, we can consider each $i' \in S$ to choose the same $t(i')$ in the constructions of both $\tau(u)$ and $\tau(v)$. In this case, $\sum_{i'=1}^{i-1} c_{i'}(u(i')) \leq \sum_{i'=1}^{i-1} c_{i'}(v(i'))$ follows from $u \leq v$, and hence $\Pr[\tau(u)(i) = j] \geq \Pr[\tau(v)(i) = j]$ holds. □

Proof of Theorem 5. The output r of Algorithm 1 satisfies $\mathbb{E}[f(r)] \geq \mathbb{E}[f(\tau(\sigma(x)))]$. We can also observe that r is always feasible.

By Lemma 3, τ is a monotone $1/2$-CRS with respect to q, where q is the probability distribution over $[B]^I$ defined in Lemma 3. Moreover, $\sigma(x) \sim q$ holds. Hence, by Theorem 4, $\mathbb{E}[f(\tau(\sigma(x)))] \geq \mathbb{E}[f(\sigma(x))]/2$. The right-hand side of this inequality is $\bar{F}(\bar{x})/2$ by Lemma 2. Therefore, $f_{\mathrm{avg}}(\pi) = \mathbb{E}[f(r)] \geq \mathbb{E}[f(\tau(\sigma(x)))] \geq \bar{F}(\bar{x})/2$. □

We note that Assumptions 1 and 2 are required for proving the monotonicity of τ in Lemma 3. In the previous studies (Gupta, Nagarajan, and Singla 2017; Ma 2014) on the stochastic knapsack problem, adaptive algorithms achieve a constant approximation guarantee without these assumptions. Hence it is interesting to investigate whether those assumptions are really necessary for CSSMP.

6 Experimental Results

As proposed algorithms, we prepared two implementations; one employs the continuous greedy algorithm to solve the continuous optimization problem (2), and the other does the stochastic continuous greedy algorithm. The step size δ was set to $o(|I|^{-3})$ in the performance guarantee given in Corollary 1, but this setting requires large computational time. Hence we set δ to $(2|I|)^{-1}$ in our implementations. The stopping time b was set to 1. Again, this is different from the

Algorithm 2 Greedy algorithm

Input: set I of items, monotone lattice-submodular function $f\colon [B]^I \to \mathbb{R}_+$, budget $C \in \mathbb{Z}_+$, costs $c_i\colon \{1,\dots,B\} \to \mathbb{Z}_+$ and probabilities $p_i\colon \{1,\dots,B\} \to [0,1]$ $(i \in I)$
Output: $r \in [B]^I$
$r \longleftarrow \mathbf{0}$, $C' \longleftarrow 0$, $S \longleftarrow \emptyset$
$I' \longleftarrow \{i \in I \setminus S\colon c_i(B) + C' \leq C\}$
while $I' \neq \emptyset$ **do**
  for $i \in I'$ **do** compute $\Delta(i \mid r)$ by (6) or (7)
  $i = \operatorname{argmax}_{i' \in I'} \Delta(i' \mid r)$
  observe $\theta(i)$, and $r(i) \longleftarrow \theta(i)$
  $C' \longleftarrow C' + c_i(\theta(i))$
  $S \longleftarrow S \cup \{i\}$, $I' \longleftarrow \{i' \in I \setminus S\colon c_{i'}(B) + C' \leq C\}$
output r and terminate

setting of Algorithm 1, but in our experiments, the setting of $b = 1$ attained the best performance with the above parameter settings.

We compare these implementations with two baseline algorithms obtained by extending the well-known greedy algorithms for the maximization of set-submodular functions. The algorithms iteratively select an item that maximizes an evaluation metric defined as follows. Let S be the set of items chosen so far, and $r \in [B]^I$ be the realization vector (i.e, $r(i) = 0$ for $i \in I \setminus S$, and $r(i)$ is the state of i for $i \in S$). Both of the algorithms evaluate an item $i \in I \setminus S$ by using the expected ratio of the function gain to the cost when i is chosen. The first baseline evaluates the gain of item i by

$$\Delta(i \mid r) = \mathbb{E}\left[\frac{f(r \vee \theta(i)\chi_i) - f(r)}{c_i(\theta(i))}\right], \tag{6}$$

whereas the second baseline does by

$$\Delta(i \mid r) = \frac{\mathbb{E}[f(r \vee \theta(i)\chi_i) - f(r)]}{\mathbb{E}[c_i(\theta(i))]}. \tag{7}$$

The details of these baseline algorithms are given in Algorithm 2.

