Correlated Stochastic Knapsack with a Submodular Objective

Sheng Yang ✉ 📧
Northwestern University, Evanston, IL, USA

Samir Khuller ✉
Northwestern University, Evanston, IL, USA

Sunav Choudhary ✉
Adobe Research, Bangalore, India

Subrata Mitra ✉
Adobe Research, Bangalore, India

Kanak Mahadik ✉
Adobe Research, San Jose, CA, USA

──── **Abstract** ────

We study the correlated stochastic knapsack problem of a submodular target function, with optional additional constraints. We utilize the multilinear extension of submodular function, and bundle it with an adaptation of the relaxed linear constraints from Ma [Mathematics of Operations Research, Volume 43(3), 2018] on correlated stochastic knapsack problem. The relaxation is then solved by the stochastic continuous greedy algorithm, and rounded by a novel method to fit the contention resolution scheme (Feldman et al. [FOCS 2011]). We obtain a pseudo-polynomial time $(1 - 1/\sqrt{e})/2 \simeq 0.1967$ approximation algorithm with or without those additional constraints, eliminating the need of a key assumption and improving on the $(1 - 1/\sqrt[4]{e})/2 \simeq 0.1106$ approximation by Fukunaga et al. [AAAI 2019].

1 Introduction

The knapsack problem is one of the most celebrated frameworks to model profit maximization with limited resources. Though well understood in its basic form, many new variants were proposed to model and target more complicated problems. One line of variants take

randomness into consideration. Such randomness may appear on item sizes only, on item profits only, or on both in a correlated fashion. A significant body of work [4, 10, 22, 25, 26] connects knapsack problem with the field of stochastic optimization, greatly broadening the spectrum of knapsack problems while introducing various challenges for theoretical analysis. Another line of variants model diminishing returns in the profit, leading to the field of submodular optimization [27, 14, 6, 7, 12, 13], which enjoys tremendous popularity both in theory and in practice. The two lines of work are connected together into stochastic submodular optimization, another fruitful field [3, 20, 9, 15, 17, 18, 21, 31]. In this work, we follow this line, and consider a correlated stochastic knapsack problem with a submodular target function. We arrived at this problem when modeling the spot scheduling problem in Yang et al. [29] (see details in the full version [30]). A slight variant of the final problem was first proposed in Fukunaga et al. [16], trying to model "performance-dependent costs of items" in stochastic submodular optimization. This problem turns out to be a very powerful framework that applies to several other real world applications, like recommendation systems [32, 1], and batch-mode active learning [23].

1.1 Formal Problem Statement

There are n items, each takes a random $\mathsf{size}_i \in \mathbb{N}$ with probability $p_i(\mathsf{size}_i)$, and gets a reward that *corresponds* to its size. In other words, for each item i, there is a reward function $R_i : \mathbb{N} \to [M]$, such that $r_i = R_i(\mathsf{size}_i)$. (For simplicity, we define $[n]$ to be the set $\{0, 1, 2, \ldots, n\}$, and M a positive integer that upper bounds the maximum reward.) We assume each R_i to be non-decreasing, i.e., the larger an item, the more reward it deserves. We are given a budget $B \in \mathbb{N}$ for the total size of items, and wish to extract as much profit as possible. The total profit is a lattice-submodular function[1] $f : [M]^n \to \mathbb{R}^+$ on the rewards of included items, and we wish to maximize its expectation[2].

Items are put in the knapsack one by one. As soon as an item is put in the knapsack, its reward and size are revealed. We halt when the knapsack overflows (not collecting the last item's reward), and proceed to add another item otherwise. We consider adaptive policies, i.e., we can choose an item to include, observe its realized size, and make further decisions based on the realized size. At first, only the reward function and the size distribution of items are known. When the policy includes an item i, its size_i is realized, and so is its reward $r_i = R_i(\mathsf{size}_i)$. In this work, we only consider adaptive policies without cancellation, i.e., the policy can make its decision based on all the realizations it has seen so far, and the inclusion of an item is irrevocable.

For a vector $q \in [M]^n$, let $\Pr_\gamma[q]$ denote the probability that we get outcome q when running policy γ. Note this probability is with respect to the randomness both in the state of items and in the policy γ. Let $f_{\mathsf{avg}}(\gamma)$ denote $\sum_{q \in [M]^n} \Pr_\gamma[q] f(q)$, i.e., the average objective value obtained by γ. Our aim is to find a policy γ that maximizes $f_{\mathsf{avg}}(\gamma)$. We say γ is an α-approximation policy if $f_{\mathsf{avg}}(\gamma) \geq \alpha f_{\mathsf{avg}}(\gamma^*)$ for any policy γ^*.

In addition to all the above, we further require that the chosen set of items S be an independent set of a partition matroid[3] $\mathcal{I} = \{\mathcal{I}_k\}_{k \in [K]}$. This is without loss of generality as we can put each item in a separate partition, and every subset of items is valid. The

---

[1] See definition of partition matroid, submodular and lattice-submodular in Section 3.
[2] Let $S \subseteq [n]$, we sample a vector $q \in [M]^n$ as follows. Each component $q(i)$ is sampled independently. For $i \in S$, $\Pr[r_i = R_i(s)] = p_i(s)$; for $i \notin S$, $r_i = 0$ with probability 1. Denote this distribution as q_S. Then the objective is to select a (random) set $S \subseteq \mathcal{I}$ of items that maximizes $\mathbb{E}_{\theta \sim q_S}[f(\theta)]$ subject to $\sum_{i \in S} \mathsf{size}_i \leq B$.
[3] See definition of partition matroid, submodular and lattice-submodular in Section 3.

additional constraint allows us to impose conflicts between items, which is needed for the modeling in Yang et al. [29]. More importantly, it is also crucial if we are to allow the attempt to include an item that could possibly overflow the knapsack, a case unsolved and left as open problem in Fukunaga et al. [16] (see details in Section 1.3). This partition matroid is also used to ensure the correctness of our approach based on a time-indexed LP.

1.2 Our Contributions

We present a pseudo-polynomial time algorithm for the correlated stochastic knapsack problem with a submodular target function. It computes an adaptive policy for this problem which is guaranteed to achieve $(1 - 1/\sqrt{e})/2 \simeq 0.1967$ of the optimal solution on expectation. It improves on the $(1 - 1/\sqrt[4]{e})/2 \simeq 0.1106$ approximation algorithm from Fukunaga et al. [16]. Furthermore, we eliminate one key assumption in Fukunaga et al. [16] which does not allow the inclusion of any item which could possible overflow the budget.

1.3 Eliminating An Assumption in Previous Work

In Fukunaga et al. [16], the authors considered a slightly different problem. They made two assumptions, and we managed to eliminate one of them. The first assumption states that larger size means larger reward for every particular job. This assumption is reasonable for general problems and remains crucial in our analysis. The second assumption states that we will never select an item which could overflow the budget, given the realization of selected items[4]. However, for many cases, selecting such an item is a desirable choice since additional value is obtained with high probability. If we are unlucky and the size goes beyond the remaining budget, we either receive a partial value, or do not get any value at all.

1.4 Our Techniques

If the target function is linearly additive, this problem becomes the correlated stochastic knapsack problem. For this problem, Gupta et al. [22] gave an $1/8$ approximation algorithm for adaptive policies based on LP relaxation. The approximation ratio was improved to $1/(2 + \epsilon)$ by Ma [26], via a different time indexed LP formulation and a more sophisticated rounding scheme. Fukunaga et al. [16] extends the $1/8$ approximation algorithm, and achieve a $(1 - 1/\sqrt[4]{e})/2$ approximation for a case with submodular target function. This is achieved via a combination of the stochastic continuous greedy algorithm [2] (for getting a fractional solution), and the contention resolution scheme [13] (for rounding). A natural idea for improvement is to take ingredients from the $1/(2 + \epsilon)$ algorithm by Ma [26]. While the LP can be easily adapted, its rounding exhibits complicated dependencies that can be hard to analyze. We also have no luck with a direct application of the *contention resolution scheme*[8, 12, 13], a powerful technique in submodular optimization. The difficulties come in two folds. First, the original scheme is based on FKG inequality, which requires an *independently* rounded solution (possibly invalid) to start with. Any attempt to enforce our partition matroid at this step will break the whole scheme. This invalid solution is later fixed by ignoring some items from the rounded solution, and we need a way to impose the additional partition matroid constraint. Second, the ignoring step needs a critical "monotone"

---

[4] For example, suppose we are left with a remaining budget of 20 at some time, and all items have a 0.001 probability of size 21. What this assumption suggests is that none of the items are allowed to be selected.

property. At a high level, the property says that the more items you choose, the lower the probability every other items will be selected (See Section 5.1 for a rigours definition). While this may seem trivially true for any reasonable algorithm, it is not. In particular, it does not hold for Ma's algorithm [26], due to its complicated dependencies. The first difficulty is not hard: if we happen to pick two items from the same partition, we just throw the later one out. Unfortunately, this makes the second obstacle even harder. The second obstacle is overcome by designing a brand-new rounding scheme which allows the direct analysis on the correlated probability of events. In order to achieve the aforementioned monotone property, we insert phantom items to block some "time slots" even when no item is there to conflict with. Such phantom items may be of independent interest for other applications of the contention resolution scheme. This alternate way of achieving monotonicity simultaneously free our analysis from one assumption mentioned in Section 1.3, which was needed in Fukunaga et al. [16] in their proof of the monotone property. A factor of $(1 - 1/\sqrt{e})$ is lost for the continuous optimization part, and another factor of 2 is lost for rounding, leading to our $(1 - 1/\sqrt{e})/2 \simeq 0.1967$ approximation algorithm.

2 Other Related Works

Stochastic Knapsack Problem.   The stochastic version of the knapsack problem has long been studied. Kleinberg et al. [24], and Goel and Indyk [19] consider the stochastic version to maximize profit that will overflow the budget with probability at most p. However, they assume deterministic profits and special size distributions. Dean et al. [11] relax the limit on size and allow arbitrary distributions for item sizes. They investigate the gap between non-adaptive policies (the order of items to insert is fixed) and adaptive policies (allowed to make dynamic decision based on the realized size of items) and give a polynomial-time non-adaptive algorithm that approximates the optimal adaptive policy within a factor of $1/4$ in expectation. They also give an adaptive policy that approximates within a factor of $1/(3 + \epsilon)$ for any constant $\epsilon > 0$. Bhalgat et al. [4] improves on this and give a bi-criteria $(1 - \epsilon)$ algorithm by relaxing the budget by $(1 + \epsilon)$. Dean et al. [10] show that if correlation between size and reward is allowed, the problem would be PSPACE-hard. Gupta et al. [22] considered the case where the size and reward of an item can be arbitrarily correlated, and give an $1/8$ approximation. Li and Yuan [25] improved on this and get a $1/(2 + \epsilon)$ approximation with correlations and cancellation when ϵ fraction of extra space is allowed. This was further improved by Ma [26], who gets the same approximation ratio but without the budget augmentation requirement.

Submodular Maximization.   Nemhauser et al. [27] studied the problem of maximizing a monotone submodular function subject to a cardinality constraint and gave the standard greedy $(1 - 1/e)$-approximation algorithm. For the case with a matroid constraint, Fisher et al. [14] showed that the standard greedy algorithm gives a $1/2$-approximation. This was improved to $(1 - 1/e)$ by Calinescu et al. [6], via the *continuous greedy algorithm*, which was originally developed by Calinescu et al. [5] for the submodular welfare problem. In this algorithm, the target function is relaxed via an exponential multilinear-extension. Though exponential, this version can be approximately solved to arbitrary precision in polynomial time. The fractional solution is then rounded via pipage rounding [5, 6, 28] or other rounding schemes [7]. In order to generalize the problem for other constraints and non-monotone submodular functions, a general rounding framework *contention resolution scheme* was proposed [6, 12, 13]. In this framework, the rounding step happens in two phases, an

independent rounding phase followed by a pruning phase, where the second phase ensures an upper bound on the probability that an element is pruned. One line of stochastic submodular optimization [3] assumes items have stochastic states, and would like to maximize a monotone submodular function on the stochastic states, under constraints on the set of chosen items. In other words, the constraints only depend on the selection of items, but not on the stochastic states of them. This is a generalization of the stochastic knapsack problem where the size of items are deterministic. Various settings of this problem are investigated by a series of follow-up works [20, 9, 15, 17, 18, 21, 31]. Asadpour and Nazerzadeh [2] considers the maximization of a monotone lattice-submodular function. In this problem, each selected item has a stochastic state (a non-negative *real* number). The target function accepts a vector of such numbers, and satisfies lattice-submodularity (defined in Section 3). In their problem, only the states are stochastic, while the matroid constraint is on the set of selected items. Fukunaga et al. [16] pushed one step further and allowed the constraints to be dependent on the state of items, but limited the set of states to be non-negative integers.

3 Preliminary

We start with some notations. The description of the spot scheduling problem [29] can be found in the full version [30], together with its modeling and reduction to the problem we consider. In Section 3.1, we explain how we reduce and manage to eliminate one critical assumption in the previous work by Fukunaga et al. [16].