We also incorporate a heuristic to pick items greedily into the implementations of the proposed algorithms; if all the budget is not spent after executing the rounding phase, the remaining items are picked greedily.

6.1 Recommendation

We first report the results on synthetic datasets constructed with a motivation to apply our algorithms to recommendations. More concretely, we consider recommending items (e.g., news article or movie) to a user so as to maximize the user's utility. If an item i is recommended, then the user evaluates it through his/her action (e.g., skipping or reading articles). We define B levels of evaluations. The history of the recommendations is represented by a vector $r \in [B]^I$; $r(i) = 0$ indicates that item i is not recommended to the user, whereas $r(i) > 0$ means that i is recommended to the user and its evaluation is $r(i)$. When the evaluation for item i is j, the user incurs a cost $c_i(j)$. Here, the cost is such as the fee or time to obtain the evaluation j. For example, in pay-per-article news platforms, e.g., Blendle, the user needs to pay the fee for reading an article completely. We assume that the budget C of the user is known or can be estimated from the user activities, and the task is to recommend items under the budget constraint defined by C.

We define the utility of the user by extending the probabilistic topic coverage function, that is often used in recommendation (e.g., (El-Arini et al. 2009)). While the function is originally defined as a set-function, we extend it to an integer lattice for modeling the levels of evaluations. Let K be the number of topics. We define weight vectors $w \in [0,1]^K$ and $\phi_i \in [0,1]^K$ $(i \in I)$ such that $\sum_{k=1}^K w(k) = 1$ and $\sum_{k=1}^K \phi_i(k) = 1$ for all $i \in I$. $w(k)$ indicates a user's preference for the kth topic, and $\phi_i(k)$ indicates the proportion of the kth topic in item i. Given w and ϕ_i $(i \in I)$, the utility function $f : [B]^I \to \mathbb{R}_+$ is defined by $f(r) = \sum_{k \in K} w(k) \left(1 - \prod_{i \in I} \left(1 - \frac{r(i)\phi_i(k)}{B}\right)\right)$ for each $r \in [B]^I$. This is monotone lattice-submodular.

In the experiments, we constructed probability distribution $p_i\colon \{1,\dots,B\} \to [0,1]$ randomly according to the symmetric Dirichlet distribution with parameter 1.0, and the vectors w and ϕ_i $(i \in I)$ were sampled from the symmetric Dirichlet distribution with parameter α. The cost $c_i(j)$ $(i \in I, j \in [B])$ was set to $\lceil \max\{Cf(j\chi_i), 1\} \rceil$. We considered 18 different settings corresponding to each combination of parameters $B \in \{3, 5\}$, $K \in \{5, 15, 30\}$, and $\alpha \in \{0.1, 0.05, 0.01\}$. In all experiments, both C and $|I|$ were set to 100. We generated three datasets randomly from a single setting. For each dataset, the algorithms performed 100 random trials of adaptive selection, and report the average of the objective values attained by the algorithms.

Figure 2 (a) compares the average objective values for 18 settings. In this figure, each point in the coordinate corresponds to a single setting. The x-coordinate of a point shows the average objective value attained by the proposed algorithm with the stochastic continuous greedy algorithm, where the first baseline algorithm is used when the budget is not spent when the proposed algorithm terminates. The y-coordinate represents the maximum of the average objective values achieved by the two baseline algorithms. Hence, a point in the lower half indicates that the proposed algorithm outperformed both of the baseline algorithms in the corresponding setting. We can observe that the proposed algorithm outperformed the baseline algorithms in 14 settings. The difference between the objective values is large when the proposed algorithm is better, while the difference is small even when one of the baseline algorithms achieved a better objective value. This proves that the proposed algorithm is more stable than the baseline algorithms.