Given two d dimensional vectors $u, v \in [n]^d$, we write $u \leq v$ if the inequality holds coordinate wise, i.e. $\forall i \in [d], u(i) \leq v(i)$. Similarly, $u \vee v$ and $u \wedge v$ are defined coordinate wise: $(u \vee v)(i) = \max\{u(i), v(i)\}$, $(u \wedge v)(i) = \min\{u(i), v(i)\}$. Consider a base set $[n]$, a matroid is defined to be an independent set $\mathcal{I} \subseteq 2^n$. This independent set needs to contain $\emptyset$, and if $A \in \mathcal{I}$, so is every $A' \subseteq A$. Furthermore, if $A, B \in \mathcal{I}$ and $|A| > |B|$, then there exists an element $x \in A \setminus B$ such that $B \cup x$ is in $\mathcal{I}$. Particularly, for a *partition matroid* $\{\mathcal{I}_k\}_{k \in [K]}$ where $\mathcal{I}_i \cap \mathcal{I}_j = \emptyset, \forall i \neq j$, its independent set $\mathcal{I}$ is $\{S | \forall k, S \cap \mathcal{I}_k \leq 1\}$.

A function $f : 2^d \to \mathbb{R}$ is *submodular* if for every $A, B \subseteq [d]$, $f(A) + f(B) \geq f(A \cup B) + f(A \cap B)$. An equivalent definition is that for every $A \subseteq B \subseteq [d]$ and $e \in [d]$, $f(A \cup \{e\}) - f(A) \geq f(B \cup \{e\}) - f(B)$. This definition is generalized to a domain of $[n]^d$, where function $f : [n]^d \to \mathbb{R}^+$ is called *lattice-submodular* if $f(u) + f(v) \geq f(u \wedge v) + f(u \vee v)$ holds for all $u, v \in [n]^d$. 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 $i \in I$, where $\chi_i \in \{0,1\}^n$, and only the i-th coordinate is 1. That is, $f(u + \chi_i) - f(u) \geq f(v + \chi_i) - f(v)$ does not necessarily hold for all $u, v \in [n]^d$ such that $u \leq v$ and $i \in [d]$ even if f is *lattice-submodular*. Function $f : [n]^d \to \mathbb{R}^+$ is called *monotone* if $f(u) \leq f(v)$ for all $u \leq v$.

3.1 Reduction and Eliminating an Assumption

In order to eliminate the second assumption mentioned in Section 1.3, we introduce the notion of a "size cap". For each item i and a size cap b, we define an item (i, b), where $p_{(i,b)}(s) = \pi_i(s)$ when $s < b$; $p_{(i,b)}(s) = \sum_{s' \geq s} \pi_i(s')$ when $s = b$; and 0 otherwise. The new reward function is exactly $R_{(i,b)}(\cdot)$.

We will be using a time-indexed LP formulation following Ma [26]. Instead of making a decision at each time step, we do it at each remaining size level. When there is enough room, we take item i itself into consideration. If the remaining size b is small, we are not able to get more reward for an item than when it has a size of b. Therefore, instead of trying to include

the original item i, we include item (i, b), which is item i with size cap b. Obviously, we can include each item at most once. To achieve this, we impose a partition matroid $\{\mathcal{I}_i\}_{i \in [K]}$ on the items, where $\mathcal{I}_i = \{(i, b) | \forall b\}$. For the remainder of this paper, we view each (i, b) as an item, and the conflict between them is captured by the partition matroid.

4   Continuous Optimization Phase

Like most submodular maximization problems, our algorithms consists of two phases, a continuous optimization phase and a rounding phase. In this section, we describe the former.

4.1   Target Function

Given a lattice-submodular function $f : [M]^n \to \mathbb{R}^+$ and a distribution q_S of elements in set $S \subseteq [n]$, we define a set-submodular function $\bar{f} : 2^n \to \mathbb{R}_+$, where $\bar{f}(S) := \mathbb{E}_{r \sim q_S}[f(r)]$. This $\bar{f}$ is guaranteed to be a monotone set-submodular function (See proof in [2]). Suppose the final selected (random) set is S, the value we are interested in would be $\mathbb{E}[\bar{f}(S)]$. Let $\bar{x}$ be a vector, where $\bar{x}(i)$ denotes the probability that item i is in S. Using the well-established multi-linear extension, we define $\bar{F} : 2^n \to \mathbb{R}^+$, where $\bar{F}(\bar{x}) = \sum_{S \subseteq [n]} \prod_{i \in S} \bar{x}_i \prod_{i' \notin S} (1 - \bar{x}_{i'}) \bar{f}(S)$. This is the target function we are maximizing. Evaluating the function $\bar{F}$ can take exponential time, but it can be approximated within a multiplicative factor of $(1 + \epsilon)$ for any constant $\epsilon > 0$. For simplicity, we assume $\bar{F}(\bar{x})$ can be evaluated exactly in this paper, which is standard in the literature (e.g. see [6]).

4.2   Stochastic Knapsack Exponential Constraints

The exponential and polynomial constraints on $\bar{x}$ are adapted from Ma [26]. A group of exponential sized constraints describes the problem exactly. They are then relaxed to have a polynomial size, losing a factor of 2. For ease of notation, we follow Ma [26] and view an stochastic item i as an equivalent Markovian bandit, a special one that can force us to keep pulling it for a certain period of time. We use state $u_i(k, s)$ to indicate that arm i has been pulled k times, and the corresponding item has size s. From its initial state ρ_i, a single pull would decide the size s of this job, and move to state $u_i(1, s)$ respectively. We are then forced to keep pulling this arm (we will be using arm and item interchangeably) for the next $s - 1$ steps, and the last of such pulls moves us to its termination state $\emptyset_i$, and we can pull a new arm. Denote the probability of moving from state u to state v with $p_{u,v}$. After the first pull of item i, it moves to state $u_i(1, s)$ (having size s) with probability $p_{\rho_i, u_i(1,s)} = p_i(s)$. Therefore, if $k < s$, a pull will transit it to state $u_i(k + 1, s)$ with probability $p_{u_i(k,s), u_i(k+1,s)} = 1$. Otherwise, when $k = s$, transit to state $\emptyset_i$ with probability $p_{u_i(k,s), \emptyset_i} = 1$, and we are allowed to pull a new arm.