Figure 2 (b) compares the average objective values attained by the continuous greedy and the stochastic continuous greedy algorithms for problem (2). The stochastic continuous greedy algorithm outperformed the continuous greedy algorithm in 15 settings. Thus, we can conclude that the modification of the continuous greedy algorithm given in the stochastic continuous greedy algorithm makes a difference even on empirical performance.

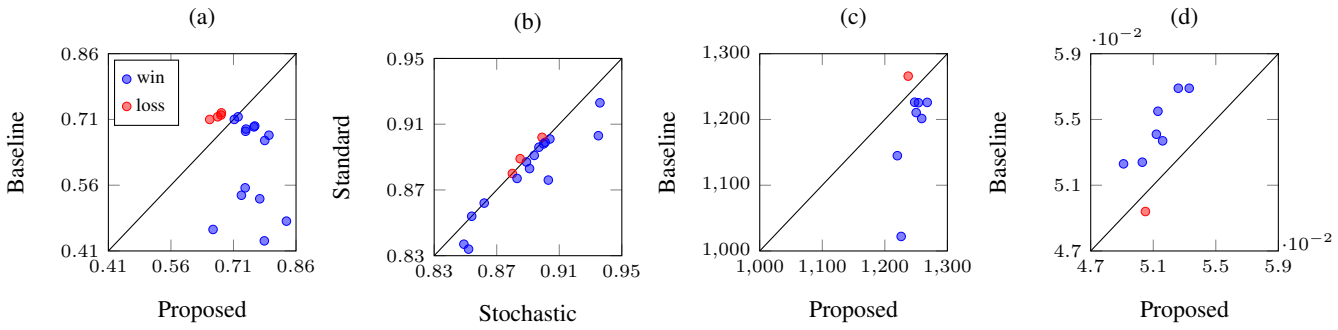

Figure 2: Summaries of experimental results. A blue (red) point denotes our proposed algorithm was better (worse) than the compared baseline algorithms in the corresponding parameter setting.

6.2 Budgeted Batch-Mode Active Learning

Next, we report the results for applying the algorithms for CSSMP to a budgeted batch-mode active learning. In this problem, we have a set D of data, wherein each data point is represented by a feature vector $x \in \mathbb{R}^d$ and is associated with a label $y \in \{-1, 1\}$. However, the labels of data are not known in advance. The purpose is to select a part of the data to label under a budget constraint so as to maximize the performance of a classifier trained from the labeled data.

In particular, we consider the following setting. The dataset D is divided into data subsets D_i ($i \in I$), and we seek to select several subsets. A subset D_i consists of data points $x_{i1}, \ldots, x_{iB}$. If subset D_i is selected, then the data points in D_i are processed sequentially from x_{i1} to x_{iB}. It is unclear how many data points are processed in advance, but we know the probability $p_i(j)$ that the processing of $x_{i1}, \ldots, x_{ij}$ is completed but that of x_{ij+1} is not. How many data points in D_i will be processed is revealed immediately after selecting D_i, and a cost $c_i(j)$ from the budget is paid if data points $x_{i1}, \ldots, x_{ij}$ will be processed. Each item in CSSMP corresponds to each subset, and the state of an item corresponds to the number of processed data points in the subset. We assume that at least one data point will be processed if D_i is selected. We consider algorithms to select subsets adaptively under the constraint that the total cost does not exceed budget C.

To obtain a better classifier, we should maximize the informativeness of the processed data points. In order to measure informativeness, we use a function derived from the Fisher information ratio of logistic regression (Hoi et al. 2006). This function is defined from an existing linear classifier parameterized by a weight vector $\beta \in \mathbb{R}^d$ and a bias term $\beta_0 \in \mathbb{R}$. Define $\eta\colon \mathbb{R}^d \to (0, 1/4]$ as $\eta(x) = \frac{1}{1+e^{\beta^\top x+\beta_0}}\left(1 - \frac{1}{1+e^{\beta^\top x+\beta_0}}\right)$. Let γ be a small positive parameter. Let $r \in [B]^I$ be the vector representing how many data points in each subset will be processed; $r(i) = j > 0$ represents that D_i is selected and data points $x_{i1}, \ldots, x_{ij}$ in D_i are processed, whereas $r(i) = 0$ means that D_i is not selected. Then, the objective function $f\colon [B]^I \to \mathbb{R}_+$ is formulated by $f(r) = \frac{1}{\gamma}\sum_{i\in I}\sum_{j=1}^{B}\eta(x_{ij}) - \sum_{i\in I}\sum_{j>r(i)}\frac{\eta(x_{ij})}{\gamma+\sum_{i'\in I}\sum_{j'=1}^{r(i')}\eta(x_{i'j'})(x_{ij}^\top x_{i'j'})^2}$. The monotonicity and the lattice-submodularity of this function follow from the monotonicity and set-submodularity of the set-function considered in Hoi et al. (2006).

For our experiments, we used the WDBC dataset (569 instances; 32 features) from the UCI machine learning repository (http://archive.ics.uci.edu/ml). We used half of the dataset for the pooled data and the other for the test data. First, we randomly selected 20 initial instances from the pooled data and learned the L_2-regularized logistic regression using the initial data to obtain β and β_0. Then, we selected additional instances from the pooled dataset by using the baseline and proposed methods for CSSMP under a budget constraint. Notice that the given labels of these instances are not used up to this point. Finally, with the labels of these instances, we again learned the classifier by using the initial and additional instances.

To implement the logistic regression, we used scikit-learn (http://scikit-learn.org). For all training of the logistic regression, the regularization parameter ρ was selected from $\{0.1, 0.5, 1.0, 2.0, 10.0\}$ by 5-fold cross-validation. All features in the dataset were standardized; it makes each feature have zero mean and unit variance. Note that we did not scale the features so that $||x||_2^2 = 1$ for improving the performance of the classifiers.

C and γ were set to 100 and 0.01 respectively. p_i ($i \in I$) was constructed randomly according to the symmetric Dirichlet distribution with parameter 1.0. To make the correlation between performance and costs, we sorted all the instances in the pool data by the objective function values, then divided the sorted instance list into the subsets. We considered eight experimental settings on B and item costs; B was selected from $\{3, 4, 5, 6\}$, and $c_i(j)$ ($i \in I$; $j \in [B]$) was set to $\lceil \max\{Cf(j\chi_i), 1\}\rceil$ or $\lceil \max\{jCf(j\chi_i)/B, 1\}\rceil$. In the same as the experiments of recommendation, we generated three datasets randomly from each setting, repeated the trial of selections 100 times, and report the average of the objective values and the error rates attained by the algorithms.

Figure 2 (c) shows the average objective values of the proposed and the baseline algorithms. The proposed algorithm showed better performances in seven settings. Figure 2 (d) shows the error rates of the trained classifiers. Note that a smaller error rate indicates better performance, and hence a

point in the upper half indicates that the proposed algorithm is better than the baseline algorithms. It can be observed that the proposed algorithm reduced the error of the classifier in seven settings.

7 Conclusion

We considered adaptive algorithms for CSSMP. In design of our proposed algorithms, we extend the framework of the contention resolution scheme, known to be useful in the maximization problem of set-submodular functions, to lattice-submodular functions. We believe these contributions to be potentially useful in other problems related to lattice-submodular functions.

Through the experiments, we verified that algorithms output better solutions compared with baseline algorithms. A disadvantage of our algorithms is their computational complexity. In particular, the continuous optimization phase (relying on the continuous greedy or the stochastic continuous greedy algorithm) is slow. There are several attempts to speed up this part (Badanidiyuru and Vondrák 2014; Chekuri, Jayram, and Vondrák 2015), and it is a future work to consider them in our algorithms.

References

Adamczyk, M.; Sviridenko, M.; and Ward, J. 2016. Submodular stochastic probing on matroids. *Mathematics of Operations Research* 41(3):1022–1038.