Let π be a vector representing a joint state/node, where π_i denotes the state on item i. Let $S_i = \{u_i(*, *)\} \cup \{\rho_i, \emptyset_i\}$ for all $i \in [n]$, the set of all states for arm i, and $\tilde{\mathcal{S}} = S_1 \times \cdots \times S_n$, the set of all possible (maybe invalid) joint states. Let $\mathcal{S}' = \{\pi \in \tilde{\mathcal{S}} | \exists i \neq j, \pi_i \notin \{\rho_i, \emptyset_i\}, \pi_j \notin \{\rho_j, \emptyset_j\}\}$, the set of states where at least two arms are in the middle of processing at the same time, and $\mathcal{S}'' = \{\pi \in \tilde{\mathcal{S}} | \pi_i \neq \rho_i \text{ and } \pi_j \neq \rho_j, i, j \in \mathcal{I}_k \text{ for some } k\}$, the set of states where some conflicting arms (due to the partition matroid) have been started. Define $\mathcal{S} := \tilde{\mathcal{S}} \setminus (\mathcal{S}' \cup \mathcal{S}'')$, which is the set of all valid states. Let $I(\pi) = \{i | \pi_i \neq \emptyset_i\}$, the set of arms that could be played from state π. Let π^u denote the joint node where the component corresponding to u is replaced by u (note u can correspond to only one component). Let $y_{\pi,t}$

be the probability that we are at state π at time t, and $z_{\pi,i,t}$ the probability that we pulled arm i at time t, when the current state was π. Recall B is the total budget, we have the following basic constraints.

$$\sum_{i\in I(\pi)} z_{\pi,i,t} \le y_{\pi,t} \qquad\qquad \pi \in \mathcal{S}, t \in [B] \tag{1}$$

$$z_{\pi,i,t} = y_{\pi,t} \qquad\qquad \pi \in \mathcal{S}, i : \pi_i \in S_i \setminus \{\rho_i, \emptyset_i\}, t \in [B] \tag{2}$$

$$z_{\pi,i,t} \ge 0 \qquad\qquad \pi \in \mathcal{S}, i \in [n], t \in [B] \tag{3}$$

Let $\mathcal{A}_i = \{\pi \in \mathcal{S} : \pi_i \notin \{\rho_i, \emptyset_i\}\}$, the joint node when arm i is in the middle of processing. Note $\mathcal{A}_i$ and $\mathcal{A}_j$ are disjoint for $i \ne j$. We call arm i the *active* arm. Let $\mathcal{A} = \bigcup_{i\in[n]} \mathcal{A}_i$, the set of all states where some arm is *active*. For a state $\pi \in \mathcal{S}$, let $\mathcal{P}(\pi)$ denote the subset of $\mathcal{S}$ that would transit to π with no play, which could happen when some arms turned inactive automatically: if $\pi \notin \mathcal{A}$, then $\mathcal{P}(\pi) = \{\pi\} \cup (\bigcup_{i\notin I(\pi)} \{\pi^u | u \in S_i \setminus \{\rho_i\}\})$; if $\pi \in \mathcal{A}$, then $\mathcal{P}(\pi) = \emptyset$. Suppose u corresponds to coordinate i, define $\mathsf{Par}(u) = \{v \in S_i : p_{v,u} > 0\}$, the nodes that have a positive probability of transitioning to u. Then y-variables are updated as follows:

$$y_{(\rho_1,\dots,\rho_n),0} = 1 \tag{4}$$

$$y_{\pi,0} = 0, \qquad\qquad \pi \in \mathcal{S} \setminus \{(\rho_1, \cdots, \rho_n)\} \tag{5}$$

$$y_{\pi,t} = \sum_{\pi'\in\mathcal{P}(\pi)} \left(y_{\pi',t-1} - \sum_{i\in I(\pi')}^{n} z_{\pi',i,t-1} \right) \qquad\qquad t > 0, \pi \in \mathcal{S} \setminus \mathcal{A} \tag{6}$$

$$y_{\pi,t} = \sum_{\rho_i\in\mathsf{Par}(\pi_i)} \left(\sum_{\pi'\in\mathcal{P}(\pi^{\rho_i})}^{n} z_{\pi',i,t-1} \right) \cdot p_{\rho_i,\pi_i}, \qquad\qquad t > 0, i \in [n], \pi \in \mathcal{A}_i, \pi_i \in \{u_i(1,*)\} \tag{7}$$

$$y_{\pi,t} = \sum_{u\in\mathsf{Par}(\pi_i)} z_{\pi^u,i,t-1} \cdot p_{u,\pi_i}, \qquad\qquad t > 0, i \in [n], \pi \in \mathcal{A}_i, \pi_i \notin \{u_i(1,*)\} \tag{8}$$

Equation (6) updates $y_{\pi,t}$ for $\pi \notin \mathcal{A}$, i.e. joint nodes with no active arms. Such a joint node π can only come from a no-play from a joint node in $\mathcal{P}(\pi)$. Equations (7) and (8) update $y_{\pi,t}$ for $\pi \in \mathcal{A}$. To get to the joint node π, we must have played arm i in previous step(s). In Equation (7), we consider the case if π_i is one of $u_i(1,*)$. We were at ρ_i right before, so it is possible that in the last step, we switched to π^{ρ_i} from some joint node in $\mathcal{P}(\pi^{\rho_i})$ without playing an arm. In Equation (8), we consider other cases, in which arm i was played at time $t-1$. These equations guarantee that at each time step, $y_{*,t}$ form a distribution, i.e. $\sum_{\pi\in\mathcal{S}} y_{\pi,t} = 1$. Combining this with Equation (1), we get $\sum_{\pi\in S} \sum_{i\in I(\pi)} z_{\pi,i,t} \le 1, \forall t \in [B]$. Equations (1)–(8) form the exponential constraints. We also need to relate these constraints with $\bar{x}$ (recall $\bar{x}(i)$ is the probability that item i is included): $\bar{x}(i) = \sum_t \sum_{u\in S_i} \sum_{\pi\in\mathcal{S}:\pi_i=u} z_{\pi,i,t}$, which is the last missing piece for our exponential program, denoted as ExpP.

4.3 Stochastic Knapsack Polynomial Constraints

Obviously, we cannot solve this exponential program directly in polynomial time. In order to solve it, we relax the exponential program by disassemble the joint distribution of items. Let $s_{u,t}$ be the probability that arm i is on node u at the beginning of time t. Let $x_{u,t}$ be the probability that we pull an arm on node u at time t. Suppose $\mathcal{S} = \bigcup_i S_i$, we have the following constraints between $x_{u,t}$ and $s_{u,t}$.

$$x_{u,t} \le s_{u,t} \qquad\qquad u \in \mathcal{S}, t \in [B] \tag{9}$$

$$x_{u,t} = s_{u,t} \qquad\qquad u \in \bigcup_{i\in[n]} S_i \setminus \{\rho_i, \emptyset_i\}, t \in [B] \tag{10}$$

$$x_{u,t} \ge 0 \qquad\qquad u \in \mathcal{S}, t \in [B] \tag{11}$$

$$\sum_{u\in\mathcal{S}} x_{u,t} \le 1 \qquad\qquad t \in [B] \tag{12}$$

We also need constraints (13) for the partition matroid of arms (recall $\mathcal{I}_k$ is a partition), and the state transition constraints (14)–(16).

$$\sum_{i \in \mathcal{I}_k} s_{\rho_i,0} \le 1, \quad \forall \mathcal{I}_k \qquad\qquad s_{\rho_i,0} \ge 0, \quad i \in [n] \tag{13}$$

$$s_{u,0} = 0 \qquad\qquad u \in \mathcal{S} \setminus \{\rho_1, \cdots, \rho_n\} \tag{14}$$

$$s_{\rho_i,t} = s_{\rho_i,t-1} - x_{\rho_i,t-1} \qquad\qquad t > 0, i \in [n] \tag{15}$$

$$s_{u,t} = \sum_{v \in \mathsf{Par}(u)} x_{v,t-1} \cdot p_{v,u} \qquad\qquad t > 0, u \in \mathcal{S} \setminus \{\rho_{1,\dots,\rho_n}\} \tag{16}$$

Relating these constraints with $\bar{x}$: $\bar{x}(i) = \sum_t \sum_{u \in \mathcal{S}_i} x_{u,t}$, we get the polynomial program PolyP. For any program $P \in \{\mathsf{PolyP}, \mathsf{ExpP}\}$, let OPT_P denote its optimal value.

4.4   Relating between the Exponential and the Polynomial Constraints

This was given in Ma [26], and we re-state for completes without proof. The direction from ExpP to PolyP is trivial.

▶ **Theorem 1** (reformation of Lemma 2.3 from Ma [26])**.** *Given a feasible solution* $\{z_{\pi,i,t}\}, \{y_{\pi,t}\}$ *to* ExpP*, we can construct a solution to* PolyP *with the same objective value by setting* $x_{u,t} = \sum_{\pi \in \mathcal{S}:\pi_i = u} z_{\pi,i,t}$, $s_{u,t} = \sum_{\pi \in \mathcal{S}:\pi_i = u} y_{\pi,t}$ *for all* $i \in [n]$, $u \in [0,1]$, $t \in B$. *Thus, the feasible region of* PolyP *is a projection of that of* ExpP *onto a subspace and* $\mathsf{OPT}_{\mathsf{ExpP}} \le \mathsf{OPT}_{\mathsf{PolyP}}$.

For the other direction, we construct a solution $\{z_{\pi,i,t}, y_{\pi,t}\}$ of ExpP from a solution $\{x_{u,t}, s_{u,t}\}$ of PolyP, which obtains half its objective value. It will satisfy

$$\sum_{\pi \in \mathcal{S}:\pi_i = u} z_{\pi,i,t} = \frac{x_{u,t}}{2} \quad i \in [n], u \in \mathcal{S}_i, t \in [B].$$

We define specific $\{z_{\pi,i,t}, y_{\pi,t}\}$ over B iterations. On iteration t:

- Compute $y_{\pi,t}$ for all $\pi \in \mathcal{S}$.
- Define $\tilde{y}_{\pi,t} = y_{\pi,t}$ if $\pi \notin \mathcal{A}$, and $\tilde{y}_{\pi,t} = y_{\pi,t} - \sum_{a \in A} z_{\pi,i,t}$ if $\pi \in \mathcal{A}_i$ for some $i \in [n]$ (if $\pi \in \mathcal{A}_i$, then $\{z_{\pi,i,t} : a \in A\}$ is already set in a previous iteration).
- For all $i \in [n]$, define $f_{i,t} = \sum_{\pi \in \mathcal{S}:\pi_i = \rho_i} \tilde{y}_{\pi,t}$.
- For all $i \in [n]$, $\pi \in \mathcal{S}$ such that $\pi_i = \rho_i$, and $a \in A$, set $z^a_{\pi,i,t} = \tilde{y}_{\pi,t} \cdot \frac{1}{2} \cdot \frac{x_{\rho_i,t}}{f_{i,t}}$.
- For all $i \in [n]$, $\pi \in \mathcal{S}$ such that $\pi_i = \rho_i$ and $\pi_j \in \{\rho_j, \phi_j\}$ for $j \ne i$, define $g_{\pi,i,t} = \sum_{\pi' \in \mathcal{P}(\pi)} z_{\pi',i,t}$.
- For all $i \in [n]$, $u \in \mathcal{S}_i \setminus \{\rho_i\}$, $\pi \in \mathcal{S}$ such that $\pi_i = u$, and $a \in A$, set $z^a_{\pi,i,t+\mathsf{depth}(u)} = g_{\pi^{\rho_i},i,t} \cdot (x^a_{u,t+\mathsf{depth}(u)})/x_{\rho_i,t}$.

4.5   Solve the Continuous Optimization Problem

In order to solve PolyP, we follow Fukunaga et al. [16] and use the Stochastic Continuous Greedy algorithm. This algorithm maximizes the multi-linear extension G of a monotone set-submodular function g over a solvable downward-closed polytope. A polytope $\mathcal{P} \subseteq [0,1]^{\mathcal{N}}$ is considered *solvable* if we can find an algorithm to optimize linear functions over it, and downward-closed if $x \in \mathcal{P}$ and $0 \le y \le x$ imply $y \in \mathcal{P}$. In our case, $\mathcal{P}$ is solvable due to its linearity, and that solving a linear program falls in polynomial time. Note $\mathcal{P}$ is down-monotone. The algorithm involves a controlling parameter called *stopping time*. For a stopping time $0 < b \le 1$, the algorithm outputs a solution x such that $x/b \in \mathcal{P}$, while $G(x) \ge (1 - e^{-b} - O(n^3\delta)) \max_{y \in Q} G(y)$, where n is the size of the set over which g is defined and δ is the step size used in the algorithm. Here $\mathcal{P}$ is assumed to include the characteristic vector of every singleton set.

▶ **Theorem 2** (reformation of Theorem 3 from Fukunaga et al. [16]). *If the stochastic continuous greedy algorithm with stopping time $b = 1/2 \in (0,1]$ and step size $\delta = o(|I|^{-3})$ is applied to program PolyP, then the algorithm outputs a solution $x \in b\mathcal{P}$ such that $\bar{F}(\bar{x}) \geq (1 - e^{-b} - o(1))f_{avg}(\pi^*) \simeq 0.3935 f_{avg}(\pi^*)$ for any adaptive policy π^*.*

5   Rounding Phase

Now that we have a fractional solution x, we proceed to round it to an integral policy (notice the fractional solution has already been scaled by a factor of 2). We need a variant of the contention resolution scheme that was introduced as a general framework for designing rounding algorithms that maximizes expected submodular functions ([8, 12, 13]). The variant is an extension from a set submodular function to a lattice-submodular function, first introduced in Fukunaga et al. [16]. We include its definition here for self-containment.