Ahmed, A.; Teo, C. H.; Vishwanathan, S. V. N.; and Smola, A. J. 2012. Fair and balanced: learning to present news stories. In *Proceedings of the Fifth International Conference on Web Search and Web Data Mining*, 333–342.

Asadpour, A., and Nazerzadeh, H. 2016. Maximizing stochastic monotone submodular functions. *Management Science* 62(8):2374–2391.

Badanidiyuru, A., and Vondrák, J. 2014. Fast algorithms for maximizing submodular functions. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, 1497–1514.

Călinescu, G.; Chekuri, C.; Pál, M.; and Vondrák, J. 2011. Maximizing a monotone submodular function subject to a matroid constraint. *SIAM Journal on Computing* 40(6):1740–1766.

Chekuri, C.; Jayram, T. S.; and Vondrák, J. 2015. On multiplicative weight updates for concave and submodular function maximization. In *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science*, 201–210.

Chen, Y., and Krause, A. 2013. Near-optimal batch mode active learning and adaptive submodular optimization. In *Proceedings of the 30th International Conference on Machine Learning*, 160–168.

El-Arini, K.; Veda, G.; Shahaf, D.; and Guestrin, C. 2009. Turning down the noise in the blogosphere. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 289–298.

Feldman, M.; Naor, J.; and Schwartz, R. 2011. A unified continuous greedy algorithm for submodular maximization. In *IEEE 52nd Annual Symposium on Foundations of Computer Science*, 570–579.

Feldman, M. 2013. *Maximization Problems with Submodular Objective Functions*. Ph.D. Dissertation, Technion – Israel Institute of Technology.

Fortuin, C. M.; Kasteleyn, P. W.; and Ginibre, J. 1971. Correlation inequalities on some partially ordered sets. *Communications in Mathematical Physics* 22(2):89–103.

Fujii, K., and Kashima, H. 2016. Budgeted stream-based active learning via adaptive submodular maximization. In *Advances in Neural Information Processing Systems 29*, 514–522.

Gabillon, V.; Kveton, B.; Wen, Z.; Eriksson, B.; and Muthukrishnan, S. 2013. Adaptive submodular maximization in bandit setting. In *Advances in Neural Information Processing Systems 26*, 2697–2705.

Gabillon, V.; Kveton, B.; Wen, Z.; Eriksson, B.; and Muthukrishnan, S. 2014. Large-scale optimistic adaptive submodularity. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, 1816–1823.

Golovin, D., and Krause, A. 2011. Adaptive submodularity: Theory and applications in active learning and stochastic optimization. *Journal of Artificial Intelligence Research* 42:427–486.

Gotovos, A.; Karbasi, A.; and Krause, A. 2015. Non-monotone adaptive submodular maximization. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence*, 1996–2003.

Gupta, A.; Krishnaswamy, R.; Molinaro, M.; and Ravi, R. 2011. Approximation algorithms for correlated knapsacks and non-martingale bandits. In *IEEE 52nd Annual Symposium on Foundations of Computer Science*, 827–836.

Gupta, A.; Nagarajan, V.; and Singla, S. 2017. Adaptivity gaps for stochastic probing: Submodular and XOS functions. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, 1688–1702.

Hoi, S. C. H.; Jin, R.; Zhu, J.; and Lyu, M. R. 2006. Batch mode active learning and its application to medical image classification. In *Proceedings of the Twenty-Third International Conference on Machine learning*, 417–424.

Ma, W. 2014. Improvements and generalizations of stochastic knapsack and multi-armed bandit approximation algorithms: Extended abstract. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, 1154–1163.

Yu, B.; Fang, M.; and Tao, D. 2016. Linear submodular bandits with a knapsack constraint. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, 1380–1386.

Yue, Y., and Guestrin, C. 2011. Linear submodular bandits and their application to diversified retrieval. In *Advances in Neural Information Processing Systems 24*, 2483–2491.

Ziegler, C.-N.; McNee, S. M.; Konstan, J. A.; and Lausen, G. 2005. Improving recommendation lists through topic diversification. In *Proceedings of the 14th international conference on World Wide Web*, 22–32.