5.1   Contention Resolution Scheme

A contention resolution scheme (CRS) accepts a *pairwise independently* rounded solution which may violate some constraints, and fixes it without losing too much on expectation. Let $f : [B]^n \to \mathbb{R}_+$ be a monotone lattice-submodular function and the probability distribution $q_i : [B] \to [0,1]$ on $[B]$ be given for each $i \in \{1, \ldots, n\}\}$. We write $v \sim q$ if $v \in [B]^n$ is a random vector such that, for each $i \in \{1, \ldots, n\}$, the corresponding component $v(i)$ is determined independently as $j \in [B]$ with probability $q_i(j)$. This is the independently rounded solution we feed into a CRS. Let $\mathcal{F} \subseteq [B]^n$ be a downward-closed subset of $[B]^n$, and let $\alpha \in [0,1]$. We have the following definition for a α-CRS, its monotonicity, and one key property.

▶ **Definition 3** (α-Contention Resolution Scheme (α-CRS)). *A mapping $\psi : [B]^n \to \mathcal{F}$ is an α-CRS with respect to q if it satisfies:*
1. *$\psi(v)(i) \in \{v(i), 0\}$ for each $i \in [n]$;*
2. *if $v \sim q$, then $\Pr[\psi(v)(i) = j | v(i) = j] \geq \alpha$ holds for all $i \in I$ and $j \in B$. The probability is based on randomness both in v and in ψ when ψ is randomized.*

▶ **Definition 4** (monotone α-CRS). *An α-CRS ψ is considered* monotone, *if, for each $u, v \in [B]^n$ such that $u(i) = v(i)$ and $u \leq v$, $\Pr[\psi(u)(i) = u(i)] \geq \Pr[\psi(v)(i) = v(i)]$ holds. The probability is based only on the randomness of ψ.*

▶ **Lemma 5** (Theorem 4 from Fukunaga et al. [16]). *If ψ is a monotone α-CRS with respect to q, then $\mathbb{E}_{v \sim q}[f(\psi(v))] \geq \alpha \mathbb{E}_{v \in q}[f(v)]$.*

5.2   Rounding Algorithm

To fit in the contention resolution scheme, we need to first round everything independently. This means for each pair (i, t), item i is scheduled at time t with probability $x_{\rho_i, t}$. Now we have a set $R' = \{(i, t)\}$ of proposed item time pairs. We sort the set according to t, and include the items one by one. Intuitively, for a pair (i, t), we will only include item i if time t is available and does not invalidate the solution, i.e. each item is scheduled at most once, and at most one item from each partition. After including it in our solution, we get its realized size, and mark the corresponding time slots unavailable.

The main problem of this naive approach is that it does not exhibit monotonicity, which is a subtle but critical requirement for a CRS. To fix it, we schedule *phantom item i* even when we cannot fit it. We *simulate* its inclusion, and sample its size size_i should it be included. We

also mark those time slots corresponding to this *phantom item* unavailable, even when they are actually unoccupied. This seemingly wasteful step ensures that the rounding scheme is monotone. The final rounding algorithm is described in Algorithm 1.

■ **Algorithm 1** Rounding Algorithm.

---

1 foreach *pair* (i, t) **do**
2 $\quad$ Sample (i, t) with probability $x_{\rho_i, t}$, and gets $\emptyset$ otherwise;
3 $\quad$ **if** *not get* $\emptyset$ **then** $\quad I \leftarrow I \cup \{(i, t)\}$;
4 Sort I according to a non-decreasing ordering of t, break ties uniformly at random;
5 $C = 0, S = \emptyset$, mark all times slots available;
6 for $(i, t) \in I$ **do**
7 $\quad$ **if** *time slot t is available* **and** *item i does not violate constraints* **then**
8 $\quad\quad$ Include item i and observe s_i;
9 $\quad$ **else**
10 $\quad\quad$ Simulate including item i, and observe s_i;
11 $\quad$ Mark time slots from t to $t + s_i$ unavailable;

The remaining of this section is devoted to proving the following theorem, which combined with Theorem 2 leads to our main result.

▶ **Theorem 6.** *Let π denote Algorithm 1, and x denote the solution we get from* PolyP*. Then* $f_{avg}(\pi) \geq \bar{F}(\bar{x})/2$.

To prove Theorem 6, we define two mappings $\sigma(\cdot)$ and $\omega(\cdot)$, where the first corresponds (roughly) to the step that maps x to I in Algorithm 1, and $\omega(\cdot)$ corresponds to the mapping (CRS) from set I to the final output. The mapping $\sigma(\bar{x})$ receives a real vector $\bar{x} \in [0, 1]^n$ and returns a random vector $v \in [B]^n$. From each partition $\mathcal{I}_k$, we pick at most one i, each $i \in \mathcal{I}_k$ is picked with probability $\bar{x}(i)$. If it is picked, the i-th component $v(i)$ independently takes value j with probability $p_i(j)$, and 0 otherwise, which happens with probability $1 - \sum_j p_i(j)$. This captures the construction of set I (only the item part, note $\Pr[\sigma(x)(i) > 0] = \Pr[\exists t, \text{s.t.} (i, t) \in I]$), together with the random outcome of the item. The mapping $\omega(\cdot)$ maps $v \in [B]^n$ to $w \in [B]^n$. To mimic Algorithm 1, we first assign time value $t(i)$ to each component $v(i)$, according to $x_{\rho_i, t}$. Based on $t(i)$, we form a precedence ordering $\prec$ between i after random tie breaking (a random tie breaking is crucial). Then, we set $\omega(v)(i) = 0$ if there exists a component $j \prec i$ such that $t(j) \leq t(i) < t(j) + v(i)$, and $w(v)(i) = v(i)$ otherwise. We can observe that given input x, Algorithm 1 outputs exactly $\omega(\sigma(x))$ if the random realized sizes of items are the same. In order to prove Theorem 6, we need the following two lemmas. The first, whose proof in Fukunaga [16], corresponding to the independent rounding step, and the second corresponding to the CRS step.

▶ **Lemma 7.** $\mathbb{E}[f(\sigma(x))] \geq \bar{F}(\bar{x})$ *holds for any $x \in P$.*

▶ **Lemma 8.** ω *is a $1/2$-CRS with respect to $\bar{x}$.*

▶ **Lemma 9.** *The $1/2$-CRS ω is monotone.*

Lemma 7 is trivially true by the definition of $\bar{F}(\bar{x})$, which is the common starting point of contention resolution scheme. We first prove ω is a $1/2$-CRS.

Proof of Lemma 8. Recall there are two properties needed for an α-CRS. The first property is obviously correct due to the definition of $\omega(\cdot)$. The second property needs to prove $\Pr[\omega(v)(i) = j | v(i) = j] \geq 1/2$. In the language of the rounding algorithm, let $\mathsf{Drop}_{i,t}$ denotes the event (respect to the randomness in ω and v) that we drop the pair (i, t). It is the same as proving

$$\Pr[\mathsf{Drop}_{i,t} | \text{item } i \text{ is selected at time } t] \leq \frac{1}{2}.$$

Due to the way we round the solution, item i may be included more than once (at different times), and more than one item from the same partition may be included. Consider an item j at time t' (maybe the same as i) that could affect the pruning of item i at time t. Define $(j, t') \prec (i, t)$ if $t' < t$, or $t' = t$ and $j \prec i$. It is clear that (j, t') will affect (i, t) if and only if $(j, t') \prec (i, t)$ We slightly abuse notation, and let $\mathsf{Drop}_{i,t}(j)$ denote the probability that the item j *can* causes the drop out of item i if a copy of it is scheduled at time t. Note this does not depend on whether item i is scheduled on t or not. We have:

▶ **Lemma 10.**

$$Drop_{i,t}(j) \leq \frac{1}{2} \sum_{u \in \{\emptyset_j\} \cup \{u_j(*,*)\}} x_{u,t} + \frac{1}{2} x_{\rho_j,t}.$$

Proof of Lemma 10. There are two cases and we bound the probability of dropping in each case.
Case 1. j belongs to the same partition as i,
Case 2. j belongs to a different partition.

For the first case, the probability that it makes (i, t) invalid is

$$\mathsf{Drop}_{i,t}(j) \leq \frac{1}{2}(s_{\rho_j,0} - s_{\rho_j,t}) + \Pr[\text{item } j \text{ is considered before } i] \cdot \frac{1}{2} x_{\rho_j,t}$$
$$\leq \frac{1}{2} \sum_{u \in \{\emptyset_j\} \cup \{u_j(*,*)\}} x_{u,t} + \frac{1}{2} x_{\rho_j,t}.$$

The first term is the probability that at least one item j is scheduled before time t. Note this is actually an union bound due to our independent rounding. The second term is the probability that it is scheduled at time t, but will invalidate i since $j \prec i$. The second equality comes from the fact that if item j is scheduled some time before t, then it must be at some state at time t that is not the starting state ρ_j. In other words, either the end state $\emptyset_j$ or some transient state $u_j(*,*)$.

For the second case, fix j, it can only drop i if it marked time slot t unavailable. The probability is

$$\mathsf{Drop}_{i,t}(j) \leq \frac{1}{2} \sum_{t'=1}^{t-1} x_{\rho_j,t'} \cdot \Pr[\mathsf{size}_j \geq t - t'] + \Pr[\text{item } j \text{ is considered before } i] \cdot \frac{1}{2} x_{\rho_j,t}$$
$$\leq \frac{1}{2} \sum_{t'=1}^{t-1} x_{\rho_j,t'} \cdot \Pr[\mathsf{size}_j \geq t - t'] + \frac{1}{2} x_{\rho_j,t}.$$

The first term is a summation of all the possible starting point of job j, times the probability that it will mark time slot t unavailable. Note this is also a union bound since there can be more than one copy of item j due to independent rounding. The second term is the probability that item j is also scheduled at time t, but is considered before i, i.e. $j \prec i$, which marks time slot t unavailable for i. We focus on the first term,

$$\sum_{t'=1}^{t-1} x_{\rho_j,t'} \cdot \Pr[\mathsf{size}_j \geq t - t'] = \sum_{t'=1}^{t-1} \sum_{\tau=t-t'}^{B-t} x_{\rho_j,t'} \Pr[\mathsf{size}_j = \tau] = \sum_{t'=1}^{t-1} \sum_{\tau=t-t'}^{B-t} x_{u_j(1,\tau),t'+1}$$

$$= \sum_{t'=1}^{t-1} \sum_{\tau=t-t'}^{B-t} x_{u_j(t-t',\tau),t} \leq \sum_{u \in \{\emptyset_j\} \cup \{u_j(*,*)\}} x_{u,t}.$$

The last inequality holds because the index set of the summation on the left is a subset of that on the right. ◀

Therefore, the total probability that item i is blocked by any item is upper bounded by the union bound:

$$\mathsf{Drop}_{i,t} = \sum_{j)} \mathsf{Drop}_{i,t}(j) \leq \frac{1}{2} \sum_j \sum_{u \in \{\emptyset_j\} \cup \{u_i(*,*)\}} x_{u,t} + \frac{1}{2} \sum_{j \in [n]} x_{\rho_j,t}$$

$$\leq \frac{1}{2} \sum_{j \in [n]} \sum_{u \in \{\emptyset_j\} \cup \{u_i(*,*)\}} x_{u,t} + \frac{1}{2} \sum_{j \in [n]} x_{\rho_j,t} \leq \frac{1}{2}(1 - \sum_{j \in n} x_{\rho_j,t}) + \frac{1}{2} \sum_{j \in [n]} x_{\rho_j,t} = \frac{1}{2}. \quad ◀$$

Lastly, we show ω is monotone in the full version [30]. With everything ready, we can now prove Theorem 6, which combined with Theorem 2 leads to the main claim.

Proof of Theorem 6. The output r of Algorithm 1 satisfies $\mathbb{E}[f(r)] = \mathbb{E}[f(\omega(\sigma(x)))]$, and its feasibility is guaranteed by the algorithm. By Lemma 8 and Lemma 9, ω is a monotone $1/2$-CRS with respect to q, where q is the probability defined in Lemma 8. Moreover, $\sigma(x) \sim q$ holds. By Lemma 5, $\mathbb{E}[f(\omega(\sigma(x)))] \geq \mathbb{E}[f(\sigma(x))]/2$. Using Lemma 7, we get $f_{\mathrm{avg}}(\pi) = \mathbb{E}[f(r)] = \mathbb{E}[f(\omega(\sigma(x)))] \geq \mathbb{E}[f(\sigma(x))]/2 \geq \bar{F}((x))/2.$ ◀

6   Conclusion

We consider the well studied correlated stochastic knapsack problem, generalizing its target function with submodularity to capture diminishing returns. An extra partition matroid constraint is added to generalize it and resolve an open question raised in a previous work to eliminate an assumption. We also make improvement on the approximation ratio. There is still a gap of 2 comparing to the variant with linear target function and we leave it as an open problem to close the gap.

——— **References** ———

1 Amr Ahmed, Choon Hui Teo, S.V.N. Vishwanathan, and Alex Smola. Fair and balanced: Learning to present news stories. In *Proceedings of the Fifth ACM International Conference on Web Search and Data Mining*, WSDM '12, pages 333–342, New York, NY, USA, 2012. Association for Computing Machinery. `doi:10.1145/2124295.2124337`.

2 Arash Asadpour and Hamid Nazerzadeh. Maximizing stochastic monotone submodular functions. *Management Science*, 62(8):2374–2391, 2016. `doi:10.1287/mnsc.2015.2254`.

3 Arash Asadpour, Hamid Nazerzadeh, and Amin Saberi. Stochastic submodular maximization. In *Proceedings of the 4th International Workshop on Internet and Network Economics*, WINE '08, pages 477–489, Berlin, Heidelberg, 2008. Springer-Verlag. `doi:10.1007/978-3-540-92185-1_53`.

4 Anand Bhalgat, Ashish Goel, and Sanjeev Khanna. Improved approximation results for stochastic knapsack problems. In *Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete Algorithms*, pages 1647–1665. SIAM, 2011.

5   Gruia Calinescu, Chandra Chekuri, Martin Pál, and Jan Vondrák. *Maximizing a Submodular Set Function Subject to a Matroid Constraint (Extended Abstract)*, pages 182–196. Integer Programming and Combinatorial Optimization. Springer Berlin Heidelberg, 2007. `doi:10.1007/978-3-540-72792-7_15`.

6   Gruia Calinescu, Chandra Chekuri, Martin Pál, and Jan Vondrák. Maximizing a monotone submodular function subject to a matroid constraint. *SIAM Journal on Computing*, 40(6):1740–1766, 2011. `doi:10.1137/080733991`.

7   Chandra Chekuri, Jan Vondrak, and Rico Zenklusen. Dependent randomized rounding for matroid polytopes and applications. *CoRR*, 2009. `arXiv:0909.4348`.

8   Chandra Chekuri, Jan Vondrák, and Rico Zenklusen. Submodular function maximization via the multilinear relaxation and contention resolution schemes. *SIAM Journal on Computing*, 43(6):1831–1879, 2014. `doi:10.1137/110839655`.

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

10   Brian C Dean, Michel X Goemans, and Jan Vondrák. Adaptivity and approximation for stochastic packing problems. In *ACM Symposium on Parallelism in Algorithms and Architectures*, pages 395–404. SIAM, 2005.

11   Brian C Dean, Michel X Goemans, and Jan Vondrák. Approximating the stochastic knapsack problem: The benefit of adaptivity. *Mathematics of Operations Research*, 33(4):945–964, 2008.

12   Moran Feldman. *Maximization Problems with Submodular Objective Functions*. Ph.d. dissertation,, Technion - Israel Institute of Technology, 2013.

13   Moran Feldman, Joseph Naor, and Roy Schwartz. A unified continuous greedy algorithm for submodular maximization. In *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, October 2011. `doi:10.1109/focs.2011.46`.

14   M. L. Fisher, G. L. Nemhauser, and L. A. Wolsey. *An analysis of approximations for maximizing submodular set functions-II*, pages 73–87. Mathematical Programming Studies. Springer Berlin Heidelberg, 1978. `doi:10.1007/bfb0121195`.

15   Kaito Fujii and Hisashi Kashima. Budgeted stream-based active learning via adaptive submodular maximization. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL: `https://proceedings.neurips.cc/paper/2016/file/07cdfd23373b17c6b337251c22b7ea57-Paper.pdf`.

16   Takuro Fukunaga, Takuya Konishi, Sumio Fujita, and Ken ichi Kawarabayashi. Stochastic submodular maximization with performance-dependent item costs. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33:1485–1494, 2019. `doi:10.1609/aaai.v33i01.33011485`.

17   Victor Gabillon, Branislav Kveton, Zheng Wen, Brian Eriksson, and S. Muthukrishnan. Adaptive submodular maximization in bandit setting. In C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013. URL: `https://proceedings.neurips.cc/paper/2013/file/f4573fc71c731d5c362f0d7860945b88-Paper.pdf`.

18   Victor Gabillon, Branislav Kveton, Zheng Wen, Brian Eriksson, and S. Muthukrishnan. Large-scale optimistic adaptive submodularity. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, pages 1816–1823, 2014.

19   Ashish Goel and Piotr Indyk. Stochastic load balancing and related problems. In *40th Annual Symposium on Foundations of Computer Science (Cat. No. 99CB37039)*, pages 579–586. IEEE, 1999.

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

21   Alkis Gotovos, Amin Karbasi, and Andreas Krause. Nonmonotone adaptive submodular maximization. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

22   Anupam Gupta, Ravishankar Krishnaswamy, Marco Molinaro, and R Ravi. Approximation algorithms for correlated knapsacks and non-martingale bandits. In *Foundations of Computer Science (FOCS), 2011 IEEE 52nd Annual Symposium on*, pages 827–836. IEEE, 2011.

23   Steven C. H. Hoi, Rong Jin, Jianke Zhu, and Michael R. Lyu. Batch mode active learning and its application to medical image classification. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pages 417–424, New York, NY, USA, 2006. Association for Computing Machinery. `doi:10.1145/1143844.1143897`.

24   Jon Kleinberg, Yuval Rabani, and Éva Tardos. Allocating bandwidth for bursty connections. *SIAM Journal on Computing*, 30(1):191–217, 2000.

25   Jian Li and Wen Yuan. Stochastic combinatorial optimization via poisson approximation. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 971–980. ACM, 2013.

26   Will Ma. Improvements and generalizations of stochastic knapsack and markovian bandits approximation algorithms. *Mathematics of Operations Research*, 43(3):789–812, 2018. `doi:10.1287/moor.2017.0884`.

27   G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functions-i. *Mathematical Programming*, 14(1):265–294, 1978. `doi:10.1007/bf01588971`.

28   Jan Vondrák. Symmetry and approximability of submodular maximization problems. In *2009 50th Annual Ieee Symposium on Foundations of Computer Science*, pages 251–270, October 2009. `doi:10.1109/focs.2009.24`.

29   Sheng Yang, Samir Khuller, Sunav Choudhary, Subrata Mitra, and Kanak Mahadik. Scheduling ML training on unreliable spot instances. In *2021 IEEE/ACM 14th International Conference on Utility and Cloud Computing (UCC '21) Companion*, 2021. `doi:10.1145/3492323.3495594`.

30   Sheng Yang, Samir Khuller, Sunav Chowdhary, Subrata Mitra, and Kanak Mahadik. Correlated stochastic knapsack with a submodular objective. *CoRR*, 2022. `arXiv:2207.01551`.

31   Baosheng Yu, Meng Fang, and Dacheng Tao. Linear submodular bandits with a knapsack constraint. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 1380–1386, 2016.

32   Yisong Yue and Carlos Guestrin. Linear submodular bandits and their application to diversified retrieval. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc., 2011. URL: `https://proceedings.neurips.cc/paper/2011/file/33ebd5b07dc7e407752fe773eed20635-Paper.pdf`.